NOTE TO USERS

This reproduction is the best copy available.

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Jared Michael Vawater
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical and Computer Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Alternative Pointing Devices :
Design and Evaluation of a System Supporting Multiple Users

TITRE DE LA THÈSE / TITLE OF THESIS

Emil Petriu
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Jean-François Lapointe
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Gabriel Wainer

Voicu Groza

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Alternative Pointing Devices: Design and Evaluation of a System Supporting Multiple Users

by

Jared M. Vawter

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-73864-1
Our file Notre référence
ISBN: 978-0-494-73864-1

Abstract

This thesis evaluates alternative pointing devices for interacting with a video display. The challenge of identifying where a user is pointing is defined in terms of either the direct mapping problem or the indirect mapping problem. In the former, users are assumed to interact with a display surface using a laser pointer. In the latter, users are assumed to interact with a display surface using a hand-held camera.

Two approaches are evaluated as potential solutions to the direct mapping problem: homography estimation, and nonlinear point-to-point mapping. These solutions use an external camera to capture the location of the laser pointer and construct a mapping between pixels of the image plane and pixels of the video display to determine where a user is pointing. Similarly, two approaches are evaluated as potential solutions to the indirect mapping problem: a variation of Tsai's extrinsic calibration algorithm, and the Nintendo Wii pointing device. These solutions, by contrast, determine where a user is pointing based on the relative position of feature points on the image plane.

To support concurrent users, techniques for differentiating among the users are evaluated. For the direct mapping solutions, two novel approaches are presented: off-the-shelf shape-based laser pointers, and custom shape-based laser pointers. In these approaches, each user is assumed to hold a laser pointer with a unique geometry. Differentiating among the users then becomes the challenge of identifying which shapes are cast on the display at a given instant. Through both simulation and trials on real image data, it is shown that custom shape-based laser pointers provide a superior solution. For indirect mapping class of solutions, a simple yet effective approach based on unique identifiers is pursued for supporting an arbitrary number of concurrent users.

Competing solutions are compared using a proposed set of evaluation metrics. Specifically, the solutions are compared with respect to measures of static accuracy, latency, jitter, drift, and reliability. The effects of Kalman state estimation on dynamic accuracy are also explored. For the context considered, an ideal solution is one providing the greatest amount of accuracy and reliability while minimizing the observed latency, jitter, and drift. Furthermore, an ideal solution would support an arbitrary number of concurrent users.

For a fixed size display, results show the static accuracy of the direct mapping solutions to be within 0.317% of the distance along the display diagonal and are independent of the user's location relative to the display surface. For the indirect mapping solutions, the change in static accuracy is measured as a function of the user's location. Results show the static accuracy decreases as the angle between the pointing device and the display normal increases. For angles beyond 30°, estimates using the variant of Tsai's extrinsic algorithm become unreliable; for the Nintendo Wii, estimates become unreliable beyond 50°. The Nintendo Wii is found to have a maximum operating range of approximately 4 m from the display - irrespective of the display size. For the variant of Tsai's extrinsic algorithm, the maximum operating range could not be established; however, accurate estimates were recorded consistently beyond a distance of three times the display diagonal.

Each approach is shown to have negligible jitter and drift. The Nintendo Wii, homography, and nonlinear mapping solutions are found to have 100% reliability for static and dynamic pointing. The reliability is also found to be independent of the speed of the pointing device - up to the maximum speed that could be generated by hand. The trajectory of motion was chosen at random. For the variant of Tsai's extrinsic algorithm, the reliability is found to decrease modestly as the speed of motion increases. Over 500 trials, the reliability decreases by at most 6% for the maximum speed considered. Using Kalman state estimation, the dynamic accuracy can be improved by up to 45% when using a three-state dynamic model. The direct mapping solutions and the variant of Tsai's extrinsic algorithm are found to have an expected latency of 33 ms. By comparison, the Nintendo Wii pointing device has an expected latency of just 16 ms.

This thesis finds that no single solution is ideal for all applications. Depending upon the performance constraints and design specifications, it is plausible for either a direct mapping solution or an indirect mapping solution to provide the most suitable implementation. When choosing among competing solutions it is important to carefully consider the advantages and shortcomings of each. In general, this thesis finds direct mapping solutions to have the greatest range of motion for a given measure of accuracy. Indirect mapping solutions are found to support the largest number of users but have a bounded range of motion. The Nintendo Wii is shown to be more robust to changes in position but, unlike the variant of Tsai's extrinsic algorithm, cannot recover the pose of the pointing device.

Acknowledgements

I would like to thank the National Research Council for giving me the opportunity to work on this challenging yet stimulating problem. I am thankful for the financial support they have provided, for the structured work hours, and for their world-class facilities.

I would also like to thank my two co-supervisors: Dr. Jean-François Lapointe of the National Research Council and Dr. Emil Petriu of the University of Ottawa. Their direction, expertise, and encouragement during my years of study have made a world of difference.

Above all, I give thanks to my amazing bride-to-be for sticking with me throughout the tough times and for providing me with my single greatest source of motivation. I owe my success to your unwaivering support and belief in my abilities. I dedicate this work to you.

Contents

List of Tables

List of Figures

List of Symbols

f	Camera focal length
O	Camera center of projection
π	Camera image plane
Oz	Camera optical axis
C	Image center
P	A point in the camera coordinate system
p	The projection of P onto the image plane
M	A point in the display coordinate system
m	The projection of M onto the image plane
q	Image pixel
$\hat{q}$	Estimated cursor position in the output video
(X, Y, Z)	World coordinate system
(x, y, z)	Camera coordinate system
(i, j, k)	Pixel coordinate system
(U, V, W)	Display surface coordinate system
$\mathbf{R}$	3x3 camera rotation matrix
$\mathbf{T}$	3x1 camera translation matrix
θ	Camera rotation about y-axis
ϕ	Camera rotation about x-axis
ψ	Camera rotation about z-axis
λ	Threshold for image segmentation
κ	Camera distortion coefficient
(x_d, y_d)	Distorted camera coordinate
R	Translation ratio
(d_x, d_y)	Camera sensor element dimensions
$h(A, B)$	Directed Hausdorff distance metric
E_x	Translation error along the x-axis

E_y	Translation error along the y-axis
E_z	Translation error along the z-axis
$\mathbf{x}_k$	System state vector
$\mathbf{z}_k$	System output vector
$\mathbf{u}_k$	System input vector
$\mathbf{w}_k$	Process noise vector
$\mathbf{v}_k$	Measurement noise vector
$\mathbf{\Phi}_k$	State transition matrix
$\mathbf{\Psi}_k$	Input dynamics matrix
$\mathbf{\Gamma}_k$	Process noise weighting matrix
$\mathbf{H}_k$	State measurement matrix
$\mathbf{R}_k$	Measurement error covariance matrix
$\mathbf{Q}_k$	Process error covariance matrix
$\mathbf{P}_k$	Estimation error covariance matrix
$\mathbf{K}_k$	Kalman gain matrix

Chapter 1

Introduction

1.1 Alternative Input Devices

Since their invention, the keyboard and mouse have dominated as the primary means for users to interact with a computer system. Quite naturally, when first introduced to a computer, most individuals will learn and become accustomed to interacting with their computer system through these two peripheral devices. An interesting by-product of the learning process is that users also become accustomed to a standard of human-computer interaction where they expect to sit in front of the computer screen with the mouse and keyboard positioned on a desk in front of them.

Historically, with the mouse and keyboard being physically tethered to the computer, this was indeed the only practical means for users to interact with the computer system. With the introduction of wireless keyboards and mice in the 1990s, the standard of human-computer interaction began to evolve as users welcomed the ability to be freed of the physical connection with their computer systems.

Breaking this physical connection proved to be a pivotal step toward changing the schema of how users can and should interact with a computer system. By this time, the field of human-computer interaction was already quite mature within the academic

community where, for years, researchers had been at work studying the effects and pondering new possibilities for human-computer interaction. Indeed, the keyboard, mouse, and computer monitor each evolved from some of the earliest efforts in the field.

The introduction of wireless keyboards and mice was by no means a great advancement in technology but it did serve to change how millions of individuals around the world view interaction with a computer system. As wireless keyboards and mice became increasingly popular and more readily available, individuals became accustomed not only to a new means of human-computer interaction but also to having a choice in their means of interaction.

Over the years, the physical design of the keyboard and mouse have also evolved, primarily to become more ergonomic and to improve the efficiency of the end-user. With a computer in nearly every western household and with an increasing number of individuals spending their workday in front of a computer, the need for devices tailored toward prolonged use has never been greater. Advancements both within academia and within industry have led to ergonomic keyboards and mice shaped specifically to fit the natural orientation of the human hand.

Custom keyboards and mice now come equipped with additional buttons which can be programmed to perform custom operations as needed by the end-user; providing increased productivity and ease-of-use. Advancements in technology coupled with demand for custom designs have even led to keyboards for right and left-hand dominant users as well as keyboards targeted toward physically disabled users. Research into assistive technologies, designed to assist disabled users to interact with a computer, has been at the forefront of revolutionizing the devices and means of human-computer interaction.

As the means of interaction have changed, so too has the amount of information that can be exchanged between a human and a computer system. Research and development efforts over the past several decades have led to highly effective voice recognition capabilities which can now be used to interact with a computer system. With these capabilities, computer systems effectively evolved from a simple two-input system to a three-input system and, as a practical extension, paved the way for multi-input systems in today's modern computers.

As a by-product of increasing the number of input methods it became reasonable to envision multiple users interacting concurrently with a single computer system. With the traditional keyboard and mouse it seemed that at most two users could interact practically with a single computer. However, as new applications and operating systems have been developed to leverage these new and alternative input methods, there has been a shift toward more collaborative human-computer interaction.

Modern operating systems and video game consoles have brought collaborative user environments to the masses and are now pushing the frontiers of alternative input devices. Operating systems, such as Windows 7, have been designed to accommodate multi-user configurations by supporting multi-touch functionality with touch-capable monitors. The Nintendo Wii is perhaps the most widely-recognized gaming console supporting up to four concurrent users using an alternative input device termed the "Wiimote". The Wiimote is a wireless device which acts much like a mouse pointer with additional buttons for increased functionality. The device uses an infrared camera to track known feature points to estimate where a user is aiming. It is equipped with gyroscopes and accelerometers to gauge the orientation as well as the force exerted along different axes which can be used to augment the experience of the user and enables developers to provide far more interactive content.

Microsoft is currently developing its next generation of gaming console with reports of project Natal being aimed at revolutionary means of human-computer interaction. Their technology is reported to eliminate the need for an input device altogether and instead uses a camera to track the gestures and body movements of users. With research and development efforts from both industry and academia, the field of human-computer interaction has arguably never seen such a large amount of interest. It is evident the demand from the consumer is high and that individuals have become accustomed to, and perhaps even expect, new changes in how they interact with a computer system.

This thesis is concerned with the pointing aspect of human-computer interaction: that is, techniques and devices which replicate the functionality of a mouse. The context envisioned is a first-person shooter combat training simulator designed to support multiple users interacting concurrently. It is assumed each user is physically detached from the host computer system and is free to move about the training environment. Several methods of alternative pointer interaction are presented and evaluated with respect to

a set of core evaluation criteria. Based on the findings, recommendations are made for an implementation of the best-performing solution given the specific context. In the concluding section a discussion of future work and possible improvements is provided.

1.2 Problem Overview and Motivation

In partnership with the Canadian Department of National Defence, the National Research Council of Canada has been conducting research in several subject areas to augment and improve combat training facilities located in Gagetown, New Brunswick. Research into areas such as voice recognition, cognitive modeling, and alternative pointing devices has led to several unique capabilities which are currently being evaluated and improved by both researchers and soldiers.

This thesis documents research and development efforts on the viability of alternative pointing devices for use in the combat training simulator. The training environment envisioned comprises a video projector and projection screen showing a first-person shooter "video game". The video projector is positioned to have an unobstructed view of the projection screen and is connected to a host computer running the training simulator. The host computer accepts input from one or more users and relays the inputs to the training simulator. Given this context, one of the primary objectives of this thesis is to develop capabilities for supporting multiple users interacting at once.

Two classes of solution are evaluated for use as alternative pointing devices: direct mapping and indirect mapping. The direct mapping solutions assume each user holds a laser pointer and generates system inputs by aiming the laser pointer directly at the projection screen. In this configuration, an external camera is positioned in the training environment and is assumed to have an unobstructed view of the projection screen. With each frame captured by the camera, the laser spots are detected on screen, processed by the host computer, and assigned to one of the users present in the environment using an identification schema. The position of the laser spot on the image plane is then used to estimate the location where a user is aiming.

For the indirect mapping configuration, a collection of identifiable feature points are

carefully positioned about the perimeter of the projection screen whose locations in a local coordinate system are accurately known. Each user is then assumed to hold a commodity camera with an unobstructed view of the feature points. With each frame captured by the camera, the relative position of the feature points on the image plane are used to estimate the position and orientation of the camera with respect to the projection screen. Once the pose of the camera has been recovered, a ray is traced from the camera's optical center to the projection screen and used to estimate the location where a user is aiming.

Each class of solution has its respective advantages and disadvantages which are explored and evaluated as part of this thesis. To assess the viability of a particular solution, a set of core evaluation metrics are proposed and used to objectively compare competing solutions. Specifically, measures of static accuracy, dynamic accuracy, latency, jitter, drift, and reliability will be utilized. Furthermore, in multi-user settings, it is critical to accurately distinguish which inputs are generated by which user. A solution which fails to address these design and performance considerations will undoubtedly face resistance from the end-user and will certainly fail to enhance the training experience. The solutions considered here are designed explicitly to support multiple users and their effectiveness to do so is critically evaluated.

1.3 Existing Solutions

1.3.1 Indirect Mapping

After a comprehensive review of existing work on direct mapping and indirect mapping solutions, there was found to be only one original effort to use indirect mapping for controlling the position of a cursor on screen: the Nintendo Wii. After its release, enthusiasts and hobbyists have collectively been able to reverse engineer the Nintendo Wii's input device [20] and have since developed many custom applications using the Wii's technology.

Efforts from the community have led to some ingenious applications of the Wii's

input device owing to it's unique all-in-one design, comprising an infra-red camera, an accelerometer, 12 buttons, and Bluetooth connectivity all at a relatively low cost. The performance of applications developed using this device are, however, inherently limited by the commodity hardware available on-board the device. Although exact specifications have not been confirmed, the internal camera is reported to have a resolution of 128x96 which is up-sampled to 1024x768 but has a respectable refresh rate of 100 Hz. Image processing is performed in hardware on the device and enables at most four feature points to be tracked at once.

Most applications developed by the community infer the cursor position based on the relative location of only two features points on the image plane. The primary reason being that the Nintendo Wii ships with a sensor bar, equipped with infra-red LEDs at opposite ends of the bar, and positioned either above or below a user's television screen. The infrared LEDs project two feature points on the image plane which can be used to infer the location of the cursor on screen.

One major drawback to these approaches is that the true position and orientation of the internal camera cannot be recovered. Even if the camera could be accurately calibrated, two (coplanar) feature points do not provide sufficient information to recover the camera pose. Indeed, [21] indicates that as many as four possible camera poses are possible when tracking three feature points and that tracking four feature points still leads to two plausible poses.

Experiments conducted with the Nintendo Wii system have demonstrated the estimated location of the cursor is reasonably accurate, provided a user stands within 4 meters of the projection screen and within about 50 degrees from the midpoint of the screen. Errors in the estimate of the cursor position are indeed noticeable but are well-masked because no visual feedback of the true location where a user is pointing is made available.

To provide a realistic training environment, the solution sought in this thesis would ideally be as accurate as possible over the widest range of distances and angles of incidence. In theory, if the pose of the camera can be accurately recovered, then the estimate for the cursor position will also be accurate. An additional advantage of pursuing a solution which estimates the pose of the camera is that it can provide invaluable feedback

to soldiers about their performance. For example, if a soldier fails to hold his or her firearm upright while firing, this information would be available and could be used assist in their training. An indirect mapping approach based solely on the Nintendo Wii would be incapable of providing this critical feedback.

1.3.2 Direct mapping

Many works in the literature [1–19] provide solutions to the direct mapping problem as an alternative means of pointer input. A large proportion of these solutions provide accurate estimates for the cursor position, most of which are implemented by constructing some form of mapping between camera coordinates and the pixel coordinates of the host computer. Based on the works reviewed, research into this class of solution dates back to the late 1990s and has progressively evolved to produce systems with greater accuracy, reliability, and reduced latency.

With the exception of the works found in [2, 10, 15–17], all solutions evaluated were designed explicitly to support just a single user. In many situations and for many practical applications, there is no need to support multiple users. However, the objectives set forth in this thesis are to support several users interacting concurrently with the training simulator. A natural extension to the single user case, as proposed in [2, 16], is to have each user hold a uniquely coloured laser pointer and to have as many cameras as there are users in the environment. If each camera is fitted with a bandpass filter allowing light from a single laser to pass through, then single user systems can easily be extended to accommodate multiple users.

An alternative approach to supporting multiple users is to apply a sort of time-division multiplexing as implemented in [10, 15–17]. In this scenario, each laser pointer is powered on then off in turn using a predefined cyclic pattern. The number of concurrent users that can be supported is a function of the pattern length and structure. The length is equivalent to the number of frames required to identify which users are present. The structure defines when each laser is turned on. For example, using a binary blink on structure of length three, there can be three concurrent users supported: 001, 010, 100. In this scenario, one pointer is on in the first frame, one in the second, and the third

pointer in the final frame.

The main disadvantage to this approach is that it causes latency to increase proportionally with pattern length. Using a binary coding schema with pattern length n, and assuming a camera frame rate of F, at most n users can be supported at a rate of F/n. In theory, with a sufficiently high frame rate and adequate pattern length, this approach could support a large number of users with limited latency. Unfortunately, the cameras selected for research and development were limited to a maximum frame rate of 30 fps. At this rate, a maximum of five users could only be identified once every 6 frames - resulting in a high degree of latency.

1.4 Contributions

This thesis presents a number novel contributions specific to direct mapping, camera pose estimation, image segmentation, state estimation, and performance evaluation.

Direct mapping: A new approach to supporting multiple users is proposed and evaluated under the assumption that each user holds a laser pointer casting a unique geometric pattern on the projection screen. A training phase is described where the system constructs templates for each laser pattern and proceeds with frame-by-frame supervised classification using two unique approaches: the Hausdorff distance metric and the circular Hough transform. A discussion of the complexities incurred to implement each classifier are then discussed.

Camera Pose Estimation: New findings on the coplanar form of the POSIT algorithm [21] for estimating camera pose from known feature points are provided. Results of the authors are showed the errors in estimating both translation and rotation depend on the so-called distance ratio. Simulations provided in this thesis illustrate the errors are additionally dependent on a new metric termed the *translation ratio* - a measure of the relative magnitudes of the components of the translation vector.

A new variation on Tsai's external camera calibration method [23] is proposed for cameras with wide-angle lenses. A novel three-step process to estimate camera pose

is then described. First, the external parameters are estimated using Tsai's method. Second, image correction is applied to reposition the feature points on the image plane. Third, the external parameters are re-estimated using Tsai's method. This approach is shown to be considerably more accurate than applying Tsai's external calibration alone under the experimental setup used in this thesis.

Image Segmentation: Two novel image segmentation algorithms are introduced which are capable of identifying and labeling unique components in an image by performing just a single pass over pixels in the image. One approach is shown to have linear performance characteristics while the second is shown to have nonlinear characteristics. These approaches each provide a novel alternative to the component labeling algorithm detailed in [22].

A novel method to aggregate pixels from multiple components is also introduced using a technique termed *successive relaxations*. The basic idea is to apply local averaging and to successively add pixels in a local neighbourhood to a component until convergence. Applications to both direct mapping as well as indirect mapping are described and shown to increase robustness to fast-moving laser spots and feature points.

State Estimation: An application of Kalman state estimation is presented to predict the future location of a laser pointer during dynamic pointing. The system dynamics are modeled using a Taylor series expansion of the laser pointer position under two specific conditions: constant acceleration and constant velocity. Using these assumptions, two state-space representations are defined and used to construct two separate Kalman systems. It is shown that state estimation can significantly improve the dynamic accuracy of a given pointing solution.

Performance Evaluation: This thesis presents a comparison of direct mapping and indirect mapping solutions for use in alternative pointing devices. Comparisons between competing solutions are made using a set of performance evaluation metrics for both static and dynamic pointing. Simulation results are also presented to evaluate the computational efficiency of each solution and to identify the most computationally demanding portions of their implementation. The effects of using Kalman state estimation to improve dynamic accuracy are also evaluated and shown to provide significant performance gains.

1.5 Outline

This thesis continues in Chapter 2 with a literature review of relevant research in the field of alternative pointing devices. A description of the common experimental configuration used in the field is first provided. Focus is then placed on techniques derived to address the direct mapping as well as indirect mapping classes of solution. A review of works designed to identify multiple users is provided and followed by a review of solutions for interpreting mouse click events. The literature review then concludes with a review of techniques for estimating the pose of a camera from a collection of known feature points.

In Chapter 3 a mathematical formulation for the direct mapping and indirect mapping classes of solution is provided. Key nomenclature and formulae are introduced which are used throughout the thesis. The direct mapping class of solutions is described in terms of a mapping problem where pixels on the image plane are directly mapped to a cursor position for input to the training simulator. The indirect mapping class of solutions is described in terms of a mapping problem where pixels on the image plane are indirectly mapped to the cursor position.

In Chapter 4 image processing techniques are described for acquiring the images, performing image segmentation, labeling components, and for detecting and identifying feature point correspondences. Two novel approaches for component labeling are described and compared with respect to their computational complexity. Also, a novel algorithm for identifying feature point correspondences is presented.

In Chapter 5 the importance of camera calibration for the indirect mapping problem is discussed, specifically using Tsai's approach for coplanar feature points. A discussion of key parameters such as focal length, image center, radial distortion, camera translation, and camera rotation is provided in the context of the indirect mapping problem. The experimental setup, procedure, and results obtained are then provided.

In Chapter 6 specific techniques for addressing the direct and indirect mapping problems are provided. For the direct problem, an approach based on homography estimation is presented. For the indirect problem, techniques using coplanar feature points for estimating camera pose are described. Specifically, the coplanar POSIT algorithm, Tsai's

extrinsic calibration algorithm, and a novel modification of Tsai's algorithm are evaluated as possible techniques for recovering the camera pose. Assuming the camera pose can be accurately recovered, an algorithm is then provided for controlling the cursor location in the output video.

In Chapter 7 techniques for distinguishing between multiple users are described. For the direct mapping class of solutions, two techniques are described and compared using simulation results: off-the-shelf laser pointers with unique geometric patterns, and custom-designed laser pointers with unique patterns. For the indirect mapping class of solutions, a simple solution using unique identifiers is described.

In Chapter 8 arguments are presented for using Kalman state estimation to improve the dynamic accuracy of a given solution. A mathematical formulation is provided for the case of two and three state estimation.

In Chapter 9 the performance evaluation criteria are introduced and formally defined. Definitions are provided for static accuracy, dynamic accuracy, latency, jitter, drift, and reliability along with a detailed description of the experimental setup and procedure. Results are provided for the performance evaluation of the direct and indirect mapping-based solutions. A performance evaluation of the Nintendo Wii pointing device is also provided for comparison.

In Chapter 10 concluding remarks and recommendations based on the findings herein are presented. The primary objectives are briefly revisited before discussing the advantages and disadvantages of the direct and indirect mapping solutions. A summary of contributions and a proposal for future work are then provided in the closing sections.

Chapter 2

Literature Review

2.1 A Common Configuration

A literature review of existing research in alternative pointing devices [1–19] highlights a common experimental configuration used for tracking user input. The configuration observed in each of the works reviewed comprises a hand-held laser pointing device, a video projector, a projection display, a host computer, and an external camera. A typical setup where the camera and video projector are positioned behind the projection screen is shown in Figure 2.1.

The research presented in this thesis uses a similar experimental configuration to the one shown. For the direct mapping class of solutions considered, a video projector and external camera are assumed positioned on the same side of the projection screen as the users. For the indirect mapping class of solutions considered, each user holds their own commodity camera - eliminating the need for an external camera. In this scenario, feature points are carefully positioned about the perimeter of the projection display and are tracked by each individual camera.

A comprehensive review of related research has found that, with the exception of the Nintendo Wii, each existing approach focuses exclusively on the direct mapping class

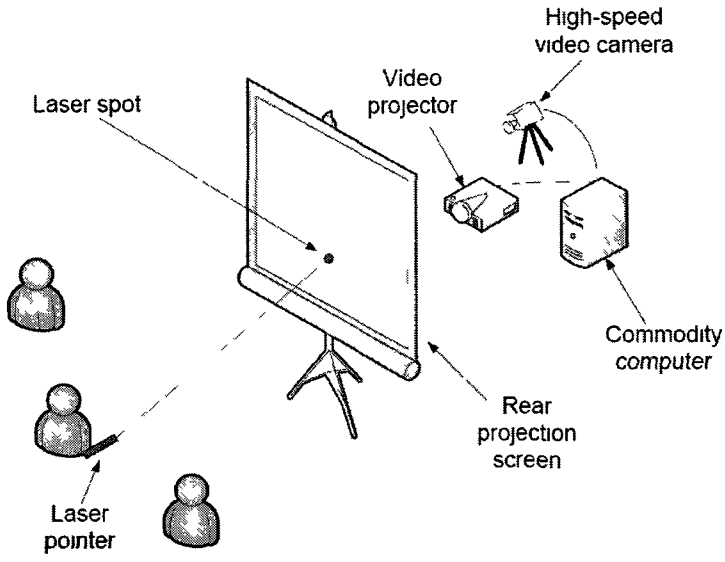

Figure 2.1: A common experimental configuration for tracking user input (reproduced from [9]).

of solution for generating user inputs. Some of the solutions reviewed have been developed for single-user environments, some have been developed for multi-user environments without identifying which input is derived from which user, and some provide "complete" solutions for tracking and identifying inputs from multiple users. A subset of these works are now reviewed in greater detail.

2.2 Laser Pointer Acquisition

In each of the works reviewed, users hold a laser pointing device directed toward the projection screen. In [4, 12, 13], the authors investigate using infra-red laser pointers. In all other works reviewed, the authors use laser pointers operating in the visible spectrum.

An approach common to all solutions uses the external camera to capture images of the environment and applies a thresholding technique to identify the location of the laser spot on the image plane. When using infra-red laser pointers, the external camera must be capable of detecting infra-red sources which is most often achieved using a bandpass filter positioned over the lens of the camera. Regardless of the wavelength used, every

solution evaluated determines the location of the laser on the image plane as the pixels with intensity above some predetermined threshold.

Specifics of the thresholding procedure were documented in [2, 5, 8, 9, 11, 13, 15–19]. In [2,5,9,11,15–19] the pixels residing above the specified threshold within an image frame are clustered into groups by aggregating adjacent pixels. The mean of the aggregated pixels is then used as a sub-pixel estimate of the location of the laser pointer on the image plane. In [8], the authors search for the pixel with greatest intensity and use it as an estimate for the laser position on the image plane. However, they provide no indication of how multiple pixels with maximum intensity are treated. In [13], the authors detect a laser pointer within a single PAL field by fitting an ellipse about the pixels above the specified threshold. The centroid of the ellipse is then used as the location of the laser on the image plane.

2.3 Estimating the Cursor Position

With the location of the laser pointer known on the image plane, each of the works reviewed proceed to estimate where the mouse cursor should be positioned with respect to the projected video of the host computer. It is important to note that the resolution of the camera and output video sequence need not be the same. Ideally, the cursor would be positioned directly at the point of intersection of the laser pointer and the projected video. However, the accuracy of the estimate depends highly on the technique established for mapping image coordinates to coordinates of the host system. In the works reviewed, a description of the mapping process was provided in [5, 6, 8, 9, 11, 13, 16, 18, 19].

The most commonly used technique, found in [5, 9, 11, 13, 18], establishes a homography matrix directly mapping image to computer coordinates. By projecting a set of feature points on the projection screen with known computer coordinates and by identifying the corresponding features on the image plane, a homography between the two coordinate systems can be established. Provided there are a sufficient number of feature points, an over-determined set of equations can be used to obtain a homography which is optimal in a least squares sense. Once a homography has been established, points in the image plane can be mapped to their corresponding computer coordinates by solving

a system of linear equations.

A slight variation of the homography approach described in [13] partitions the output video sequence into a series of sub-regions and computes a homography matrix for each region. The authors contend their approach accounts for nonlinear distortions introduced by most commodity cameras and, as a result, can increase the accuracy of image to computer coordinate mappings.

In [6] the authors project a series of 25 points on the projection screen whose coordinates in the output video sequence are known. The corresponding points on the image plane are then identified to establish 25 pairs of points. A least squares approximation is then used to compute the coefficients for polynomials with degree -1 to 3. The learned polynomials are then used to map image coordinates to computer coordinates. This approach is similar, in a sense, to a homography in that a system of equations is used to construct the mapping between image and computer coordinates. It is the formulation of the equations that differentiates the techniques from one another.

Alternative techniques for estimating the cursor position are mentioned in [8, 16, 19] but without details of their implementation. In [8], a mapping between image and computer coordinates is established using the known correspondence between a line in the world and its projection on the image plane. In [16] and [19] the authors indicate a perspective transformation is used to establish the mapping but do not specify how this is achieved.

2.4 Distinguishing Multiple Users

In a multi-user configuration, it is essential to identify which users are interacting with the computer system and at which location they are pointing. Distinguishing between multiple users can be performed independently from, but should be performed prior to, estimating the cursor position. This leads to a simple three-step process for generating user input: acquire, distinguish, estimate. Of all the works reviewed, only a few efforts were designed with the objective of concurrently supporting multiple users [2, 10, 15–17, 20] from which three unique solutions were found.

In the first solution, a rather simple yet effective means for supporting multiple users is to have each user hold a laser pointer with a different colour [2,16]. Such a configuration requires there to be as many cameras as there are users and for each camera to be fitted with a bandpass filter allowing the light from one laser pointer to pass through it. Effectively, for each laser, there must be a camera with a bandpass filter at the same wavelength as the laser itself. In many situations this is an effective approach, provided a sufficient number of lasers with different spectra are available.

The second solution found in the literature uses a blinking pattern to turn the laser pointers on and off over a series of frames [10, 15–17]. The basic idea is that, over the set of frames, each laser pointer will be on for a fraction of the frames and in a specific order. At the end of the set, a decoding schema is used to determine which laser pointers were on during the set. For example, a blinking pattern could be 001 for one laser, 010 for a second, and 100 for a third. The number of frames in the set would be three, with the first laser pointer on for the first frame, the second pointer on for the second frame, and the third pointer on for the final frame.

The number of frames required for a set is determined by the number of users being supported and the coding schema. Table 2.1 provides three coding schemes, each of length three, and the patterns used to blink each pointer. The patterns are read in order from right to left, with a 1 indicating the laser is on during that frame, and a 0 indicating it is off.

Laser Pointer	Blink-On	Binary	Blink-Off
Laser Pointer 1	001	001	110
Laser Pointer 2	010	010	101
Laser Pointer 3	100	011	011
Laser Pointer 4	N/A	100	N/A
Laser Pointer 5	N/A	101	N/A
Laser Pointer 6	N/A	110	N/A
Laser Pointer 7	N/A	111	N/A

Table 2.1: Blinking patterns for three coding schemes (reproduced from [16]).

The primary disadvantage to using a blinking pattern is that it introduces additional latency proportional to the length of the pattern being used. Depending on the frame

rate of the camera, this may be unnoticeable to the user and may still provide a highly responsive system. A disadvantage specific to the binary coding schema is that only a single laser pointer can be active over the set of frames. For example, if laser pointer 1 and 2 were active in the same set, it would be impossible to distinguish between these from laser pointer 3 being active.

One additional disadvantage is that it requires the blinking pattern to be sequenced with the external camera to ensure all on / off events are detected. In [15] the authors propose using a frame capture operating at twice the blinking rate and introducing a small, random amount of delay at the start of each laser pattern to mitigate detection errors. The purpose of the random delay is to evenly distribute the likelihood of missing an on / off event over many frames instead of them coming in at once.

The third solution, used by the Nintendo Wii and documented in [20], implements an indirect mapping approach to estimating the cursor position and distinguishes multiple users by assigning a unique identifier to each hand-held camera. Because the Nintendo Wii pointing devices communicate with the host over Bluetooth, each device is automatically assigned a Bluetooth ID which the host uses to distinguish among the data arriving from different users. This approach enables a theoretically large number of users to be supported and a definitive way to distinguish between them.

2.5 Interpreting Input Events

An alternative pointing device, designed with the objective of replacing a standard mouse, should provide functionality beyond simply controlling the cursor position. Actions such as right, left, and double-clicking are essential components in many desktop applications. In the context of a training simulator, a mouse click can be considered analogous to pulling the trigger on a firearm and constitutes an essential part of an effective solution. Many works [1, 3–6, 11, 12, 14, 15, 17, 19] in the literature have addressed this design consideration leading to solutions for mouse clicking, scrolling, and controlling zoom among others.

In [1], the authors describe a technique called "goal-crossing" and demonstrate its

usage for controlling a slide show and for interacting with a map viewer. The technique uses the four edges of a projection screen as the goals and interprets movement of the laser pointer across these goals as different interaction methods. In their work they define three specific crossing events: in-out, out-in, and in-out-in. By "in" the authors mean the laser pointer is within the boundaries of the projection screen. They also propose linking the events together to obtain further functionality, such as: out-in, in-out, out-in, etc.

Depending upon the application, these crossing events can have different interpretations. When controlling a slide show, the out-in crossing is used to alternate between pages. When used on the left portion of the display it changes to the next page, and when used the right portion of the page it changes to the previous page. When controlling a map viewer, the out-in crossing is used to control the pan. Depending on which edge the goal-crossing occurs, the map will pan in different directions.

In [3, 5, 15, 17] the authors use wireless buttons to replicate the left and right buttons of a standard mouse. In [3] the authors use two buttons mounted on the pointing device connected by radio frequency to a receiver which communicates standard PS/2 events the host computer. In [15] a wireless link is established between the pointing device and a USB base station connected to the host computer. Buttons are incorporated into the laser pointer and events are interpreted on the host machine to generate the appropriate inputs.

In [5] the authors utilize an infra-red transmitter and receiver to communicate mouse click events between the pointer and host machine. The transmitter is a standard remote control unit and interfaces with a Linux infra-red controller. It is not specifically stated whether the remote control is physically connected to the laser pointer or not. In [17] the authors connect a custom laser pointer fitted with a button to a pocket PC which interprets events from the laser pointer and communicates them to the host computer.

The works presented in [4, 12] make use of mouse gestures to generate inputs to the host computer. In [4], the authors design a system to recognize gestures over objects and sweeps across action bars to generate the inputs. Each action is initiated by turning the laser pointer on, following a specific path on the projection screen, and finishing by turning the laser pointer off. In their work, the gestures are context-dependent and have

different effects depending on which objects are interacted with. The use of action bars is similar to the goal-crossing technique where directing the laser pointer across a given bar initiates a unique action such as click or select.

In [12] the authors define an interaction schema using gestures and what they term "hotspots". Hotspots are defined as areas surrounding an object which change appearance when the cursor is positioned over them. In their work, the hotspots are used as a way to select objects without having to issue a click command. They define gestures as specific paths traced by the laser pointer on the projection screen. For example, circling around an object with the laser pointer initiates a select command.

The solution presented in [6] makes use of dwell times and what they term "interactors" to interpret context-specific mouse events. Their approach is unique in that they focus on the information available from a specific desktop application to define the interactive techniques. For example, the interactive techniques defined for a button would be different from those of a drop-down menu. Using a custom architecture, they enable buttons, list menus, and text items to implement widgets defining the possible interaction techniques.

For buttons, when a laser pointer dwells in a bounded area for several frames, the system interprets the action as a button click. For a list menu, dwelling on the menu will show the list of choices and initiate a scrolling or tracking phase to enable the user to make a selection. This approach presents a unique interaction environment but requires significantly more coding as each application needs to be modified to implement the set of widgets.

In [11] the authors interpret dwell times and on / off events to generate mouse events. If the laser pointer is detected in the same approximate location for several frames, a mouse click event is generated. If the laser pointer is detected as being turned off for several frames, a button release event is generated. Using this approach the authors are able to simulate basic clicking as well as selection capabilities.

A technique implementing a laser pointer and pocket PC is presented by the authors in [14]. Their solution is to either have a pocket PC in one hand and a laser pointer in the other, or to use a pocket PC fitted with an internal laser pointer. Using the laser

to position the mouse cursors, the pocket PC is then used to generate and communicate input events to the host computer. A novel "snarfing" schema is defined by the authors where portions of the projection display are captured by a separate application and transmitted to the pocket PC to enable additional interaction. With this approach, they are able control the position of the cursors, navigate drop down menus, enter text, and control the zoom of regions transmitted to the pocket PC.

2.6 Estimating Camera Pose

An essential component of solutions based on indirect mapping is the ability to accurately estimate the translation and rotation of a camera with respect to the display surface. This pose estimation problem has been the subject of much research since the early 1980s and has led to a number of solutions for the cases of both coplanar and non-planar feature points. The coplanar case is concerned with feature points whose relative depths can be considered zero. In this thesis, the chosen approach is to determine camera pose from coplanar feature points and, therefore, the solutions presented in [21, 23–26] are of greatest interest.

In [21] the authors describe an iterative method for estimating camera pose from N feature points. Their approach begins by computing a scaled orthographic projection of the known feature points under the assumption the focal length of the camera is known a priori. A further assumption is made that, when the relative depths of the feature points are small, scaled orthographic projection provides a good approximation to perspective projection - which indeed holds for coplanar feature points.

Upon computing an initial approximate pose, the algorithm iteratively re-computes more exact poses until convergence is achieved. When the camera is far from the projection screen, the algorithm converges within just a few iterations. If, however, the camera is closer to the projection screen, the image will have strong perspective and the algorithm will require more iterations to converge. Due to the nature of how pose is estimated, this approach always produces two possible camera poses.

In [23] the author describes a complete camera calibration procedure: estimating both

the intrinsic as well as extrinsic (translation and rotation) parameters of the camera from N known feature points. Provided the intrinsic parameters are known a priori, specifically the camera center, focal length, and distortion coefficient, the author describes a straightforward technique for computing the extrinsic parameters alone. The approach is founded on a novel radial alignment constraint which imposes that vectors in the plane of the feature points are parallel to their projections on the image plane.

Using the pin-hole camera model and the equations of perspective projection, the author describes a technique for efficiently recovering the camera pose provided $N \geq 7$. With $N > 7$ an overdetermined system of linear equations can be established which allows the extrinsic parameters to be refined through nonlinear optimization. The author presents an error measure for the complete calibration procedure (extrinsic and intrinsic) which is shown to be proportional to $1/\sqrt{N}$. Accordingly, increasing N tends to decrease the error in estimating the camera pose.

In [24] the authors describe techniques for recovering camera pose from 4, 5, and N feature point correspondences. Their approach constructs vectors x_i from the known feature points to the unknown location of the camera's center of projection in world coordinates. Pairs of these vectors (x_i, x_j) are then used to construct a series of quadratic equations of the form $f_{ij}(x_i, x_j) = 0$. Using the Sylvester resultant, a series of eighth-degree polynomials in each of the x_i and of the form $g(x_i) = 0$ can be derived.

The equations $g(x_i)$ can then be used to recover the translation of the camera, which equates to the unknown location of its center of projection, as well as the rotation. Simulation results presented by the authors show the 5-point algorithm consistently outperforms the 4-point algorithm with respect to relative error measures in both translation and rotation.

The random sample consensus algorithm (RANSAC), widely used in many fields beyond machine vision, is presented by the authors in [25]. Contrary to classical optimization techniques, the RANSAC algorithm does not attempt to fit the best model to *all* data observations. Instead, the smallest number of data points n required to estimate the parameters of a model are sampled at random N times and used to estimate the parameters of the model. In the context of camera pose, the model parameters are the translation and rotation vectors, while the model itself could be any pose estimation

solution.

For each set of n data points sampled, the camera pose is estimated and the number of data points consistent with the estimate is recorded. The points consistent with the pose estimate are considered to be the inliers. The trial N' producing the largest number of inliers is deemed to provide the best estimate for the camera pose over the N trials. If there are a sufficient number of points $n \in N'$, least squares optimization of the translation and rotation components is then applied.

In their paper, the authors estimate camera pose from the perspective projection of three known feature points. This approach is a 3-point version of the technique described in [24] which results in four possible pose solutions. The primary advantage of this approach is that it enables any model to be used for estimating pose and enables the computational complexity to be controlled by fixing the parameter N. For the case of a large number of feature points n, this approach may be significantly quicker than performing an optimization over all n points.

In [26] the author presents an iterative solution for estimating camera pose. The solution requires an initial estimate for the pose then iteratively converges to a more accurate pose using Newton's method. To avoid converging on a local minimum, the author describes stabilization methods which make the approach much more effective in practice. By writing the translation and rotation parameters collectively as $\mathbf{p}$, Newton's method iteratively computes a vector of corrections $\mathbf{x}$ to be subtracted from the estimate of $\mathbf{p}$ at iteration i:

$$\mathbf{p}^{i+1} = \mathbf{p}^{i} - \mathbf{x}$$

The primary disadvantage of this technique is that it requires an initial estimate $\mathbf{p}^0$ for the camera pose. The author indicates that Monte Carlo simulations for this approach produce correct results provided the errors in the initial rotation estimates are less than 90 degrees and that errors in the initial translation estimates have little impact. In many situations, assuming the camera is facing an object of interest and is oriented approximately upright is sufficient to ensure the initial estimates are within these error bounds. When the errors are near 90 degrees, the approach converges within just 6 iterations. Using the stabilization methods proposed by the author, convergence is achieved within even fewer iterations.

Owing to the popularity and availability of open-source implementations of the approaches described in [21, 23], these two techniques will be implemented and evaluated as potential indirect mapping solutions for use in the combat training simulator.

Chapter 3

Problem Formulation

3.1 Experimental Setup and Notation

The experimental setup used throughout this thesis for testing and evaluation is shown in Figure 3.1. The camera used for capturing images for both the direct mapping and indirect mapping solutions is shown mounted on a tripod in the left portion of the image. The projection screen is positioned at the end of the room and is clearly visible in the center of the image. The video projector is sat atop a small wooden table in the lower right portion of the image. Both the camera and video projector are connected to a desktop PC which is positioned outside the limits of the image.

The classic pinhole camera model is assumed as shown in Figure 3.2. The camera has an image plane π and a center of projection O which serves as the frame of reference for the camera coordinate system. The distance between the image plane and center of projection is the focal length f. The center point of the image C denotes the point of intersection between the image plane and the optical axis Oz. A point $P = (X, Y, Z)$ in the camera coordinate system is projected onto the image plane at $p = (x, y, z)$. Because $z = f$ is a constant in the pinhole model, the projection of P on the image plane is commonly written in short form as $p = (x, y)$.

Figure 3.1: Experimental setup used for testing and evaluation.

The focal length f, points P, and projections p are all measured in metric units. The location of the image center C is defined in terms of pixel units $(C_x, C_y) = (i, j)$ with respect to a coordinate system defined on the camera sensor array. For a camera positioned upright and facing the projection screen, the upper-left pixel $(0, 0)$ is used as the frame of reference. All pixels are then defined in terms of integer units i along the horizontal and j along the vertical. The number of horizontal and vertical sensor elements provides the upper bounds on i and j.

A camera is assumed to have N_{cx} sensor elements along its horizontal and N_{cy} sensor elements along its vertical. The host computer is assumed to be running at a native resolution of N_{hx} by N_{hy} which is illuminated by the video projector onto the projection screen at a resolution of N_{vx} by N_{vy}. Points $M = (U, V, W)$ on the projection screen are described in a local metric coordinate system, relative to the top left of the screen, as shown in Figure 3.2. Because the screen is assumed to be rigid and planar, all points M can be considered to have constant depth $W = 0$. The projection of M on the image plane is given by $m = (x, y)$. Pixel coordinates in the output video will be denoted as (u, v).

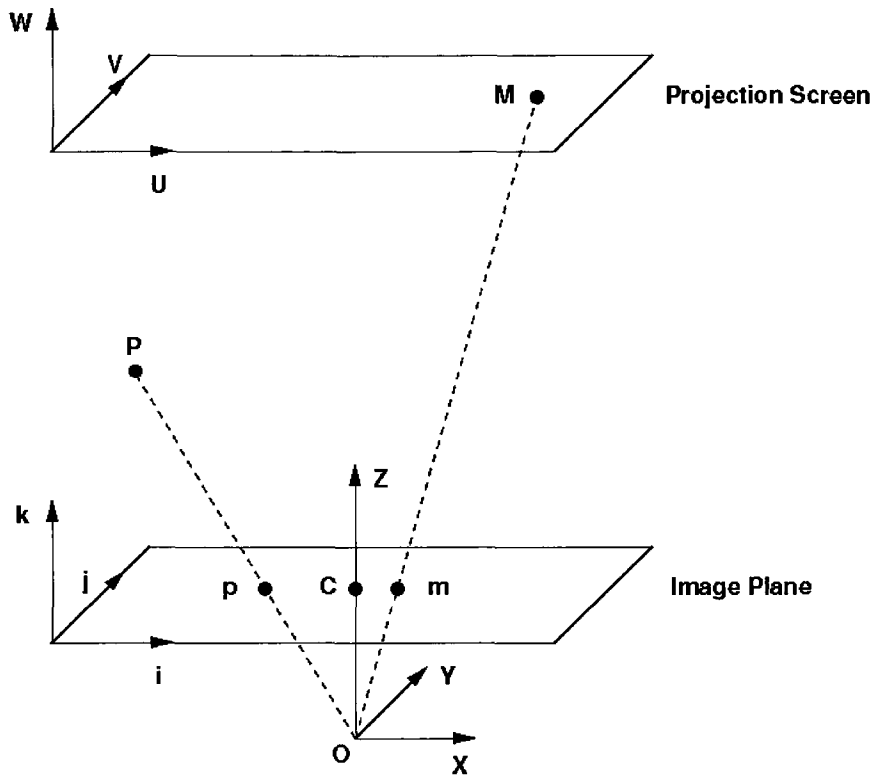

Figure 3.2: Pinhole camera model, projection screen, and coordinate axes.

In certain situations, it is convenient to abuse notation and allow m or p to refer to the pixel coordinates of a projected point on the image plane instead of the camera coordinates. In such cases, camera coordinates will be denoted (x, y) and pixel coordinates as (i, j). If the metric dimensions (d_x, d_y) of a sensor element are known, it is trivial to convert between camera coordinates and pixel coordinates by noting:

$$i = \frac{x}{d_x} + C_x \qquad j = \frac{y}{d_y} + C_y \tag{3.1}$$

3.2 The Direct Mapping Problem

Given the notation and experimental configuration described, the direct mapping class of solutions can now be presented. When a camera captures an image of the laser pointer directed at the projection screen, the laser spot produces a distinctive collection of bright pixels on the image plane. Assuming the laser pattern forms points $\mathbf{M} =$

$\{M_1, M_2, \ldots, M_n\}$ on the projection screen, the camera should observe the projections of these points on the image plane as $\mathbf{m} = \{m_1, m_2, \ldots, m_n\}$.

If a single laser pointer is used, the mean of the points $\bar{m} = \frac{\sum_{i=1}^{n} m_i}{n}$ can be computed to obtain a sub-pixel estimate of the laser projection on the image plane. If N laser pointers are used, the points $m_i \in \mathbf{m}$ will need to be clustered into non-overlapping components leading to a set of centroids $\bar{\mathbf{m}} = \{\bar{m}_1, \bar{m}_2, \ldots, \bar{m}_N\}$.

The direct mapping problem then becomes the challenge of establishing a function $\mathcal{F}$ which, for a given centroid $\bar{m}_i$, produces an estimate $\hat{q} = (u, v)$ of which pixel the laser pointer is directed at with respect to the output video sequence. The position of the mouse cursor can then be controlled by positioning the cursor at the estimated location. Because $\hat{q}$ is computed directly from $\bar{m}_i$ using $\mathcal{F}$, this will be referred to as the direct mapping problem.

3.3 The Indirect Mapping Problem

The indirect mapping problem aims to identify a function indirectly mapping points on the image plane to pixels in the output video. For this class of solution, a collection of feature points $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$ are assumed to be positioned about the projection screen. As before, the projection of these feature points on the image plane are given by $\mathbf{m} = \{m_1, m_2, \ldots, m_n\}$.

In a multi-user environment with N users, each user holds their own respective camera, making it important to distinguish between the projections of the feature points on different image planes. Let $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ denote the image planes of each user such that, for user k, the projections of the feature points can be written as $\mathbf{m}|_{\pi_k}$. From these projections, a function $\mathcal{G}$ is sought which estimates a transformation aligning the camera frame of reference with the frame of reference of the projection screen. The correct transformation, shown in Figure 3.3, comprises a translation component $\mathbf{T}$ and a rotation component $\mathbf{R}$ which, collectively, define the pose of the camera.

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\phi + \cos\psi\sin\theta\cos\phi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

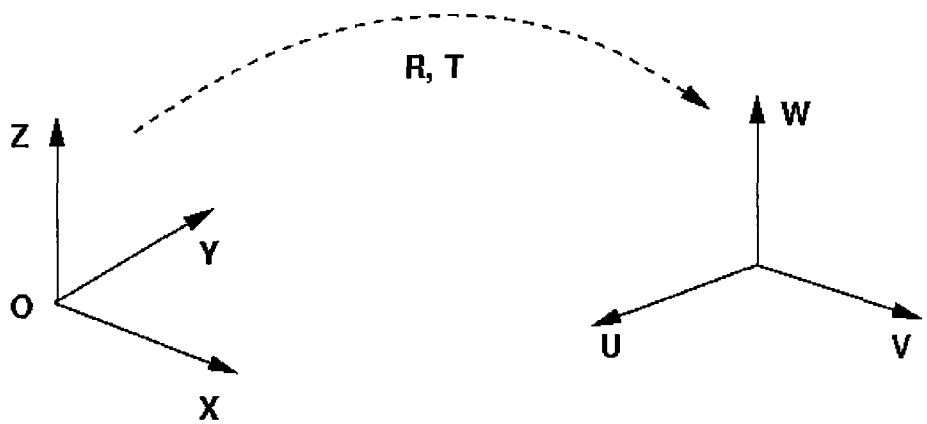

Figure 3.3: Transformation aligning the camera and projection screen frames of reference.

The translation component defines a vector connecting the point O with the origin of the projection screen coordinate system. The rotation component describes a matrix aligning all three coordinate axes of the camera and projection screen frames of reference. The rotation component can be considered the result of three independent rotation matrices: a rotation about y by an angle θ, a rotation about x by an angle ϕ, and finally a rotation about z by an angle ψ. These rotations correspond to the standard yaw, pitch, and roll angles.

Once the translation and rotation components have been recovered, a function $\mathcal{H}$ is sought which maps the estimated camera pose to an estimate $\hat{q} = (u, v)$ of where the cursor should be positioned with respect to the output video sequence. The complete mapping can be expressed as a composite function $\mathcal{F} = \mathcal{H} \circ \mathcal{G}$. Because $\hat{q}$ is computed indirectly from $\bar{m}_i$ using $\mathcal{F}$, this will be referred to as the indirect mapping problem.

Chapter 4

Image Processing for the Mapping Problem

4.1 Image Acquisition

The cameras used for testing produce greyscale images with a maximum resolution of 1024x768 and frame rate of 30 fps when using 8-bit intensity images. Each camera is also fitted with a variable focus lens.

The camera is connected to a host PC through an IEEE-1394 connection capable of a maximum data throughput of 400 Mbps. When operating at a resolution of 1024x768 and with 8-bit intensity, a single image occupies 6.29 Mbps of bandwidth. This means transferring a single image from the camera to the host takes a minimum of 15.73 ms. With the camera operating at 30 fps, this leaves just 17.60 ms to process the image before another one is ready to be transferred. If images are captured continuously, and if the image processing requires more than 17.60 ms, then the system will not be able to produce a frame-by-frame estimate for the cursor position.

Each camera comes with a software development kit (SDK) containing a C/C++ application programming interface (API), device drivers, and a camera utility application

for configuring the camera parameters. The camera utility application provides control over many of the camera's parameters, including: frame rate, resolution, brightness, and exposure time among others. The application also provides access to advanced features such as region of interest selection and custom image formats. Using the pixel binning feature, the camera can be set to acquire images at a resolution of 512x384 with 8-bit intensity and a maximum frame rate of 60 fps.

Using the pixel binning feature and assuming the same IEEE-1394 connection, each image would occupy just 1.57 Mbps of bandwidth and could be transferred to the host computer in 3.93 ms. Fixing the frame rate at 30 fps as before, a total of 29.40 ms can now be allocated to the image processing routines. This provides considerably more time to allow the system to provide frame-by-frame estimates for the cursor position. This, of course, comes at the expense of decreasing the accuracy of the system as the camera resolution is effectively halved.

Images acquired through the SDK are stored in an `Image` structure which stores the image in an unsigned character array with length equivalent to the number of pixels. The structure also provides ready access to information about the image such as the number of pixels per row and per column, the number of bytes per row, the video mode of the image, a timestamp for when the image was captured, a flag indicating whether the image is greyscale or colour, and the pixel format of the image.

```
struct Image {
   unsigned char *pData;
   int iRows;
   int iCols;
   int iRowInc;
   VideoMode videoMode;
   Timestamp timeStamp;
   bool stippled;
   PixelFormat pixelFormat;
}
```

4.2 Thresholding and Successive Relaxations

Once an image has been captured it is essential to locate the projection of the laser pointers or the feature points on the image plane. These points of interest are assumed to produce pixels which are brighter than all other objects in the scene. This requires the ambient lighting in the target environment to be carefully controlled to ensure this criteria is met. Internal lighting, as well as light coming from outdoors, tends to produce pixels near the maximum intensity on the image plane. By changing the iris setting on the lens and the shutter time of the camera utility application, it is possible to ensure the lighting criterion is met.

Image thresholding is a simple process with complexity of $\mathcal{O}(n)$ in the number of image pixels. By using the camera utility application to configure the camera, it is possible to manually select a threshold λ where each pixel $q \geq \lambda$ on the image plane belongs to either a laser spot or one of the known feature points. When setting the camera shutter time to be 15 ms, fixing the frame rate at 30 Hz, and setting the gain to be 15 dB, a threshold of $\lambda = 230$ is sufficient to ensure accurate thresholding when the ambient lighting is around 10 Lux. Points $q \geq \lambda$ which are adjacent to one another in an 8-connected sense will collectively be referred to as a component.

Because the shutter time of the camera is finite and non-zero, interest points on the image plane tend to become "smeared" from fast movement of the pointing device. As a result, some pixels which would normally be above the threshold actually fall below it. When this occurs, straightforward application of the thresholding procedure tends to produce a larger number of means $\bar{\mathbf{m}}$ than desired. To correct for this effect and to produce more accurate estimates of where a user is pointing, a technique known as successive relaxations is applied immediately after thresholding.

Successive relaxations is a recursive algorithm that iteratively grows the image components until there is no change in any of the components (i.e. until convergence). Let the pixels found through straightforward thresholding be denoted as $\mathbf{q}^{(0)} = \{q_1, q_2, \ldots, q_N\}$ and define the 8-neighbourhood of a pixel to be the pixels $\mathbf{p} = \{p_1, p_2, \ldots, p_8\}$ adjacent to q_i.

In the first iteration, the average value over the 8-neighbourhood of each pixel q_i is computed. If any pixel p_j in the 8-neighbourhood is above the average it is added to a new set $\mathbf{q}^{(1)}$. At the next iteration, the average value over the 8-neighbourhood of the pixels $q_i \in \mathbf{q}^{(1)} \setminus \mathbf{q}^{(0)}$ is computed and the pixels above the average are added to a new set, and so on. This process continues recursively until, at iteration k, $\mathbf{q}^{(k)} \equiv \mathbf{q}^{(k-1)}$. An example showing the original image, the image after thresholding, and the image after successive relaxations is shown in Figure 4.1.

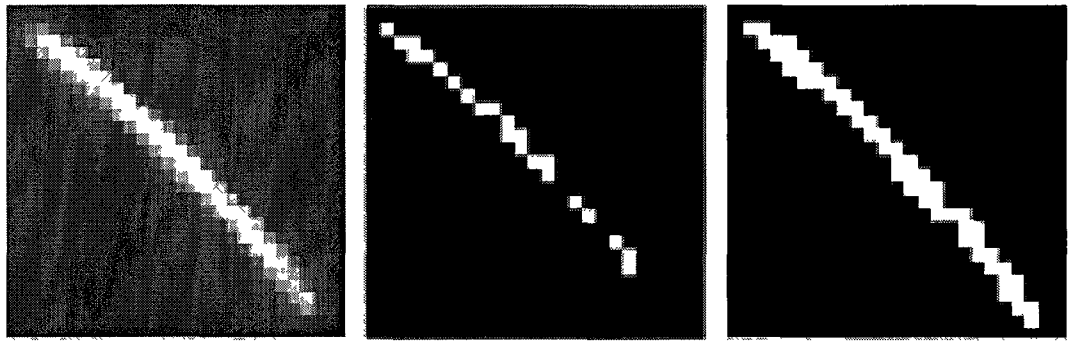

Figure 4.1: Sample images showing the effects of thresholding (center) and successive relaxations (right) applied to the original image (left).

From the figure shown, applying thresholding alone produces an image with three components which could be easily misinterpreted as three unique laser pointers active on screen at the same time. Even if it were known a priori that a single laser pointer was active, taking the centroid of the three means would produce a less accurate estimate of the cursor's true position when compared to using successive relaxations. It is evident that applying successive relaxations helps to maintain the original structure of the laser pointer's projection on the image plane which leads to a more accurate estimate of the centroid of the laser spot.

An important point to note is that the increase in accuracy comes at the cost of increased computational complexity. Depending upon the number of interest points on the image plane after thresholding, the added computation time may or may not be justifiable. For the case of a simple laser spot on screen, the number of interest points tends to be quite low; so too does the number of recursive function calls needed to implement the relaxation algorithm. However, for indirect mapping solutions, the number and size of the feature points impacts how many pixels lie above λ and may indeed add significant latency.

At the end of this chapter, simulations are provided for the amount of latency introduced by a variable number of interest points. It is shown that as the number of interest points increases, the computation time required to apply successive relaxations increases beyond what is available before a new image is transferred from the camera to the host computer. Evidently, as this point it becomes a design trade-off between accuracy and latency as to whether or not implementing successive relaxations is justified.

4.3 Component Labeling

Once thresholding and, if chosen, successive relaxations have been applied to an image, it is necessary to aggregate adjacent pixels into components. In image processing and machine vision this is often called component labeling because a unique label is assigned to each of the aggregate pixel clusters. The application of component labeling is useful primarily from a software implementation perspective. If each component is labeled and stored in an appropriate structure, or object, then each can be accessed by its label and specific operations can be applied individually.

In this thesis, two novel component labeling algorithms were implemented and used for testing. Details of their implementation are, however, currently protected under intellectual property rights. These approaches will hereafter be referred to as the "linear-time" and "nonlinear-time" component labeling algorithms.

4.4 Simulation Results

To evaluate the computational complexity of the two component labeling approaches, 500 simulated images with random component locations are generated and used to record the computation time. Images of the same resolution as the camera are simulated by allocating an array of dimension 1024x768 with entries set to zero. Pixels within a grid positioned at the center of the image and of length W are then populated with values chosen from a Gaussian distribution with mean μ and standard deviation 3. Successive

relaxations are then applied to pixels $q \geq \lambda$ where the threshold is fixed at $\lambda = \mu + 2$. The computation time is recorded for $\mathbf{W} = \{5, 10, 20, 30, 40, 50, 60, 70, 80\}$ and plotted against $\mathbf{W}^2$. Box plots for the nonlinear-time approach are shown in Figure 4.2 and for the linear-time approach in Figure 4.3.

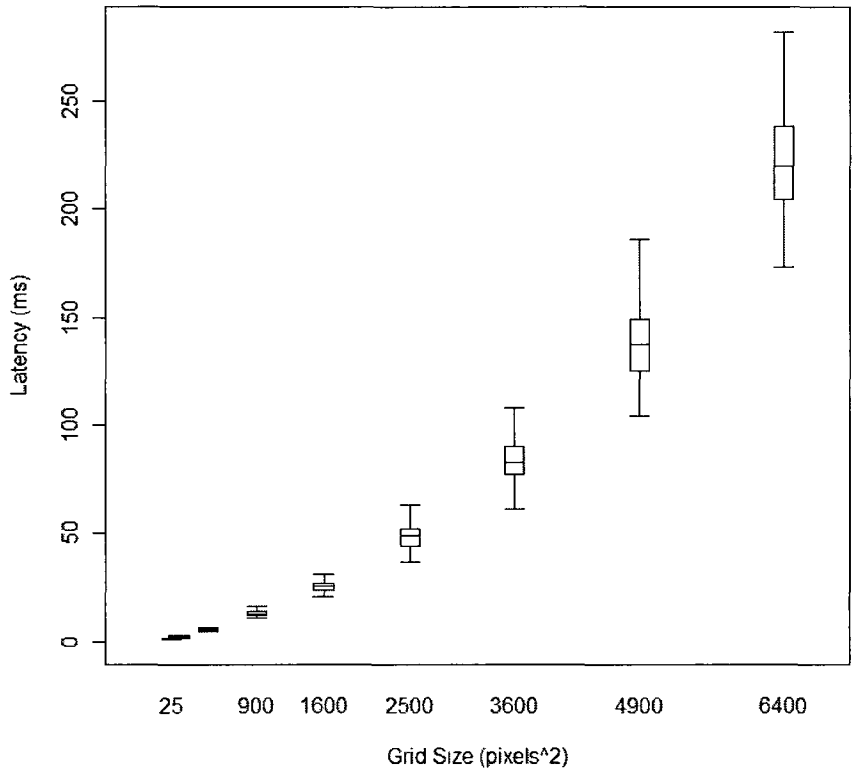

Figure 4.2: Computation time of component labeling using the nonlinear-time algorithm.

Figure 4.2 illustrates the computation time of the first approach follows a nonlinear trajectory as the grid size increases. Figure 4.3 shows how the computation time of the second approach follows a more linear trajectory. The box plots also indicate that the variance in computation time remains constant for the second approach but increases, quite noticeably, for the first approach.

An interesting characteristic shows that the first approach provides a computational advantage over the second approach for grid sizes below $W \approx 60$. Indeed, a plot of the mean computation times provided in Figure 4.4 helps to highlight this characteristic.

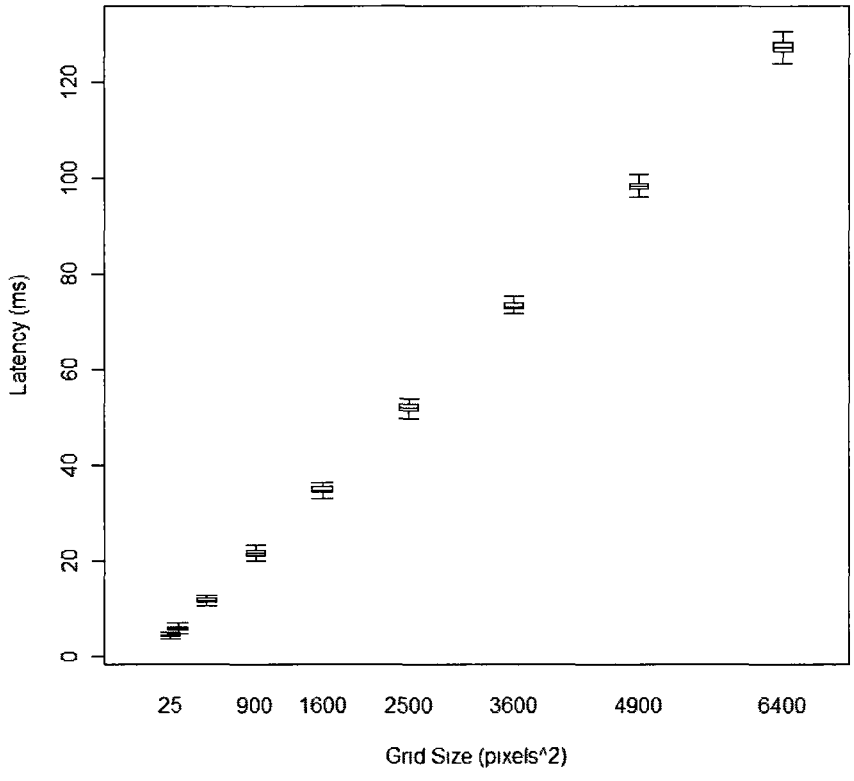

Figure 4.3: Computation time of component labeling using the linear-time algorithm.

Using a Gaussian distribution to generate the pixel intensities and running the simulations over 500 images ensures the number of pixels $q \geq \lambda$ is equivalent, on average, for the two approaches. This also ensures the number of components arising should, on average, also be equivalent. The most plausible explanation for the difference arises from how hash maps are used to assign and re-assign component labels.

In the first approach, a placeholder hash map is used which increases in size with every new component label assigned. It also must be searched every time two or more neighbour labels have different values. Using a Gaussian distribution to generate pixel intensities, the number of components and the number of pixels $\geq \lambda$ increase linearly as a function of the grid size. Accordingly, there will be a greater number of neighbour labels that disagree. This implies the hash map both increases with grid size and requires searching more often - leading to the nonlinear characteristic observed.

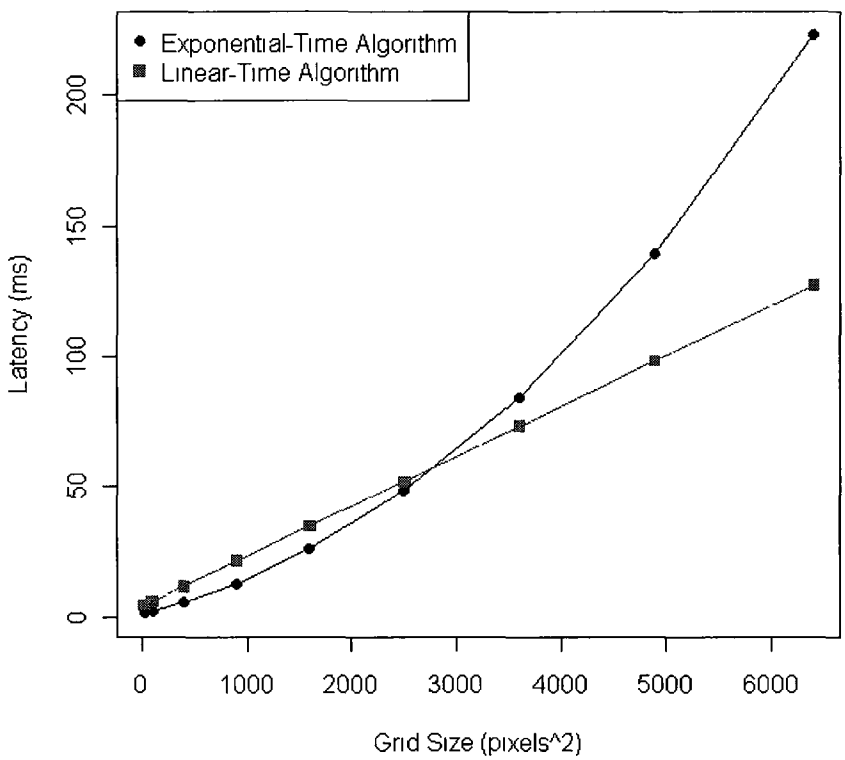

Figure 4.4: Computation time as a function of grid size for the two component labeling algorithms.

The computation time required to apply successive relaxations as a function of grid size is shown in Figure 4.5. The plot illustrates that both the mean and variance of the computation time are linear in the size of the grid. For grid sizes below $W = 20$, the mean computation time is less than 15.56 ms. As introduced at the beginning of this chapter, 17.60 ms are available for image processing before another is ready to be transferred from the camera. Clearly, for $W > 20$, applying successive relaxations would introduce at least one complete frame of latency.

With $W = 20$, it is important to point out an average of seven recursive function calls were made to the successive relaxations algorithm. At iteration k, this led to an average of 35 pixels belonging to $\mathbf{p}^k$ whose neighbours needed to be considered for inclusion in $\mathbf{p}^{k+1}$. Fortunately, the laser pointers used in this thesis project a pattern of fewer than 35

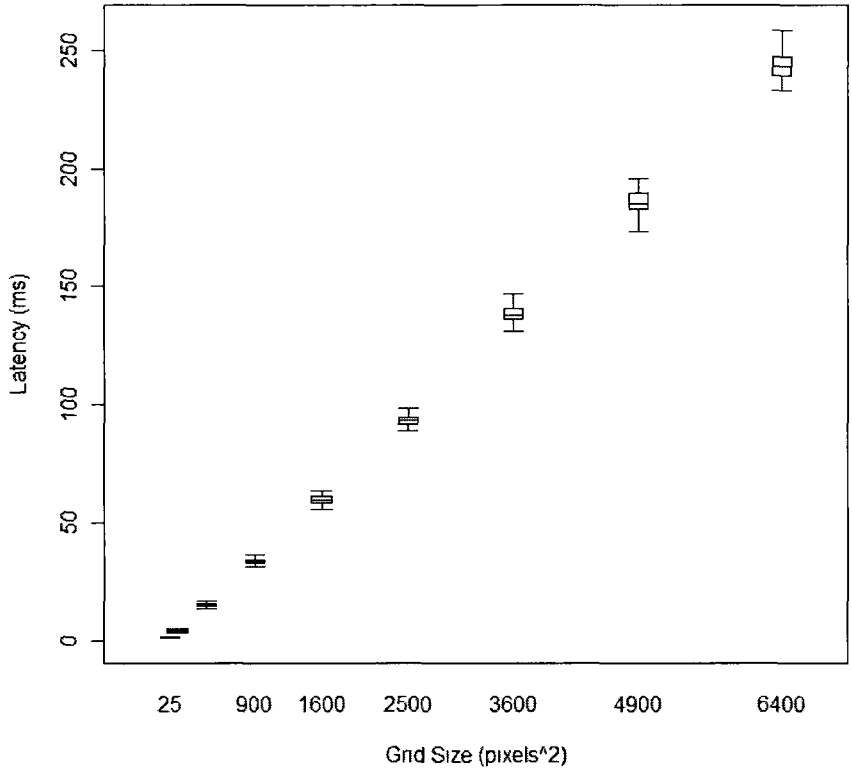

Figure 4.5: Computation time as a function of grid size for computing successive relaxations.

pixels and, because the distribution of pixel intensities is more uniform and concentrated above λ, only one or two function calls to the successive relaxations algorithm are required - reducing the actual computation time needed from that found through simulation. In fact, the simulation results represent a worst-case scenario for the expected computation time which is rarely, if ever, observed in practice.

4.5 Feature Detection and Identification

For solutions implementing indirect mapping, it is essential to establish a correspondence between the location of the features in the world coordinate system and the projection of these features on the image plane. By using specific features that can be positioned at arbitrary locations, it is possible to construct geometries which cast a distinctive pattern on the image plane regardless of the translation or rotation of the camera. The approach taken here assumes the features points are arranged in the world coordinate system as illustrated in Figure 4.6 and that each feature point is visible by the camera.

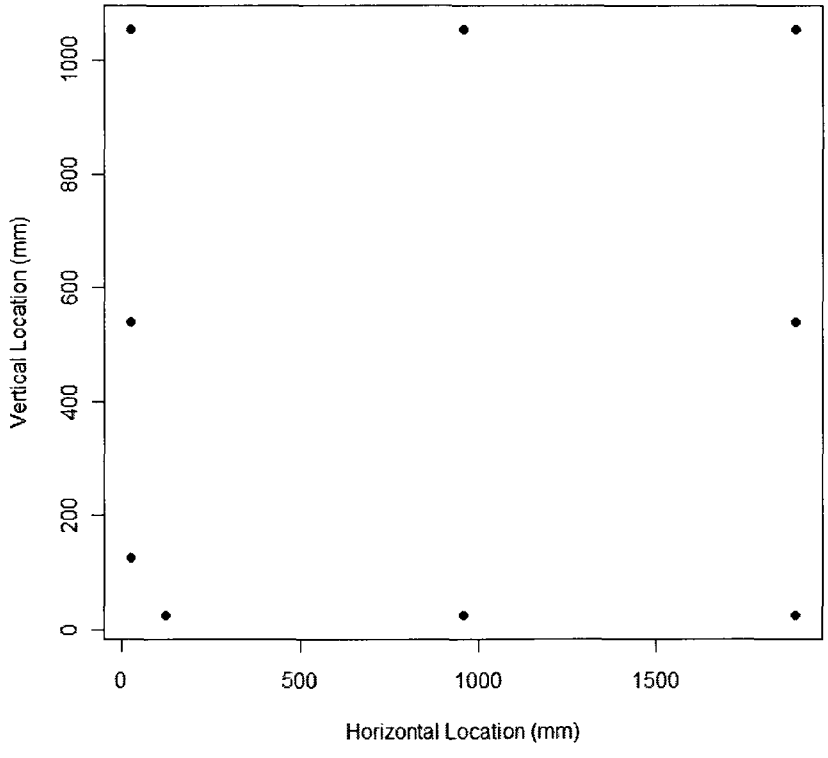

Figure 4.6: World coordinate system feature point arrangement.

As shown, there are two feature points clustered in the lower left portion of the image. The point furthest left of these two serves as an anchor point from which all others are defined in clockwise order. To obtain the correspondences between the world and image coordinates, it is sufficient to identify this anchor point within the image and then order

all other points clockwise relative to it. To obtain this ordering, the centroid $B = (B_i, B_j)$ of all components is computed from the centroids $b_k = (b_{ik}, b_{jk})$ of each individual one. For each component, a vector v_k joining B to the centroid of the component is computed as $v_k = (b_{ik} - B_i, b_{jk} - B_j)$.

To order the points in clockwise order, a simple "right turn" predicate is used. For each pair of vectors v_i and v_j, the cross product between the two vectors is computed. If the sign of the cross product is negative, then vector v_j is to the right of vector v_i in a clockwise direction. By computing the cross products for all pairs of vectors it is possible to order the vectors in clockwise order. Since the angle between two vectors can also be obtained from the cross product, it is possible to identify the anchor as belonging to vector v_j forming the smallest clockwise angle with vector v_i over all pairs of vectors.

Chapter 5

Camera Calibration

Camera calibration is a critical component of the indirect mapping solutions evaluated in this thesis. Calibration enables specific parameters of the camera to be recovered so that a composite function $\mathcal{F} = \mathcal{H} \circ \mathcal{G}$ can be established. Feature points on the image plane are related to camera pose via $\mathcal{G}$ and to a cursor position on the projected video sequence via $\mathcal{H}$. The camera calibration technique described here is Tsai's calibration algorithm [23] which enables the focal length, image center, radial distortion, aspect ratio, camera translation vector, and camera rotation matrix to be recovered from a single image containing feature points with known world and image correspondences.

5.1 The Pinhole Camera Model

The classic pinhole camera model introduced in Section 3.1 is assumed. To briefly recap, the camera has a center of projection O which serves as the frame of reference for the camera coordinate system. The distance between the image plane and O is the focal length f. The center point of the image is C. A point $P = (X, Y, Z)$ in the camera coordinate system is projected onto the image plane at $p = (x, y, z)$. With $z = f$ a constant in the pinhole model, the projection of P on the image plane can be written as $p = (x, y)$. Points p are defined with respect to the center point of the image.

Given the pinhole model and notation described, the fundamental equations of perspective projection can now be introduced. Perspective projection is a technique for mapping points defined in three dimensions (X, Y, Z) with respect to an origin O onto a plane with fixed depth, also defined with respect to O. For the pinhole camera model, this equates to mapping points P defined with respect to the center of projection onto the image plane as follows:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{5.1}$$

Because of the factor $1/Z$ in the perspective projection equations, these expressions are inherently nonlinear and do not preserve distances between points or angles between lines [27]. Assuming the relative distances between the points are small compared to the average distance from the center of projection $\bar{Z}$, these equations can be approximated with a linear expression known as the weak-perspective model. For the case of points which are essentially coplanar, this approximation becomes especially useful and enables the equations to be re-written as:

$$x = f\frac{X}{Z} \approx f\frac{X}{\bar{Z}} \qquad y = f\frac{Y}{Z} \approx f\frac{Y}{\bar{Z}}$$

This expression combines two well-known and important transformations: an orthographic projection and isotropic scaling. Using the orthographic projection, points P are projected onto a common plane located at a distance $\bar{Z}$ from the camera's center of projection. Isotropic scaling then projects these points onto the image plane as p through perspective projection using $f/\bar{Z}$ as a scaling factor.

The perspective projection and weak-perspective equations have been presented here because they form a fundamental basis in camera calibration, camera pose estimation, and the solutions presented for the composite function $\mathcal{F}$. Tsai's camera calibration procedure relies on perspective projection for the case of non-planar feature points and on the weak-perspective model for feature points which are essentially coplanar. The coplanar POSIT algorithm, described in Section 6.2.1, derives an estimate for the camera pose using the weak-perspective model. Finally, the solution for $\mathcal{F}$ uses coordinates on the image plane (x, y) to estimate their coordinates (X, Y) by inverting the perspective projection equations.

5.2 Camera Parameters

Camera calibration enables the *extrinsic* and *intrinsic* parameters of the camera to be estimated. By knowing the locations of specific feature points **M** in a local coordinate system and by establishing their corresponding projections **m** on the image plane, camera calibration can be readily achieved. The extrinsic parameters refer to the camera translation vector **T** and camera rotation matrix **R** as described in Section 3.3. To simplify some notation, let the entries of the rotation matrix be written as:

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{5.2}$$

The intrinsic parameters are used to characterize the optical, geometric, and digital characteristics of the camera. The focal length f, required for the projection equations previously described, is clearly an important intrinsic parameter of the camera. From the equations presented in Section 3.1, it is possible to express the camera coordinates in terms of the pixel coordinates provided the metric dimensions (d_x, d_y) of the sensor elements and the image center (C_x, C_y) are known. The dimensions of a sensor element are often provided in the camera specifications, which leaves just the focal length and image center to be estimated.

An additional intrinsic parameter to recover arises from the distortion introduced by the camera lens. Depending upon the field of view of the camera, the amount of distortion introduced will vary significantly. For the lenses used in this thesis, the camera field of view is quite wide leading to a noticeable amount of distortion, specifically near the periphery of the images as shown in Figure 5.1.

The most common model for distortion assumes the magnitude of distortion increases as the radial distance from the center of the image increases. The image shows this is indeed a reasonable assumption for the camera and lens being used. Clearly, the amount of distortion increases along radial lines extending toward the periphery of the field of view. The sides of the projection screen, which are indeed perpendicular, appear bowed outward - a characteristic known as *barrel* distortion. If the sides were bowed inward, this would be known as *pincushion* distortion. Both barrel and pincushion distortion

can be modeled according to the relations:

$$x = x_d(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{5.3}$$

$$y = y_d(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{5.4}$$

Figure 5.1: Sample image showing significant radial distortion.

In the expression for radial distortion, (x_d, y_d) refer to the coordinates of the distorted points on the image plane, (x, y) refer to their undistorted representation, and $r^2 = x_d^2 + y_d^2$. The parameters κ_1 and κ_2 are additional intrinsic parameters which need to be recovered through camera calibration. The difference between barrel and pincushion distortion arises from the sign of these parameters. Using the convention from Tsai's calibration method, positive values denote barrel distortion while negative values denote pincushion distortion. Both parameters are often quite small even when the amount of distortion is large. Moreover, it is often the case that $\kappa_2 << \kappa_1$, and is therefore customary to assign κ_2 equal to zero. In the derivations that follow, it is assumed κ_1 is the only distortion coefficient to estimate through calibration.

5.3 Tsai's Calibration Algorithm

Tsai's calibration algorithm arose from the deficiencies found in state-of-the-art camera calibration techniques available at the time. He argued that existing techniques suffered from a few basic disadvantages: initial estimates were required for techniques based on nonlinear optimization, lens distortion was not modeled by techniques based on geometric construction or perspective transformations, and that techniques based on solving systems of linear equations were not linearly independent in the underlying parameters.

Tsai's novel approach to addressing these disadvantages was to seek out a constraint which would reduce the dimensionality of the parameters and to solve for only a subset of the calibration parameters. The solution presented by Tsai recovers each of the camera parameters discussed in the prior section along with an additional parameter s_x denoting an uncertainty scale factor due to scanning and acquisition timing errors. In his work, Tsai presents solutions based on observing a set of known feature points from either a single camera pose or from multiple camera poses. Tsai also presented solutions for both coplanar and non-planar feature points. The coplanar algorithm for a single image, described here, assumes $s_x = 1$.

The first stage of Tsai's algorithm is to compute the camera rotation matrix $\mathbf{R}$ and two components of translation : T_x and T_y. It is assumed there are n feature points $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$ each with coordinates $M_k = (U_k, V_k, W_k)$ defined with respect to a local coordinate system. Furthermore, a distinction between the number of sensor elements N_{cx} and the number of *effective* image pixels N_{fx} is assumed. This distinction is necessary because manufacturing of the camera CCD results in an array of sensor elements larger than the effective resolution of the camera. Both values are often available from the camera specifications. Let the revised sensor element dimension along the horizontal direction be written as $d'_x = d_x\,(N_{cx}/N_{fx})$.

The first stage of the algorithm proceeds as follows. Let $C = (N_{fx}/2, N_{fy}/2)$ be an initial estimate for the image center then, for each feature point k, the distorted camera coordinates of its projection on the image plane are computed from (3.1) as:

$$x_{dk} = d'_x(i_k - C_x) \qquad y_{dk} = d_y(j_k - C_y) \tag{5.5}$$

Using the coordinates of the feature points and the distorted camera coordinates, a system of linear equations for each k can be established. Provided $n >> 5$, the system of linear equations are indeed overdetermined allowing for the unknowns to be solved by least squares optimization. Let the unknowns of interest be given as $r'_1 = {T_y}^{-1} r_1$, $r'_2 = {T_y}^{-1} r_2$, ${T_y}^{-1} T_x$, $r'_4 = {T_y}^{-1} r_4$, and $r'_5 = {T_y}^{-1} r_5$ leading to the following linear equation for each k:

$$\begin{bmatrix} y_{dk}U_k & y_{dk}V_k & y_{dk} & -x_{dk}U_k & -x_{dk}V_k \end{bmatrix} \begin{bmatrix} r'_1 \\ r'_2 \\ {T_y}^{-1}T_x \\ r'_4 \\ r'_5 \end{bmatrix} = x_{dk} \tag{5.6}$$

Upon solving for the five unknowns, Tsai derived the following expression for the squared magnitude of T_y:

$$T_y^2 = \frac{S_r - [S_r^2 - 4(r'_1 r'_5 - r'_4 r'_2)^2]^{1/2}}{2(r'_1 r'_5 - r'_4 r'_2)^2} \qquad S_r = r'^2_1 + r'^2_2 + r'^2_4 + r'^2_5$$

It remains now to determine the sign of T_y which can be obtained as follows. A feature point k whose projection on the image (i_k, j_k) is far from the image center is chosen. The sign of T_y is then initially chosen to be positive and the following expressions are evaluated:

$$\begin{aligned} r_1 &= r'_1 T_y \\ r_2 &= r'_2 T_y \\ r_4 &= r'_4 T_y \\ r_5 &= r'_5 T_y \\ T_x &= (T_y^{-1} T_x) T_y \\ \alpha &= r_1 U_k + r_2 V_k + T_x \\ \beta &= r_4 U_k + r_5 V_k + T_y \end{aligned}$$

If α and x_{dk} have the same sign and β and y_{dk} also have the same sign, then the sign of T_y is positive. Otherwise, the sign of T_y is negative.

The next step of the first stage is to determine the rotation matrix $\mathbf{R}$ from one of two possible solutions. Let $s = -\text{sgn}(r_1 r_4 + r_2 r_5)$ where $\text{sgn}(\cdot)$ denotes the sign of its argument such that the two possible solutions $\mathbf{R_1}$ and $\mathbf{R_2}$ can be expressed as:

$$\mathbf{R_1} = \begin{bmatrix} r_1 & r_2 & (1 - r_1^2 - r_2^2)^{1/2} \\ r_4 & r_5 & s(1 - r_4^2 - r_5^2)^{1/2} \\ r_7 & r_8 & r_9 \end{bmatrix} \qquad \mathbf{R_2} = \begin{bmatrix} r_1 & r_2 & -(1 - r_1^2 - r_2^2)^{1/2} \\ r_4 & r_5 & -s(1 - r_4^2 - r_5^2)^{1/2} \\ -r_7 & -r_8 & r_9 \end{bmatrix}$$

where

$$r_7 = r_2 r_6 - r_3 r_5 \qquad r_8 = r_3 r_4 - r_1 r_6 \qquad r_9 = r_1 r_5 - r_2 r_4$$

The second stage of Tsai's algorithm is to compute the effective focal length f, the distortion coefficient κ_1, and the third component of translation T_z. To determine which of $\mathbf{R_1}$ or $\mathbf{R_2}$ should be chosen, it is necessary to first compute an approximate value for the focal length and z-component of translation. Once these approximate values have been obtained and the appropriate solution for $\mathbf{R}$ determined, it is then possible to obtain exact values for f, κ_1, and T_z.

If the approximate value for f is positive, then $\mathbf{R_1}$ provides the correct solution for the rotation matrix. Otherwise, $\mathbf{R_2}$ provides the correct solution. Let $P_k = (X_k, Y_k, Z_k)$ be the location of feature point M_k with respect to the camera coordinate frame of reference. Then, for each feature point k, and provided that $n >> 5$, an overdetermined system of linear equations can be established and solved by least squares optimization from the following:

$$\begin{bmatrix} Y_k & -y_{dk} \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} = w_k y_{dk}$$

where

$$Y_k = r_4 U_k + r_5 V_k + T_y \qquad w_k = r_7 U_k + r_8 V_k$$

Finally, using the equations of perspective projection in (5.1) and noting that $P_k = \mathbf{R}M_k + \mathbf{T}$, it is possible to recover exact values for f, κ_1, and T_z. By combining the radial distortion expressions with the perspective projection equations, it follows that:

$$x_k = x_{dk}(1 + \kappa_1 r_k^2) = f\frac{X_k}{Z_k} = f\frac{r_1 U_k + r_2 V_k + r_3 W_k + T_x}{r_7 U_k + r_8 V_k + r_9 W_k + T_z} \tag{5.7}$$

$$y_k = y_{dk}(1 + \kappa_1 r_k^2) = f\frac{Y_k}{Z_k} = f\frac{r_4 U_k + r_5 V_k + r_6 W_k + T_y}{r_7 U_k + r_8 V_k + r_9 W_k + T_z} \tag{5.8}$$

With approximate values for f and T_z known, and assuming an initial estimate of $\kappa_1 = 0$, it is possible to apply standard optimization techniques (such as the Levenberg-Marquardt algorithm) to estimate more exact values for these parameters. In fact, since the distorted camera coordinates are a function of the image center, optimization can also be used to obtain more accurate estimates for C_x and C_y. When applying the optimization procedure, the error e_k to minimize over all k is specified as:

$$e_k = \sqrt{e_{kx}^2 + e_{ky}^2} \qquad e_{kx} = f\frac{X_k}{Z_k} - x_k \qquad e_{ky} = f\frac{Y_k}{Z_k} - y_k$$

5.4 Calibration Setup and Procedure

The software implementation of Tsai's calibration procedure is a custom-written C++ wrapper for the open-source version maintained and written in C by Reg Wilson [28]. A checkerboard pattern with 20 squares, as shown in Figure 5.2, is positioned within view of the camera. The squares are evenly spaced and each one has dimension 25.4 mm by 25.4 mm. The camera is positioned to ensure the image plane and the plane of the checkerboard are not parallel. The feature points required by Tsai's algorithm are taken to be the corners of the squares - leading to a total of 80 feature points.

An image of the pattern is first captured then the corners of the squares are detected using a C++ implementation of the classic Harris corner detector [29]. Starting with the original image IP, gradient images I_x^2, I_y^2, and $I_x I_y$ are computed by applying the Prewitt operators to every 8-neighbourhood in the image. Effectively, this equates to convolving

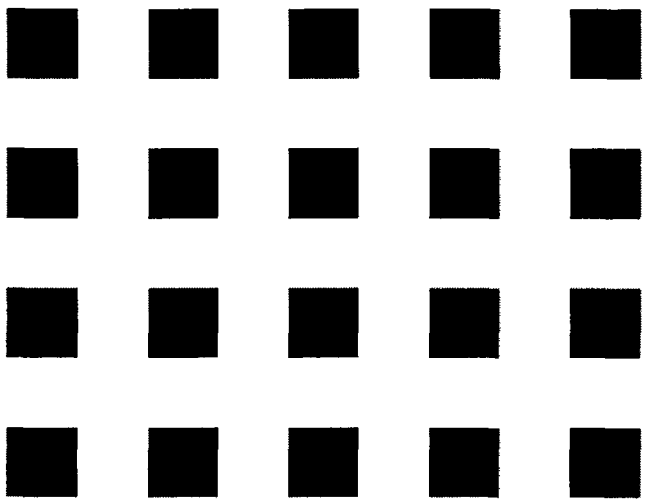

Figure 5.2: Camera calibration pattern.

two separate 3x3 kernels with the original image:

$$I_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * IP \qquad I_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * IP \qquad I_{xy} = I_x I_y$$

Once the gradient images have been obtained, Gaussian smoothing is applied by convolving a 3x3 kernel over the gradients as follows:

$$G_x = G * I_x^2 \qquad G_y = G * I_y^2 \qquad G_{xy} = G * I_x I_y$$

where

$$G = \begin{bmatrix} .075114 & .123841 & .075114 \\ .123841 & .204180 & .123841 \\ .075114 & .123841 & .075114 \end{bmatrix}$$

From the Gaussian-smoothed gradient images, a "cornerness" measure is computed for each pixel p in the original image. The idea is that if a pixel lies at the intersection of two perpendicular edges, the cornerness measure for that pixel will be large. Furthermore,

by knowing the number of expected corners n in the original image, it is possible to sort all measures in decreasing order and to select the top-n as belonging to the feature points of interest. Let $G_{(\cdot)}(p)$ denote the Gaussian-smoothed gradient along $(\cdot)$ in the 8-neighborhood of p such that the cornerness measure $R(p)$ can be expressed as:

$$R(p) = G_x^2(p)G_y^2(p) - G_{xy}(p)G_{xy}(p) - 0.04\left[G_x(p) + G_y(p)\right]^2$$

Prior to sorting the cornerness measures, non-maxima suppression is applied in each neighbourhood to ensure the corner is properly located. Due to the finite resolution of the camera, several pixels may exhibit large cornerness measures near the true location of a corner. As a result, it is necessary to isolate the pixel most likely to be at the intersection of the two perpendicular lines. This is achieved by scanning all 3x3 neighbourhoods and locating the pixel with maximum cornerness value. The cornerness measure of this pixel is maintained while all others are set to zero. Finally, the cornerness measures are sorted by decreasing value and the top-n are used to identify the corner locations.

5.5 Calibration Results

For completeness, the camera calibration results for the intrinsic parameters obtained using the setup and procedure described are included. The results provided are an average of five separate calibration estimates. An observation noted during testing is that estimates of the intrinsic parameters tend to be affected by the location of the camera relative to the calibration pattern. Therefore, the camera was positioned in five unique orientations while calibration was performed at each.

$$\begin{aligned} f &= 3.357157 \text{ mm} \\ \kappa_1 &= 0.0477783 \text{ mm}^{-2} \\ C_x &= 522.630049 \\ C_y &= 367.275437 \end{aligned}$$

Clearly, as the camera is moved to different locations, the extrinsic parameters will also change. However, the intrinsic parameters of the camera should remain constant

provided the zoom of the lens also remains constant. One practical application of this observation is that the camera pose, i.e. the extrinsic parameters, can be recovered without needing to re-estimate the intrinsics and leading to a more efficient implementation. In the following section it will be shown how this observation enables near real-time ray tracing to be achieved so that a hand-held camera can be used as an alternative input device.

Chapter 6

Potential Solutions to the Mapping Problem

Having described the mathematical framework, image processing routines, and camera calibration procedure, it is now possible to describe how the cursor position can be estimated using direct and indirect mapping solutions. The following sections introduce the homography estimation approach as a possible direct mapping solution, and two potential indirect solutions: the coplanar POSIT algorithm and a modification of Tsai's extrinsic algorithm. For each technique, the necessary theory is first introduced followed by a description of a complete implementation.

6.1 Direct Mapping Solutions

6.1.1 Homography Estimation

Homography estimation is a popular technique used to describe the transformation aligning two planes. In the context of a direct solution estimating the cursor location from a laser pointer, this approach was implemented in the works of [5, 9, 11, 13, 18]. The most

common approach assumes an experimental configuration as shown in Figure 3.1.

Theory

A homography is established by first projecting a series of feature points $\mathbf{M}$ onto the projection screen whose pixel coordinates (u, v) are known with respect to the output video resolution. An image of the feature points is taken and the centroid of their projections on the image plane $\bar{\mathbf{m}}$ are identified. For the case of coplanar feature points, a point $M_k = (u_k, v_k)$ and its image $\bar{m}_k = (i_k, j_k)$ are related via a homography $\alpha \bar{m}_k = \mathbf{H} M_k$ where $\alpha \neq 0$ is a scaling factor. Using a homogeneous coordinate representation, this relation can be written as:

$$\alpha \begin{bmatrix} i_k \\ j_k \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix}$$

which leads to the following expressions:

$$\begin{aligned} \alpha i_k &= h_{11}u_k + h_{12}v_k + h_{13} \\ \alpha j_k &= h_{21}u_k + h_{22}v_k + h_{23} \\ \alpha &= h_{31}u_k + h_{32}v_k + h_{33} \end{aligned}$$

and finally,

$$i_k = \frac{h_{11}u_k + h_{12}v_k + h_{13}}{h_{31}u_k + h_{32}v_k + h_{33}} \qquad j_k = \frac{h_{21}u_k + h_{22}v_k + h_{23}}{h_{31}u_k + h_{32}v_k + h_{33}}$$

This implies that, for each point correspondence, a pair of independent homogeneous equations is available. It turns out that, because of the scaling factor α, the homography matrix has only eight degrees of freedom. Therefore, only eight of the entries can be uniquely determined with the final entry being a linear combination of the other entries. The most popular way to address this issue is to simply set $h_{33} = 1$ and to solve for the remaining entries in $\mathbf{H}$. Taking this approach, it is clear that at least four point correspondences are needed to solve for the independent entries of the homography matrix.

By writing the independent entries as a column vector $\mathbf{h}$, the following expression can be obtained:

$$\mathbf{Ah} = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -i_1u_1 & -i_1v_1 & -i_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -j_1u_1 & -j_1v_1 & -j_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -i_2u_2 & -i_2v_2 & -i_2 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -j_2u_2 & -j_2v_2 & -j_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -i_3u_3 & -i_3v_3 & -i_3 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -j_3u_3 & -j_3v_3 & -j_3 \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -i_4u_4 & -i_4v_4 & -i_4 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -j_4u_4 & -j_4v_4 & -j_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = 0 \qquad (6.1)$$

whose solution can be obtained from the nullspace of $\mathbf{A}$. If the number of point correspondences available is $n >> 4$, then an overdetermined system of linear equations is obtained and the solution to the system solved by least squares optimization.

Implementation

The homography solution implemented in this thesis uses just four feature points to solve for the homography matrix. The feature points are simulated by projecting an image of four white squares on top of a black background to the projection screen as shown in Figure 6.1. The dimensions of the image were chosen to match the resolution of the projector and the centroids M_k of the feature points were carefully measured.

The correspondence between feature points and their projection on the image plane is then obtained using the right-turn predicate described in Section 4.5. Once the homography has been established, it is possible to invert the problem and convert image coordinates into pixels in the output video.

Consider a user interacting with a computer system where the video is sent to the projector and shown on the projection screen. Furthermore, assume the mouse cursor is controlled by a laser pointer and that the cursor position is determined to be the point of intersection between the laser and the output video. This point of intersection corresponds to a specific pixel, $\hat{q}$, in the output video. By using the camera to capture an

Figure 6.1: Simulated feature point arrangement for homography computation.

image of the scene, the homography can be used to recover $\hat{q} = (u, v)$ from its projection $\bar{m} = (i, j)$ on the image plane. From (6.1), the solution for a single point correspondence can be written as:

$$u(ih_{31} - h_{11}) + v(ih_{32} - h_{12}) + (ih_{33} - h_{13}) = 0 \quad (6.2)$$

$$u(jh_{31} - h_{21}) + v(jh_{32} - h_{22}) + (jh_{33} - h_{23}) = 0 \quad (6.3)$$

Let

$$\begin{aligned} a &= ih_{31} - h_{11} \\ b &= ih_{32} - h_{12} \\ c &= ih_{33} - h_{13} \\ d &= jh_{31} - h_{21} \\ e &= jh_{32} - h_{22} \\ f &= jh_{33} - h_{23} \end{aligned}$$

and substitute (6.2) into (6.3) to yield:

$$u = \frac{fb - ec}{db - ea} \qquad v = \frac{af - dc}{db - ae}$$

These results provide a sufficient framework to control the mouse cursor using a laser pointer. When the homography is computed, it is important the camera and projection screen remain in a fixed orientation. If either is moved, a new homography must be computed to ensure an accurate mapping between image and video coordinates is maintained.

6.2 Indirect Mapping Solutions

6.2.1 The Coplanar POSIT Algorithm

The POSIT algorithm [32] is a well-known solution for recovering a camera's pose from a collection of non-planar feature points. The algorithm was first proposed in the mid-1990s and, owing to its accuracy, is available in the widely-used open-source machine vision library *OpenCV*. The POSIT algorithm has also received much attention because it enables the camera pose to be estimated from just a single image - making it a practical solution for real-time machine vision applications. In the context of a training simulator as envisioned in this thesis, such a solution would enable users to interact in near real-time, subject to the frame rate of the camera, with the simulator. One downfall, however, is the training environment does not lend itself well to an arrangement of non-planar feature points.

For solutions based on indirect mapping, each user is assumed to hold a camera aimed toward the projection screen. It is reasonable to assume that, in a large proportion of the images captured by each camera, the projection screen itself will be the most consistently viewed object in the scene. This implies the most logical positioning of the feature points would be on or around the perimeter of the screen. However, because most projection screens are flat, it becomes difficult to affix feature points extending outward from the screen (non-planar features). A much more simple configuration is to simply place the feature points in the plane of the projection screen. For this reasoning, the standard POSIT algorithm does not provide a practical solution.

Shortly after publishing the POSIT algorithm, the authors published a follow-on

paper describing a coplanar form of the POSIT algorithm [21] along with an open-source implementation of the technique. In their paper, they present results based on a series of simulated images and demonstrate the approach to be quite accurate, subject to what they term the "distance ratio". Owing to these promising results, a C++ wrapper for the open-source C implementation was developed and tested using both real and simulated images. A novel contribution presented here is that the accuracy also depends on a new measure termed the "translation ratio".

Theory

A description of the coplanar POSIT algorithm is now presented for completeness. Derivation of the necessary theory and mathematical framework follows closely the work presented in [21]. As presented in Section 3.3, the objective of an indirect mapping solution is to recover the position and orientation of a camera relative to some frame of reference defined on the projection screen. The position is described by a 3x1 vector $\mathbf{T}$ while the rotation of the camera is described by a 3x3 matrix $\mathbf{R}$. The position vector defines the transformation aligning the origins of the two coordinate systems. The rotation matrix describes the transformation aligning the unit vectors $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ of the camera coordinate system with the unit vectors $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ of the feature point coordinate system:

$$\mathbf{R} = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix} = \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{bmatrix}$$

Following the notation provided in [21], let the origin of the camera coordinate system be O and be located at the camera's center of projection. Let M_k denote the local coordinates of feature point k relative to a reference feature point M_0 located in the plane of the display. It is assumed the local coordinates (U_k, V_k, W_k) of each feature point are known. The projection of the feature points on the image plane are denoted as m_k and have known camera coordinates (x_k, y_k) relative to the center of projection. Note that these conventions are slightly different from those introduced in Section 3.1.

According to the authors, since point M_0 has the known image point m_0, the translation vector is aligned with the vector $\mathbf{Om_0}$ and is equivalent to $(Z_0/f)\mathbf{Om_0}$. The feature

points $P_k = (X_k, Y_k, Z_k)$ expressed in the camera frame of reference can be written in terms of the local feature coordinates and the translation vector $\mathbf{T} = (T_x, T_y, T_z) = (X_0, Y_0, Z_0)$ as:

$$\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix} = \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \\ \mathbf{k}^T \end{bmatrix} \begin{bmatrix} U_k \\ V_k \\ W_k \end{bmatrix} + \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \tag{6.4}$$

This formulation requires the focal length of the camera as well as the center of projection of the image plane to be known a priori, which can be recovered using Tsai's calibration method.

Using this formulation, the equations forming the pose estimation algorithm can now be derived. The algorithm is based on the notion that approximating a perspective projection with a scaled orthographic projection is suitable when the depths Z_k of the feature points are nearly equivalent. When this notion holds, the depths of each feature point can be considered equal to the depth Z_0 of the reference feature point M_0. The algorithm begins by computing a scaled orthographic projection approximation and iteratively refines up to two possible solutions for the camera pose. Using the equations from perspective projection (5.1), the coordinates of the image points can be written as:

$$x_0 = f\frac{X_0}{Z_0} \qquad x_k = f\frac{X_k}{Z_k} \qquad y_0 = f\frac{Y_0}{Z_0} \qquad y_k = f\frac{Y_k}{Z_k}$$

Expanding the equation for x_k using (6.4), and noting similar results hold respectively for y_k, the following relation is obtained:

$$x_k = f\frac{\mathbf{M_0M_k} \cdot \mathbf{i} + X_0}{\mathbf{M_0M_k} \cdot \mathbf{k} + Z_0} = \frac{(f/Z_0)\mathbf{M_0M_k} \cdot \mathbf{i} + x_0}{(1/Z_0)\mathbf{M_0M_k} \cdot \mathbf{k} + 1}$$

from which an exact pose is defined by $\mathbf{i}$, $\mathbf{j}$, x_0, y_0, and Z_0, provided for each point M_k, the following equations are satisfied:

$$\mathbf{M_0M_k} \cdot \mathbf{I} = x_k(1 + \epsilon_k) - x_0 \qquad \mathbf{I} = \frac{f}{Z_0}\mathbf{i} \tag{6.5}$$

$$\mathbf{M_0M_k} \cdot \mathbf{J} = y_k(1 + \epsilon_k) - y_0 \qquad \mathbf{J} = \frac{f}{Z_0}\mathbf{j} \tag{6.6}$$

with

$$\epsilon_k = \frac{1}{Z_0}\mathbf{M_0M_k} \cdot \mathbf{k} \qquad \mathbf{k} = \mathbf{i} \times \mathbf{j}$$

The POSIT algorithm states that if values are given to ϵ_k, then (6.5) and (6.6) are systems of linear equations in which $\mathbf{I}$ and $\mathbf{J}$ are the only unknowns. Starting with $\epsilon_k = 0$, these unknowns can be computed and used to recover $\mathbf{i}$, $\mathbf{j}$, and Z_0. Once $\mathbf{i}$ and $\mathbf{j}$ have been computed, an iterative process can be applied to compute more accurate values for ϵ_k. In (6.5) and (6.6) it is important to note the left-hand side of the equations contain vector dot products which can be expressed in the local coordinate system of the feature points as:

$$\begin{bmatrix} U_k & V_k & W_k \end{bmatrix} \begin{bmatrix} I_u & I_v & I_w \end{bmatrix}^T = x_k(1+\epsilon_k) - x_0 = x\prime$$
$$\begin{bmatrix} U_k & V_k & W_k \end{bmatrix} \begin{bmatrix} J_u & J_v & J_w \end{bmatrix}^T = y_k(1+\epsilon_k) - y_0 = y\prime$$

Writing these equations for the n known feature points $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$ leads to the following matrix representation:

$$\mathbf{AI} = \mathbf{x}\prime \qquad \mathbf{AJ} = \mathbf{y}\prime \tag{6.7}$$

If matrix $\mathbf{A}$ had rank 3, then $\mathbf{I}$ and $\mathbf{J}$ could be solved using the pseudo inverse of $\mathbf{A}$. However, since the feature points are all coplanar, the matrix has rank 2 and cannot be solved even with an overdetermined set of equations. The solutions for (6.7) are best described using a geometric interpretation which allows the problem to be formulated as the roots of a complex number [21]. In their work, the authors impose the condition $|\mathbf{I}| = |\mathbf{J}|$. Once these solutions have been obtained, the depth Z_0 of the reference point can be computed by dividing the focal length by the norm of $\mathbf{I}$ or $\mathbf{J}$. Using equations (6.5) and (6.6), the first two rows of the rotation matrix can be recovered and the third row obtained from the cross-product $\mathbf{i} \times \mathbf{j}$. Finally, the remaining components of the translation vector can be obtained from the reference point $m_0 = (x_0, y_0)$ as $X_0 = (Z_0/f)x_0$ and $Y_0 = (Z_0/f)y_0$.

Implementation

The coplanar POSIT algorithm implemented for real images uses nine feature points to estimate the camera pose. The feature points are simulated by projecting an image of nine white squares on top of a black background to the projection screen as shown in Figure 6.2. The dimensions of the image were chosen to match the resolution of the projector, fixed at 1920x1080. The centroids of the feature points were then carefully measured to within an error of ± 7 mm. The reference feature point M_0 is taken to be the upper-left square shown in the image. Let the pixel coordinates, with respect to the output video, of the feature points be $M_k = (u_k, v_k)$.

Figure 6.2: Simulated feature point arrangement for the coplanar POSIT algorithm.

To obtain high accuracy measurements, the projector was carefully positioned so the image on the projection screen measured 1920$\pm$ 5mm by 1080$\pm$ 5mm. This ensured each pixel occupied an area of approximately 1 mm^2. By knowing the centroid locations of each square in the output video, and noting that one pixel unit was equivalent to 1mm, the local coordinates could be obtained trivially as $M_k = (U_k, V_k, 0) = (u_k, v_k, 0)$.

The correspondence between feature points and their projections on the image plane are obtained using the approach described in Section 4.5. Since the horizontal and vertical coordinates of the feature points are readily obtained (in mm) as previously described, it only remains to compute the metric coordinates of their projections from the known

pixel coordinates. Using equation (3.1), it follows that:

$$x = d_x(i - C_x) \qquad y = d_y(j - C_y)$$

where the pixel dimensions specified by the manufacturer are $d_x = d_y = 4.65 * 10^{-3}$ mm. Once the camera pose has been recovered, a ray is traced from the camera's center of projection to a point on the projection screen and converted into a pixel $\hat{q}$ in the output video sequence.

Since the estimated pose is defined with respect to the reference point M_0, it is quite simple to determine the point of intersection. Let the rotation about the x-axis be given by ϕ where a negative angle denotes a clockwise rotation. An angle of $\alpha = 0$ implies the camera is parallel to the projection screen. Similarly, let θ denote a rotation about the y-axis where a negative angle again denotes a clockwise rotation. An angle of $\theta = 0$ also indicates the camera is parallel to the projection screen.

Let the components of the translation vector $\mathbf{T} = (T_x, T_y, T_z)$ be positive along the positive direction of the unit vectors $(\mathbf{u}, \mathbf{v}, \mathbf{w})$. Because the physical dimensions of the output video match its resolution, one millimeter is equivalent to one pixel unit. This ensures $\hat{q}$ can be computed directly using the metric values for the translation and rotation components. If $\hat{q} = (u, v)$ denotes the point of intersection, its entries can be computed as follows:

$$u = \begin{cases} T_x - |T_z| \tan(|\theta|) & \theta \leq 0 \\ T_x + |T_z| \tan(|\theta|) & \theta > 0 \end{cases} \tag{6.8}$$

$$v = \begin{cases} T_y + |T_z| \tan(|\phi|) & \phi \leq 0 \\ T_y - |T_z| \tan(|\phi|) & \phi > 0 \end{cases} \tag{6.9}$$

This point of intersection serves as the location at which to position a user's cursor when interacting with the host computer.

Simulation Results

To evaluate the accuracy of the implementation, the average orientation and relative translation errors for a series of simulated images were recorded. The approach to generating the images is identical to the approach described in [21]. In their work, the authors distribute 10 feature points evenly about a 50 by 50 meter grid and simulate a variety of camera poses by applying different transformations to the feature points. By fixing the z-component of translation, the projections of the feature points on the image plane are simulated by applying the perspective projection equation given by (5.1).

Images are simulated by fixing the translation vector of the camera then varying its rotation. The rotation ϕ was fixed at 0 and the rotation θ was varied from 10° to 90° in increments of 5°. For each value of θ, the rotation ψ about the z-axis was varied from 0° to 360° in increments of 5°. Using this approach, 72 rotations of ψ were applied to the camera for each value of θ. Errors were then computed as the average over the 72 rotations for each θ.

The average orientation and relative translation errors were then computed for each rotation about the y-axis. Because the true translation and rotation of the camera are known a priori, the values estimated by the algorithm are used to determine the errors. Let $\hat{\mathbf{R}}$ and $\mathbf{R}$ denote the estimated and true rotation matrices, respectively. Similarly, let the estimated and true translation vectors be $\hat{\mathbf{T}}$ and $\mathbf{T}$. The orientation error is computed by first determining the axis of rotation aligning the true and estimated frames of reference by applying singular value decomposition on the matrix $\Delta\mathbf{R} = \mathbf{R} - \hat{\mathbf{R}}$. Next, the angle of rotation required to align them is computed from the angles aligning each individual axis. The relative translation error is computed by first determining the translation aligning the origins of the true and estimated frames of reference and then diving by the norm of $\mathbf{T}$. Since the algorithm refines up to two possible solutions, $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$ are chosen from the pose which minimizes these errors.

Based on simulation results provided by the authors, a distance ratio of 5 was chosen to evaluate the accuracy of the implementation. In their work, the distance ratio is defined as the z-component of translation divided by the size of the feature point grid. Through simulation, they showed the distance ratio has a significant impact on the

accuracy of the coplanar POSIT algorithm. Based on their findings, a distance ratio of 5 was shown to provide the best performance characteristics over the entire range of simulated camera poses. A novel finding of this thesis is that the performance of the algorithm is also influenced by the translation ratio R defined as:

$$R = \frac{\sqrt{T_x^2 + T_y^2}}{T_z}$$

Simulations results for various translation ratios are shown for the average orientation error in Figure 6.3 and for the relative translation error in Figure 6.4.

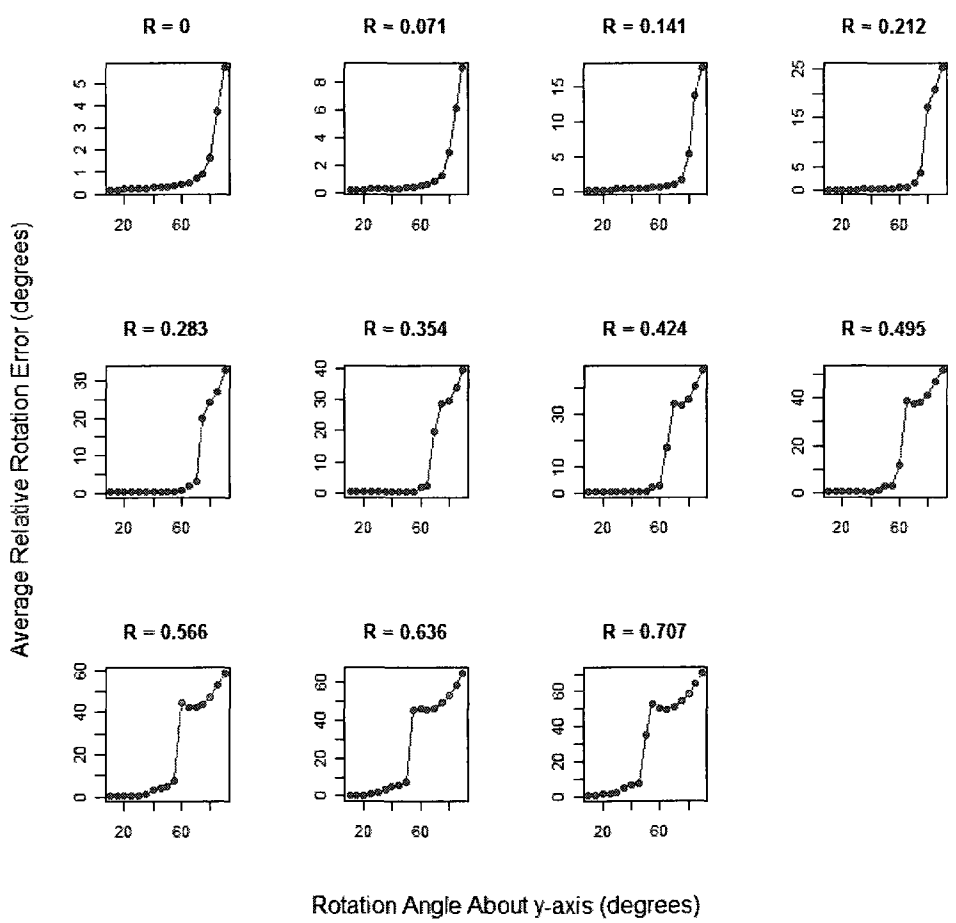

Figure 6.3: Average orientation error versus angle of rotation θ for 10 coplanar feature points

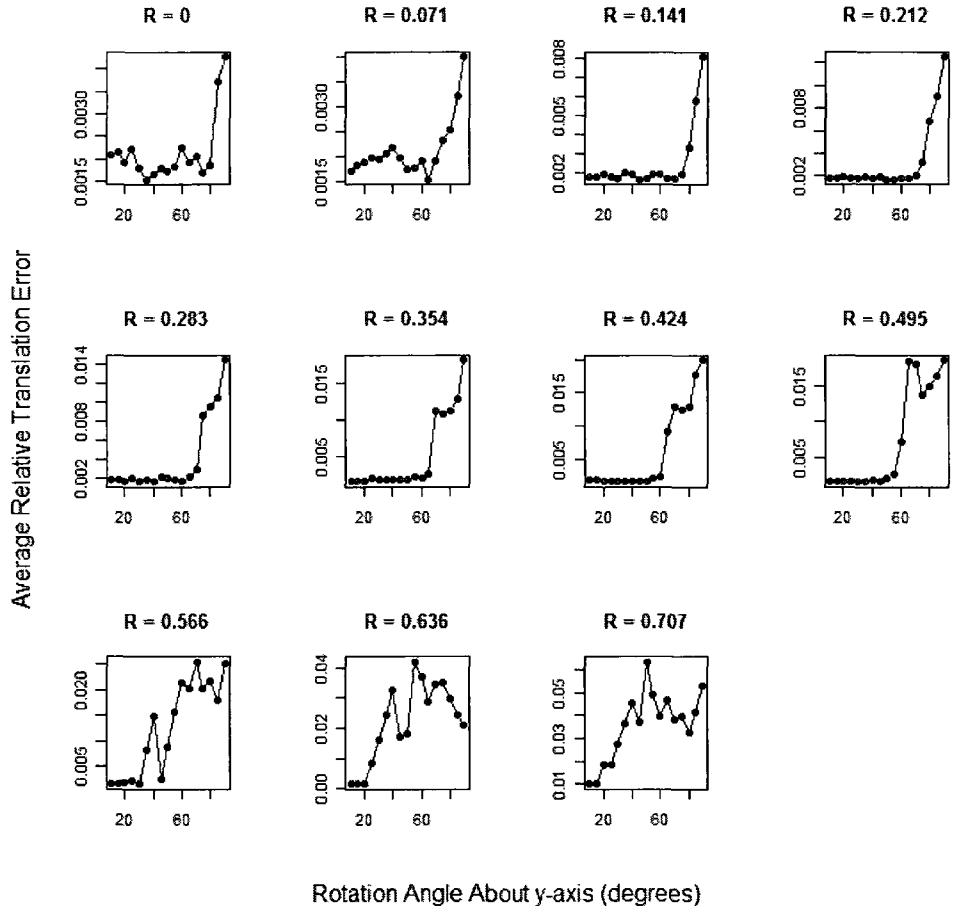

Figure 6.4: Relative translation error versus angle of rotation θ for 10 coplanar feature points

As shown, when the translation ratio is set to zero, both the rotation and translation errors are reasonably small over the entire range of rotations θ. Increasing the translation ratio tends to have little effect on the relative translation error over the full range of applied rotations. However, from Figure 6.3, it appears as though an inflection point exists where the average rotation error suddenly jumps. This is not a result of the C++ implementation; the same behaviour was observed using the open-source implementation. Increasing the translation ratio evidently causes the inflection point to shift toward lower values of θ - decreasing the range of rotations where acceptable error performance can be obtained.

Although the performance characteristics illustrate the rotation errors are susceptible to the physical location of the camera, it is important to note there is a finite range of values for θ over which both the translation and rotation are accurately estimated. Specifically, below 45° the average rotation error is fewer than 10° while the relative translation error is less than 5% for all translation ratios considered.

In theory, for a camera with a wide angle lens and for a user located sufficiently far from the projection screen, this solution should enable users a wide range of motion in the training environment while still providing accurate pose estimation. However, results obtained using real images consistently failed to produce acceptable estimates for either the translation or rotation components of pose - regardless of the translation ratio. Table 6.1 provides results for the absolute and relative translation errors obtained from five separate camera poses and real image data.

(T_x, T_y, T_z) (m)	E_x (m)	E_y (m)	E_z (m)	E (%)	R
(0, 0, 3.0)	0.15	0.08	0.31	11.79	0
(0, 0, 4.0)	0.15	0.09	0.31	8.89	0
(0, 0, 5.0)	0.16	0.10	0.30	7.09	0
(0.50, 0.50, 4.0)	0.31	0.36	0.42	15.61	0.177
(1.0, 1.0, 3.0)	0.80	0.85	0.68	40.73	0.471

Table 6.1: Absolute and relative translation errors for the coplanar POSIT algorithm on real image data.

The rotation of the camera was fixed as accurately as possible to point directly toward the center of the projection screen. The translation of the camera was then measured manually with an accuracy of approximately ±5 cm. The absolute errors were computed as the difference between the true and estimated components of the translation vector:

$$E_x = |T_x - \hat{T}_x| \qquad E_x = |T_y - \hat{T}_y| \qquad E_z = |T_z - \hat{T}_z|$$

The relative translation error E is the norm of the absolute errors divided by the norm of the true translation vector:

$$E = \frac{\sqrt{E_x^2 + E_y^2 + E_z^2}}{\sqrt{T_x^2 + T_y^2 + T_z^2}}$$

These results are provided to illustrate the discrepancy between simulated and real image data. The simulation results indicate the translation ratio has very little impact on the relative translation error; however, real image data clearly indicates increasing the translation ratio increases both the absolute and relative translation errors. Even when the translation ratio is fixed at zero, the relative error is nearly 12% at a distance of 3 m from the projection screen. When the translation ratio approaches 0.471, the relative error is greater than 40%. These errors will undeniably lead to inaccurate estimates for the cursor position.

The results observed indicate that, based on the chosen implementation, the coplanar POSIT algorithm does not provide a sufficiently accurate solution to the indirect mapping problem. In the context of a combat training simulator, it is imperative that users be free to move about the training environment to provide a more realistic and more interactive learning experience. However, the accuracy of pose estimation is clearly limited by the translation ratio, which in turn limits a user's mobility throughout the training environment. These observations led to the conclusion that a more robust indirect mapping solution is required.

6.2.2 A Modification of Tsai's Extrinsic Algorithm

Tsai's coplanar camera calibration algorithm can be considered an indirect mapping solution because, in addition to recovering intrinsic parameters of the camera, it also estimates the translation and orientation of the camera relative to the calibration grid. Indeed, once the intrinsic parameters of the camera have been recovered, Tsai's calibration procedure can be reduced to simply estimating the pose of the camera. This is what is referred to as Tsai's extrinsic algorithm.

Much like the coplanar POSIT algorithm, Tsai's extrinsic algorithm is capable of estimating the camera pose from just a single image containing feature points with known local coordinates. Furthermore, because the algorithm is designed for the case of coplanar feature points, it is well-suited for use in the training simulator environment.

The theory behind Tsai's extrinsic calibration procedure is first presented building

upon the theory and equations introduced in Section 5.3. A brief description of how the algorithm is implemented is then discussed and followed by results obtained on real image data. The algorithm is implemented using a C++ wrapper to Reg Wilson's open-source software [28]. It is shown the accuracy of the algorithm is greatly affected by the field-of-view of the camera lens as well as the relative position of the camera in the training environment. Owing to these observations, a modification of Tsai's extrinsic algorithm is then proposed and shown to provide significantly greater accuracy when estimating the pose of a camera with wide-angle lenses.

Theory

Tsai's extrinsic calibration algorithm assumes a series of feature points $\mathbf{M}$ are distributed in a planar arrangement and have known coordinates $M_k = (U_k, V_k, W_k)$ with respect to a local coordinate system. An image of the feature points is captured and the corresponding centroids $\bar{\mathbf{m}}$ on the image plane are identified. It is further assumed the focal length, image center, and distortion coefficient have been recovered using the complete camera calibration procedure. Starting with the pixel coordinates $\bar{m}_k = (i_k, j_k)$, the distorted camera coordinates (x_{dk}, y_{dk}) are computed from (5.5) as:

$$x_{dk} = d'_x(i_k - C_x) \qquad y_{dk} = d_y(j_k - C_y) \qquad \forall k$$

where d'_x and d_y are obtained from the camera specifications. Next, the undistorted camera coordinates are computed from (5.7) and (5.8) using the known distortion parameter κ_1 as follows:

$$\begin{aligned} x_k &= x_{dk}(1 + \kappa_1 r_k^2) \\ y_k &= y_{dk}(1 + \kappa_1 r_k^2) \\ r_k^2 &= x_{dk}^2 + y_{dk}^2 \end{aligned}$$

From the undistorted coordinates, a linear equation similar to (5.6) is formed. With $n \geq 5$, a system of linear equations can be formed and solved for the five unknowns. Following the notation used in Section 5.3, let the unknowns of interest be $r'_1 = {T_y}^{-1} r_1$, $r'_2 = {T_y}^{-1} r_2$, ${T_y}^{-1} T_x$, $r'_4 = {T_y}^{-1} r_4$, and $r'_5 = {T_y}^{-1} r_5$ where r_1, r_2, r_4, and r_5 are entries in the

rotation matrix given by (5.2). Using this notation, the linear equation of interest can be written as:

$$\begin{bmatrix} y_k U_k & y_k V_k & y_k & -x_k U_k & -x_k V_k \end{bmatrix} \begin{bmatrix} r'_1 \\ r'_2 \\ T_y^{-1} T_x \\ r'_4 \\ r'_5 \end{bmatrix} = x_k \tag{6.10}$$

In solving for the five unknowns, the rotation matrix can then be recovered using the procedure described in Section 5.3. Knowing the entries of the rotation matrix, and the undistorted projections of the feature points, (5.7) and (5.8) can be re-arranged to obtain the following system of linear equations:

$$\begin{bmatrix} f & 0 & -x_k \\ 0 & f & -y_k \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x_k(r_7 U_k + r_8 V_k + r_9 W_k) - f(r_1 U_k + r_2 V_k + r_3 W_k) \\ y_k(r_7 U_k + r_8 V_k + r_9 W_k) - f(r_4 U_k + r_5 V_k + r_6 W_k) \end{bmatrix} \tag{6.11}$$

The only unknowns in the above expression are the components of the translation vector and, provided $n \geq 3$, there exists a unique solution for these unknowns. As with the coplanar POSIT algorithm, the pose is estimated with respect to a reference feature M_0. With $n >> 3$ an overdetermined system of linear equations is obtained and the unknowns can be solved by least squares optimization. Solving equations (6.10) and (6.11) is sufficient to completely recover the camera pose using Tsai's extrinsic calibration algorithm.

Implementation

As with the coplanar POSIT algorithm, Tsai's extrinsic calibration algorithm is implemented using a series of nine simulated feature points as shown in Figure 6.2. The reference point is taken to be the top-left square. The projected centroids of the feature points $\bar{\mathbf{m}}$ are located on the image plane and, for each point, the pixel coordinates $\bar{m}_k = (i_k, j_k)$ are recorded. With the focal length and image center recovered through

calibration, the distorted camera coordinates (x_k, y_k) are computed. From the nine feature point projections, the system of linear equations in (6.10) is then solved to recover the rotation of the camera. Finally, with nine feature points, eighteen linear equations can be formed from (6.11) and solved by least-squares optimization for the unknowns.

Once the camera pose has been recovered, a ray can be traced from the camera's center of projection to the projection screen. The point of intersection can then be converted into a pixel coordinate $\hat{q} = (u, v)$ with respect to the projected video sequence and used to control the position of the mouse cursor. By ensuring the dimensions of the projected video sequence match its resolution, one pixel unit can be made equivalent to one millimeter. Thus, the point of intersection can be computed directly from the metric values of $\mathbf{R}$ and $\mathbf{T}$ using equations (6.8) and (6.9). From Section 3.3, the rotation matrix can be written as:

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\phi + \cos\psi\sin\theta\cos\phi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \\ &= \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}\end{aligned}$$

where the rotations θ and ϕ can be computed as:

$$\theta = \arcsin(-r_3) \qquad \phi = \arcsin\left(\frac{r_6}{\cos\theta}\right)$$

Results on Real Image Data

In testing, Tsai's extrinsic calibration algorithm demonstrated exceptional performance on simulated image data. Using the procedure described in Section 6.2.1 for generating simulated images, both the open-source and C++ implementations recovered the camera pose with 100% accuracy. These results are clearly very encouraging and provide the potential for a truly effective indirect mapping solution. To confirm the accuracy for real image data, a camera was positioned in five separate locations and the true translation vector $\mathbf{T} = (T_x, T_y, T_z)$ was measured manually with an accuracy of approximately ± 5

cm. The rotation of the camera was fixed as accurately as possible to point directly toward the center of the projection screen. The absolute and relative translation errors, reported in Table 6.2, were then recorded based on the estimated translation vector. For comparison, the translation ratio at each position is also included.

(T_x, T_y, T_z) (m)	E_x (m)	E_y (m)	E_z (m)	E (%)	R
(0.95,0.75,3.50)	0.082	0.158	0.223	7.70	0.348
(1.60,0.75,3.45)	0.561	0.153	0.415	18.51	0.504
(2.15,0.75,3.30)	1.192	0.165	0.650	34.16	0.694
(2.70,0.75,3.00)	1.814	0.166	1.074	51.15	0.928
(3.20,0.75,2.70)	2.402	0.168	1.573	67.69	1.229
(3.65,0.75,2.25)	2.943	0.173	2.149	83.96	1.652

Table 6.2: Absolute and relative translation errors for Tsai's extrinsic algorithm on real image data.

Once again, increasing the translation ratio is shown to increase both the absolute and relative translation errors. However, when compared with Table 6.2, the accuracy of Tsai's extrinsic algorithm is influenced to a much lesser degree by the translation ratio. For example, with $R = 0.504$, Tsai's method results in a relative translation error of 18.51%, whereas the coplanar POSIT algorithm exhibits a relative translation error of 40.73% for $R = 0.471$. To be fair, neither of these results is overly encouraging. The comparably high absolute and relative translation errors severely limit the range over which the solutions can be used. If a user must remain confined to a small area in order to achieve accurate pointing, it would be infeasible to accomodate more than a single user at a time - thus defeating the objectives of this thesis.

If an indirect mapping solution fails to accurately estimate the rotation matrix as well, the error in estimating where a user is pointing will be further compounded. Although not reported, the rotation estimates for the coplanar POSIT algorithm were consistently observed to be quite poor for non-zero translation ratios. Combined with the inaccurate translation estimates, this provided sufficient evidence to abandon it as a practical solution. One interesting observation noted during testing was that Tsai's extrinsic algorithm actually appeared to recover the camera pose with reasonable accuracy. In fact the translation ratio was observed to have little impact on the estimated rotation over most of the translations considered. This is a promising observation because it implies

one half of the pose estimation problem can be solved. It only remains to determine how to accurately recover the translation component of pose.

It turns out a simple modification to Tsai's approach can be made to greatly improve the accuracy of the estimated translation. Through tedious testing and observation, it became apparent Tsai's algorithm was greatly impacted by the wide-angle lens being used on the camera. With a wide-angle lens, much of a user's interaction with the projection screen causes the feature points to be concentrated about the center of the image plane. This observation is most noticeable when the z-component of translation is large compared to the separation of the feature points. When a user is closer to the projection screen, this effect is mitigated but it may cause some of the feature points to lie outside of the captured image if a user is too close. By imposing a minimum bound on the z-component of translation so that all feature points are always visible, it was possible to investigate techniques for mitigating the effects of wide-angle lenses.

Assuming the camera rotation can be accurately recovered, a novel and rather simple variation to Tsai's extrinsic algorithm was developed for wide-angle lenses. The basic premise is to generate an estimate for the camera pose, use the estimated rotation to reposition the feature points on the image plane, and finally re-estimate the camera translation. Let the initial coordinates of a feature point on the image plane be given as $\bar{\mathbf{m}}' = (i'_k, j'_k)$ and the repositioned coordinates as $\bar{\mathbf{m}} = (i_k, j_k)$. To reposition the coordinates, correction factors are first computed:

$$\Delta x = \begin{cases} -(f/d_x)\sin(|\theta|) & \theta \geq 0 \\ (f/d_x)\sin(|\theta|) & \theta < 0 \end{cases} \tag{6.12}$$

$$\Delta y = \begin{cases} -(f/d_y)\sin(|\phi|) & \phi \leq 0 \\ (f/d_y)\sin(|\phi|) & \phi > 0 \end{cases} \tag{6.13}$$

and then used to reposition the feature points according to:

$$\begin{aligned} i_k &= i'_k + \Delta x \quad \forall k \\ j_k &= j'_k + \Delta y \quad \forall k \end{aligned}$$

The purpose of these correction factors is to "undo" the rotation of the camera and to position the feature points on the image plane as they would be seen with $\theta = \phi = 0$.

This approach was motivated by the observation that Tsai's algorithm provided reasonably accurate estimates for $\mathbf{T}$, over wide range of locations, when these conditions held. From the repositioned coordinates, the translation component of pose is re-estimated by forming and solving equations (6.10) and (6.11) in turn. The absolute and relative translation errors using the revised implementation of Tsai's extrinsic algorithm for the same five camera locations are recorded in Table 6.3. Again, the rotation of the camera was fixed as accurately as possible to point directly toward the center of the projection screen. The dimensions of the projected video were also ensured to match its resolution.

(T_x, T_y, T_z) (m)	E_x (m)	E_y (m)	E_z (m)	E (%)	R
(0.95,0.75,3.50)	0.313	0.092	0.088	9.12	0.348
(1.60,0.75,3.45)	0.226	0.149	0.027	7.02	0.504
(2.15,0.75,3.30)	0.171	0.099	0.102	5.55	0.694
(2.70,0.75,3.00)	0.125	0.078	0.039	3.71	0.928
(3.20,0.75,2.70)	0.095	0.052	0.191	5.16	1.229
(3.65,0.75,2.25)	2.884	0.082	1.070	70.69	1.652

Table 6.3: Absolute and relative translation errors for the modification of Tsai's extrinsic algorithm on real image data.

Comparing these results to the ones in Table 6.2, it is evident the proposed variation to Tsai's approach provides significantly better accuracy for nearly every camera location considered. Furthermore, increasing the translation ratio appears to have no corresponding impact on the absolute or relative translation errors. This is a highly desirable property as it will allow a user greater movement throughout the training environment while still affording acceptable pointing accuracy. The results also confirm the assumption that Tsai's extrinsic algorithm can accurately recover the camera rotation despite providing inaccurate translation estimates. If the rotation estimates were incorrect, applying the correction step and re-estimating the translation would not have improved the accuracy so significantly.

At the final camera location, the relative translation error abruptly increases from 5.19% to 70.69%. Although this is indeed an improvement over the previous approach, it still indicates the camera pose was incorrectly estimated. The most notable increases are the absolute error E_x which jumps from an error of about 9 cm to nearly 3 m, and E_z which jumps from about 19 cm to over 1 m. Interestingly, the camera was only moved

an additional 45 cm further along the horizontal and 45 cm toward the projection screen when compared to the fourth camera position. Considering these were relatively minor movements, it was surprising to see the errors increase so dramatically. Nevertheless, the proposed variation to Tsai's approach has been shown to be the most accurate indirect mapping solution of those considered.

Chapter 7

Multiple-User Distinction

When multiple users are interacting concurrently with the display, it is important to identify which inputs are being generated by which user. For solutions based on a direct mapping approach, this equates to identifying which laser patterns are present on screen, which user the pattern belongs to, and identifying where these users are pointing. For solutions based on indirect mapping, this equates to identifying which cameras are actively viewing the scene, which users they belong to, and to identify where each of them is pointing.

For solutions implementing an indirect mapping approach, a single and simple technique using unique identifiers is described for distinguishing among multiple users. Effectively, each hand-held camera is assigned a unique identifier such that data communicated between the host PC and one of the cameras can easily be differentiated from another. For cameras tethered to the PC through an IEEE-1394 connection, it is possible to enumerate the cameras based on which port they are connected to. Using the camera's suite of APIs it is trivial to enumerate the cameras on the bus as needed.

7.1 Direct Mapping Distinction

For solutions implementing a direct mapping approach, two techniques to distinguish among multiple users are presented: off-the-shelf laser pointers with unique geometries, and laser pointers with custom designed geometries.

7.1.1 Off-the-Shelf Shape-Based Laser Pointers

One approach to distinguishing between users is to assume each user holds a laser pointer with a different geometric pattern. In this configuration only a single camera needs to view the projection screen and therefore only a single map $\mathcal{F}$ needs to be established between the camera and projection screen. If each user holds a laser pointer with a different geometric pattern, then distinguishing between multiple users becomes the challenge of classifying image components into one of N possible patterns.

One technique to enable such classification would first construct a template for each geometric pattern and then search over images captured by the camera to determine if one or more of the templates was present. If the templates are stored as image masks, or kernels, the matching process can be achieved by convolving the mask with the acquired image and computing some measure of similarity. One such measure, called the Hausdorff distance metric, has been shown to be both accurate and robust to occlusions in the image.

A metric which is robust to occlusions is highly desirable because it ensures multiple users can be pointing at the same location on the projection screen yet still be differentiated from one another. Let $\mathbf{A} = \{a_1, a_2, \ldots, a_N\}$ be a template for one of the laser patterns where entry $a_k = A(i_k, j_k)$ is set to 1 if it belongs to the template foreground, and set to 0 if it belongs to the background. Let $\mathbf{B} = \{b_1, b_2, \ldots, b_M\}$ denote the acquired image after thresholding and relaxation averaging. Entries $b_k = B(i_k, j_k)$ are then set to 1 if $b_k \geq \lambda$, and set to 0 otherwise. The *directed* Hausdorff metric $h(\mathbf{A}, \mathbf{B})$ can then be expressed as:

$$h(\mathbf{A}, \mathbf{B}) = \max_{a \in \mathbf{A}} \left(\min_{b \in \mathbf{B}} ||a - b||_2 \right)$$

where $||\cdot||_2$ denotes the Euclidean norm of its argument. This equation states that all pixels within the set $\mathbf{A}$ are at most $h(\mathbf{A}, \mathbf{B})$ distance from any pixel in $\mathbf{B}$. When this measure is small, the template $\mathbf{A}$ is highly probable to be within the set of pixels in $\mathbf{B}$. The primary disadvantage to this metric is that it requires a template and its projection on screen to be close in a Euclidean sense. That is, the scale, rotation, and translation of the template must match its projection.

In order to definitively conclude that a template is not present in an image, the entire class of transformations which could align the template with its projection must be exhausted. This implies the template must be positioned at all possible translations with respect to the image - which is achieved through convolution. For each translation, the set of all possible rotations in $[0, 2\pi]$ must be also applied to the template. Finally, a reasonable range of scales κ must be searched for each translation and for each rotation because the size of a laser pattern at, say distance d, will be different from its size at κd for $\kappa \neq 1$. Clearly, the computational complexity required to search the entire transformation space is immense.

Another disadvantage to this approach is that when patterns are projected from small incident angles relative to the projection screen, the projected pattern illustrates a large amount of nonlinear skew. From experimentation, the nonlinear skew causes some shapes (such as the '+' crosshatch shape) to be distorted so much that it would be practically impossible to achieve an accurate match - even when searching over the entire transformation space. An image showing the crosshatch pattern with and without skew is provided in Figure 7.1.

An additional problem encountered during testing has shown that the projected intensity of off-the-shelf patterns decreases dramatically as the distance from the projection screen increases. This, in turn, causes the projected pattern to fall well below reasonable intensity thresholds used for component labeling. A brief summary of the "detection" and "effective" ranges for five different laser pointer patterns is provided in Table 7.1. Detection range denotes the maximum distance at which the pattern is detected on screen. This is in contrast to the effective range which is the maximum distance where the projected pattern can be used to effectively control the position of the cursor. Measurements were accurate to within about ± 5 cm.

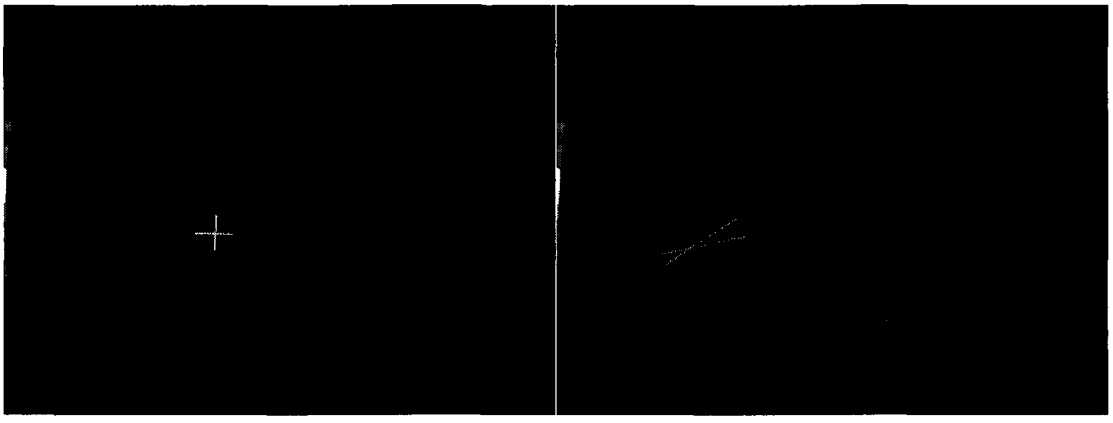

Figure 7.1: Image showing ideal (left) and skewed (right) crosshatch pattern.

All detection range measurements were obtained when the pattern was illuminated directly at the projection screen (i.e. perpendicular to the screen plane). These distances are not deemed effective, however, because minor deviations from perpendicular cause the intensity of the patterns to decrease and fall below the threshold used for component labeling. For the single dot pattern, the detection and effective ranges were equivalent and bounded only by the length of the room where testing was performed. Tests were conducted with the threshold λ set at 100, with a frame rate of 30 Hz, and with an exposure time of 15 ms. This configuration enabled the detection of very faint projection intensities and ensured the maximum possible distances were evaluated. When the tests were conducted, the ambient lighting measured 5.17 Lux at the projection screen and 7.29 Lux at the camera lens.

Pattern	Detection Range (m)	Effective Range (m)
Circle	0.60	0.15
Crosshatch	1.20	0.60
1 Line	1.50	1.20
3 Lines	1.20	0.60
Single Dot	$\geq$ 12.20	$\geq$ 12.20

Table 7.1: Detection and effective ranges for several laser pointer patterns.

From the table it is evident the effective ranges are quite poor, with the exception

of the single dot laser pattern. This results exclusively from the decrease in projected intensity as the distance from the projection screen increases. It is possible to achieve minor increases in effective range by decreasing the threshold λ. However, in doing so, additional objects in view of the camera begin to be remain in view after thresholding. One encouraging observation is the formidable effective range of the single dot pattern. This led to the idea that creating custom patterns from an array of single dot lasers may provide a more suitable approach to distinguishing between users.

7.1.2 Custom Shape-Based Laser Pointers

Owing to the idea that, in theory, users should be distinguishable by different geometric patterns, a novel approach based on custom-designed patterns is presented. The approach builds upon the encouraging results obtained for a single dot pattern while mitigating the computational complexity of searching a large transformation space. The basic idea investigated is for each user to have a laser pointer casting a circular pattern on screen but, for each user, the radius of the circle is unique. Differentiating among multiple users then becomes the challenge of determining which circles of what radius are present in the captured images - a task readily accomplished using the circular Hough transform.

An advantage to using circles as the only pattern is that they are, by definition, invariant to rotation. This means that searching an image for a particular circle does not require a search over rotation space. Furthermore, if the circles are approximated by *parallel* single dot lasers, the pattern projected on screen would be invariant to changes in scale. The reason this holds is that parallel lines remain parallel at any distance which ensures the scale of the projected pattern will remain constant. An added advantage of the circular Hough transform is that it negates the need to search over translation space which further reduces the computational burden.

To evaluate the effectiveness of using custom shaped-based laser pointers, the classification accuracy using the circular Hough transform and the Hausdorff distance metric are compared for a set of simulated images. Let N denote the number of parallel lasers used to approximate a circle and let S denote the radial separation, in pixels, between

circles. Let R' denote the smallest radius such that circle k has radius $R_k = R' + (k-1)S$.

The two approaches were evaluated for values of $N \in \{4, 8, 16, 32\}$ and for radial separations of $S \in \{5, 10, 25, 50\}$ pixels. Simulations were run 500 times for each scenario. At each iteration, the center point of a given circle was sampled from a uniform distribution bounded by a resolution of 1024x768. The location of the individual laser pointers p_k along the circle were then evenly spaced at angles of $2\kappa\pi/N$ for $\kappa \in \{1 \ldots N\}$. These locations were then subject to Gaussian noise $\sim \mathcal{N}(0, 1)$ and to a random rotation sampled uniformly from the interval $[0, 2\pi]$.

The circular Hough transform can be computed as follows. For every laser pointer location p_k in the input image, a circle of radius r is formed in discrete space. That is, points along the perimeter of the circle must have integer coordinates (i, j). For each coordinate, a counter $C(i, j, r)$ is maintained and initially set to 1. When circles formed from individual laser pointers intersect, say at (u, v), the counter at the coordinate of intersection $C(u, v, r)$ is incremented. The range of radii to consider is a parameter of the implementation.

If a range of radii $r \in \{R_1, R_2, \ldots, R_N\}$ are considered then it is possible to determine if a circle of radius $r = R_k$ exists in the image. This is achieved by examining all counter values for the given radius. If a circle of radius R_k does exist, there will be a large counter value in C at the corresponding point of intersection. Therefore, by examining all entries in the counter, it is possible to identify which circles of what radius are present in an image. Effectively, this provides a solution to the challenge of distinguishing among multiple users.

Classification accuracy based on simulated images were obtained for five circular patterns projected concurrently on screen. A classification is considered correct if a circle of radius R_k exists in an image and was identified by one of the approaches. Results for the circular Hough transform are provided in Table 7.2 and for the Hausdorff distance metric in Table 7.3.

As shown in the tables, both approaches perform comparably well when the patterns are constructed using $N = \{16, 32\}$ parallel lasers. For the case of $N = 16$, the circular Hough transform outperforms the Hausdorff distance metric by nearly 30% when the

Radial Separation (S)	Number of Pointers (N)	Percent Correct
5	4	9.56
5	8	33.28
5	16	88.64
5	32	99.16
10	4	10.68
10	8	36.64
10	16	95.52
10	32	99.84
25	4	12.72
25	8	45.08
25	16	97.20
25	32	99.64
50	4	9.00
50	8	29.44
50	16	91.60
50	32	99.76

Table 7.2: Classification accuracy using the circular Hough transform.

radial separation between each circle is only 5 pixels. For the case of $N = \{4, 8\}$ it is evident the Hausdorff distance metric provides superior performance for all radial separation values and that the classification accuracy appears to converge for radial separations beyond 25 pixels.

They both indicate that increasing the number of lasers approximating the circle tends to increase the classification accuracy for any given radial separation. Further, the results show that for a radial separation of 25 pixels and $N = 16$, the classification accuracy approaches 100%.

For the circular Hough transform, the primary source of classification error arises from the Gaussian noise applied to the center point of each individual laser. Tests performed without the addition of noise indicate classification accuracy of nearly 100% for all values of N and S under consideration. For the Hausdorff distance metric, the primary source of classification error arises from the circles overlapping one another.

Radial Separation (S)	Number of Pointers (N)	Percent Correct
5	4	30.20
5	8	42.28
5	16	61.12
5	32	96.40
10	4	40.44
10	8	56.40
10	16	86.20
10	32	100.00
25	4	64.24
25	8	87.84
25	16	100.00
25	32	100.00
50	4	63.48
50	8	87.24
50	16	99.96
50	32	100.00

Table 7.3: Classification accuracy using the Hausdorff distance metric.

Since the Hausdorff distance metric is based on a Euclidean distance, the addition of noise and random rotations can lead to mis-classifications. This was observed by bounding the circle centers to be sampled within a resolution of 256x256, ensuring the circles overlapped more often for each trial run. Although this did indeed decrease the classification accuracy, it was only by a matter of percentage points - confirming the metric is quite robust to occlusion.

The most important assumption made for these approaches is that each of the N lasers are exactly parallel. This criteria ensures the projected patterns will maintain a constant scale and that the two template matching techniques can be used reliably. If the assumption fails to hold, then both approaches will inevitably become unreliable. Depending upon the chosen value of N, the techniques may be robust to some of the lasers being anti-parallel. Reasonably, for a larger value of N, the techniques will be more robust provided only a few of the lasers are anti-parallel.

Another point to consider is the physical size of a device constructed from N individual laser pointers. With $N = 16$ and assuming a cylindrical-shaped laser pointer with radius 5 mm, the diameter of a single custom laser pointer would need to be approximately 60 mm. With each additional user to accommodate, the size of the device would need to increase according to the required separation of the projected radii.

Although it is fair to compare the two techniques with respect to classification accuracy, the most important point to consider is the computational expense of each. The circular Hough transform is orders of magnitude quicker than the Hausdorff distance metric. Taking into consideration that users must be identified on a frame-by-frame basis, the Hausdorff metric is far too intensive to compute using a software-only implementation. By contrast, the circular Hough transform is quite efficient and lends itself well to multi-threaded implementations.

Chapter 8

Kalman State Estimation

In Section 4.2, the concept of successive relaxations was introduced to improve the localization of interest points on the image plane. By applying successive relaxations, the original structure of a laser pattern or, respectively, feature points was recovered. This enabled the centroid of these objects to be estimated with greater accuracy. In what follows, the context considered assumes a direct mapping solution; however, the arguments indeed hold for indirect mapping solutions as well. For dynamic pointing, it is shown how the accuracy in estimating the cursor position can be greatly improved by estimating its location in future image frames.

When a laser pointer is used to control the mouse cursor, fast movements of the pointer lead to a noticeable amount of lag between the cursor and the laser spot. This occurs because the cursor location is estimated to be at the centroid of the laser pattern projected on screen. In actuality, the correct location at which to position the cursor would be at the endpoint of the laser stroke. The problem with this is that differentiating between the start of a stroke and its endpoint is ambiguous. Conceptually, if the dynamics of the laser pointer could be modeled such that a prior could be placed on the direction of travel, it would be possible to obtain an estimate of where the endpoint *should* be located. Using Kalman state estimation provides a way to model these dynamics and to improve the dynamic accuracy of a given solution.

In the following sections, the theory and mathematics of Kalman state estimation are introduced assuming the dynamics of a laser pointer are modeled using a two or three state system. The works of [30] and [31] are followed closely to derive the Kalman expressions.

8.1 System Dynamics

Let $s(t)$ denote the position of a laser pointer on the image plane at time t. Using the Taylor series expansion, the position $s(t)$ can be expressed as:

$$s(t) = s(0)t + \dot{s}(0)t + \frac{1}{2}\left[\ddot{s}(0) + w(t)\right]t^2 + \ldots$$

where $\dot{s}(0)$ and $\ddot{s}(0)$ indicate the velocity and acceleration, respectively, of s at initial time $t = 0$. In the above expression a noise term $w(t)$ is coupled to the acceleration to indicate that, at any time t, there is also a random component of the acceleration. This term is included to account for random perturbations of the laser spot on screen resulting from a user's hand tremor. If the acceleration component is assumed constant then the higher order terms can be ignored and the expression reduced to:

$$s(t) = s(0) + v(0)t + \frac{1}{2}\left[a(0) + w(t)\right]t^2$$

In the case of a discrete-time model, where events at time k and $k+1$ are separated by an interval ΔT (the reciprocal of the camera frame rate), the above expression can be re-written as:

$$s_{k+1} = s_k + v_k\Delta T + \frac{1}{2}\left[a_k + w_k\right]\Delta T^2 \tag{8.1}$$

If the velocity component is also assumed constant, the above expression can be further reduced to:

$$s_{k+1} = s_k + v_k\Delta T + \frac{1}{2}w_k\Delta T^2 \tag{8.2}$$

To apply Kalman state estimation using the dynamics presented above, it is necessary to express (8.1) and (8.2) in terms of a state-space representation. The classic approach

is to model these equations using a linear, dynamic, discrete-time system as follows:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{\Psi}_k \mathbf{u}_k + \mathbf{\Gamma}_k \mathbf{w}_k \tag{8.3}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{8.4}$$

where $\mathbf{x}_k$ denotes the full system state and $\mathbf{u}_k$ the system inputs. This will hereafter be referred to as the "generic" state-space representation. The system outputs are given by $\mathbf{z}_k$ and contain the measurable states of the system. The matrix $\mathbf{\Phi}_k$ encodes the dynamics of the system and $\mathbf{\Psi}_k$ describes how the inputs drive the system dynamics. The vector $\mathbf{w}_k$ describes the process noise which accounts for unmodeled disturbances. The matrix $\mathbf{\Gamma}_k$ describes the weighting of the process noise on each of the states. The measurement noise $\mathbf{v}_k$ is included to account for noise in the measurement of outputs. Both noise terms are assumed to be white with zero mean and independent of one another.

For the model described by (8.1), the system states comprise the position, velocity, and acceleration of the laser pointer. The only measurable output of the system is the position of the laser pointer on the image plane at frame k. This implies the output and measurement noise are in fact constants and can be written as $\mathbf{z}_k = z_k$ and $\mathbf{v}_k = v_k$. It is assumed there are no driving forces $\mathbf{u}_k$ affecting the system dynamics. Under these assumptions and observations, a state-space representation assuming constant acceleration can be written as:

$$\mathbf{x}_k = \begin{bmatrix} s \\ v \\ a \end{bmatrix}_k \qquad \mathbf{\Phi}_k = \begin{bmatrix} 1 & \Delta T & \Delta T^2/2 \\ 0 & 1 & \Delta T \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{\Gamma}_k = \begin{bmatrix} \Delta T^2/2 \\ \Delta T \\ 1 \end{bmatrix} \qquad \mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

For the model described by (8.2), the system states comprise the position and velocity alone. Again, the position of the laser pointer is the only measurable state and it is assumed there are no driving forces affecting the dynamics. This leads to the following two-state representation under the assumption of constant velocity:

$$\mathbf{x}_k = \begin{bmatrix} s \\ v \end{bmatrix}_k \qquad \mathbf{\Phi}_k = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \qquad \mathbf{\Gamma}_k = \begin{bmatrix} \Delta T^2/2 \\ \Delta T \end{bmatrix} \qquad \mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

One final point to make regarding these representations is that the dynamics of the laser pointer are equally valid for movement along the horizontal and vertical directions.

In fact, the dynamics along the two directions of travel can be separated into two distinct state-space representations. Let $\mathbf{x}_{k,x}$ denote the state-space representation for movement along the horizontal direction and let $\mathbf{x}_{k,y}$ denote the representation along the vertical direction. Because the dynamics are equivalent, the matrices $\mathbf{\Phi}_k$, $\mathbf{\Psi}_k$, $\mathbf{\Gamma}_k$, and $\mathbf{H}_k$ remain as given and equations (8.3), (8.4) can be written as:

$$\begin{aligned}
\mathbf{x}_{k+1,x} &= \mathbf{\Phi}_k \mathbf{x}_{k,x} + \mathbf{\Psi}_k \mathbf{u}_k + \mathbf{\Gamma}_k \mathbf{w}_k \\
z_{k,x} &= \mathbf{H}_k \mathbf{x}_{k,x} + v_k \\
\mathbf{x}_{k+1,y} &= \mathbf{\Phi}_k \mathbf{x}_{k,y} + \mathbf{\Psi}_k \mathbf{u}_k + \mathbf{\Gamma}_k \mathbf{w}_k \\
z_{k,y} &= \mathbf{H}_k \mathbf{x}_{k,y} + v_k
\end{aligned}$$

8.2 Error Covariance Estimates

Kalman state estimation requires certain qualities of the process and measurement noise to be satisfied. Specifically, it requires that

$$\begin{aligned}
E\left[\mathbf{w}_k \mathbf{w}_i^T\right] &= \begin{cases} \mathbf{Q}_k & i = k \\ 0 & i \neq k \end{cases} \\
E\left[\mathbf{v}_k \mathbf{v}_i^T\right] &= \begin{cases} \mathbf{R}_k & i = k \\ 0 & i \neq k \end{cases} \\
E\left[\mathbf{w}_k \mathbf{v}_i^T\right] &= 0 \qquad \forall i, k
\end{aligned}$$

where $\mathbf{Q}_k$ is the process covariance and $\mathbf{R}_k$ is the measurement error covariance. Because the measurement noise is a constant for the models considered here, the measurement error covariance reduces to a constant $\mathbf{R}_k = R_k$. Similarly, the measurement noise reduces to a constant $\mathbf{v}_k = v_k$.

An assumption made earlier was for the noise terms to be independent of one another and to have the characteristics of white noise. By definition, white noise is a stationary process with zero autocorrelation for any two time steps $i \neq k$. Also by definition, two independent stationary processes will have zero cross-correlation for all time steps i, k.

This ensures that $E\left[\mathbf{w}_k\mathbf{w}_i^T\right] = 0$ and $E\left[v_k v_i^T\right] = 0$ for $i \neq k$, and that $E\left[\mathbf{w}_k v_i^T\right] = 0$ for all i, k.

Let $\mathbf{Q}_{k,x}$ and $\mathbf{Q}_{k,y}$ denote the process covariance for the horizontal and vertical representations, respectively. Similarly, define $R_{k,x}$ and $R_{k,y}$ to be the corresponding measurement error covariances. In measuring the horizontal and vertical coordinates of the laser pattern on the image plane there is a finite amount of quantization error introduced by rounding the estimate of the centroid to the nearest integer pixel coordinates. It is therefore reasonable to assume that 1/3 of the time rounding will produce the correct estimate, 1/3 of the time there will be an error of negative one pixel, and 1/3 of the time there will an error of positive one pixel. On average, the mean error is zero. Using a straightforward computation of the variance gives:

$$R_{k,x} = R_{k,y} = \frac{1}{3}((-1)^2 + (0)^2 + (+1)^2) = \frac{2}{3} = \sigma_z^2 \qquad \forall k$$

To obtain an accurate expression for the process covariance, the works of [30] and [31] are followed. According to [31], the process covariance can be expressed as $\mathbf{Q}_k = \sigma_a^2 \mathbf{\Gamma}_k \mathbf{\Gamma}_k^T$, where σ_a^2 denotes the variance of the random acceleration component caused by a user's hand tremor. In [30] the authors investigate applying Kalman state estimation to predict the motion of human body parts in a sequence of images. More specifically, the authors aim to predict the motion of the hands and arms for gesture recognition.

The experimental configuration described by the authors can be considered analogous to a user interacting with the training simulator. In the context of the training simulator, a user's hands and arms produce a corresponding movement of the laser pattern on the image plane. Owing to these similar experimental configurations, the results obtained in [30] are used to guide the derivation of the process covariance. According to their experiments they estimate $\sigma_a < 1225 \frac{pixel}{s^2}$ which, at a rate of 15 fps, leads to $\sigma_a = 5.44 \frac{pixel}{frame^2}$.

Although the camera used in the training simulator operates at 30 fps, the results obtained by the authors at a rate of 15 fps will be used as an approximation. This decision is motivated by noting that a standard deviation of $5.44 \frac{pixel}{frame^2}$ in the random component of acceleration seems very reasonable for cameras operating at a resolution of 1024x768 and with a frame rate of 30 fps. Using the values derived for $\mathbf{\Gamma}_k$, the process

covariance for the three-state system is:

$$\mathbf{Q}_{k,x} = \mathbf{Q}_{k,y} = \sigma_a^2 \begin{bmatrix} \Delta T^4/4 & \Delta T^3/2 & \Delta T^2/2 \\ \Delta T^3/2 & \Delta T^2 & \Delta T \\ \Delta T^2/2 & \Delta T & 1 \end{bmatrix}$$

and for the two-state representation is:

$$\mathbf{Q}_{k,x} = \mathbf{Q}_{k,y} = \sigma_a^2 \begin{bmatrix} \Delta T^4/4 & \Delta T^3/2 \\ \Delta T^3/2 & \Delta T^2 \end{bmatrix}$$

The final error covariance matrix which must be described before continuing is the state estimation error covariance $\mathbf{P}_k^-$. It is assumed, through Kalman state estimation, that an estimate $\hat{\mathbf{x}}_k^-$ of the states at time t_k is available based on current knowledge of the system dynamics and error characteristics. Because this (prior) estimate is derived based on a noisy process and on noisy measurements, the estimate of the states differs from that of the true system states by $\mathbf{x}_k - \hat{\mathbf{x}}_k^-$. The covariance associated with the estimation error is thus:

$$\mathbf{P}_k^- = E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)^T\right]$$

which, for the three-state system described, gives:

$$\mathbf{P}_{k,x}^- = \mathbf{P}_{k,y}^- = \begin{bmatrix} \sigma_{s,s}^2 & \sigma_{s,v} & \sigma_{s,a} \\ \sigma_{v,s} & \sigma_{v,v}^2 & \sigma_{v,a} \\ \sigma_{a,s} & \sigma_{a,v} & \sigma_{a,a}^2 \end{bmatrix}$$

and for a two-state system leads to:

$$\mathbf{P}_{k,x}^- = \mathbf{P}_{k,y}^- = \begin{bmatrix} \sigma_{s,s}^2 & \sigma_{s,v} \\ \sigma_{v,s} & \sigma_{v,v}^2 \end{bmatrix}$$

As will be shown in the following section, only an initial estimate $\mathbf{P}_0^-$ of the state estimation error covariance is required because the Kalman estimation process will iteratively refine a more accurate representation for $\mathbf{P}_k^-$. The experimental results obtained

in [30] are again used to guide the derivation of the initial estimation error covariance $\mathbf{P}_0^-$. In their work, the authors assume the position, velocity, and acceleration estimation errors are independent and uncorrelated - leading to $\mathbf{P}_0^-$ being a diagonal matrix with entries given by:

$$\sigma_{s,s}^2 = \frac{9}{16} \qquad \sigma_{v,v}^2 = \frac{25}{16} \qquad \sigma_{a,a}^2 = \left(\frac{49}{9}\right)^2$$

8.3 State Estimation

With the system dynamics and error covariance matrices now defined, the Kalman state estimation process can be described in greater detail. A generic state-space model, as described by equations (8.3) and (8.4), is assumed. For the case of two and three-state estimation, the initial velocity and acceleration are assumed to be zero. With the first image acquired by the camera, there is no prior knowledge of where the laser pointer would be on the image plane. Therefore, the center of the image plane is taken as the initial position estimate. With each frame k captured by the camera, the following five-step process is carried out:

1. Compute the Kalman gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + R_k\right)^{-1}$$

   The Kalman gain acts as a sort of weighting matrix when computing the a posteriori state estimates $\hat{\mathbf{x}}_k$ (described below). As shown, the Kalman gain is influenced by the current state estimation error covariance as well as the measurement error covariance. This implies the Kalman gain is a function how reliable state measurements are and how reliable the a priori state estimates are.

2. Update the state estimates

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-\right)$$

   The a posteriori state estimates are obtained by superimposing the current estimates with a weighting of the residual $\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-$. Effectively, an estimate $\hat{\mathbf{x}}_k^-$ of the states is made, a measurement of the system outputs $\mathbf{z}_k$ is obtained, and a weighted version of the error between these is used to update the estimate.

3. Update the estimation error covariance

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^-$$

This step computes the a posteriori estimation error covariance matrix by weighting the a priori one. Because the Kalman gain is a function of both the a priori state estimation error covariance and the measurement error covariance, this update step reflects how reliable state estimation is at frame k.

4. Project the state estimate ahead

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k\hat{\mathbf{x}}_k$$

Using the a posteriori state estimate at frame k, the a priori state estimate at frame $k+1$ can be obtained using the state transition matrix $\mathbf{\Phi}_k$. The estimate $\hat{\mathbf{x}}_{k+1}^-$ becomes the anticipated position of where the laser pointer will be in the next frame based on the system dynamics and the current error characteristics.

5. Compute the estimation error covariance at frame $k+1$

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{Q}_k$$

Finally, the a posteriori estimation error covariance is propagated to frame $k+1$. As shown, the propagation is achieved using the state transition matrix as well as the process covariance matrix. This ensures there is always a current assessment of the state estimation accuracy available for each newly captured frame. This enables a new Kalman gain can to be computed and for the entire Kalman state estimation process to be applied recursively.

By applying this process at each frame it is possible to estimate where the laser pointer should be in the current frame based on its position in the previous frame. This enables an estimate of where to position the cursor with respect to the output video sequence to be obtained at frame $k+1$, given its position at frame k. By modeling the dynamics of the laser pointer using an appropriately chosen model, and by carefully specifying the error covariance matrices, these estimates can significantly improve the dynamic accuracy of the system.

To be clear, the Kalman state estimation process needs to be applied along both the horizontal and vertical directions to estimate the two coordinates of the laser pointer

location. This can be readily achieved by creating two separate Kalman systems initialized with two a priori state estimates: $s_{0,x}^- = 1024/2$ and $s_{0,y}^- = 768/2$. Then, with each frame, Kalman state estimation is applied independently on each of the estimates. For the three-state representation, the initial estimates can be written as:

$$\hat{\mathbf{x}}_{0,x}^- = \begin{bmatrix} s_{0,x}^- \\ v_{0,x}^- \\ a_{0,x}^- \end{bmatrix} = \begin{bmatrix} 1024/2 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{\mathbf{x}}_{0,y}^- = \begin{bmatrix} s_{0,y}^- \\ v_{0,y}^- \\ a_{0,y}^- \end{bmatrix} = \begin{bmatrix} 768/2 \\ 0 \\ 0 \end{bmatrix}$$

and for the two-state representation as:

$$\hat{\mathbf{x}}_{0,x}^- = \begin{bmatrix} s_{0,x}^- \\ v_{0,x}^- \end{bmatrix} = \begin{bmatrix} 1024/2 \\ 0 \end{bmatrix} \qquad \hat{\mathbf{x}}_{0,y}^- = \begin{bmatrix} s_{0,y}^- \\ v_{0,y}^- \end{bmatrix} = \begin{bmatrix} 768/2 \\ 0 \end{bmatrix}$$

This approach uses Kalman state estimation to independently estimate where the horizontal and vertical coordinates of the laser pointer on the image plane will be at frame $k+1$. With an estimate of the laser location, one of the forward mapping solutions described in Section 6.1 can be used to transform the image coordinates into a cursor location with respect to the output video sequence. This is how the cursor location is estimated and how, effectively, the apparent lag of the system can be reduced.

For an indirect mapping solution, the system states and measured outputs need to be redefined. An indirect mapping solution provides an estimate for the cursor position by recovering the camera pose and intersecting a ray from the camera's center of projection with the projection screen. In this scenario, the system states are taken directly as the position, velocity, and acceleration of the physical point of intersection. Indeed, the measurable outputs of the system are the locations m_i of the feature points on the image plane. However, the point of intersection computed using the indirect mapping approach is taken to be the *inferred* measurable output z_k of the system.

Using this approach, the locations of the feature points on the image plane are measured, the camera pose is recovered, and the point of intersection is computed. A state vector $\mathbf{x}_k$ is then constructed from the position, velocity, and acceleration of the physical point of intersection. Two separate Kalman systems are created: one for the horizontal component of movement and one for the vertical. Kalman state estimation is then

applied to obtain an estimate, at frame $k+1$, of where the new point of intersection would be. This estimate, s_{k+1}^{-}, is then converted to a cursor position $\hat{q}$ with respect to the output video sequence.

Chapter 9

Performance Evaluation Criteria and Results

Building upon the theory and results obtained to this point, an objective comparison of solutions to the mapping problem is now presented. In Chapter 6, a single direct solution and three indirect solutions were introduced. It was shown through simulation and experiments on real image data that Tsai's extrinsic algorithm and the coplanar POSIT algorithm did not provide sufficient accuracy to implement an effective indirect solution. However, it was shown that a simple modification to Tsai's approach enabled a far more accurate solution to be achieved.

Owing to these observations, the proposed modification to Tsai's approach is evaluated more thoroughly based on a set of core evaluation metrics. For comparison, the Nintendo Wii pointing device (Wiimote) is also subject to the same evaluation criteria. The encouraging results provided in [9] indicate the homography solution can indeed constitute an effective alternative pointing system. An implementation of the homography solution is included here utilizing the novel image processing routines described in Chapter 4. For comparison, results using a nonlinear direct mapping solution[1] are also included.

[1]The nonlinear mapping solution evaluated was developed by the National Research Council of Canada. Details of the solution have been omitted due to intellectual property rights.

The criteria used for evaluation and comparison are measures based on accuracy, latency, jitter, drift, and reliability. Each of these metrics is described in greater detail in the sections that follow. To determine whether the measures are global across the entire projection screen or whether they are local to specific regions, the pointing area is logically divided into nine separate regions, as shown in Figure 9.1, and the measures are recorded for each.

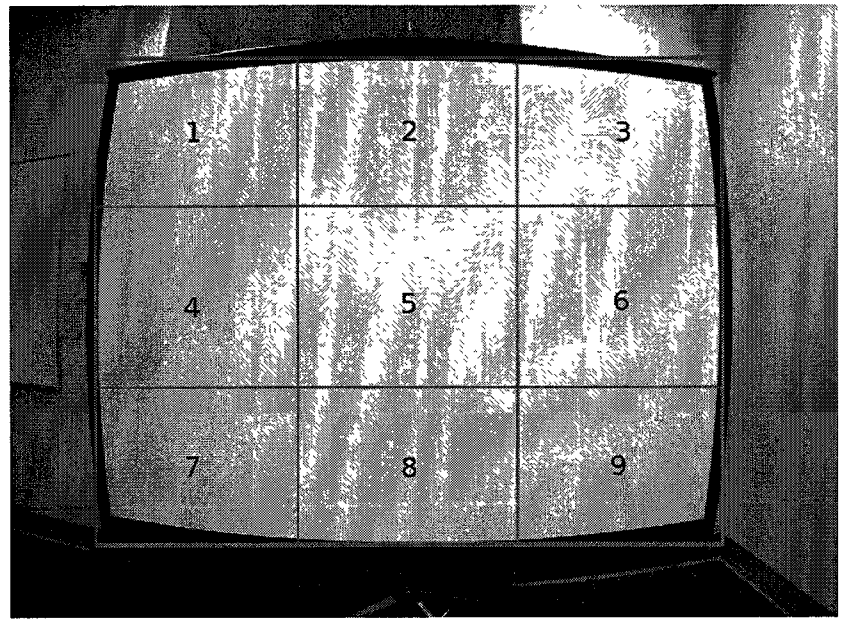

Figure 9.1: Logical partitions used to determine if metrics are global or local across the projection screen.

If a measure is indeed local, it is important the individual solutions are evaluated at the exact same location within the region to obtain a fair comparison between them. In order to address this concern, an "alignment image" is shown on the projection screen containing nine simulated feature points and nine control points (Figure 9.2). As described in Section 6.2, the nine white squares projected on screen are used to simulate the feature points needed by indirect mapping solutions. The control points serve to ensure a pointing device is directed at the exact same location within a region. For direct mapping solutions, the laser pointer is positioned over top of the control points and fixed while measurements are recorded. For Tsai's modified algorithm and for the Nintendo Wiimote, a laser is first affixed to the camera (respectively Wiimote) and then is oriented to be positioned directly over a control point.

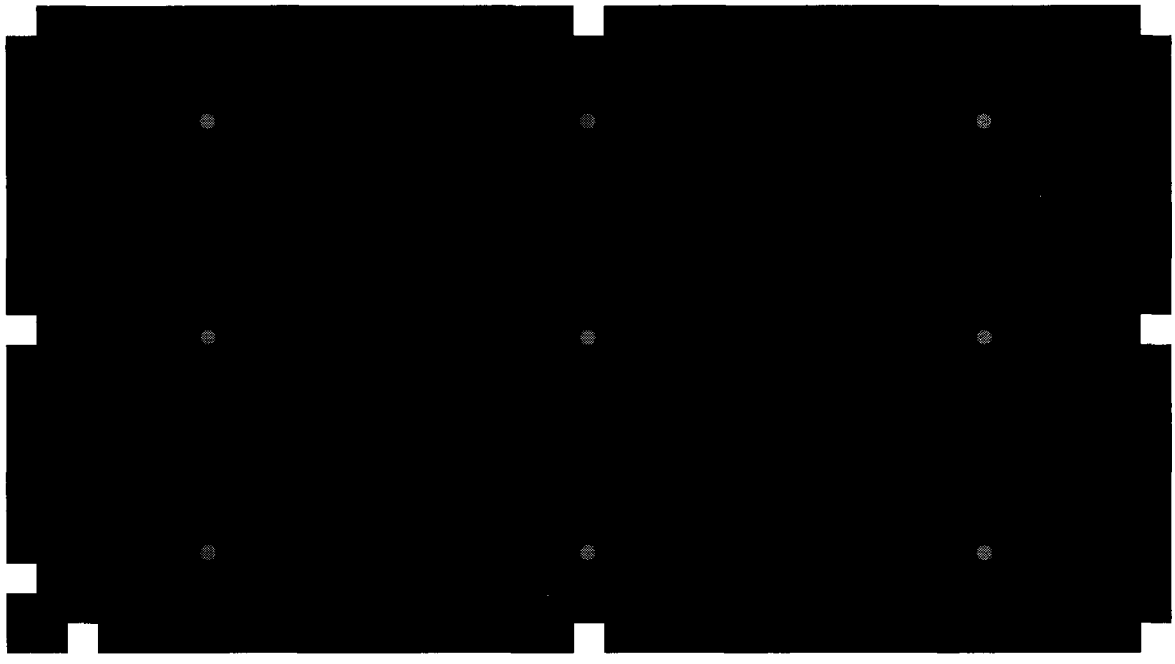

Figure 9.2: Alignment image showing simulated feature points (white squares) and control points (red circles).

The proposed approach to ensure the hand-held camera and Nintendo Wiimote point at the same intra-region location is admittedly flawed. The ideal configuration would be for the hand-held camera and for the internal camera of the Wiimote to be oriented such that a direct line of sight is made between their centers of projection and the control points. Clearly, unless the true camera orientations were known a priori, this simply could not be achieved in practice. The proposed approach ensures, at the very least, the camera and Wiimote will aim at the same intra-region location when recording the different measures. Thus, when measures are reported for a specific region, it is guaranteed the measures were recorded at the same intra-region location.

Simulation results and experiments on real image data have shown that pointing accuracy for indirect mapping solutions is dependent upon the relative location of the camera in the training environment. Direct mapping solutions are, however, invariant to a user's location because the accuracy of homography and nonlinear mapping solutions depends only on the accuracy of the map established between the projection screen and image plane. Therefore, provided the camera viewing the projection screen is fixed at a single location, the accuracy would be invariant to a user's location. Based on this

argument, the evaluation criteria will be recorded at various locations for the two indirect mapping solutions but not for the two direct mapping solutions.

All tests were conducted in a room measuring approximately 4 m by 7 m which limited the maximum range of operation to an arc of about 6.5 m from the projection screen. The lighting conditions were carefully controlled by blocking out both internal and external light sources as best as possible. Since tests were conducted over several days, the weather conditions impacted the amount of external light entering the room. As described in Chapter 4, the amount of ambient lighting in a room can impact the image thresholding and successive relaxation algorithms so it is important to ensure lighting conditions are at least consistent from one set of measurements to the next. Over a period of five days, the ambient lighting measured was in a very narrow range: 4.86 to 7.19 Lux, which ensured measurement results were consistent from one day to the next.

The projection screen used for testing measured about 2.10 m by 1.50 m. A high-definition video projector was used to project images onto the screen at a resolution of 1920x1080 pixels with a refresh rate of 60 Hz. The projector was carefully positioned to ensure the physical dimensions of the projected images measured 1920± 5mm by 1080± 5mm. This ensured each pixel had an area of 1 mm^2 as required by the implementation of Tsai's modified algorithm. The camera resolution was fixed at 1024x768 and set to capture images with 8-bit intensity. Using the camera utility application, the shutter time was set to 15 ms and the gain was fixed at 30 dB. Based on the internal lighting conditions and camera configuration, a threshold of $\lambda = 230$ was chosen. The red laser pointer used for testing had a wavelength of 642 nm and an average transmit power of 0.84 mW.

For the homography and nonlinear mapping solutions, the external camera is fixed at a location 3.30 m from the projection screen. The height of the camera was fixed at 0.75 m below the top of the projected video sequence and the horizontal position was fixed at 0.95m from the left hand edge of the projected video. In terms of the (U,V,W) coordinate system described in Section 3.1, this equates to a point $P = (0.95, 0.75, -3.30)$ with a measurement error of ± 0.05 m. A laser pointer was then fixed on a tripod behind the camera at a position measured to be $Q = (0.95, 0.80, -3.50) \pm 0.05$ m. From there, the laser pointer was aimed at each of the control points while measurements were recorded.

To evaluate the performance of the Nintendo Wiimote, the sensor bar is positioned just below the bottom edge of the projected video and centered between the left and right edges. The Nintendo Wii's video output was then connected to the video projector and displayed at a resolution of 640x480 pixels, the maximum for the Nintendo Wii. The Wiimote is mounted on a tripod at a fixed point $P = (0.95, 0.75, -3.50) \pm 0.5\text{m}$. To simulate moving the Wiimote to different locations, the tripod is fixed while the sensor bar itself is rotated. By placing a compass on top of the sensor bar, it is possible to mimic rotations to within an error of $\pm 1°$. By varying the rotation from 0° to 50° in increments of 10°, it was possible to simulate placing the Wiimote in 6 different locations as listed below. For clarity, a plot of the locations in the U-W plane is provided in Figure 9.3.

$$
\begin{aligned}
P_1 &= (0.95, 0.75, -3.50) \pm 0.05 \text{ m} \\
P_2 &= (1.60, 0.75, -3.45) \pm 0.05 \text{ m} \\
P_3 &= (2.15, 0.75, -3.30) \pm 0.05 \text{ m} \\
P_4 &= (2.70, 0.75, -3.00) \pm 0.05 \text{ m} \\
P_5 &= (3.20, 0.75, -2.70) \pm 0.05 \text{ m} \\
P_6 &= (3.65, 0.75, -2.25) \pm 0.05 \text{ m}
\end{aligned}
$$

When taking measurements within a region, the alignment image was first shown on the projection screen and the affixed laser oriented to intersect the control point. Once the alignment was achieved, the Nintendo Wii video was projected on screen and measurements were recorded.

To evaluate the proposed variation of Tsai's algorithm, a camera was mounted on a tripod and positioned at each of the P_k locations described for evaluating the Wiimote. In this configuration, however, the tripod was physically moved to each of the aforementioned locations. Each location can be considered a point in polar coordinates with a fixed radius of 3.50 m and angles ω varying from 0° to 50° in increments of 10°. Each angle is essentially the angle of incidence between the pointing device and the normal of the projection screen. An angle of zero indicates the tripod is located directly between the right and left hand edges of the projected video. Again, prior to taking measurements within a region, the camera was oriented so the affixed laser intersected the appropriate control point. Throughout testing the alignment image was continuously projected onto the projection screen while measurements were recorded.

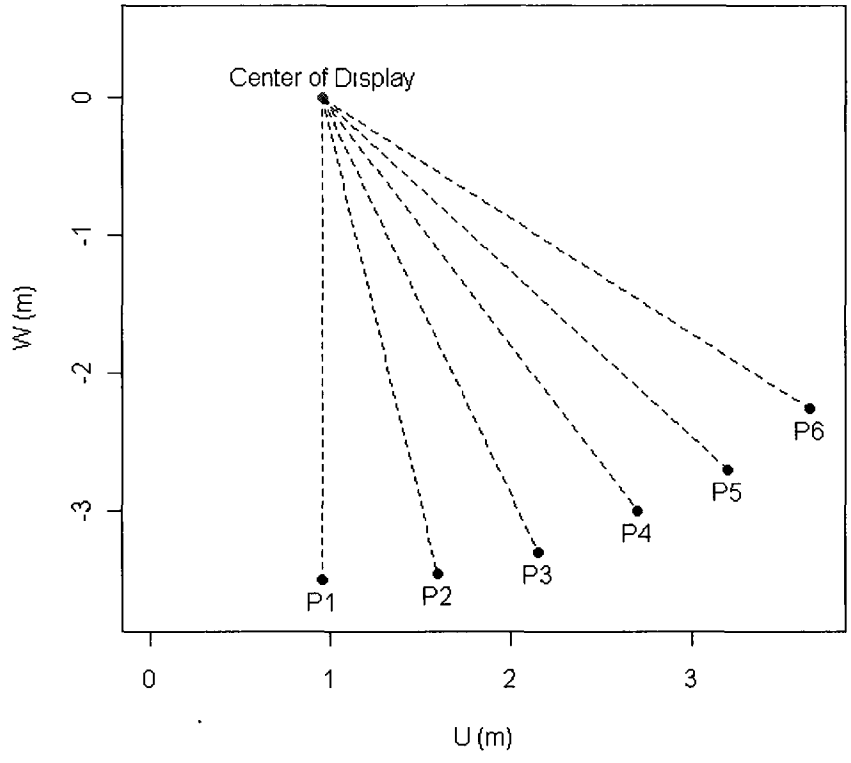

Figure 9.3: Coordinate locations used for testing plotted in the U-W plane.

9.1 Static Accuracy

Test Condition Summary - Solutions Evaluated: Homography estimation, nonlinear map, Nintendo Wii, and the variant of Tsai's extrinsic algorithm. Image processing applied: Image segmentation using the nonlinear-time component labeling algorithm, image thresholding, and successive relaxations. Kalman State Estimation: Not applied. Ambient lighting: Less than 8 Lux.

The static accuracy of a solution refers to the difference in pixels, at a specific resolution, between the estimated cursor position and the true location a user is pointing when the pointing device is *stationary.* For a direct mapping solution, this equates to the difference in pixels between the tip of the cursor and the centroid of the laser spot

cast on the projection screen. For an indirect mapping solution the static accuracy must be redefined because the true camera pose is not known a priori. In this case, a baseline accuracy is first constructed for each region at position P_1 by determining the difference between the tip of the cursor and the centroid of the affixed laser pattern cast on the projection screen. The relevant measure is then the change in static accuracy as a function of the angle of incidence ω.

To measure the static accuracy, an external camera was used to capture an image of the projection screen showing the estimated cursor position and the location of the projected laser pattern. From the resulting image the coordinates of the cursor and laser pattern were manually extracted, using the open-source Gimp image manipulation software, and used to computed the $\mathcal{L}_2$-norm between them. The resolution of the external camera was fixed at 1920x1080 and was positioned as accurately as possible to ensure the projected video occupied the entire image plane. Using this approach, each pixel of difference in the captured image was equivalent to approximately 1mm of separation between the tip of the cursor and the centroid of the projected laser pattern. The measurement error in estimating the centroid of the laser pattern was about ± 2 pixels. The static accuracy, in pixels, for the two direct mapping solutions is reported in Table 9.1. The static accuracy, as a percentage of the display diagonal, is reported in Table 9.2.

Region	Homography (pixels)	Nonlinear Map (pixels)
1	3±2	3±2
2	3±2	3±2
3	3±2	0±2
4	3±2	3±2
5	5±2	3±2
6	5±2	3±2
7	5±2	0±2
8	5±2	3±2
9	3±2	3±2

Table 9.1: Static accuracy, in pixels, measured for the homography estimation and nonlinear mapping approaches.

Taking measurement errors into consideration, the results obtained show the homog-

Region	Homography (% display)	Nonlinear Map (% display)
1	0.137±0.045	0.137±0.045
2	0.137±0.045	0.137±0.045
3	0.137±0.045	0.137±0.045
4	0.137±0.045	0.137±0.045
5	0.227±0.045	0.137±0.045
6	0.227±0.045	0.137±0.045
7	0.227±0.045	0±0.045
8	0.227±0.045	0.137±0.045
9	0.137±0.045	0.137±0.045

Table 9.2: Static accuracy, as a percent of the display diagonal, measured for the homography estimation and nonlinear mapping approaches.

raphy approach is accurate to within at most 7 mm for a user positioned 3.50 m from the projection screen. The nonlinear mapping approach is shown to be accurate to within 5 mm at the same location; a marginal but nonetheless important improvement. In certain instances, and subject to the margin of error, the nonlinear mapping approach is even shown to be 100% accurate. With regard to the question of locality, both approaches demonstrate a negligle amount variance between regions - confirming static accuracy is indeed a global measure for the two approaches considered. Clearly, the results from both approaches are very encouraging and either one could serve as a satisfactory solution to the direct mapping problem. However, given a choice between the two, the nonlinear mapping solution is consistently as accurate, if not more so, than the homography estimation approach.

A comparison of the change in static accuracy versus angle of incidence is shown in Figure 9.4 for the modification of Tsai's extrinsic algorithm and for the Nintendo Wii pointing device. Recall the values reported are the *change* in accuracy observed at a specific angle of incidence relative to the accuracy at 0°. Therefore, the change in static accuracy at 0° is always identically zero as confirmed by the plots. The general trend shows that increasing the angle of incidence causes the static accuracy to decrease, i.e. the separation between the affixed laser pointer and estimated cursor position becomes greater.

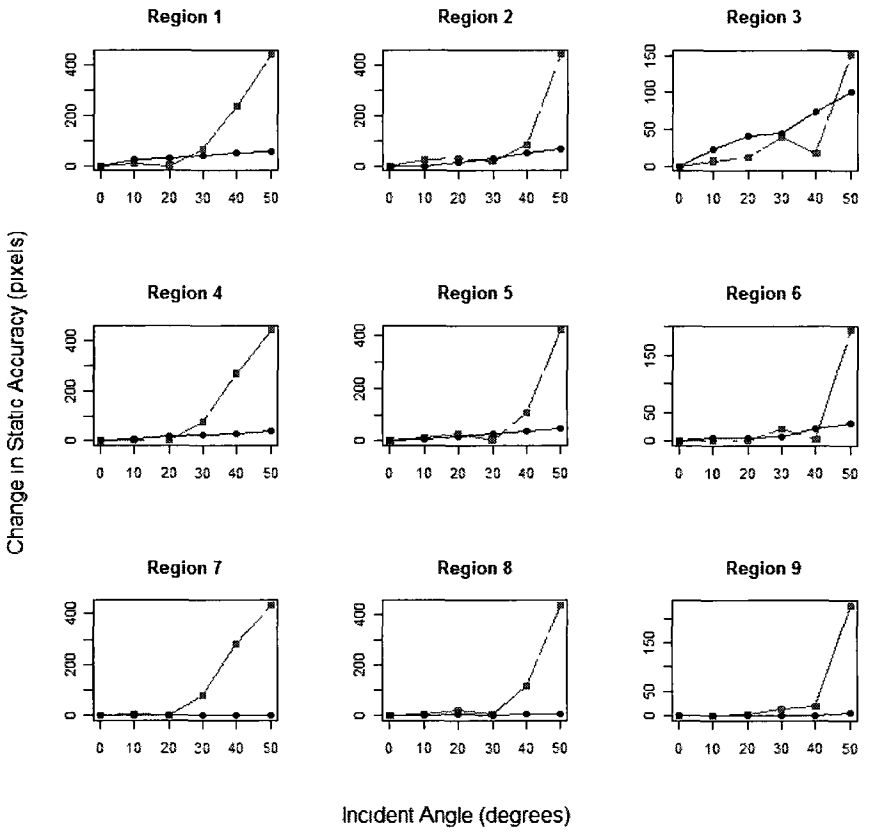

Figure 9.4: Change in static accuracy versus incident angle for the modification of Tsai's extrinsic algorithm (red squares) and the Nintendo Wii pointing device (blue circles).

An interesting observation is how the accuracy of the proposed modification to Tsai's algorithm tends to increase column-wise along the projection screen. Consider, for example, the middle row of plots. The static accuracy of the two approaches is comparable in region 4 up to incident angles in the neighbourhood of 20°. In region 5 the static accuracy of the two approaches is comparable up to roughly 30° while, in region 6, the static accuracies are comparable up to 40°. This trend is also visible in the first and third row of plots. The most plausible explanation for this observation has to do with the correction factor Δx applied to the image and the locations P_k chosen for testing.

When moving the camera to different locations and attempting to view the same fixed point, the camera's orientation must be changed to ensure the affixed laser intersects the desired point. The locations chosen for testing have the camera initially positioned at

the center of the projection screen and progressively moving toward to the right hand side (column) of the screen. As this occurs, the rotation angle θ of the camera must also increase in order to remain fixed on the same point. However, the change in θ is much greater when viewing the left column of the projection screen when compared to viewing the right column. Accordingly, the magnitude of correction Δx is also greater. Evidently, as the required correction factor increases there is a corresponding decrease in the observed static accuracy.

Of the two approaches, the Nintendo Wii pointing device is clearly more robust to changes in the angle of incidence. Based on the locations evaluated, the Nintendo Wii can be considered to have a wider range of operation for a comparable level of accuracy. The primary advantage of the modification to Tsai's algorithm is that, over its limited range of operation, the camera pose can be accurately recovered and user to provide valuable feedback to users. With respect to locality, the static accuracy must be considered a local measure for both approaches since it varies over the surface of the projection screen.

9.2 Dynamic Accuracy

> **Test Condition Summary -** Solutions Evaluated: Nonlinear map. Image processing applied: Image segmentation using the nonlinear-time component labeling algorithm and image thresholding. Kalman State Estimation: Two-state and three-state estimation applied. Ambient lighting: Less than 8 Lux.

In Chapter 8, Kalman state estimation was proposed as a means to improve the dynamic pointing accuracy of a given solution. As discussed in the previous section, each approach to estimating the cursor position has its respective advantages, disadvantages, and expected baseline static accuracy. Since each solution has its respective baseline characteristics, the relevant measure considered here for dynamic accuracy is the expectation in improvement over the baseline. Therefore, it is sufficient to evaluate the improvement in dynamic accuracy for a single solution and extrapolate that this improvement could be obtained over the baseline static accuracy of any given solution.

To evaluate the impact on dynamic accuracy, the nonlinear mapping solution was

used to estimate the cursor position and compared to the location predicted by both a two-state and three-state Kalman system. The decision to use the nonlinear mapping solution was motivated primarily by its accuracy and low latency. Using an external camera operating at 60 fps and with a resolution of 1920x1080, a series of images were captured showing the true laser pointer location, the cursor position estimated by the nonlinear map, and the two cursor positions estimated by the Kalman systems. Each image was then manually processed to recover the $\mathcal{L}_2$ norm between the true and estimated cursor locations. The improvement in dynamic accuracy using state estimation was then computed relative to the accuracy obtained using the nonlinear map alone. Figure 9.5 shows the results obtained from a series of 60 images. All images were captured with the camera and laser pointer co-located at P_1.

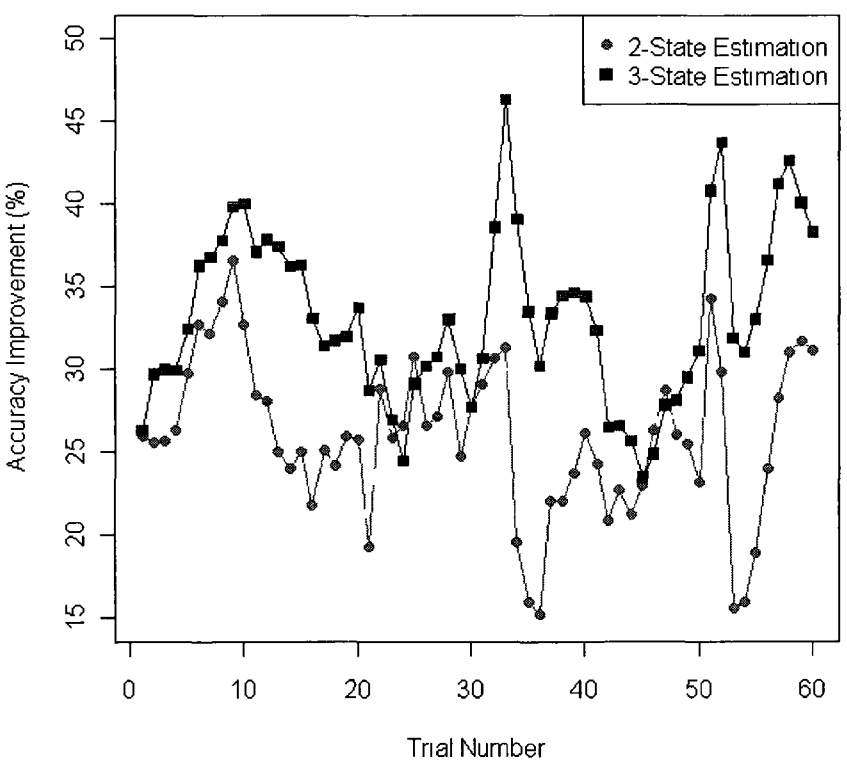

Figure 9.5: Expected improvement in dynamic accuracy using two-state and three-state Kalman estimation.

To obtain these results a series of 500 images were captured, from which 60 were

chosen at random. The trajectory of the laser pointer was essentially chosen at random, alternating between smooth strokes and sharp turning motions. The speed of movement was also varied between slow, natural, and rapid movements to capture a range of dynamic pointing scenarios. Although 60 images is a relatively small sample size, the results obtained do illustrate state estimation provides an expectation of improvement over the baseline accuracy of a given solution.

As shown, the relative improvement using a two-state estimator was found to range from about 15-36%, and from roughly 23-46% for the three-state estimator. In nearly every trial, the improvement in accuracy using a three-state model outperformed the improvement of a two-state model. For the few cases where the two-state model performed better it was only by a matter of one or two percentage points. Over the 60 trials, the average improvement in accuracy was 26% for the two-state model and 33% for the three-state model.

Clearly, it is beneficial to model the dynamics of the laser pointer as having constant acceleration as opposed to having constant velocity. Most interaction using a mouse consists of rapidly moving the cursor toward a given area, followed by a deceleration in the area of interest to improve the pointing accuracy. Once the cursor has been positioned in the region of interest, there are often small and erratic movements to position the cursor at its final destination. These small movements are akin to the random acceleration components captured by the three-state model. Based on the expected, natural interaction of a user, the three-state predictor intuitively provides the better model - an assumption confirmed by the results observed.

9.3 Function and System Latency

Test Condition Summary - Solutions Evaluated: Homography estimation, nonlinear map, and the variant of Tsai's extrinsic algorithm. Image processing applied: Image segmentation using the nonlinear-time component labeling algorithm, image thresholding, and successive relaxations. The variant of Tsai's approach is also evaluated without successive relaxations applied. Kalman State Estimation: Not applied. Ambient lighting: Less than 8 Lux.

For a solution to be truly effective, it should not only be accurate but should be responsive to input from the end-user. That is, if a user moves the pointing device, there should be an almost instantaneous corresponding movement of the cursor on screen. To capture the responsiveness of a given solution two measures of latency are proposed: function latency, and system latency. The function latency is a measure of the computation time required to acquire, process, and generate an estimate for the cursor position from an image. Function latency is a white-box analysis of the internal software implementations. The Nintendo Wii implementation, however, must be considered a black-box since the cursor estimation routine cannot be profiled directly. System latency is proposed as a measure to analyze the responsiveness of a black-box system.

9.3.1 Function Latency

The function latency of the homography, nonlinear map, and modified Tsai algorithms was measured by profiling the cursor estimation routine over 500 captured images. Using the `QueryPerformanceCounter()` routines available in the Windows API, the function latency was measured to within microsecond resolution. All images were captured with the camera and/or laser pointer located at P_1. Figure 9.6 shows a boxplot for the computation time of the three solutions assuming the nonlinear-time component labeling algorithm is implemented and successive relaxations are applied to each image. For comparison, the computation time of the modified Tsai algorithm without successive relaxations is also included. The mean computation times are recorded in Table 9.3.

Approach	Mean Function Latency (ms)
Homography	33.20
Modified Tsai with Relaxations	75.09
Modified Tsai without Relaxations	33.22
Nonlinear Map	33.16

Table 9.3: Mean function latency for the homography, nonlinear map, and modification of Tsai's extrinsic algorithm.

From the mean computation times, three of the approaches evaluated are able to run within the frame rate of the camera. This indicates that for each image captured, an

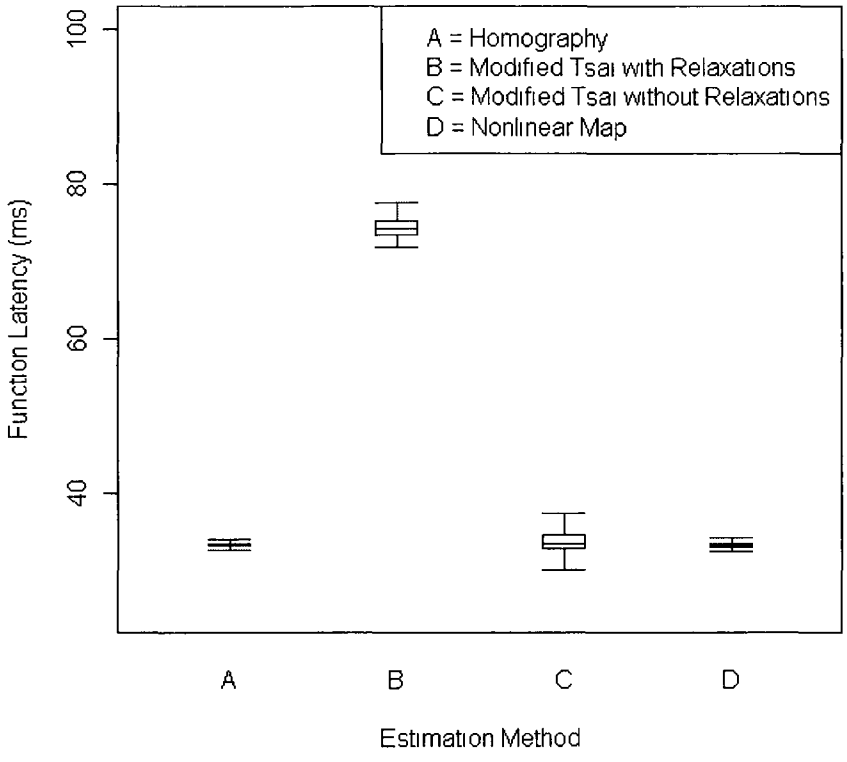

Figure 9.6: Function latency for the homography, nonlinear map, and modification of Tsai's extrinsic algorithm.

estimate of the cursor position can be made prior to the arrival of a new image resulting in a system as responsive as possible. For the modification to Tsai's extrinsic algorithm, the function latency is only within the camera frame rate if successive relaxations is not applied to the acquired images. The reason this is observed has to do with the area occupied by the projection of the feature points on the image plane. With the camera located at P_1 each feature point projects a component of approximately 400 pixels on the image plane. With nine simulated feature points in total, this equates to a total of 3600 pixels.

From Figure 4.4, the expected computation time for the successive relaxation algorithm was approximately 75 ms when applied to a grid of 3600 pixels^2. Recall, however, those results were obtained assuming the pixel intensities were chosen from a Gaus-

sian distribution and represented a worst-case scenario. When applied to the simulated feature points used for the modified Tsai algorithm, it is reasonable to assume the computation time would be within 75 ms because there would be fewer function calls needed to complete the relaxation process. Estimating the true computation time for applying successive relaxations to be anywhere from 1/2 to 3/4 of the simulated time, the latency of this function alone could take 37.50 to 56.25 ms. Taking into account there is a finite amount of computation time needed to capture an image, apply thresholding, and estimate the cursor position, it is easy to reason the mean computation time would be near the value of 75.09 ms reported.

9.3.2 System Latency

Similar to the function latency, the system latency can be used to evaluate the responsiveness of a given solution. The system latency measures the amount of time elapsed between the movement of a pointing device and the corresponding movement of the mouse cursor. Using a high-speed camera operating at 500 fps, it becomes possible to capture the instant these two events occur. At this rate, the system latency can be measured to within a resolution of 2 ms. A bar chart of the system latency for the homography, nonlinear map, Nintendo Wii, and modified Tsai algorithms is included in Figure 9.7. Again, the latency is reported for the modification to Tsai's algorithm with and without successive relaxations. The results were obtained by focusing the [affixed] laser pointer on a single point, moving the pointer, and then recording the elapsed time. The bar chart shows the mean system latency over 10 trials for each approach.

Comparing the system latencies to the values in Table 9.3, and noting the high-speed camera has a time resolution of 2 ms, the values obtained are as expected. With the mean function latency of the homography, nonlinear map, and modified Tsai approach (without relaxations) being just under the frame rate of the camera, the time resolution causes these system latencies to be rounded toward 34 ms. For the modification to Tsai's algorithm with relaxations, the system latency is slightly greater than the mean function latency recorded but can reasonably be accounted for by the measurement variance. For the Nintendo Wii pointing device, the system latency recorded was only 16 ms. According to [20], the Nintendo Wii uses an infra-red camera operating at 100 Hz which

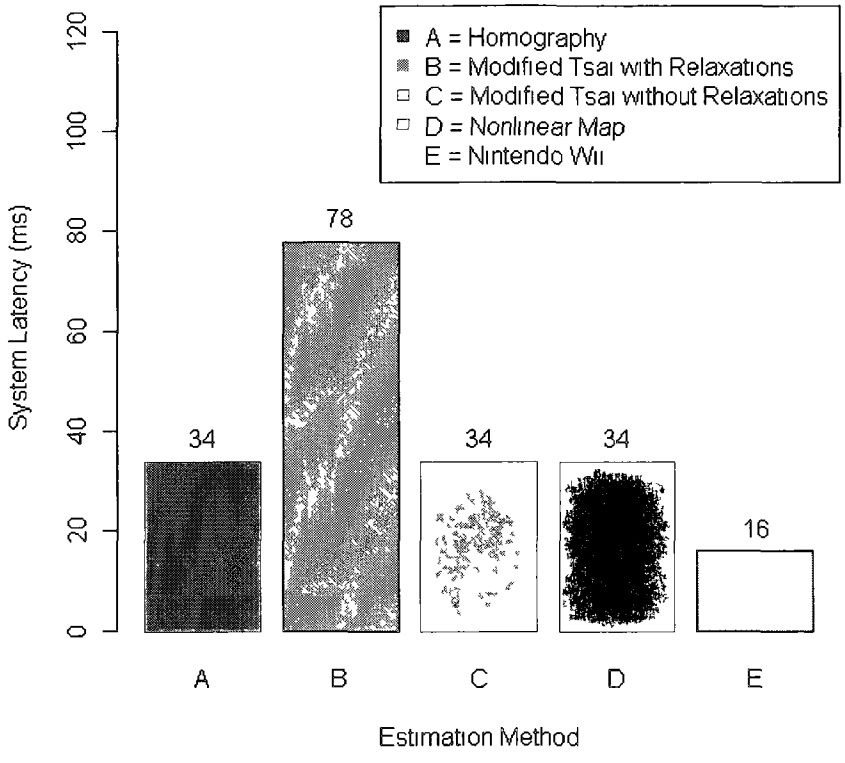

Figure 9.7: System latency for the homography, nonlinear map, Nintendo Wii pointing device, and modification of Tsai's extrinsic algorithm

enables it to provide such a significant improvement in system latency over the other four approaches.

9.4 Jitter, Drift, and Reliability

Test Condition Summary - Solutions Evaluated: Homography estimation, nonlinear map, Nintendo Wii, and the variant of Tsai's extrinsic algorithm. Image processing applied: Image segmentation using the nonlinear-time component labeling algorithm, image thresholding, and successive relaxations. The variant of Tsai's approach is also evaluated without successive relax-

ations applied. Kalman State Estimation: Not applied. Ambient lighting: Less than 8 Lux.

In addition to accuracy and latency, jitter and drift are common measures used to characterize the performance of three-dimensional tracking systems [33]. Although the proposed modification of Tsai's approach is the only "true" three-dimensional tracker evaluated, these measures are equally valid for the homography, nonlinear map, and Nintendo Wii pointing solutions. Jitter and drift are particularly useful for characterizing the *stability* of a given solution while reliability is self-explanatory.

Jitter refers to the change in output of a system when the tracked object remains stationary. For the approaches evaluated, jitter corresponds to change in the estimated cursor position, from one frame to the next, when the pointing device is stationary. This can be considered the noise component of static accuracy. Drift refers to the steady increase in tracker error over time. For the approaches considered here, this refers to a steadily increasing error in the observed static accuracy. Reliability is an additional measure that characterizes what percentage of frames a cursor estimate is generated compared to the number of frames captured by the camera. If a system is less than 100% reliable, it most likely indicates one of two problems has occurred: the centroid of the laser spot was not detected on the image plane or; the feature point correspondences were not correctly identified on the image plane.

As with static accuracy, jitter and drift are measured within each of the nine projection screen partitions. Both jitter and drift are measured using an experimental setup similar to that described for measuring static accuracy. An external camera operating at 30 fps with a resolution of 1920x1080 was used to record a video sequence comprising 50 frames. During the video sequence, a given solution was used to continuously generate an estimate for the cursor position. Each of the captured frames was then processed manually to determine the amount of cursor movement (in pixels) from one frame to the next. The maximum jitter computed over the 50 frames is provided, in pixels, for each of the four approaches in Table 9.4. The maximum jitter, as a percentage of the display diagonal, is given in Table 9.5.

In the following section label A is used to denote the jitter for the homography approach, label B for the modification of Tsai's algorithm with relaxations, label C for

the modification of Tsai's algorithm without relaxations, label D for the nonlinear map, and label E for the Nintendo Wii device. All reported values have a measurement error of ± 1 pixel and were recorded for the pointing device located at P_1.

Region	A (pixels)	B (pixels)	D (pixels)	E (pixels)
1	0±1	0±1	0±1	0±1
2	1±1	0±1	0±1	0±1
3	2±1	0±1	0±1	0±1
4	0±1	0±1	0±1	0±1
5	2±1	0±1	0±1	0±1
6	0±1	0±1	0±1	0±1
7	1±1	0±1	1±1	0±1
8	2±1	0±1	0±1	0±1
9	1±1	0±1	1±1	0±1

Table 9.4: Jitter measurements, in pixels, for the homography (A), modification of Tsai's extrinsic algorithm with relaxations (B), nonlinear map (D), and Nintendo Wii pointing device (E).

Region	A (% display)	B (% display)	D (% display)	E (% display)
1	0±0.045	0±0.045	0±0.045	0±0.045
2	0.045±0.045	0±0.045	0±0.045	0±0.045
3	0.090±0.045	0±0.045	0±0.045	0±0.045
4	0±0.045	0±0.045	0±0.045	0±0.045
5	0.090±0.045	0±0.045	0±0.045	0±0.045
6	0±0.045	0±0.045	0±0.045	0±0.045
7	0.045±0.045	0±0.045	0.045±0.045	0±0.045
8	0.090±0.045	0±0.045	0±0.045	0±0.045
9	0.045±0.045	0±0.045	0.045±0.045	0±0.045

Table 9.5: Jitter measurements, as a percentage of the display diagonal, for the homography (A), modification of Tsai's extrinsic algorithm with relaxations (B), nonlinear map (D), and Nintendo Wii pointing device (E).

The jitter observed for each of the approaches is clearly quite low. For the two indirect mapping solutions, the jitter was identically zero. For the nonlinear map, there was a

small amount of jitter recorded in just two of the nine regions. For the homography approach, a small amount of jitter was seen in six of the nine regions. The reason the two direct mapping solutions show any jitter at all can be accounted for in part by fluctuations in the intensity of the laser pointer. As the intensity varies, the projected centroid of the laser pointer also varies by a small amount. This variation, in turn, can lead to centroids which are slightly different from one frame to the next. As a result, the estimated cursor location can also vary - leading to a finite amount of jitter.

Additional sources of jitter for the two direct mapping solutions could arise from minor movements of the laser pointer. When measuring jitter, the laser pointer was physically taped to a camera tripod. It is entirely possible the laser pointer moved slightly during testing as the tape became less adhesive. One final source of jitter could arise from finite quantization of the established mappings. Since the camera resolution is lower than the resolution of the projected video sequence, it is indeed possible for one pixel on the image plane to be mapped to one or more pixels in the output video. Rounding errors when computing the centroid $\bar{m}$ of a laser spot on the image plane, combined with finite quantization of the coefficients of the homography map, could contribute to different cursor estimates $\hat{q}$ for the same $\bar{m}$ - thus resulting in jitter. This effect is partly mitigated by the nonlinear map because it attempts to establish sub-pixel mappings between the image and output video sequence. However, quantization and rounding errors can still lead to a finite amount of jitter as observed.

The drift, measured as the steady increase in tracker error over time, is reported in pixels for each of the four solutions in Table 9.6. The drift, measured as a percentage of the display diagonal, is reported in Table 9.7. As shown, the drift is identically zero for all solutions evaluated - indicating the static accuracy is a stable measure centered about some finite mean. If the static accuracy had decreased over time, the drift would have been finite and non-zero. However, this clearly was not the case. Taking into consideration the jitter, drift, and static accuracy, each of the four solutions can be considered highly stable.

The reliability of the approaches is assessed under three pointing conditions: static pointing, dynamic pointing under natural motion, and dynamic pointing under accelerated motion. Pointing under natural motion refers to guiding the cursor along a random trajectory with a speed of approximately 0.5 m/s. This was achieved by holding the

Region	A (pixels)	B (pixels)	D (pixels)	E (pixels)
1	0±1	0±1	0±1	0±1
2	0±1	0±1	0±1	0±1
3	0±1	0±1	0±1	0±1
4	0±1	0±1	0±1	0±1
5	0±1	0±1	0±1	0±1
6	0±1	0±1	0±1	0±1
7	0±1	0±1	0±1	0±1
8	0±1	0±1	0±1	0±1
9	0±1	0±1	0±1	0±1

Table 9.6: Drift measurements, in pixels, for the homography (A), modification of Tsai's extrinsic algorithm with relaxations (B), nonlinear map (D), and Nintendo Wii pointing device (E).

Region	A (% display)	B (% display)	D (% display)	E (% display)
1	0±0.045	0±0.045	0±0.045	0±0.045
2	0±0.045	0±0.045	0±0.045	0±0.045
3	0±0.045	0±0.045	0±0.045	0±0.045
4	0±0.045	0±0.045	0±0.045	0±0.045
5	0±0.045	0±0.045	0±0.045	0±0.045
6	0±0.045	0±0.045	0±0.045	0±0.045
7	0±0.045	0±0.045	0±0.045	0±0.045
8	0±0.045	0±0.045	0±0.045	0±0.045
9	0±0.045	0±0.045	0±0.045	0±0.045

Table 9.7: Drift measurements, as a percentage of the display diagonal, for the homography (A), modification of Tsai's extrinsic algorithm with relaxations (B), nonlinear map (D), and Nintendo Wii pointing device (E).

pointing device in hand, locking the elbow and wrist, and moving the cursor by rotating the shoulder only. This provides a simple yet effective way to physically limit how quickly the cursor can be moved. Pointing under accelerated motion was achieved by allowing the elbow and wrist to move freely, resulting in a more agile trajectory and a speed of approximately 2 m/s. Results were recorded for the homography, nonlinear map, Nin-

tendo Wii, the modification of Tsai's extrinsic algorithm with successive relaxations, and the modification of Tsai's extrinsic algorithm without relaxations.

For the homography, nonlinear map, and implementations of the modified Tsai approach, 500 images were captured by the camera and used to generate an estimate for the cursor position. The reliability was measured as the percentage of frames where an estimate was generated compared to the total number of frames captured. For the direct mapping solutions, an estimate is not generated if the number of centroids on the image plane is not one. For the modified Tsai algorithm, an estimate is not generated if the number of centroids found on the image plane is not identically nine.

For the Nintendo Wii, a high-speed camera operating at 500 fps was used to capture 500 images showing the output video. Each image was manually verified for the presence or lack of a cursor estimate. Results under static pointing are shown in Figure 9.8, under natural motion in Figure 9.9, and under accelerated motion in Figure 9.10. All results were compiled with the pointing device positioned at P_0 and aiming directly at region 5.

For the homography, nonlinear map, and Nintendo Wii, the reliability was found to be 100% across all trials and therefore independent of the pointing speed. The modified version of Tsai's extrinsic algorithm, however, demonstrates a noticeable decrease in reliability as the speed of motion increases. Applying successive relaxations clearly helps to mitigate this effect by providing improvements of 0.04% for static pointing, 9.40% for natural motion, and 9.80% for accelerated motion when compared to the implementation using a straighforward application of image thresholding. Intuitively, these results are as anticipated since the successive relaxation algorithm was designed specifically for this purpose.

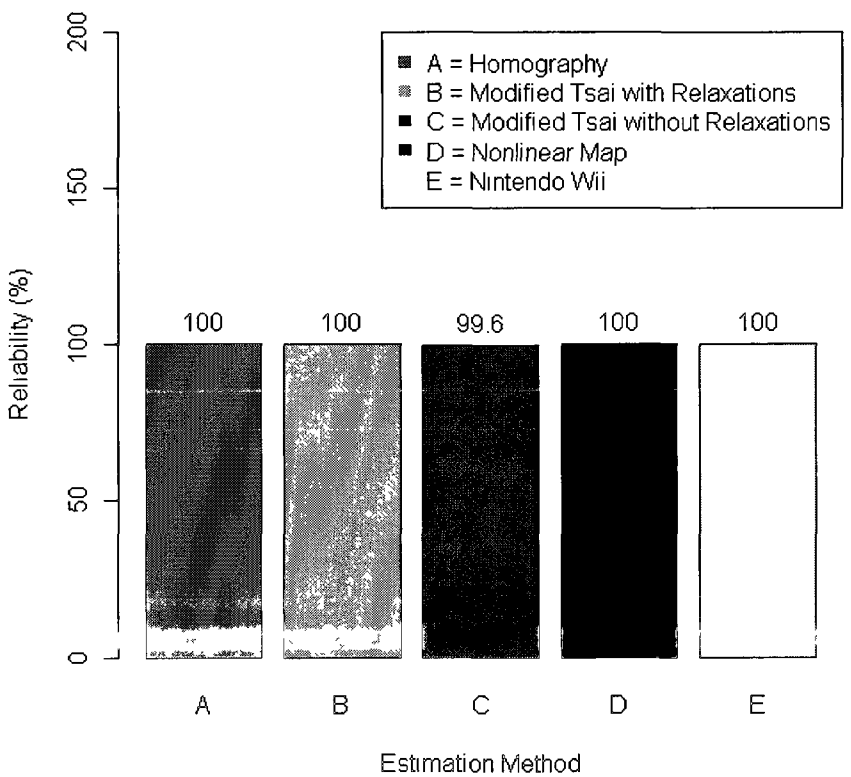

Figure 9.8: Reliability measures under static pointing.

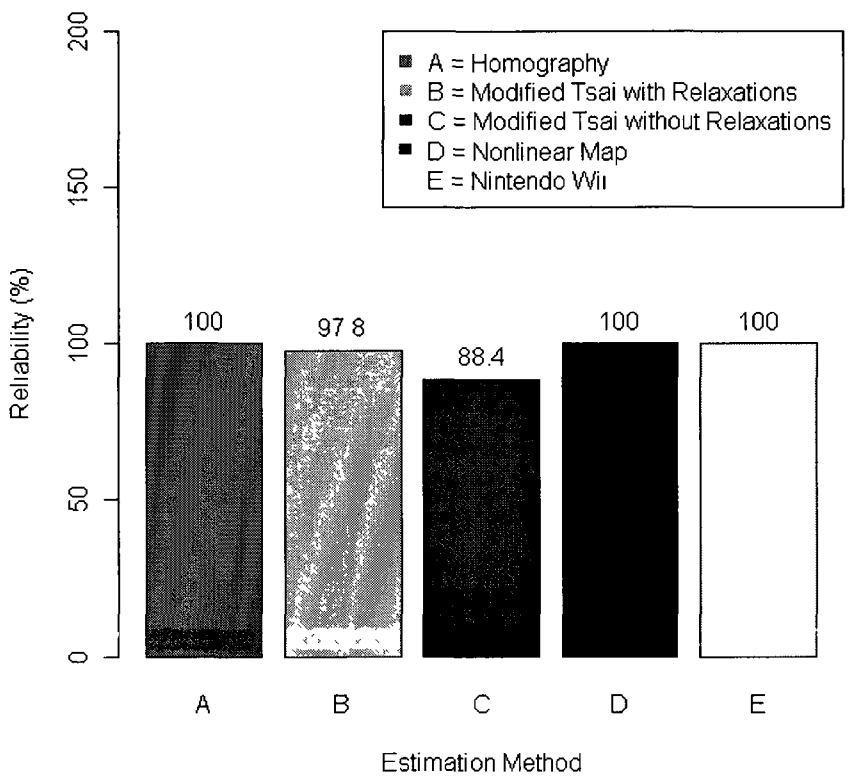

Figure 9.9: Reliability measures under natural motion.

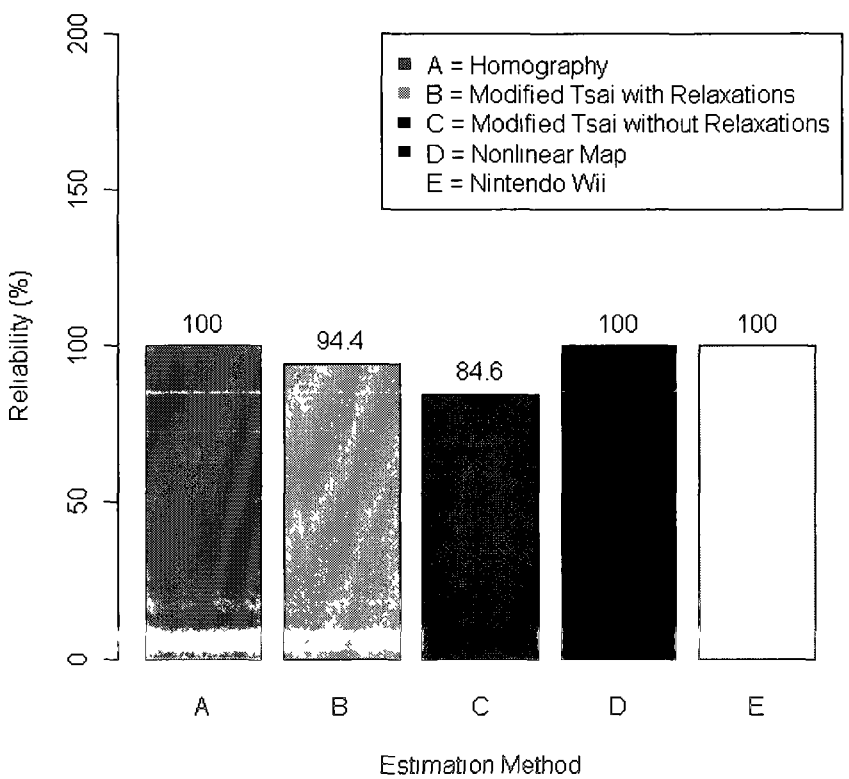

Figure 9.10: Reliability measures under accelerated motion.

Chapter 10

Conclusions

10.1 Summary of Contributions

This thesis has investigated the viability of alternative pointing devices for interacting with a combat training simulator. The challenge of identifying where a user is aiming has been presented in the context of a direct mapping problem and an indirect mapping problem. Two approaches have been investigated as potential solutions to the direct mapping problem: homography estimation, and nonlinear point-to-point mapping. Similarly, two approaches have been investigated as potential solutions to the indirect mapping problem: a modification to Tsai's extrinsic algorithm, and the Nintendo Wii pointing device. In the development and implementation of these potential solutions, a number of contributions have been made in the areas of image processing, cursor position estimation, multi-user distinction, state estimation, and performance evaluation.

Image Processing: An algorithm has been described for implementing what is termed successive relaxations. It has been shown that applying successive relaxations can provide a better representation of a laser stroke on the image plane when compared to applying image thresholding alone. The importance of this algorithm is that it serves to provide more accurate estimates for the centroid of image components which, in turn, leads to more accurate estimates of the true location where users are aiming. Through

simulation, the computational complexity of applying successive relaxations was shown to be linear in the size of image components provided the intensities of pixels are drawn from a normal distribution.

Two novel algorithms have been described for labeling components in binary images. These algorithms both require just a single pass over the image and make liberal usage of C++ vectors and hash maps. Through simulation, it was found that one of the algorithms exhibits nonlinear computational complexity while the other exhibits a linear characteristic. It was shown that the nonlinear-time algorithm outperforms the linear-time algorithm when image components are sufficiently small and have intensities drawn from a normal distribution.

A novel approach to identifying world and image feature point correspondences has been described. By carefully arranging feature points in the world coordinate system, it was shown how it is possible to construct geometries which are invariant to changes in scale, translation, and rotation. Furthermore, it was shown that one of the feature points can serve as an easily identifiable anchor point. With the anchor point known, an algorithm was described for arranging the feature points in a clockwise ordering relative to the anchor point using what is termed a right-turn predicate. By knowing the anchor point in the world and image coordinate systems, and by ordering the feature points clockwise relative to this point, it was shown how feature point correspondences can be readily obtained.

Cursor Position Estimation: By establishing the correspondence between four known feature points, it has been shown how a homography between the image plane and the projection screen can be obtained. It was described how the homography can be used to determine a correspondence between pixels on the image plane and pixels in the simulation video sequence. Using the homography, an approach was described for converting the centroid of an image component (formed from a laser pointer) into a cursor position with respect to the video sequence. This cursor position then served as the location where a user was aiming.

In the context of an indirect mapping solution, a function was defined for intersecting a ray between the focal point of a camera and the projection screen. It was then shown how this point of intersection could be converted into a cursor location, with re-

spect to the output video sequence, and used as the location where a user was aiming. An implementation of the coplanar POSIT algorithm was also described by simulating feature points about the perimeter of the projection screen and identifying their correspondences on the image plane. A series of images were simulated to replicate how the feature points would appear on the image plane with the camera fixed at various orientations. By knowing the actual camera orientation, it was possible to observe the error between the true and recovered orientations. Simulation results demonstrated the coplanar POSIT algorithm was susceptible to a novel measure termed the translation ratio. It was shown how estimating the translation component of pose was robust to this ratio but that errors in estimating the rotation of the camera increased significantly as the translation ratio increased.

An implementation of Tsai's extrinsic algorithm was described by, again, simulating feature points about the perimeter of the projection screen and identifying their correspondences on the image plane. Using an approach similar to the evaluation of the coplanar POSIT algorithm, a series of images were simulated to replicate how the projection of feature points on the image plane at various camera orientations. Simulation results comparing the true and recovered camera orientations showed the algorithm to be 100% accurate. Owing to these results, experiments were conducted on real image data. By carefully measuring the translation component of pose and comparing it to the estimated translation vector, it was found that errors did indeed arise when recovering the pose. As with the coplanar POSIT algorithm, it was shown that increasing the translation ratio led to an increase in the amount of error when estimating the translation.

One observation noted during testing on real image data was that Tsai's extrinsic algorithm appeared to accurately recover the rotation component of pose for various camera orientations. Based on this observation, a modification to Tsai's extrinsic algorithm was proposed under the assumption the rotation parameters were accurately estimated. The approach described consists of repositioning the feature points on the image plane to effectively "undo" the rotation incurred. Once the points are repositioned, the algorithm re-estimates the translation component of pose. It was shown that this approach provides significantly improved results when compared to the original implementation of Tsai's extrinsic algorithm. Furthermore, results on real image data showed the modified algorithm to be robust to the translation ratio.

Multi-User Distinction: For the case of direct mapping solutions, two approaches to distinguishing between concurrent users have been evaluated. Each approach assumes users hold a laser pointer casting a unique geometry and that determining which geometries are present on a display is sufficient to determine which users are interacting at a given time. One of the approaches is to use off-the-shelf laser pointer designs which force the light from a laser through a mask to generate the shapes. An evaluation of various off-the-shelf designs has shown their effective range to be greatly limited. It was also shown that these geometries can undergo significant nonlinear transformations depending on where a user is located with respect to the projection screen - thus limiting the effective range of use.

Owing to the idea that using unique shapes to identify users could provide a viable solution, an approach using custom-designed laser patterns was investigated. It was argued that using an array of laser pointers to construct the geometries would significantly mitigate the nonlinear transformation effects observed with off-the-shelf laser pointers. A novel approach was presented where each user holds an array of lasers casting a circular pattern but with a unique radius for each user. It was shown how the Hausdorff distance metric could be used to determine which circles of what radius were present at a given time on the projection screen. Simulation results demonstrated this to be an effective approach but that using the circular Hough transform could provide comparable results with significantly less computational effort. Simulation results indicated that if a sufficient number of lasers were used to approximate the circles, and if the radial separation between two consecutive circles was sufficiently large, then the proposed approach could indeed distinguish between multiple users with a high degree of accuracy.

For the case of indirect mapping solutions, a simple yet effective approach to distinguishing among multiple users was described. The premise was to assign each camera a unique identifier and to process the images from each one separately. By knowing which user was holding which camera, it was trivial to determine where a given user was aiming and hence trivial to distinguish between the users.

State Estimation: The dynamics of a laser pointer, or similarly a hand-held camera, were modeled by expanding the position estimate using a Taylor series expansion. Using this approach, it was possible to derive two separate state-space representations: one assuming the acceleration component of movement was constant, and the second as-

suming the velocity component of movement was constant. This led to the derivation of a three-state and two-state representation, respectively. By making assumptions about the expected measurement error and by making assumptions about the natural tremor in a users hand, it was possible to derive estimates for the process and measurement error covariance matrices. Using the dynamic model and covariance matrices, a complete implementation of the two and three-state Kalman estimators was described.

Performance Evaluation: A comprehensive performance evaluation of four alternative pointing devices was provided based on both simulated and real image data. The homography estimation and nonlinear mapping approaches were evaluated as potential solutions to the direct mapping problem, while the modification to Tsai's extrinsic algorithm and the Nintendo Wii pointing device were evaluated as potential solutions to the indirect mapping problem. The proposed evaluation criteria used to compare the competing solutions consisted of measures derived from static accuracy, dynamic accuracy, latency, jitter, drift, and repeatability.

The two direct mapping solutions were first compared with respect to their static accuracy characteristics. It was shown that both approaches were highly accurate across the entire projection screen surface with the nonlinear mapping approach being moderately more accurate. Subject to measurement errors, the nonlinear mapping approach was shown to provide a static accuracy of, at worst, 0.317% of the distance along the display diagonal. The two indirect solutions were then compared with respect to the change in static accuracy as a function of user position. It was shown that both approaches become less accurate as the angle of incidence between the hand-held camera and projection screen increases. Through experiments on real image data, it was shown the two approaches had comparable accuracy within a range of incident angles; however, the Nintendo Wii pointing device provided greater accuracy over the full range of incident angles considered.

Simulation results were then provided to illustrate the effects of Kalman state estimation on dynamic pointing accuracy. Assuming that each solution had its respective baseline static accuracy performance characteristics, it was argued that evaluating the effects of Kalman state estimation on the nonlinear mapping solution was sufficient to illustrate the performance gains that could be obtained for any direct or indirect solution. Results on real image data showed that using a three-state estimator could provide

a relative improvement in dynamic accuracy of 23-46%, while the two-state estimator could provide improvements of 15-36%.

Each of the solutions were then compared with respect to measures of function and system latency. The function latency was used to measure the computation time required to generate an estimate of the cursor position for four different approaches: homography estimation, nonlinear mapping, the modification to Tsai's extrinsic algorithm with successive relaxations, and the modification to Tsai's algorithm without the application of successive relaxations. With the exception of the latter, each of the approaches was found to have a mean latency of approximately 33 ms which guaranteed that an estimate for the cursor could be generated for each image captured by the camera. For the implementation of the modification to Tsai's algorithm with successive relaxations the mean function latency was found to be about 75 ms - beyond the frame rate of the camera.

The latency of the four approaches and the Nintendo Wii pointing device were then compared using a measure termed system latency. The system latency was recorded with the aid of a high-speed external camera and measured the elapsed time between the movement of an input device and the corresponding movement of the mouse cursor. Results confirmed the performance measures of the four approaches found using the measure of function latency and demonstrated the Nintendo Wii had an expected system latency on the order of 16ms.

The jitter and drift of the homography, nonlinear map, modification to Tsai's algorithm with relaxations, and the Nintendo Wii were then measured under the condition of static pointing. None of the approaches were observed to have measurable drift. The modification to Tsai's algorithm and the Nintendo Wii were also found to exhibit zero jitter. For the homography, a maximum of two pixels of jitter was observed over a series of 50 frames while a maximum jitter of one pixel was observed for the nonlinear map. The reliability of these approaches, as well as the modification to Tsai's algorithm without successive relaxations, was then measured under three pointing conditions: static, natural motion, and accelerated motion. For the homography, nonlinear map, and Nintendo Wii approaches, the reliability was measured to be 100% under all conditions. The reliability observed for the modification to Tsai's algorithm (with and without successive relaxations), however, was shown to descrease as the speed of motion increased. Under accelerated motion, the modification to Tsai's algorithm with relaxations was shown to

provide an increase in reliability of nearly 10%.

10.2 Recommendations and Future Work

The primary objectives of this thesis have been to investigate and to develop alternative pointing solutions which demonstrate good performance characteristics while at the same time supporting multiple users interacting concurrently. Based on the performance measures considered here, the nonlinear map has been shown to provide the best direct mapping solution. When compared to the homography, it provides greater accuracy and provides nearly identical measures of latency, jitter, drift, and reliability. When compared to each of the solutions evaluated, it in fact provides the greatest amount of pointing accuracy. With respect to supporting multiple users, the most practical solution would be to use a different coloured laser for each individual user. Provided a sufficient number of laser pointers with different colours can be obtained, and provided optical filters are available at the corresponding wavelengths, this provides an effective solution meeting the primary objectives.

Of the two indirect solutions considered, the Nintendo Wii has been shown to provide better resilience to changes in the angle of incidence when compared to the modification of Tsai's extrinsic algorithm. However, it was shown these two approaches provide comparable measures of static accuracy within a range of incident angles. Within this range, the modification of Tsai's algorithm has the added advantage of recovering the camera pose which can be used to provide valuable feedback to users of the training simulator. The Nintendo Wii, by contrast, cannot provide such information to the user. Both solutions are capable of supporting a comparably large number of concurrent users. If it is more important to provide users with feedback related as to how they hold their firearm, then the modification to Tsai's algorithm provides a suitable solution provided users remain within $\pm 30^\circ$ relative to the center point of the projection screen. If, however, a greater range of motion throughout the training environment is a more important factor, then a solution using the Nintendo Wii pointing device would provide the appropriate solution.

Going forward, it would be important to improve the computational efficiency of the

successive relaxation algorithm as well as the two component labeling algorithms. If a choice is made to proceed with an implementation of the modification to Tsai's extrinsic algorithm, it is especially important to focus on optimizing the successive relaxation algorithm. If more than a single user is to interact with the simulator at a given time, it may be possible to parallelize the implementation such that a new thread is created for each camera connected to the bus. Then, with each image captured from different cameras, the corresponding thread could be dedicated to process the image and independently estimate the camera pose.

An additional avenue of future work would be to develop novel feature point arrangements which would enable correspondences to be obtained when only a subset of the feature points are within the camera's field of view. Such a condition could arise when the camera is near and facing the projection screen but only captures a subset of the feature points. Under such a condition it would be desirable to still be able to identify where the camera is aiming to generate an estimate for the cursor position.

Perhaps the most critical area of future research would be into the implementation of non-planar camera pose estimation techniques. Techniques such as the POSIT algorithm and Tsai's extrinsic calibration algorithm for non-planar feature points would be excellent initial candidates to evaluate. It would be critical to evaluate these techniques to determine how accurate they are as well as how robust they are to changes in the angle of incidence formed with the projection screen. Open-source implementations of these algorithms are readily available and would therefore only require the construction of a suitable arrangement of feature points.

One final area of research worth pursuing would be to investigate methods for extending the direct mapping solutions to provide feedback of the orientation of a laser pointer. If a device could be constructed comprising, for example, a laser pointer fitted with gyroscopes and several accelerometers, it could be possible to benefit from the high degree of accuracy afforded by a direct mapping solution while at the same time providing users with feedback on the orientation of their firearm - as provided by indirect mapping solutions. Such a solution would have excellent performance with respect to the measures previously described and, if a colour-based user distinction schema was used, could enable multiple users to be supported concurrently.

10.3 Final Remarks

This thesis finds, in general, that no single solution is ideal for all applications. Depending upon the performance constraints and design specifications, it is plausible for either a direct mapping solution or an indirect mapping solution to provide the most suitable implementation. When choosing among competing solutions it is important to carefully consider the advantages and shortcomings of each. In general, this thesis finds direct mapping solutions to have the greatest range of motion for a given measure of accuracy. Indirect mapping solutions are found to support the largest number of users but have a bounded range of motion. The Nintendo Wii is shown to be more robust to changes in position but, unlike the variant of Tsai's extrinsic algorithm, cannot recover the pose of the pointing device.

Bibliography

[1] B. Shizuki, T. Hisamatsu, S. Takahashi, and J. Tanaka, "Laser Pointer Interaction Techniques using Peripheral Areas of Screens", In *Proceedings of the Working Conference on Advanced Visual Interfaces*, Venezia, Italy, May 23-26, 2006, pp. 95-98

[2] J. Davis and X. Chen, "LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays", In *Displays*, vol. 23, no. 5, November, 2002, pp. 205-211

[3] D. Cavens, F. Vogt, S. Fels, and M. Meitner, "Interacting with the Big Screen: Pointers to Ponder", In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, Minneapolis, Minnesota, United States, April 20-25, 2002, pp. 678-679

[4] T. Winograd and F. Guimbretiere, "Visual Instruments for an Interactive Mural", In *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, May 15-20, 1999, pp. 234-235

[5] B. Ahlborn, D. Thompson, O. Kreylos, B. Hamann, and O. Staadt, "A Practical System for Laser Pointer Interaction on Large Displays", In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, Monterey, California, United States, November 07-09, 2005, pp. 106-109

[6] D. Olsen and T. Nielsen, "Laser Pointer Interaction", In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Seattle, Washington, United States, March 31 - April 05, 2001, pp. 17-22

[7] C. H. Peck, "Useful parameters for the design of laser pointer interaction techniques", In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, Seattle, Washington, United States, March 31 - April 05, 2001, pp. 461-462

[8] B. Myers, R. Bhatnagar, J. Nichols, C. H. Peck, D. Kong, R. Miller, and A. Long, "Interacting at a Distance: Measuring the Performance of Laser Pointers and Other Devices", In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, United States, April 20-25, 2002, pp. 33-40

[9] J.-F. Lapointe and G. Godin, "On-Screen Laser Spot Detection for Large Display Interaction", In *Proceedings of the IEEE International Workshop on Haptic Audio Environments and their Applications*, Ottawa, Ontario, Canada, October 01-02, 2005, pp. 72-76

[10] F. Vogt, J. Wong, B. Po, R. Argue, S. Fels, and K. Booth, "Exploring Collaboration with Group Pointer Interaction", In *Proceedings of the Computer Graphics international*, Crete, Greece, June 16-19, 2004, pp. 636-639

[11] C. Kirstein and H. Muller, "Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer", In *Proceedings of the 1998 Conference on Multimedia Modeling*, Lausanne, Switzerland, October 12-15, 1998, pp. 191-192

[12] K. Cheng and K. Pulo, "Direct Interaction with Large-Scale Display Systems using Infrared Laser Tracking Devices", In *Proceedings of the Asia-Pacific Symposium on information Visualisation*, Adelaide, Australia, February 03-04, 2003, pp. 67-74

[13] S. Matveyev, M. Gobel, and P. Frolov, "Laser Pointer Interaction with Hand Tremor Elimination", In *Proceedings of HCI International 2003*, Crete, Greece, June 22-27, 2003

[14] B. Myers, C. H. Peck, J. Nichols, D. Kong, and R. Miller, "Interacting At a Distance Using Semantic Snarfing", In *Proceedings of the 3rd international Conference on Ubiquitous Computing*, Atlanta, Georgia, United States, September 30 - October 02, 2001, pp. 305-314

[15] F. Vogt, J. Wong, S. Fels, and D. Cavens, "Tracking Multiple Laser Pointers for Large Screen Interaction", In *UIST '03 Extended Abstracts*, Vancouver, British Columbia, Canada, November 02-05, 2003, pp. 95-96

[16] J.-Y. Oh and W. Stuerzlinger, "Laser Pointers as Collaborative Pointing Devices", In *Proceedings of Graphics Interface*, Calgary, Alberta, Canada, May 27-29, 2002, pp. 141-150

[17] A. Pavlovych and W. Stuerzlinger, "Laser Pointers as Interaction Devices for Collaborative Pervasive Computing", In *Advances in Pervasive Computing*, vol. 176, April, 2004, pp. 315-320

[18] D. Laberge, J.-F. Lapointe, E. Petriu, "An Auto-Calibrated Laser-Pointing Interface for Large Screen Displays", In *Proceedings of the Seventh IEEE international Symposium on Distributed Simulation and Real-Time Applications*, Delft, The Netherlands, October 23-25, 2003, pp. 190-195

[19] K. Fukuchi, "A Laser Pointer/Laser Trails Tracking System for Visual Performance", In *Proceedings of the International Conference on Human-Computer Interaction*, Rome, Italy, September 12-16, 2005, pp. 1050-1053

[20] J. Lee, "Hacking the Nintendo Wii Remote", In *IEEE Pervasive Computing*, vol. 7, no. 3, July, 2008, pp. 39-45

[21] D. Oberkampf, D. DeMenthon, and L. Davis, "Iterative Pose Estimation Using Coplanar Feature Points", In *Computer Vision and Understanding*, vol. 63, no. 3, May, 1996, pp. 495-511

[22] D. Ballard and C. Brown, Computer Vision, Prentice Hall, Englewood Cliffs, New Jersey, United States, 1982

[23] R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", In *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, August, 1987, pp. 323-344

[24] L. Quan and Z. Lan, "Linear N-Point Camera Pose Determination", In *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, August, 1999, pp. 774-780

[25] M. Fischler and R. Bolles, "Random Sample Concensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", In *Communications of the ACM*, vol. 24, no. 6, June 1981, pp. 381-395

[26] D. Lowe, "Fitting Parameterized Three-Dimensional Models to Images", In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, May, 1991, pp. 441-450

[27] E. Trucco and A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall, Upper Saddle River, New Jersey, United States, 1998

[28] R. Wilson, *Tsai Camera Calibration Software* [Online], October 28, 1995, Available: http://www.cs.cmu.edu/~rgw/TsaiCode.html

[29] C. Harris and M. Stephens, "A Combined Corner and Edge Detector", In *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147-151

[30] M. Kohler, "Using the Kalman Filter to Track Human Interactive Motion - Modelling and Initialization of the Kalman Filter for Translational Motion", Technical Report, Informatik VII, University of Dortmund, 1997

[31] D. Salmond, "Target Tracking: Introduction and Kalman Tracking Filters", In *IEE Target Tracking: Algorithms and Applications*, vol. 2, October, 2001, pp. 1-16

[32] D. DeMenthon and L. Davis, "Model-based Object Pose in 25 Lines of Code", In *International Journal of Computer Vision*, vol. 15, May, 1995, pp. 123-141

[33] G. Burdea and P. Coiffet, Virtual Reality Technology, 2nd ed., John Wiley & Sons, Hoboken, New Jersey, United States, 2003