MARVIN: a Mobile Automatic Realtime Visual and INertial tracking system

Andrew Hogue

Technical Report CSE-2003-13

May 15, 2003

Department of Computer Science and Engineering

4700 Keele Street Toronto, Ontario M3J 1P3 Canada

MARVIN: a Mobile Automatic Realtime Visual and INertial tracking system

Andrew Hogue

A thesis submitted to the Faculty of Graduate Studies
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Graduate Program in Computer Science
York University
Toronto, Ontario, Canada

May, 2003

Abstract

Six-sided fully-enclosed projective displays present complex and novel problems for tracking systems. The fully-enclosed nature of these displays limits the use of existing tracking technologies which typically require a tether or a line-of-sight to the user which is unavailable in this context. This thesis presents a hybrid Inertial/Optical tracking system for fully-enclosed projective displays. The inertial system uses linear acceleration measurements to estimate relative head motion but the estimate is subject to drift due to sensor misalignment and calibration errors. To compensate for this drift, a vision-based tracking technology is used to estimate the absolute pose of the operator's head. The optical tracking technology relies on the operator wearing a set of laser diodes arranged in a specific configuration and then tracking the projection of these lasers on the external walls of the immersive display. This approach places minimal hardware on the user and no visible tracking equipment is placed within the immersive environment. The inertial and vision-based pose estimates may be combined via a recursive least-squares filter to provide the pose of the operator.

ACKNOWLEDGEMENTS

Throughout the development of this thesis, many people have helped me through the time I was stuck or needed some insight. Many of them offered their technical support, advice, or let me bounce ideas off of them. Without them, this thesis could never have been finished.

I wish to thank my primary supervisor Dr. Michael Jenkin. Without his tremendous help and guidance this thesis would never have been completed. The respect I have for him as a researcher, supervisor, and person cannot be expressed deeply enough in words. I also wish to thank Dr. Robert Allison, my co-supervisor, for all of his help and support. He has always been available and has made time to answer all of my questions, no matter how small or large.

A huge thanks goes to Matt Robinson, co-worker and friend, who has helped me tremendously throughout the span of this thesis. He has always been available and willing to give advice and help out technically (even when he was busy with other things he would still make time). Our discussions on many different aspects of this thesis were extremely helpful, and his amazing problem-solving skills were necessary to point me in the right direction many times.

I thank my good friend Gilles Pouliquen for helping me strengthen my mathematical background. His engineering background was a large wealth of knowledge that helped me when I was stuck in derivations. Thanks to Sergey Parilov who was also very helpful during our lengthy late-night discussions that put me on the right track at times.

Others that I would like to thank are Wolfgang Stuerzlinger, Kosta Derpanis, Jeff Laurence, James Zacher, Andrew German, and of course my parents Rene and Carolee Hogue.

Last, but certainly not least, I thank Urszula, my love, for always being there for me. She has given me more emotional support and help than I could ever imagine, without her this thesis would never have been possible.

Contents

Chapter 1

Introduction

Immersive displays have become a popular technology for scientific visualization, psychological research, tele-operation, task training/rehearsal, and entertainment. Advances in projection technology have facilitated the development of immersive displays ranging from large single wall projections (e.g. the PowerWall[54]), three-wall displays (e.g. the Immersion Square[34]), four-wall displays (e.g. the CAVE^{TM}[16]), five-wall displays (e.g. the CABIN[55]), and more recently six-sided displays (e.g. the Immersive Visual environment at York – IVY[65]). The technology is beginning to move out of the lab into the commercial arena, and vendors such as Fakespace^{TM} Systems and TAN Projektionstechnologie GmbH have begun to design and build immersive projective displays.

Currently, most existing projective immersive displays are not fully-enclosed and are designed with relatively small numbers of walls. Although there are various reasons why the non-fully-enclosed immersive environments have been constructed, the lack of "full enclosure" simplifies a number of design and construction details. As the number of walls increases, many of the problems that can be solved "easily" in immersive projective displays with small numbers of walls become much more complex. Entry/egress, projector placement and perhaps most importantly, head tracking, become very complex issues. The limiting case of a fully-enclosed (six-sided) environment is certainly the most challenging

(see [65] for an examination of some of the details associated with the construction of a fully-enclosed immersive environment).

To date, at least seven fully-enclosed immersive environments have been developed:

1. **COSMOS**[81, 25]. COSMOS (Cosmic Multimedia of Six Screens) was perhaps the first 6-sided projective VR environment. Built in 1998 at the VR Techno Centre in Gifu, Japan, COSMOS (see Figure 1.1(a)) was constructed in an extremely large space that provided considerable simplifications in terms of construction. The throw distance to the walls was sufficiently large that the wall surfaces could be projected directly. Given the tall height of the physical enclosure, ceiling and floor were projected via a single reflected mirror. Each projection surface consisted of a $9m^2$ vinyl film, with the floor supported by a sandwich of three acrylic panels. Each projection surface was projected by two projectors in order to enhance the brightness of the display. Video was generated at 1024x768 at 96Hz, and stereo viewing was available through CrystalEyes LCD shutter glasses. Head tracking was performed via a Polhemus head tracker. Given the sensitivity of this tracker to the presence of metal, much of the construction was of wood.

2. **VR-CUBE**[21]. At almost the same time as COSMOS was being built in Japan, the VR-CUBE (see Figure 1.1(c)) was constructed at the Centre for Parallel Computers at the Royal Institute of Technology in Stockholm, Sweden. The VR-CUBE was built by TAN Projektionstechnolgie and is 3m(w) x 3m(d) x 2.25m(h). Fabric projection

surfaces were used, with a 40mm acrylic glass surface used to provide structural support on the floor. The fabric projection surface was stretched above this glass surface. As with COSMOS, the PDC VR-CUBE was also built in an extremely large physical space, and this provides a number of simplifications in terms of construction and video projection. A large wooden structure provides physical support for the screens and mirrors. Most surfaces are projected via mirrors in order to reduce the total physical volume of the device. The wooden structure permits the use of standard magnetic tracking technology to track the user within the display. The floor and ceiling were configured to run with a resolution of 1024x1024 at 96Hz. The walls are projected at a resolution of 1024x856 pixels to keep the pixels square. An SGI Onyx2 was used to generate content and Barco projectors provided the video. The door to the VR-CUBE used a hinge on one side, unlike the sliding door in COSMOS.

3. **ALICE**[71]. ALICE (see Figure 1.1(d)) is a VR cube constructed at the Beckman's Integrated Systems Laboratory, University of Illinois at Urbana Champaign. As with the VR-CUBE, ALICE was built by TAN Projektionstechnolgie. Mirrors were used to display the video onto the solid screen material to reduce the overall physical size. The construction of ALICE used non-metal materials permitting the use of standard magnetic trackers within the virtual environment.

4. **HyPi-6**[38, 62]. The HyPi-6, installed at the Fraunhofer Institute for Industrial Engineering IAO, was the first six walled projective environment to work with standard

PC's. Completed in May 2001 (see Figure 1.1(e)), HyPi-6 operates in one of two modes. Using 12 Barco 909 projectors, and driven by 12 PC's, a passive stereo system using polarization filters is used. An active stereo system driven by an SGI Onyx2 is also available. The environment is 2.9m(w) x 2.9m(d) x 2.7m(h).

5. **C6**[39]. The C6 (see Figure 1.1(f)) at Iowa State University's Virtual Reality Applications Center (VRAC) became operational in 2000. The C6 relies on Barco 909 projectors at 1024x1024 resolution at 96Hz. Ascension Technology's MotionStar® Wireless system is used for head and body tracking.

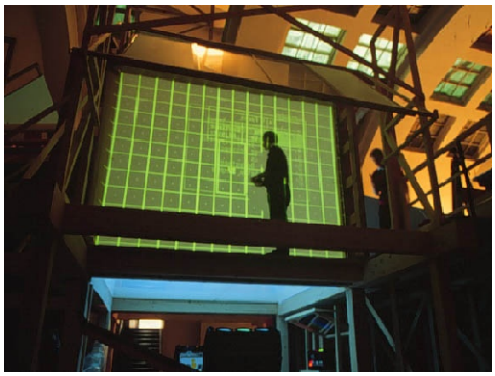
6. **VR-CAVE**[70]. The VR-CAVE (see Figure 1.1(b)) at the VR-CENTER NORD at Aalborg University is a six-sided cube measuring 2.5m x 2.5m x 2.5m. Electromagnetic tracking is used to maintain the user's viewpoint, and video is generated using an SGI Onyx2. The VR-CAVE was built to study the interplay between a user and their 3D environment.

7. **IVY**[65]. The Immersive Visual environment at York University (IVY, Figure 1.2) in Toronto, Ontario, Canada, became operational in September 2002. IVY was built as a tool to aid in the investigation of human perception. A detailed description of IVY can be found in [65].

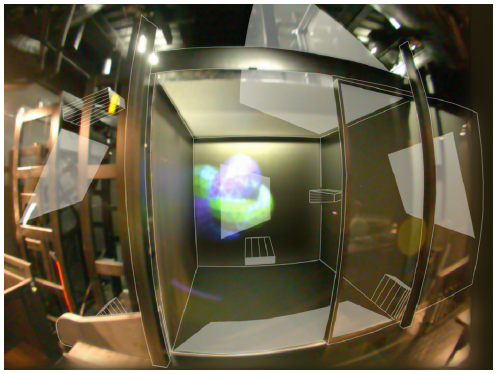
Although these fully-enclosed environments were built for different reasons, and under different environmental conditions, the goal of each is to build a compelling immersive visual environment.

(a) COSMOS

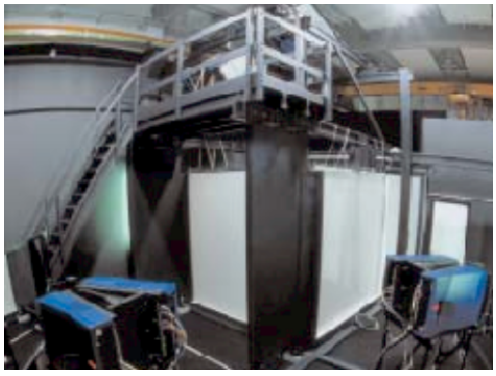
(b) VR-CAVE



(c) PDC-VRCUBE



(d) ALICE



(e) HyPi-6

(f) C6

Figure 1.1: Fully-Enclosed Immersive Projective Displays.

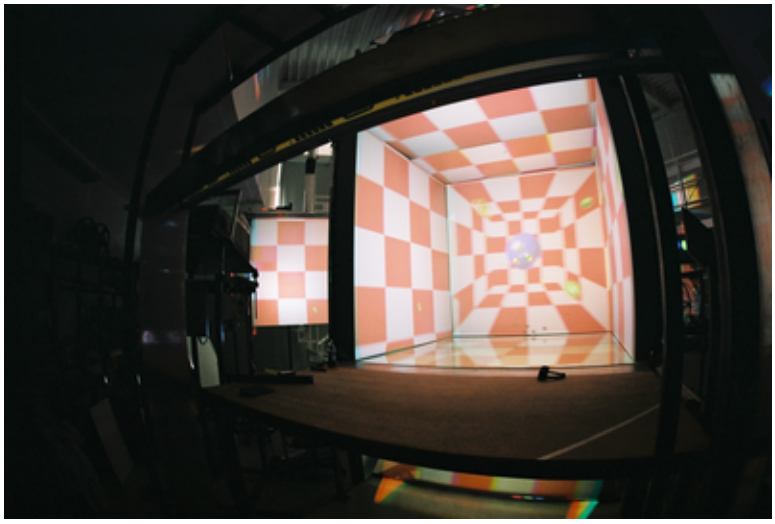


Figure 1.2: IVY at York University

1.1 Why build an immersive environment at York?

When we move within an environment, we are presented with a range of multi-modal cues to our motion within it. Our visual system provides optic flow, perspective, occulsion and other cues. Our vestibular system provides cues to the actual motion of our head, and the direction of gravity. Knowledge of our intent to move, feedback from our muscles, and many other direct and indirect cues are also present. Presenting self-motion cues in isolation or in unusual combinations has been used to assess their contribution to our overall sensation of self-motion (see [28, 63]). Presenting conflicting visual and vestibular cues to a subject is a common strategy for exploring the roles of these sensory systems. At York University, specifially at the Center for Vision Research (CVR), many different devices have been built to generate specific perceptual cue conflicts. The York "Tumbling Room" (see [3] and

Figure 1.3(a)) has been shown to be extremely effective at modifying an observer's perception of the direction of gravity. The Tumbling Room is a large physical room capable of rotating about a horizontal axis. Since the visual display is isolated from the outside world (and rotates with the user), the user experiences no visual changes while the room rotates. The resulting visual/vestibular cue conflict provides insight into the mechanisms behind the perception of our body orientation with respect to gravity (see [2, 41, 36, 37]). A static version of the Tumbling Room (known as the "Tilted Room" due to its permanent 90 degree tilted axis) has also been developed at York (see Figure 1.3(b)). Various experiments conducted within these rooms have demonstrated that subjects' perception of the direction of "up" can be manipulated within these rooms by presenting subjects with appropriately polarized visual displays. The Tumbling and Tilted Rooms have proven to be useful tools for studying human perception of self-orientation but are not without their problems. Since the Tumbling Room rotates to generate different vestibular sensations, powerful motors are employed which are loud and can even be dangerous. Also, since the Tumbling and Tilted rooms are physical rooms, it is very time consuming to change the visual displays; real objects and wall panels must be removed and replaced. These difficulties have inspired researchers to find more malleable large-scale tools that can be used to study human perception of self-motion/orientation. VR Technology has been found to be an effective alternative for generating perceptual cue conflicts and presenting combinations of proprioceptive, visual, and vestibular cues to the subject[28, 63, 1].

In order to be able to explore fundamental questions related to human perception in

(a) The Tumbling Room (b) The Tilted Room

Figure 1.3: Physical Immersive Displays. Here the outside of the Tumbling room (a) is shown. The motors are used to rotate the room along a horizontal axis. Also, the inside of the Tilted Room (b) is shown with subject lying on his side.

real and virtual environments, a fully-enclosed six-sided projective display named IVY (the Immersive Visual environment at York, see [65]), has been constructed. IVY is a six-sided, eight-foot fully-enclosed cube in which every surface including the floor and ceiling displays a rear-projected stereoscopic visual image to the observer inside of the cube. The construction of IVY posed interesting challenges including screen/projector calibration, usable input devices for interaction and operator tracking in a fully-enclosed volume. IVY itself is situated within the two-story Vision, Graphics and Robotics Laboratory (VGR Lab) at York University. Total effective ceiling height is approximately 16'. Given the limited horizontal physical space for IVY, three of the four walls are projected using mirrors to bend the light path within the available footprint in the VGR Lab. IVY's floor is located

four feet above the ground. This means that the mirrors, projectors, and wall surfaces are positioned well above the ground. The wall projectors are mounted approximately 8' above the surface of the floor of the lab (in line with the centre of the wall surfaces) with mirrors mounted in a similar fashion. In order to simplify construction, the wall projectors are mounted on separate "projector mounts" that locate the projectors 8' in the air. Each projector is mounted on an adjustable table that permits fine adjustment of the projector orientation.

Fabric rear-projection screens are hung in aluminum frames attached to IVY's floor outside of the projective environment. The same material is used for the floor and ceiling, although different techniques were required in order to deal with the need for physical support (floor) and for the limited spacing above the ceiling and below the floor.

Entry and exit to IVY is via one of the walls that can be slid back away from the interior of the cube. With this wall slid back, people and equipment can enter and exit IVY. With this wall in place, a user within IVY cannot distinguish between the opening/closing wall and the other three fixed walls. Stereo imagery is presented on IVY's six walls and decoded using CrystalEyes glasses. Four long range emitters have been found to be sufficiently powerful to provide full infrared coverage anywhere within IVY. Applications for IVY can be written in any graphical software package that runs on the SGI Onyx2 used to drive the displays. In order to simplify software development, a package has been built that abstracts the various display and input technologies required by applications to make full use of the tracking and display systems. This software package, known as VE, is described in [65].

1.2 Head-Tracking in Fully-Enclosed Displays

In a VR environment, if the view of the scene does not correspond to a given head position and orientation (pose), the generated image is incorrect for the user's viewpoint: stereo, perspective, vection, and motion parallax cues are incorrect, and the subject is more likely to experience discomfort (headaches, nausea, disorientation, collectively known as cybersickness[74]). In non-fully-enclosed environments, it is straightforward to use commercial head-tracking systems to obtain this pose information since the tracking equipment can be positioned in such a way that it does not interfere with the user's view of the scene (i.e. behind the user). However, six-sided displays impose a unique constraint: in a fully-enclosed volume there is no reasonable place for the tracking equipment except outside the working volume.

Since the user is fully-enclosed, state-of-the-art optical trackers and acoustical trackers which require a line-of-sight to the user are inappropriate unless the tracker is positioned within the working volume where it can be seen by the user thereby inhibiting the user's sense of presence. Currently, magnetic tracking is the technology of choice for fully-enclosed displays. The COSMOS, PDC VR-CUBE and the VR-CAVE all use the Polhemus FASTRAK® magnetic tracker while the C6 and ALICE both employ Ascension Technology's MotionStar® Wireless magnetic tracking system. The Polhemus FASTRAK® requires the user to be tethered to equipment used to compute the pose of the sensor, requiring long visible cables, while the MotionStar® Wireless system uses a mag-

netic field emitter outside of the working volume with an extended range of influence and wireless/wearable computer technology to eliminate the need for a tether. Unfortunately, magnetic tracking systems have a number of disadvantages (see [44] and also Chapter 2). They suffer from a large amount of latency due to the need for noise filtering. They are dependent on the local ambient electromagnetic environment and thus are subject to distortion and noise when used in close proximity to metallic objects or stray magnetic fields. A number of fully-enclosed immersive displays are constructed out of wood to reduce this interference. The quality of the magnetic tracking measurements is a function of the magnetic signal strength. Thus, as the user moves further away from the magnetic field emitter, the precision decreases. This implies inconsistent tracking throughout the working area and is illustrated in [45] where a comparison is given between a hybrid inertial-ultrasonic tracking system and an electromagnetic tracking system.

1.3 Problem Statement

The immersive experience within a VR environment is adversely affected by generating improper visual displays. The proper view of the scene is dependent upon the position and orientation of the user's head with respect to the display screen which gives rise to the need for accurate head pose tracking. The "fully-enclosed" nature of six-sided displays reveals the inherent issues with magnetic trackers and makes them inappropriate for use in this context. This thesis describes the design, development and evaluation of a 6 DOF head

tracking system for fully-enclosed immersive projective environments.

In order to overcome the limitations imposed by existing magnetic trackers, a novel "outside-in" vision-based tracking system for tracking a user within a fully-enclosed projective immersive environment is developed. This optical tracker utilizes commercial cameras and computers and is capable of obtaining 6 DOF pose estimates of the user within the environment at 15Hz. This vision-based tracking system is designed to be used either as a standalone tracker or as part of a hybrid system with an inertial tracking component.

The optical system is augmented with an inertial system that utilizes six accelerometers to obtain fast relative pose estimates of the user. It complements the optical system by providing accurate data between consecutive pose updates and increases the performance of the tracking system as a whole.

Combining the relative inertial information with the absolute pose data recovered from the vision-based system provides more reliable tracking results than either system alone. The hybrid tracking system is wireless, freeing the user from a physical tether which allows them to roam freely within the immersive display. The tracking system is named "MARVIN" which is an acronym for a **M**obile **A**utomatic **R**ealtime **V**isual and **IN**ertial tracking system.

1.4 Structure of this Thesis

The remainder of this thesis is organized as follows. Chapter 2 surveys existing tracking technology and related work on tracking in virtual environments. Chapter 3 provides an overview of the MARVIN tracking system. Chapters 4 and 5 describe in detail the inertial and vision-based tracking subsystems and provide a discussion and evaluation of each system respectively. Chapter 6 shows how both systems are integrated into one wearable unit. Finally Chapter 7 provides a discussion and possible directions for future work. Overviews of the recursive least-squares filtering techniques used, and hardware specifications pertaining to the system can be found in the Appendices.

Chapter 2

Virtual Reality Tracking Technology

There have been several recent in-depth surveys of tracking technology published in the literature (see [23, 66]). These surveys discuss different types of tracking technology, explaining briefly the mechanisms behind each method, and their applicability in VR. However, since fully-enclosed projective displays are a recent development, these surveys focus primarily on tracking systems applicable only to HMDs, personal computer systems, or non-fully-enclosed immersive projective displays. Each survey fails to address the unique constraints related to fully-enclosed environments. This chapter describes VR related tracking technology in general and the technology presented in these earlier surveys from the unique perspective of applying them in fully-enclosed immersive projective displays. The advantages and disadvantages of each technology are described and examples of application areas for these technologies are given. Further details on VR tracking for HMD's and more simple projective environments can be found in [23] and [65].

For the purposes of this thesis, tracking is defined as *the process of estimating human body motion, or motion of individual body parts for the purpose of interacting with 3D computer generated environments.* A valid representation of orientation and position is needed. Any free-floating 3D object has six degrees-of-freedom (6DOF), enabling it to rotate about either of the 3 axes $(\hat{X}, \hat{Y}, \hat{Z})$ as well as translating along these axes. This in

turn requires a mathematical representation of orientation and linear translation. This thesis will focus on rigid body motion as this representation normally proves sufficient for virtual environments.

2.1 Transformations

A translation vector, $T_{3\times1} = [t_x, t_y, t_z]^T$ in three-dimensional space defines a simple transformation that when applied to a point, $P_{3\times1} = [p_x, p_y, p_z]^T$, moves the point along a straight line the distance denoted by this vector. The new translated point

$$P'_{3\times1} = [p'_x, p'_y, p'_z]^T \tag{2.1}$$

is defined as

$$p'_x = p_x + t_x \tag{2.2}$$

$$p'_y = p_y + t_y \tag{2.3}$$

$$p'_z = p_z + t_z \tag{2.4}$$

This can be stated in vector notation as $P'_{3\times1} = P_{3\times1} + T_{3\times1}$.

A rotation is a transformation that when applied to a vector will produce a new vector rotated by an angle, θ, about a given axis. A simple 2D example is shown in Figure 2.1. In this two-dimensional rotation, the point $P_{2\times1} = [p_x, p_y]^T$ is rotated by angle θ about the

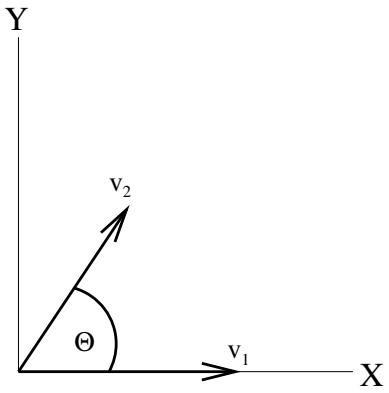


Figure 2.1: Example of 2D rotation. Vector $\vec{v}_1$ is rotated by angle θ around the Z-axis to obtain vector $\vec{v}_2$

origin to compute a new point $P'_{2\times1} = [p'_x, p'_y]^T$. The transformation is easily determined:

$$p'_x = p_x \cos(\theta) + p_y \sin(\theta) \tag{2.5}$$

$$p'_y = -p_x \sin(\theta) + p_y \cos(\theta) \tag{2.6}$$

which can be written in matrix notation as

$$R_{2\times2} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.7}$$

and the full transformation can be denoted by

$$P'_{2\times1} = R_{2\times2} P_{2\times1} \tag{2.8}$$

Even though this is enough to transform a two-dimensional vector, we exist in a three-

dimensional world and must generalize the above to 3D. The above rotation can also be thought of as a rotation of a point around a vector that is perpendicular to this paper. Thus, $R_{2\times 2}$ actually denotes a rotation on a plane around the plane normal, $\vec{n} = [0,0,1]^T$. In order to perform the transformation of a 3D point on this plane, let $P_{3\times 1} = [p_x, p_y, p_z]^T$, and the rotation is now specified by a 3×3 rotation matrix $P'_{3\times 1} = R_{3\times 3} P_{3\times 1}$. Since this rotation matrix is a rotation on the X-Y plane about the Z-axis, let it be denoted by $R_Z(\theta)$. Rotation matrices around the X,Y, and Z axes can be determined and are summarized below:

$$R_X(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) \\ 0 & -\sin(\psi) & \cos(\psi) \end{bmatrix} \tag{2.9}$$

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.10}$$

$$R_Z(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

In order to rotate a 3D point concurrently about all three axis, care must be taken as to the order in which the rotations are applied. The typical Euler angle rotation sequence used in aerospace applications is the Z-Y-X sequence[46]. This means that the vector is rotated

first by the Z-axis by angle ϕ, then around the Y-axis by angle θ, and finally rotated around the X-axis by angle ψ. The final 3×3 rotation matrix is written as

$$R_{3\times3} = R_X(\psi)R_Y(\theta)R_Z(\phi) \tag{2.12}$$

The Z-axis points downward towards the earth and the other two orthonormal axes are defined using the right-hand rule. This sequence is also called Yaw-Pitch-Roll which denotes the deviations of the heading and attitude of an aircraft.

In three-dimensional space, a transformation of vectors through rotation and translation can be written as:

$$P'_{3\times1} = R_{3\times3}P_{3\times1} + T_{3\times1} \tag{2.13}$$

This approach is rather cumbersome. By using homogenous coordinates[31, 14], this transformation can be expressed as a single 4×4 matrix multiplication. By denoting the vector to be transformed as $P_{4\times1} = [p_x, p_y, p_z, 1]^T$, the rotation and translation can be combined into a single 4 by 4 transformation matrix:

$$\mathbf{T}_{4\times4} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{T}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{2.14}$$

and the final homogeneous transformation is defined as :

$$P'_{4\times1} = \mathbf{T}_{4\times4}P_{4\times1} \tag{2.15}$$

The choice of representing the rotation of a vector in 3D as a matrix is useful since matrix multiplication can be implemented in a straightforward manner on a computer. There are other representations of rotation that are worth mentioning[46]. Euler angles, or (Yaw, Pitch, Roll), are very commonly used to represent rotations of frames in 3D. This representation has been used in physics and aerospace for many years, however it suffers from representational issues in certain configurations that compare to "gimbal-lock" in mechanical systems (where a rotation of 90^o on one axis makes the other 2 axes coincident, thus losing a degree of freedom). One way to address this issue is to use the quaternion representation: $\vec{q} = [s, \vec{v}]$ where s is a scalar defining the magnitude of the quaternion and $\vec{v}$ is a 3-vector. It can be shown (see [46]) that a special quaternion can be defined that represents a rotation, namely $[\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\vec{v}]$ where θ is the angle by which the frame is rotated and $\vec{v}$ is a unit vector representing the axis of rotation. It is important to note that the quaternion must be normalized to unit length in order to represent a rotation. This representation is extensively used in computer graphics and is explained in more detail in Appendix A. Another way to represent the orientation of a frame is to express the basis vectors of the frame directly. Using this, one would define 2 vectors and a position in space, namely $\vec{P}$ the position, $\vec{\text{Up}}$ (a unit vector in the direction of the $\hat{Y}$-axis), and $\vec{\text{Dir}}$ (a unit vector in the direction of the positive $\hat{Z}$-axis). This representation is common in the 3D computer graphics field since it is possible to specify these vectors in OpenGL$^{\text{TM}}$ for rendering purposes. There are methods to convert these different representations to one another (see Appendix A). There are many different ways of representing the orientation and position, known as *pose* but

since they are fundamentally equivalent in nature, the technologies discussed here do not confine themselves to one representation.

2.2 Tracking Applications

Accurate tracking technology has become a requirement in many research and industrial fields. There exist a wide range of different applications that require the body, parts of the body, or other objects, to be tracked accurately for a given period of time. Applications range from military use (training, missile guidance, on-board navigation, etc.) to the entertainment industry (tracking hockey pucks, digital characters in movies and games, etc.). The review provided in this chapter focuses mainly on applications in Virtual Reality (VR) and Augmented Reality (AR) which require accurate tracking of human body parts, and the user's head in particular.

The entertainment industry has shown a significant interest in human body tracking. It has become commonplace for movies and games to showcase fully digital characters. Instead of animating every motion a single frame at a time, more recent systems require an actor to perform in front of a camera while several parts of their body are tracked. The resulting captured body motion is then applied to a 3D model already prepared for the final rendering (e.g. Gollum in Lord of the Rings, Jar Jar Binks in Star Wars Episode I, etc.)[72]. Typically, the data is recorded and processed off-line since the focus is on obtaining smooth and accurate motion instead of real-time interaction.

The sport therapy, biomechanics and gait analysis fields also require accurate motion capture and tracking technologies[51, 67]. The quantification and analysis of human/animal body movement requires significant amounts of data and data analysis in order to assess physical rehabilitation processes and methodologies. Motion capture systems provide medical doctors with insight into the progress of a patient's rehabilitation. Athletes also utilize motion information to monitor and maximize their performance and efficiency. Here, the data is processed off-line since the focus is on obtaining highly accurate motion data for analysis.

Ergonomics and human-computer interaction researchers are also interested in the motion of humans. Studying the pose and motion of different body parts while using existing input devices may help researchers to develop more efficient techniques for interaction with computer systems.

Virtual Reality applications require accurate knowledge of the user's head pose in order to generate the proper visual view of the world. Tracking systems for VR must be able to provide head pose information in real-time as opposed to the primarily off-line data needed for motion analysis described above. A VR display, whether it be an HMD, LCD screen, CRT display or an IPT (Immersive Projection Technology) display can be thought of as a window looking into the virtual world. In a real window, different parts of the world should become visible as the user moves relative to the window. In a VR display, head tracking is required to generate the proper view "through" this window. Appropriate head pose information ensures that the geometry of the display is correct. Correct geometry permits

the presentation of correct perceptual cues in the simulated display. Perspective, parallax, and stereo cues are critically dependent upon the simulated scene geometry and vantage point. Parallax cues are important since they enable the user to judge distances according to the relative motion of objects in the scene. Perspective is dependent on head pose as well, and incorrect pose estimates may provide clues to the fact that the user is looking at a "virtual" world rather than a "real" one. Many VR systems require stereo information to be present, which requires different left eye and right eye views to be properly presented to the user. The disparities must be correct and correspond to the orientation of the user's head. If not, the 3D effect is lost due to the inability to fuse the two views, eye strain may occur, and the user is more likely to experience cybersickness.

Current tracking technology can be divided into several categories according to the sensing principle employed: mechanical, electro-magnetic, ultrasonic, inertial, optical, and hybrid tracking systems. Each of these systems has their advantages and disadvantages which are discussed below.

2.3 Ultrasonic Tracking Systems

Ultrasonic (Acoustic) Tracking was possibly one of the earliest attempts at tracking in VR[68]. There are mainly two different methods of using ultrasound to track users, "phase-coherent" and "time-of-flight". Phase-coherent methods utilize a continuous wave source that emits a sinusoid of known wavelength. When the microphone receives the sound, the

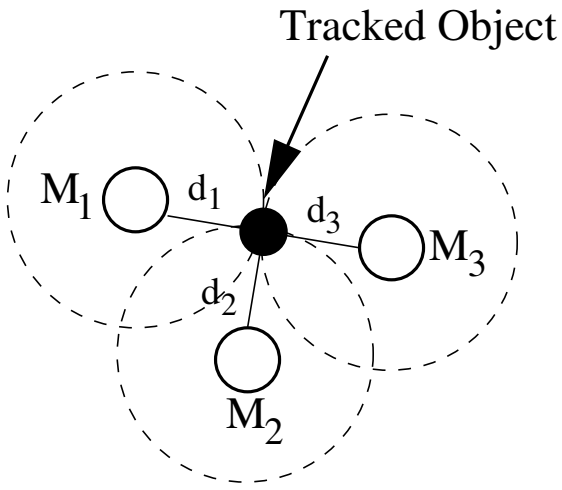
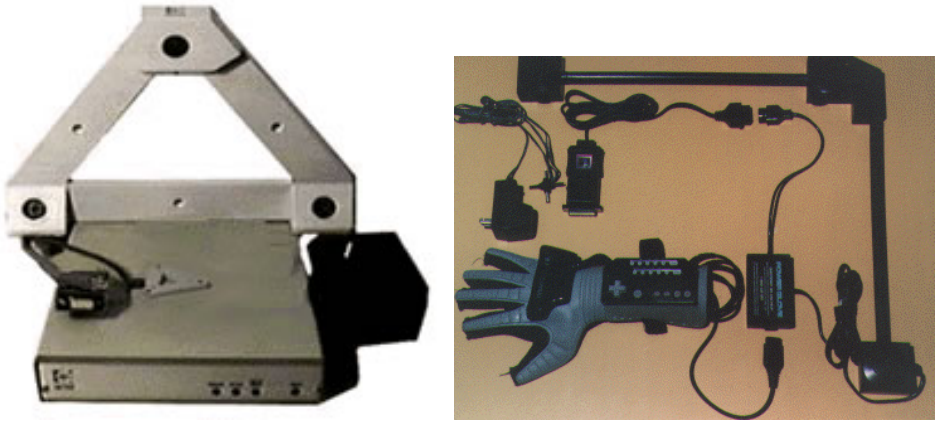


Figure 2.2: The Time of Flight Principle. Attached to the tracked object is an ultrasonic emitter which pulses an acoustical *chirp* at a known time, t. Stationary microphones, M_1, M_2 and M_3 placed in the environment receive these signals at times t_1, t_2, and t_3 respectively. The distance travelled by the ultrasonic chirp can be determined using the known speed of sound, v_s, by $d_i = v_s * (t_i - t)$ (v_s = speed of sound, d_i =distance to ith microphone, and t_i = time M_i received the chirp). Since we can accurately calibrate the 3D locations of the stationary microphones, the distance travelled by the chirp can be thought of as the radius of a sphere centered around each microphone. Using at least 3 microphones allows us to calculate a unique intersection point which is precisely the desired position of the emitter attached to the tracked object (2D case shown above).

signal will have a phase-shift that is proportional to the distance of the sound source and thus position can be determined by phase detection. However, this method only allows the changes in distance to be determined within a cycle. A more common approach in acoustical tracking systems (and all commercially available ultrasonic trackers) is to use the "time-of-flight" principle to determine the position and orientation of a sensor worn by the user (see Figure 2.2).

By emitting a known ultrasonic frequency at a specified time, it is possible to measure the time it takes for the sound *chirp* to reach the sensor. This time delay is proportional to



(a) Logitech 6DOF Tracker (b) Mattel/Nintendo PowerGlove

Figure 2.3: Common Acoustic Tracking Systems

the distance travelled from the emitter to the receiver. Using an estimate of the speed of sound, the position of the sensor can be determined using triangulation techniques. Multiple emitters allows for orientation to be computed as well. The time-of-flight approach is used by the Logitech 2D/6D mouse (Figure 2.3(a)), the Mattel/Nintendo Powerglove (Figure 2.3(b)) and by the acoustic component of the Intersense trackers. Although time-of-flight systems are relatively inexpensive, the approach has many problems. The speed of sound is dependent on the known room temperature and air pressure thus increasing the error if these are not known accurately. Since the system must wait until at least 3 microphones receive the chirp, there are latency issues. The update rate is typically low, and can be as low at 10Hz. There is a line-of-sight constraint since sound must travel from the emitter to the microphone. Occlusions are issues to contend with; the tracker will lose the tracked object if it travels behind another object acoustically. Reflections from surfaces in

the environment produce echoes and these may be confused with valid measurements increasing the system error. Background noise is also an issue, jingling keys or running water produce frequencies in the ultrasonic range that may be mistaken as valid tracker data. Finally, in fully-enclosed immersive displays, there is no adequate way to place the array of microphones without compromising the user's sense of presence. The microphones cannot be placed outside of the environment since the screen material may occlude, absorb, and attenuate the ultrasonic signals making this type of system unusable in a fully-enclosed display.

2.4 Mechanical Tracking Systems

Mechanical tracking systems employ a mechanical armature to track the user's head or body. Standard forward kinematics techniques are used to estimate the end-effector pose from the measured relationships between the mechanical links of the system[14].

The BOOM (Binocular Omni-Orientation Monitor) from Fakespace Labs (see Figure 2.4) is a prime example of a mechanical head tracker for virtual environments. The cumbersomeness of this tracking system limits its use to very specific applications.

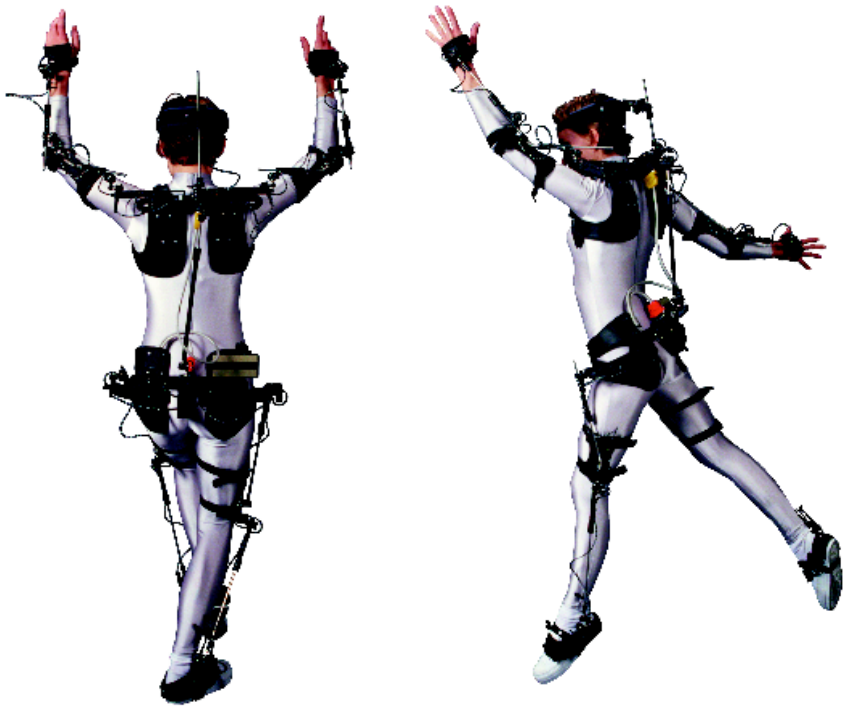
Several types of mechanical arms and exoskeletons have been developed using this forward kinematic technique. Commercially available exoskeletons track limb pose relative to a point on the user's body and are available from vendors such as Puppetworks and Metamotion (see Figure 2.5). Exoskeleton trackers have the advantage of being highly

(a) BOOM®3C (b) BOOM®HF

Figure 2.4: The BOOM (Binocular Omni-Orientation Monitor). Images courtesy of Fakespace Labs.

accurate, having low latency, and being unaffected by magnetic, acoustic, or light energy noise sources although their mass can affect subject performance.

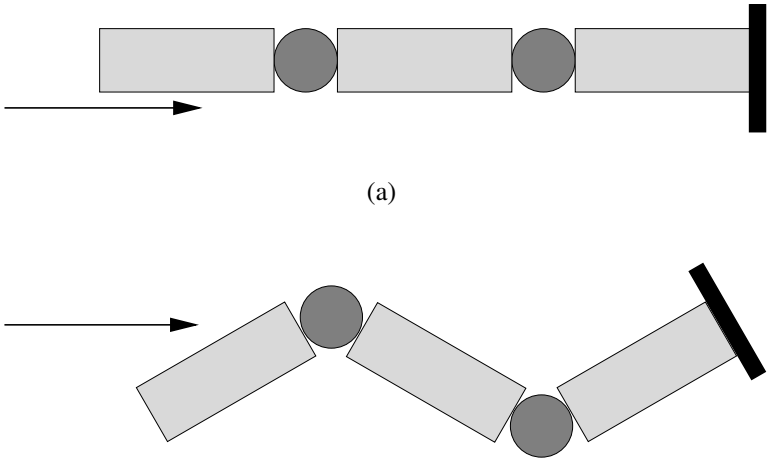
Mechanical tracking systems are extremely intrusive to the user and difficult to set up. They limit the range of motion, induce unnatural motions, and typically require the user to be tethered to a calibrated reference point to obtain absolute measurements. Another major problem of mechanical trackers is gimbal lock which occurs in mechanical systems when the joints align. This creates a situation in which a degree of freedom is lost and the effector must move in a constrained manner (see Figure 2.6). Obviously, this is undesirable for head-tracking in a virtual environment. Since a fully-enclosed display does not allow



(a) (b)

Figure 2.5: The Gypsy Jr.® Mechanical Tracking System. Images Courtesy of Meta Motion, San Fransico, CA

the user to be tethered to a calibrated reference point, mechanical trackers are inappropriate for use in fully-enclosed immersive displays.



(a)

(b)

Figure 2.6: Gimbal Lock. As can be seen in (a), when the joints align in a kinematic chain, a degree of freedom is lost and you cannot move the end effector along the direction of the arrow. (b) shows a valid configuration of the joint angles needed so the end effector can move in that direction.

2.5 Optical Tracking Systems

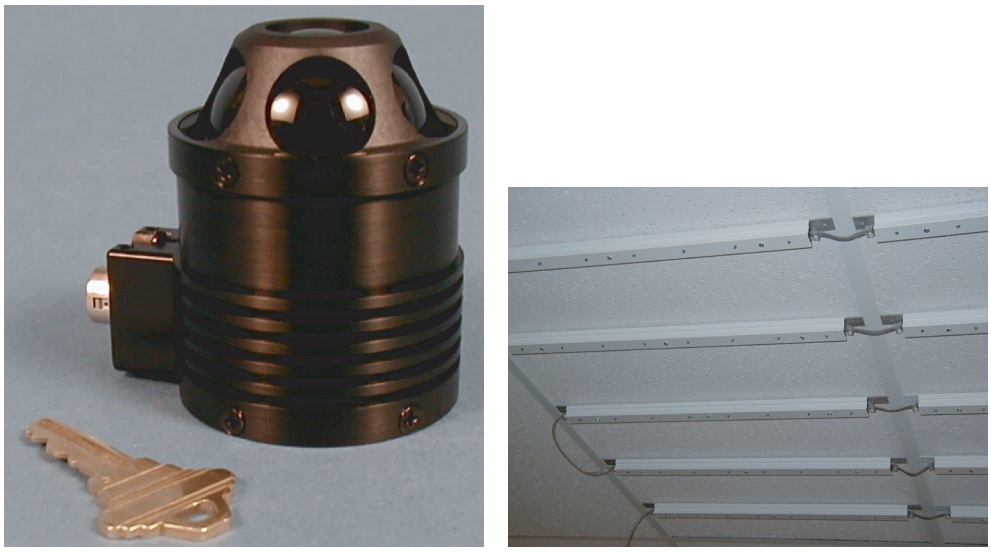
Optical tracking systems use digital cameras to acquire an image representation of the scene. The images are then processed to extract features or other known objects and this information is used to compute the pose of the objects (or of the cameras). There are essentially two different types of optical tracking systems:

- **Outside-In**. Outside-In optical trackers use (multiple) static cameras to view a dynamically changing scene. The relationship between the cameras is accurately calibrated, and matching points in the images are used to determine 3D information. Placing identifiable markers, either active (LEDs flashing in a known constellation) or passive (retroreflective markers, ping-pong balls, etc.), on the moving object at known 3D object locations simplifies the tracking process. The 2D locations of each marker is localized in all of the images and then mapped into the corresponding locations on the known kinematic structure. This results in the ability to recover the final 3D pose of the object. Trackers using this principle are commonly found in the entertainment industry (movies and games) where markers are placed on salient parts of the actor's body who is filmed while performing. The image sequence is analysed off-line to estimate the motion of the performer and this estimated motion is applied to a digital character, however recently developed systems have emerged [72] which are able to optically track the user in real-time.

- **Inside-Out**. Inside-Out tracking is essentially the above algorithm in reverse. Cam-

eras are placed on the object to be tracked and static markers are placed at known locations in the real world. By localizing four or more of the projections of the 3D points in the images, the accurate pose of the camera can be established. This is the principle behind the optoelectronic tracking system developed at the University of North Carolina, and commercially available as the HiBall tracker[77] (see Figure 2.7(a)). Active markers, flashing infrared LEDs, are embedded in the ceiling of the area to be tracked(Figure 2.7(b)) and multiple cameras connected to the user's head view the scene. By localizing the 2D locations of many of these LEDs, the pose of the cameras can be computed and used in the VR simulation. This tracking system has the advantage of being fast, accurate and can cover a wide area (as long as the ceiling LEDs are in sight).

The LaserBIRD® tracking system from Ascension Technology (see Figure 2.8) is an optical inside-out tracking system using fan-shaped laser beams that continuously scan a conical-like tracking region. A set of optical sensors (photodiodes) are placed on the object to be tracked and report the arrival of the laser beam back to a central computer to compute the pose of the sensors. This system has a very fast update rate of 240Hz due to the use of photodiodes that respond to a specific wavelength (795nm), eliminating all need for image processing. This system also incurs a low 5.17ms lag. The accuracy approaches 1^o RMS but requires the user to be tethered to the laser scanner making this system unusable in a fully-enclosed display.



(a) The HiBall^{TM} Optical/Inertial Sensor (b) Active LED Ceiling Panel

Figure 2.7: The HiBall^{TM}Optoelectronic Tracking System. Images courtesy of 3rdTech^{TM}, Inc., Chapel Hill, NC. (http://www.3rdtech.com/)

Optical tracking systems are relatively non-intrusive, usually do not require the user to be tethered to some base computer, and do not restrict the natural motion of the user. Inexpensive digital cameras can be used to reduce the cost of these trackers and inexpensive LEDs can be used as landmarks. However, optical systems are not without their problems. The update rate is limited by the framerate of the cameras (typically 30Hz) and precise camera calibration is usually necessary. Lens distortions lead to a degradation in the accuracy and dynamically changing illumination is a problem which may result in the loss of tracking for several frames. If the camera is in motion, then known landmarks must be placed within the environment for triangulation purposes. Occlusion with other objects in the environment is also a major concern for optical systems.

Figure 2.8: The LaserBIRD®. LaserBIRD® Image courtesy of Ascension Technology, Burlington, VT.

2.6 Electromagnetic Tracking Systems

Sending an electric current through a coil of wire creates a magnetic field to form around the wire. By using three orthogonal coils, different magnetic fields are produced which can be sensed by three other orthogonal passive coils placed in the sensor on an object to be tracked. The strength of the magnetic field can be measured through these passive sensor coils. The signal strength of each field is proportional to the distance of the receiver from the emitter and it also changes with the sensor orientation. These signal strengths can then be used to compute the receiver's position and orientation with respect to the transmitter(see [60, 61]).

Ascension Technology's *Flock of Birds*® and the Polhemus *FASTRAK*® are the leading commercial electromagnetic trackers (see Figure 2.9). These systems have the advan-

tage that they provide high accuracy pose data (when in close proximity to the emitter), they are easy to use, and the sensors are small and lightweight. However, they also suffer from a number of problems. They require the user to be tethered to a base computer system and have a very limited operational range, typically under one meter. These trackers suffer from magnetic interference from metallic objects and other sources that disrupt or emit magnetic fields, which causes a large amount of noise to be injected into the pose measurements. In practice it is necessary to filter the measured signal to reduce the noise level which introduces a slight latency. Ascension Technology has recently introduced a product called the MotionStar® Wireless, which is a magnetic tracker with an extended range of influence and can be completely untethered. However, the user must wear a backpack to carry the computer and the data is streamed through a wireless 802.11b connection. In [80], several tests were performed of this system which show an overall average static error of 24.2cm and a 42.18^o orientation error. The system's best performance in these tests were 1.04cm for static position and 0.24^o static orientation. In [80], the authors state that the poor average performance may have been due to the presence of metallic materials in the test space.

In summary, the main difficulties in using electromagnetic tracking systems in fully-enclosed immersive displays are the need to tether the user to a base station, their poor performance in the presence of metallic material, and the decrease in accuracy as a function of distance.

(a) Flock of Birds®

(b) MotionStar®Wireless

Figure 2.9: Electromagnetic Trackers. Flock of Birds® and MotionStar® Wireless Images Courtesy of Ascension Technology Corporation, Burlington, VT.

2.7 Inertial Navigation Systems

Inertial navigation systems (INS) use sensors that exploit Newton's laws of motion to estimate pose[69]. Newton's laws state that a moving body will continue to move uniformly in a straight line unless there is an external force or disturbance acting on the body. The force applied will produce a proportional acceleration of the body:

$$F = ma \tag{2.16}$$

$$a = \frac{F}{m} \tag{2.17}$$

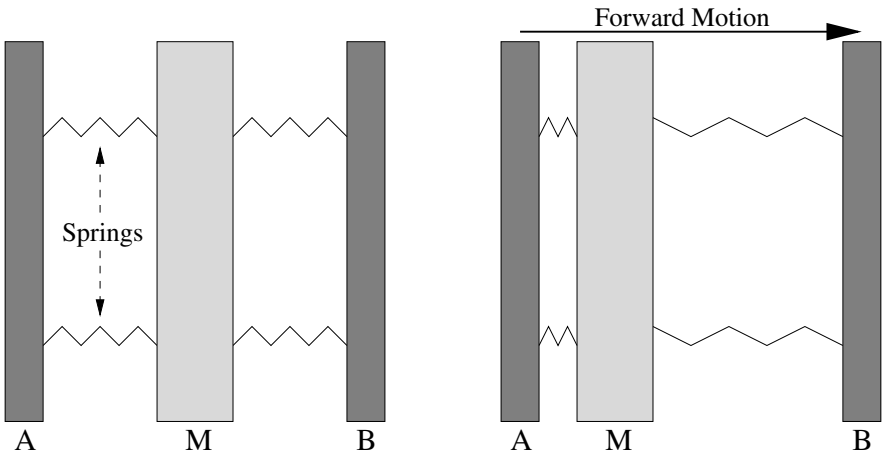
where m is the mass of the object, F is the force applied to the object, and a is the object's acceleration. If it is possible to directly measure this acceleration, then it is possible to estimate accurately the change in position and velocity by performing successive integrations of the measured acceleration with respect to time since

$$v(t) = \int_0^t \alpha \, dt \tag{2.18}$$

$$r(t) = \int_0^t v \, dt \tag{2.19}$$

where $v(t)$ is the velocity at time t, $r(t)$ is the position at time t, and α is the linear acceleration of the object.

A device called an accelerometer (see Figure 2.10), is used to measure the acceleration along the three orthogonal axes. For position estimation in three-space, three accelerome-



(a) Equlibrium (b) Moving

Figure 2.10: The Principle of a Mass-Spring Accelerometer. A mass, M, is suspended between two plates, A and B by springs. When the casing moves, the mass will resist owing to its own inertia moving closer to one of the other plates. The acceleration is proportional to the distance of M between the two plates. An electronic circuit that follows this principle can be manufactured which outputs a change in voltage that is proportional to the distance between the two plates thus measuring the acceleration of the casing.

ters are placed orthogonally to each other at known positions. After integrating the accelerations twice, the vehicle's position relative to its starting pose can be estimated. Accelerometers measure linear acceleration along their sensitive axes and their measurements are also affected by acceleration due to rotational motion which cannot easily be decoupled from the acceleration reported by the sensor. Linear accelerometers cannot distinguish between a constant acceleration due to motion and the gravitational field (since gravity is a constant acceleration). A different type of sensor (or multiple sensors) must be used to estimate the rotational motion to remove this effect from each accelerometer measurement and obtain usable linear acceleration values. Gyroscopes are sensors that directly measure rotational

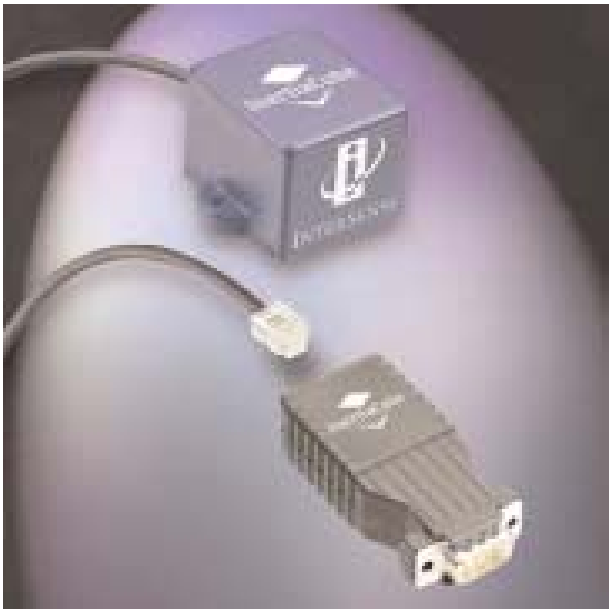


Figure 2.11: The InertiaCube from Intersense. Images courtesy of Intersense, Burlington, MA.

velocity. Estimates of rotational velocity can be used to transform accelerometer data into a reference frame in which the rotational motion is removed, before the integration takes place.

Inertial systems have the advantage that they are fully self-contained (sourceless) and are not affected by magnetic interference, radio transmissions or jamming signals, and can provide data at extremely fast update rates. However, they rely heavily on the accuracy of the starting pose of the vehicle since the sensors simply report changes in the acceleration and velocity. They are plagued by drift caused by integration of the signals, errors in calibration constants for each sensor, and their sensitivity to temperature changes.

2.7.1 Inertial Sensor Errors

Since inertial sensors are physical devices, they are prone to errors which must be modelled accurately in order to provide a usable result[69]. The most common sensor errors for gyroscopes and accelerometers are:

- **Fixed bias**. The sensor provides an output even in the absence of motion. Error in the estimate of the constant bias introduces error and drift after integrating.

- **Scale factor errors**. Errors in the ratio that transforms the signal output to a usable angular velocity or linear acceleration provide incorrect measurements.

- **Cross-coupling errors**. The sensor provides erroneous output resulting from its sensitivity to motions about an axis normal to the measured axis.

- **Noise**. Sensor outputs are typically of very small magnitude (usually in the millivolt range) and thus must be amplified for most analog-to-digital convertors. The amplification of the signal also amplifies any noise in the signal making it more pronounced. Noise will undoubtedly affect the integration and estimation process and standard signal filtering results in a phase shift observed as a latency in the integrated output.

In order to obtain a usable result from inertial sensors, these errors must be taken into account and modelled accurately. Even with precise calibration, there may be unpredictable errors which restrict the use of the device for very short periods of time. In [24], an inertial

orientation-only tracking system was developed that minimizes the drift using other sensors (inclinometers and a fluxgate magnetometer) which report absolute orientation. Even though inertial sensors accumulate error over time, due to sensor errors or miscalibration, the estimate is accurate for a short period. Since inertial sensors provide a very fast update rate, they are a logical choice as a secondary tracking system that provides accurate data for short intervals. Also, the sourceless nature (no tethers or long cables are required) makes them ideal for use in a fully-enclosed immersive display.

2.8 Summary of Existing Tracking Technology

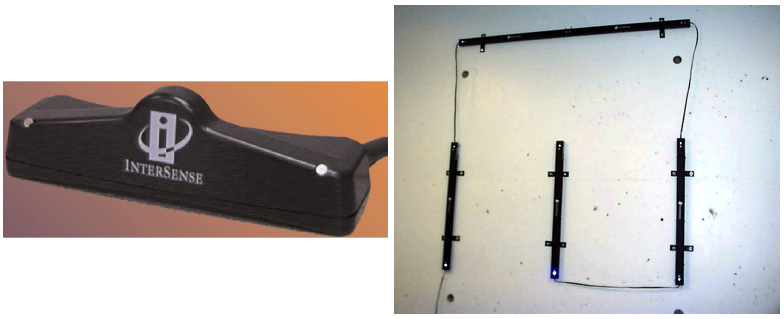
The specifications of the previously described tracking systems are provided in Table 2.1. Given that no single perfect tracking system exists, many researchers have developed hybrid tracking approaches that use data fusion to combine the estimates of multiple complementary systems for more robust and accurate pose estimation.

2.9 Hybrid Tracking Systems

Each of the tracking systems discussed above has a host of advantages and disadvantages. In order to gain accurate and reliable tracking of a user, using one tracking system alone is generally not enough. Because of this, there has been considerable interest in hybrid tracking approaches which uses two or more different tracking systems together to solve the pose estimation/tracking problem. The most common hybrid tracking systems are the

Tracking System	Mfgr.	Type	Range	Accuracy		Resolution		Latency	Update Rate
				Angular	Position	Angular	Position		
GPSMAP196	Garmin	GPS	WIDE	n/a	< 15m	n/a	n/a	15sec	1Hz
GPSMAP196	Garmin	DGPS	WIDE	n/a	3-5m	n/a	n/a	15sec	1Hz
BOOM3C	FakeSpace	Mechanical	1.8m H 0.8m V		0.038cm			200ns	> 70Hz
Flock of Birds	Ascension	Magnetic	$\pm$1.2m $\pm 180^o$ θ, γ $\pm 90^o$ ϕ	0.5^o	1.8mm	0.1^o	0.5mm		144Hz
MotionStar Wireless	Ascension	Magnetic	$\pm$3.05m $\pm 180^o$ θ, γ $\pm 90^o$ ϕ	0.5^o	0.8cm	0.1^o	0.8cm		120Hz
FASTRAK	Polhemus	Magnetic	5"	0.15^o	0.03"	0.025^o	0.0002"	4ms	120Hz
3D Bird	Ascension	Inertial	$\pm 180^o$ θ, γ $\pm 90^o$ ϕ	2.5^o	n/a	0.2^o	n/a n/a	15ms	160Hz
InertiaCube^2	Intersense	Inertial	360^o θ, ϕ, γ	1^o	n/a	0.01^o	n/a	2ms	180Hz
LaserBIRD	Ascension	Optical	$\pm 0.25 - 1.83$m $\pm 85^o$ θ, ϕ $\pm 180^o$ γ	1^o	0.1mm	0.05^o	0.1mm	5.17ms	240Hz
HiBall	3rdTECH	Hybrid	WIDE AREA $\pm 180^o$ θ $0 - 90^o$ ϕ	0.01^o	0.2mm			<1ms	<2000Hz

Table 2.1: Commercially Available Tracker Specifications. θ =Azimuth, ϕ =Elevation, γ=Roll, Manufacturer Specifications, latency reported without filtering signal.



(a) (b)

Figure 2.12: A Hybrid Inertial/Ultrasonic Tracker from Intersense. (a) the emitter that is worn by the user and (b) the sensor unit in a calibrated position.

combination of inertial sensors with optical or acoustic sensors. Inertial information is frequently used since these systems provide precise data at very fast update rates. However, inertial systems only provide relative information that must be mathematically integrated (in practice this is a source of instability), and the estimated pose tends to drift as time increases. Thus, the starting point of the integration must be reset frequently by a tracking system that measures absolute position and orientation. Intersense provides inertial trackers which use a secondary ultrasonic tracking system to measure the absolute pose of the tracked device and continually update the parameters of the inertial system.

Using a hybrid system has advantages, merging the "best" attributes of each system and using information from one system to compensate for deficiencies of the other. In an Inertial/Ultrasonic hybrid tracker, the inertial data will provide extremely fast updates but after some time it will drift, thus the Ultrasonic tracker which works at a much lower update rate is used to compensate for this drift. However, there are still challenges involved

in developing a hybrid system. A highly accurate multi-step calibration process must be developed to calibrate each system independently, as an integrated whole, or both. Since each tracker uses different types of sensors with much different sensor characteristics, the data must be "fused" into a single estimate of position and orientation. This is typically done through the use of a recursive least squares approach such as the Kalman filter[76] or a SCAAT filter[75] which are discussed in Appendix B.

2.9.1 Data Fusion

Data fusion has been researched extensively in the signals[15] and robotics field[17]. Sensor fusion is a large field and a complete discussion is beyond the scope of this thesis. The interested reader is directed to [43, 42, 48, 75, 4, 6, 17] and Appendix B for more details. Typically multiple sensors with different characteristics are used. It is necessary to combine these measurements in some statistically robust way to obtain more information than from a single sensor alone. Given that it is possible to measure the sensor characteristics off-line prior to use, the "best" a system can do to maintain an estimate using these sensors is to take a weighted average of the measurements taken up to and including a given time. As an example, take a static situation where you wish to estimate some variable $\hat{x}$ and measurements with equal uncertainty are available from n sensors, then the simplest "best" estimate of the variable is:

$$\hat{x} = \frac{\sum x_i}{n}$$

Usually, it is appropriate in these situations to evaluate the "goodness-of-fit" of the confidence in the estimated variable. This can be accomplished by computing the covariance matrix for the sensor system:

$$COV = \begin{bmatrix} \sum x_1^2 & \sum x_1 x_2 & \dots & \sum x_1 x_n \\ \sum x_2 x_1 & \sum x_2^2 & \dots & \sum x_2 x_n \\ & & \dots & \\ \sum x_n x_1 & \sum x_n x_2 & \dots & \sum x_n^2 \end{bmatrix}$$

In a system where measurements from the sensors have different uncertainties, different approaches must be used. The Kalman filter (see [76] and Appendix B) is set of equations precisely designed for this purpose. In a Kalman filter, the system to be estimated is described as a set of variables combined into a vector called the *state vector*. The technique can fuse measurements from (possibly) multiple sensors with different characteristics (noise levels, certainties, etc.) are available at the same or different times. By applying the filtering equations, the estimate of the state is updated with every measurement using a Predictor-Corrector feedback loop. The Kalman filter incorporates a known dynamical model of the system to predict the next state. A measurement model based on the predicted state is used to predict the sensor measurement. The difference between this prediction and the actual measurement can be used as an error metric for the model. The state vector is then updated using this error to evaluate the sensor measurement and weigh it appropriately limiting its influence on the updated state. The covariance matrix of the sys-

tem is also updated at each step. Measurements are combined with the current estimate by weighing these estimates by their uncertainties. This technique has been employed extensively in almost every field in computer science and engineering and has several extensions to evaluate non-linear systems(a technique known as the extended Kalman filter, EKF[75]).

A system is said to be *observable* when the state vector can be fully determined from a single measurement (i.e. the system can generate a new estimate of the state vector with every measurement). This is the way most tracking systems employ the Kalman filter. Multiple sensors give estimates of the postion and orientation of an object and the Kalman filter is used to keep track of this information appropriately. However, an interesting extension to this is to use a Kalman filter to estimate a *globally observable* system from multiple *locally unobservable* systems. This modification to the extended Kalman filter was named SCAAT which stands for *single-constraint-at-a-time*[75]. This technique was explored in [75] for the HiBall Optoelectronic tracking system, where each measurement of a single LED in the ceiling is used to update the state of the system. Each measurement is insufficient to fully estimate the pose of the HiBall thus leading to a locally unobservable system. However, SCAAT algorithm allows each individual measurement to contribute to the final estimate of the system leading to a globally observable state after multiple measurements.

2.10 Tracking in other domains

As discussed above, there has been much research and development of technology and sensors that allow the user to be tracked within virtual environments. Inertial sensor technology has been used effectively in vehicle tracking and navigation [9, 11, 18, 47, 58]. More recently, inertial sensors have been used to track body parts for inserting humans into virtual worlds[6]. A hybrid inertial/optical system developed by [82] uses inertial information (six accelerometers) to track the fast head motion of the user wearing a see-through HMD. A camera is also placed on the HMD to track known landmarks in the environment. By tracking these landmarks, an estimate of the head motion is computed. They use an extended Kalman filter to fuse this data with the inertial data to obtain a robust estimate of the head pose. Given the current head-pose, objects may be placed in the scene at the appropriate locations. Since the inertial data gives very fast updates, they are able to compensate very quickly for the head motion so that the 3D object being superimposed on the scene stays in the same apparent position. A hybrid inertial/vision-based tracking system was also developed in [84, 83]. They use a differential-based optical-flow calculation to track the camera motion and use commercially available inertial trackers to predict the motion of feature vectors in the images. For the inertial/optical HiBall tracking system developed at the University of North Carolina at Chapel Hill, a Kalman filter-based predictive tracking algorithm was developed in [4], which was used to track head motions for use in augmented reality applications. Another hybrid optical/inertial tracker was developed in [52] that used

a camera to localize binary coded fiducial marks placed in the environment to compute the absolute pose of the camera and is then used to compensate for the inertial drift. The system is placed on the user's glasses to track their head motion.

2.11 Summary

In a fully-enclosed immersive display, the head tracking system must operate robustly under the constraint that the user is fully-enclosed in the workspace. Since is not desirable to have any physical mechanism inside the workspace (since it would detract from the immersive experience) the tracking system must be able to provide estimates from outside the display. Ultrasonic tracking systems are inappropriate for use in a fully-enclosed immersive environment since their acoustic *chirps* would be absorbed by the screen material and/or reflected within the environment causing echos. Inertial systems are promising for fully-enclosed displays since they do not require a tether, minimal processing is required and can be self-contained in a wearable device. Mechanical trackers are too cumbersome and inhibit natural motion. Even though electromagnetic trackers have been used in many fully-enclosed displays, tethering the user inhibits their motion and immersive experience. Also, the noise characteristics in the presence of metallic objects and the accuracy falloff with distance makes these systems unusable within IVY. Perhaps the most promising approach for tracking a user within fully-enclosed projective displays is to consider a hybrid tracking approach based on visual and inertial sensors.

Chapter 3

The MARVIN Tracking System

There are several different criteria to evaluate a tracking technology for virtual reality. In [23] the following possible criteria are introduced:

- User feels *present* in the virtual world.
- Perceptual stability: fixed virtual objects appear stationary, even during head motion.
- No simulator sickness occurs.
- Task performance is unaffected by any tracking artifacts.
- Tracking artifacts are below the detection threshold of a user who is looking for them.

These criteria are extremely relevant to keep in mind when choosing or developing a tracking technology for VR. If the tracking technology is insufficient under any of these metrics, the immersive experience of the user will be degraded.

In Chapter 2 different technologies available for human body motion tracking were discussed, placing emphasis on each system's advantages and disadvantages. Currently, no single system is sufficient for general tracking, nor does there currently exist a non-invasive tracking system for fully-enclosed projective environments.

A tracking system for use in a fully-enclosed projective display must satisfy the following constraints:

- It must be tetherless. The user must be able to roam freely within the environment.
- It must have low latency. The user's actions should trigger the appropriate change in the display immediately, or within a sufficiently short period of time that the lag is unnoticeable.
- It must have a high update rate. The system must update at least as fast as the framerate of the presented visual display.
- Sensors attached to the body must be small and lightweight and must not affect the user's natural motions.
- It must have high rotational accuracy. Rotation is extremely important to generate the proper stereo vision cues.
- It must be jitter/glitch free.

With these constraints and the above criteria in mind, a wireless hybrid inertial/optical tracking system has been developed for use in a fully-enclosed immersive projective environment. This system is named **MARVIN**: a **M**obile **A**utomatic **R**ealtime **V**isual and **IN**ertial tracking system. The framework developed here has been implemented for the IVY environment at York University but is general enough to use in any fully-enclosed immersive projective display.

3.1 Overview

A hybrid tracking system has the potential to be extremely robust and accurate for VR systems and in particular for fully immersive displays. The use of inertial information for fast updates and a slower optical system to provide accurate absolute data is probably the best candidate for tracking in fully immersive displays. The hybrid system developed in this thesis satisfies the unique constraints imposed by fully-enclosed immersive environments. MARVIN allows the user to roam freely within the display without being physically tethered or being encumbered by a large device, and the accuracy acheived is consistent throughout the physical range as well as being resistant to external interference. Using recursive least squares filtering techniques, MARVIN provides smooth, jitter-free pose estimates with a high rotational and positional accuracy throughout the entire working volume.

MARVIN's inertial component, builds on the inertial tracking system developed in [82], and uses acceleration information to compute relative position and orientation of the user at a very fast update rate (greater than 200Hz). By imposing geometric constraints on the placement of six accelerometers, the inertial device is able to provide relative motion information on all axes (as discussed in Chapter 4). The recovered accelerations are integrated twice to recover angular and linear displacements. This lightweight device is mounted on a bicycle helmet that is worn by the user. Calibration of the sensors is of utmost concern since any bias in the accelerations will produce a quadratically increasing error in our

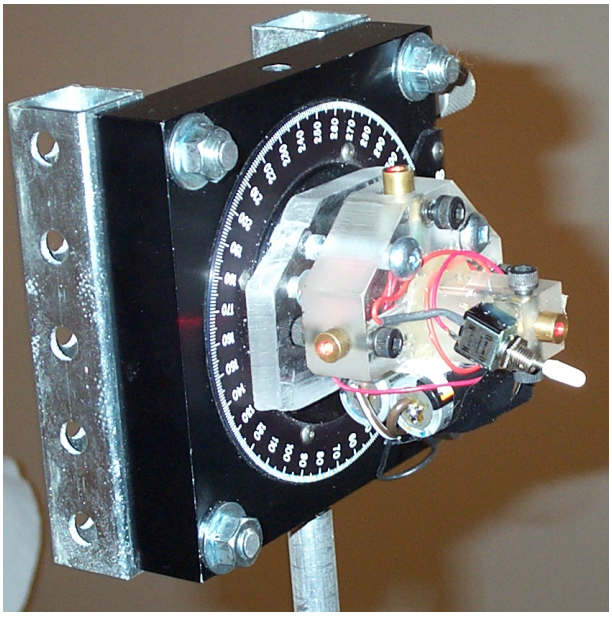


Figure 3.1: The optical tracking system hardware. A lightweight acrylic housing holds the 4 laser diodes used to determine the user's head pose.

computed displacements. These errors, and other unpredictable sensor errors, subject the inertial pose estimate to drift rapidly. To compensate for the drift, a novel "outside-in" optical tracker was developed[35] to provide absolute pose data at a reliable but slower update rate(15-20Hz).

MARVIN's optical tracking component consists of a lightweight acrylic housing for four low-power visible light laser diodes. The diodes are mounted in a known geometric configuration such that the projections of the laser beams emitted from the device provides geometric constraints on the pose of the device (See Figure 3.1). Ensuring that the lasers point behind the user's viewing direction keeps the tracking system from interfering with the user's visual experience. Cameras situated outside of IVY aimed at the rear-projection screens provides tracking of the visible laser projections using standard computer vision

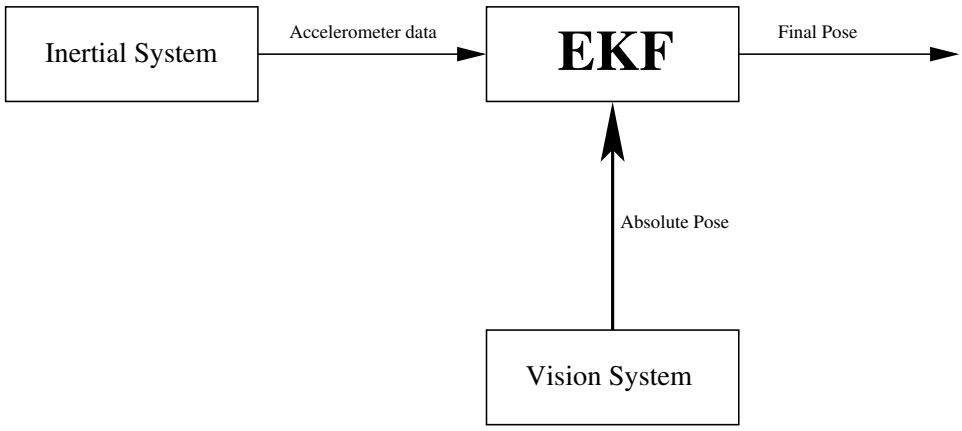


Figure 3.2: Overview of the MARVIN system. An extended Kalman filter (EKF) is used to fuse the accelerometer and vision data into a single pose estimate. The EKF continually estimates the pose from the inertial system and the vision system provides absolute pose data as a control input to the filter.

techniques. The optical component complements the inertial system by providing absolute pose data at a slower but reliable update rate. As with the inertial system, calibration of the optical system is extremely important. The cameras must be fully calibrated in order to provide an accurate description of the scene. Moreover, since the cameras are not perfectly aligned with the screen, a transformation from image space to screen space is required. This transformation is modeled using a planar projective homography[29] that maps image coordinates into screen coordinates. Finally, the transformation from the coordinate system defined by the laser device to the user's head must be determined for the system to be used.

A major constraint imposed by existing magnetic tracking systems is the need for the user to be physically tethered. MARVIN requires no physical tether through the use of 802.11b standard wireless ethernet technology in conjunction with a lightweight PC/104 wearable computer. PC/104 technology was chosen since it is very popular in the wearable

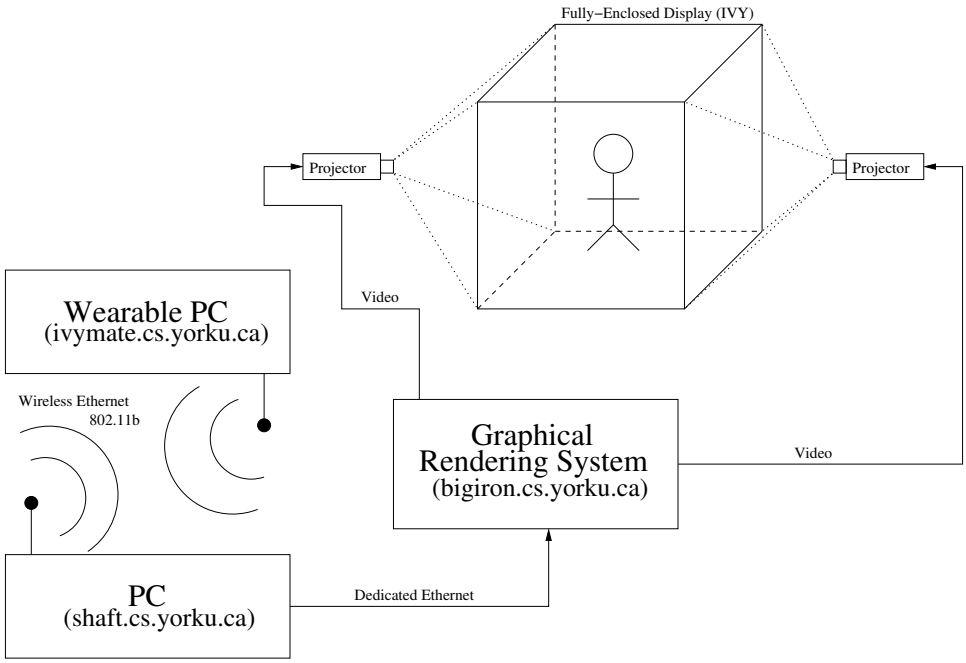


Figure 3.3: Layout of current implementation of MARVIN. The wearable PC is connected to the input PC via a wireless ethernet link to send input events and accelerometer data. The input PC computes the pose and relays the events off to the calling program on the rendering machine which generates the appropriate video sent to the projectors.

computer realm (see [64]) and there is a significant hardware and software infrastructure available. Using this standard provides a straightforward upgrade path when new functionality is needed in the future. Standard USB and serial ports are also available which makes it possible to connect many different readily available input devices for the user. Even though PC/104 devices are small and lightweight, they are sufficiently powerful to collect and transmit inertial data from the user to a base station, while providing some local data analysis. Data is transmitted through the wireless network to a base computer outside of IVY, where it is integrated and filtered to maintain an estimate of the user's relative pose.

The MARVIN software infrastructure was built as a driver for use with the VE library

(see http://www.cs.yorku.ca/∼ivy/ve and also [65]). The decision to use VE as the software infrastructure permits any program written in VE to take advantage of this tracking system with minimal additional effort. In order to use the MARVIN system, a single line of code is added to the program's device file: "use marvin". This initializes the tracking system when VE is initialized and routes the generated tracking events to the calling application.

A typical device file for VE that uses the MARVIN tracking system is shown below:

```
use keyboard
use marvin
use gamepad { optional 1 }

filter keyboard.q { exit }
filter marvin.pose { rename tracker.frame }
filter *.* { dump }
```

The driver acquires images of all projective surfaces, performs some image processing to localize the laser dots, and performs the pose calculation. A separate thread in the driver acquires inertial data from the wireless network and combines this information with the absolute pose estimate using an extended Kalman filter. At each update of the Kalman Filter, an event is triggered with the network input device layer (NID) of VE and the pose data is delivered to the appropriate calling program.

The following chapters discuss the details of each of MARVIN's subsystems, describing the theory behind each, issues addressed in developing these systems, performance characteristics, and the calibration processes employed.

Chapter 4

The Inertial System

The ability of inertial sensors to provide accurate relative motion data within short time intervals at a high update rate is invaluable for a tracking system in virtual reality. Since these types of sensors are able to operate without a physical tether to a calibrated reference point, they are ideal candidates for tracking in fully-enclosed immersive projective displays.

Linear accelerometers sense the linear acceleration of a moving body. By attaching multiple accelerometers in a specific configuration, it is possible to use the differential measurements from the sensors to also estimate the angular acceleration[51, 50]. This chapter provides the mathematics and theory behind the configuration of accelerometers used in the MARVIN tracking system. First, the mathematics of rigid body dynamics will be reviewed providing insight into the use of acceleration to estimate the position and orientation of a moving body. These equations are then applied to the six accelerometer configuration used in MARVIN followed by a discussion on the calibration of the device. A method to estimate the pose of the device is described and results from a simulation undergoing rotation and translation will be shown. Finally, an evaluation of the inertial tracking system will be presented.

In 1686-87, Sir Isaac Newton published his Principia Mathematica[53] which defined the concepts of mass, momentum, inertial forces, centripetal forces and proposed the three

laws of motion. An exhaustive review of these concepts is beyond the scope of this thesis and the reader is directed to [78] for more detail. However, the equations of motion that are relevant to the development of this thesis are developed in this chapter.

4.0.1 Accelerometers

Accelerometers transduce the specific force acting on an accelerating body. Since the acceleration output from the sensor is a change in voltage, this is directed into an analog-to-digital (A/D) converter and analyzed by a digital computer. The changes in voltage are directly proportional to the force applied to the body. From Newton's second law of motion, this output is therefore directly proportional to the acceleration of the body. The sensor output must then be converted into a usable acceleration value.

Sensors do not provide perfect data. The output is corrupted by random noise, nonlinearity, and cross-coupling errors due to sensitivity on transverse axes. Calibration data such as biases, and scalings also influence the measurements and must be modelled in order to transform the voltage output into the proper acceleration.

In [69], the output of the accelerometer, $\hat{a}_x$, along its sensitive axis, x, is modelled as

$$\hat{a}_x = (1+S_x)a_x + M_y a_y + M_z a_z + B_f + B_v a_x a_y + n_x \tag{4.1}$$

where S_x = scale factor error

M_y, M_z = cross-coupling factors along perpendicular axes

B_f = measurement bias

B_v = vibro-pendulous error-coefficient

n_x = random noise bias

This model can be simplified by using the calibration results from the manufacturer. The cross-coupling factors are negligable and the vibro-pendulous error-coefficient is inapproprate for use in a mass-spring accelerometer. Under these simplifications, Equation 4.0.1 becomes

$$\hat{a}_x = (1+S_x)a_x + B_f + n_x \tag{4.2}$$

Note however that this only provides the proper acceleration measurement of a single accelerometer. In a typical inertial navigation system, multiple accelerometers are used to provide acceleration along each of the axes in 3D Euclidean space and the calibration parameters for each must be estimated. Also, it is practically impossible to build a housing that ensures the sensors are perfectly orthogonal, the misalignments between the sensors must be taken into account. The calibration of a multiple accelerometer tracking system

will be discussed later.

4.0.2 Rigid Body Motion

Rigid body dynamics is a long studied field and an exhaustive review is beyond the scope of this thesis, however the concepts that are key to this thesis will be discussed and the reader is urged to see [14, 78] and Appendix A for more details.

For the following discussion on rigid body dynamics, a few key concepts need to be defined;

- Frame of Reference.
- Inertial frame of reference.
- Linear acceleration of a reference frame.
- Angular acceleration of a reference frame.
- Angular velocity of a reference frame.

Frame of Reference

A frame of reference is an orthogonal set of unit vectors with a common origin (see Figure 4.1) that constitutes a coordinate system used to describe the position, velocity and acceleration of an object. In the context of this thesis, a frame of reference A will be denoted

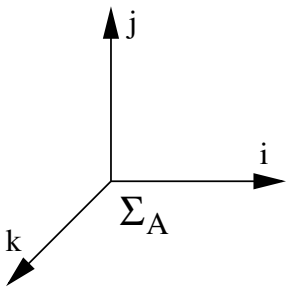


Figure 4.1: Frame of Reference. The frame of reference denoted as $\sum_A$ is composed of three unit vectors i, j, k which are orthogonal to each other and constitute an orthonormal basis.

as $\sum_A$. Any vector within the frame of reference can be expressed as a linear combination of the basis vectors of the frame.

Inertial Frame of Reference

An inertial frame of reference ($\sum_I$) is a non-accelerating reference frame. It is also typically described as a reference frame in which Newton's laws of motion hold, or as a reference frame with constant velocity. As an example, take an accelerating car that has its own reference frame attached to it (Figure 4.2). That is to say that any object within the car can be expressed in this reference frame as a linear combination of the basis vectors of the frame. However since this reference frame is accelerating, it cannot be considered as an inertial frame. A good choice for the inertial frame in this scenario would be one that is attached to a stationary observer looking onto the accelerating vehicle, or the center of the earth since each of these are moving at a constant velocity.

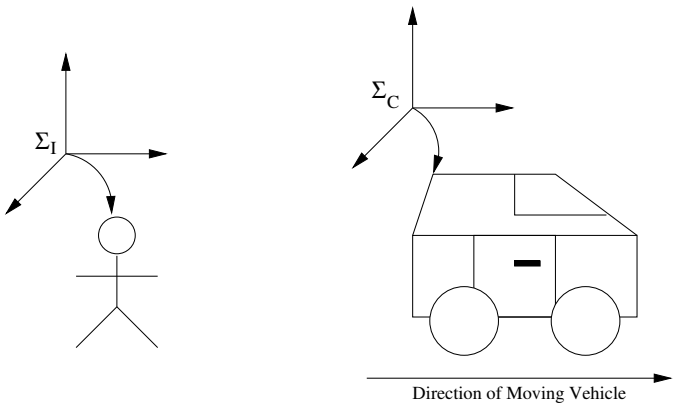


Figure 4.2: Inertial Frame of Reference.

Linear Motion of a Reference Frame

Let $\sum_I$ denote the inertial frame of reference, and let $\sum_A$ denote a second frame of reference attached to object A (see Figure 4.3). Here the object is moving with respect to the inertial frame but not rotating. The origin of $\sum_A$ can be expressed as a vector in the inertial frame; let the origin of frame $\sum_A$ be denoted by vector $\vec{R}$. The motion of the frame can be described by how the vector $\vec{R}$ changes over time. Thus, the velocity of $\sum_A$ can be described as the change of $\vec{R}$ with respect to the inertial frame over time:

$$\vec{v} = \frac{^I d\vec{R}}{dt} \tag{4.3}$$

where the superscripted I denotes the frame in which the derivative is taken with respect to.

Similarly the linear acceleration, α, of the frame is the change in velocity over time, or

rather the second derivative of the position vector:

$$\vec{\alpha} = \frac{{}^{I}d^2\vec{R}}{dt^2} \tag{4.4}$$

Now, let $\vec{r}$ denote an object expressed within the moving reference frame (i.e. imagine a passenger within the moving car in the example above). The total linear acceleration of this object is the linear acceleration of the object itself with respect to the moving frame added to the linear acceleration of the frame itself. Likewise for the velocity of the object. Note however in the following that $\vec{r}$ is expressed in the inertial frame as is the origin of its associated moving reference frame.

$$\vec{v} = \frac{{}^{I}d\vec{R}}{dt} + \frac{{}^{I}d\vec{r}}{dt} = \frac{{}^{I}d}{dt}(\vec{R}+\vec{r}) \tag{4.5}$$

$$\vec{\alpha} = \frac{{}^{I}d^2\vec{R}}{dt^2} + \frac{{}^{I}d^2\vec{r}}{dt^2} = \frac{{}^{I}d^2}{dt^2}(\vec{R}+\vec{r}) \tag{4.6}$$

Angular Motion of a reference frame

When a body rotates about a given axis, the angular velocity can be computed and expressed as a vector, $\vec{\omega}$. As seen in Figure 4.4, if ω is a scalar and $\vec{v} = [0,1,0]^T$ is a unit vector around the Y-axis, then the angular velocity about this axis can be described as $\vec{\omega} = \omega\vec{v} = [0,\omega_y,0]^T$. The axis of rotation is perpendicular to the rotating vector. This gives

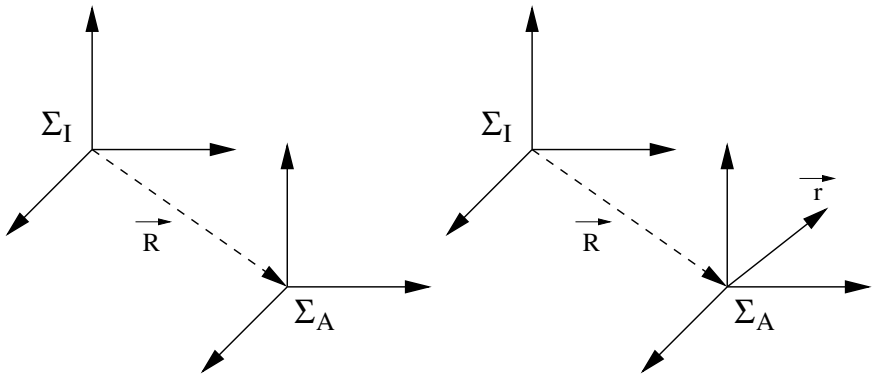


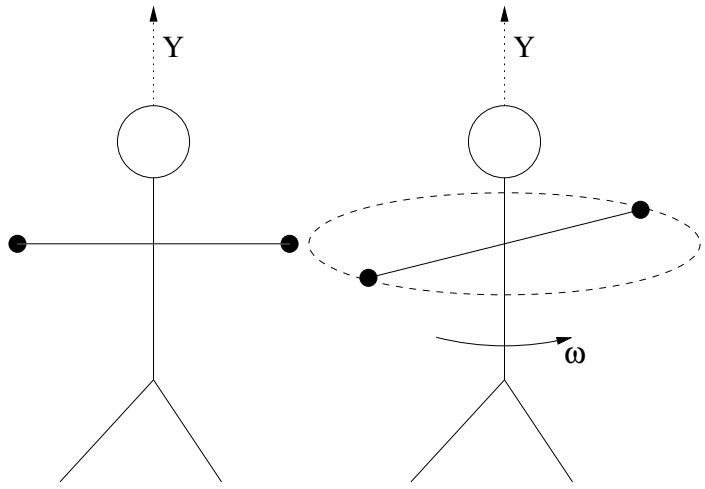
Figure 4.3: Linear Motion of a Reference Frame. (a) shows that the motion of the origin of frame $\sum_A$ can be described as a vector $\vec{R}$ in an inertial frame of reference. (b) shows that the motion of the vector $\vec{r}$ within the non-inertial frame is described as the motion of itself in addition to the motion of the entire frame with respect to the inertial frame.

an important result, the axis of rotation is perpendicular to the differential change of the vector.

Figure 4.5 shows a vector $\vec{r}$ undergoing a rotation around a vector $\vec{\omega}$. The radius of the circle illustrated has a magnitude of $|\vec{\omega}|\sin(\theta)$. The change in a vector $\vec{r}$ undergoing a rotation about a unit vector $\vec{v} = \frac{\hat{\omega}}{|\hat{\omega}|}$ with angular velocity $|\vec{\omega}|$ can be expressed in terms of the cross product of the vector $\vec{r}$ with the angular velocity vector $\vec{\omega} = |\vec{\omega}|\vec{v}$,

$$\frac{d\vec{r}}{dt} = \vec{\omega} \times \vec{r} \tag{4.7}$$

Note however that this describes the rotational change of the vector only.



(a) Initial Pose

(b) Rotating about Y-axis with angular velocity ω

Figure 4.4: Angular Velocity Example. As a person rotates about their body axis (Y-axis), they induce a rotational velocity ω about the axis of rotation.

Final Motion of a Rigid Body

Both the linear and rotational change of a vector must be accounted for in order to accurately describe the motion of the vector with respect to another frame of reference. In general, to describe the motion of a vector expressed in reference frame $\sum_A$ as it changes with respect to reference frame $\sum_B$, the following relationship can now be defined[78] which accounts for all linear and rotational motion of the vector:

$$\frac{^B d\vec{\mathbf{r}}}{dt} = \frac{^A d\vec{\mathbf{r}}}{dt} + {}^B\vec{\omega}_A \times \vec{\mathbf{r}} \tag{4.8}$$

where $\vec{\mathbf{r}}$ is the vector describing the particle in frame A, ${}^B\vec{\omega}_A$ is the angular-velocity vector of frame A with respect to frame B, and $\frac{^A d}{dt}\vec{\mathbf{r}}$ describes the change of the vector itself in

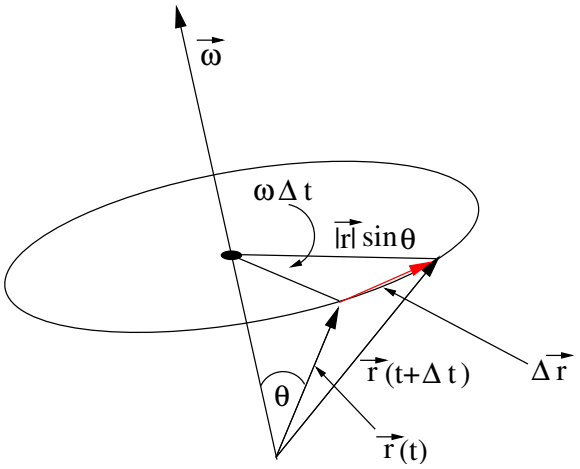


Figure 4.5: Angular velocity of a vector.

frame A. Note that this relationship only accunts more a moving particle in a stationary reference frame, i.e the reference frame is not rotating and moving itself. However, the motion of the reference frame can be taken into account in the same fashion as will be seen later.

This rule can be used to transform derivatives in one frame of reference into derivatives with respect to a different frame of reference. The general rule is written as (following the notation in [78]):

$$\frac{^{B}d}{dt}(\,) = \frac{^{A}d}{dt}(\,) + {}^{B}\vec{\omega}_{A} \times (\,) \tag{4.9}$$

Using this equation, it is possible to derive the acceleration of a point with respect to another frame of reference, in particular the inertial frame. Applying Equation 4.9 twice, and explicitly representing the motion of the reference frame as well, results in:

$$\frac{^{B}d^{2}}{dt^{2}}\left(\vec{\mathbf{R}} + \vec{r}\right) = \frac{^{B}d^{2}}{dt^{2}}\vec{\mathbf{R}} + \frac{^{B}d}{dt}\left(\frac{^{A}d}{dt}\vec{\mathbf{r}} + {}^{B}\vec{\omega}_{A} \times \vec{\mathbf{r}}\right) \tag{4.10}$$

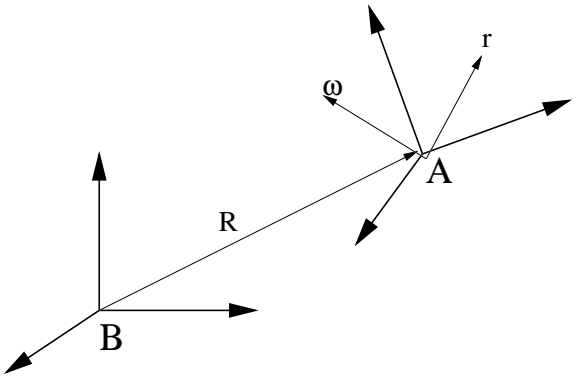


Figure 4.6: Transformation between two frames.

Where $\vec{\mathbf{R}}$ is the displacement between the two frames (see 4.6).

Equation 4.10 can be expanded into

$$\frac{{}^{B}d^2}{dt^2}\left(\vec{\mathbf{R}}+\vec{\mathbf{r}}\right)=\frac{{}^{B}d^2}{dt^2}\vec{\mathbf{R}}+\frac{{}^{A}d^2}{dt^2}\vec{\mathbf{r}}+2\left({}^{B}\vec{\omega}_A\times\frac{{}^{A}d}{dt}\vec{\mathbf{r}}\right)+\left(\frac{{}^{A}d}{dt}{}^{B}\vec{\omega}_A\right)\times\vec{\mathbf{r}}+\left({}^{B}\vec{\omega}_A\times\left({}^{B}\vec{\omega}_A\times\vec{\mathbf{r}}\right)\right) \tag{4.11}$$

Let $\dot{\vec{a}}$ denote the first derivative of a vector $\vec{a}$ and let $\ddot{\vec{a}}$ denote the second derivative. Using this shorthand to simplify the above relation, the acceleration of a vector $\vec{\mathbf{r}}$ expressed in an inertial frame B is defined as:

$$\frac{{}^{B}d^2}{dt^2}\left(\vec{\mathbf{R}}+\vec{\mathbf{r}}\right)={}^{B}\ddot{\vec{\mathbf{R}}}+{}^{A}\ddot{\vec{\mathbf{r}}}+2\left({}^{B}\vec{\omega}_A\times\dot{\vec{\mathbf{r}}}\right)+({}^{B}\dot{\vec{\omega}}_A\times\vec{\mathbf{r}})+{}^{B}\vec{\omega}_A\times({}^{B}\vec{\omega}_A\times\vec{\mathbf{r}}) \tag{4.12}$$

For the rest of the discussion, the superscripts to denote the reference frames are dropped for convenience since the transformations discussed will always be in reference to an inertial reference frame.

4.0.3 Rigid Body Dynamics in the MARVIN tracking system

In the MARVIN tracking system, the aluminum housing that holds all of the accelerometers constitutes the sensor frame housing coordinate system. The accelerometers are held rigid in place and do not move with respect to this origin, thus $\frac{d\vec{\mathbf{r}}}{dt} = 0$. This removes the Coriolis force term, $2\left(\vec{\omega} \times \dot{\vec{\mathbf{r}}}\right)$, from Equation 4.12. Finally, the acceleration of point $\vec{\mathbf{r}}$ in the MARVIN tracking system is described as

$$\vec{A}_{\mathbf{r}} = \vec{\alpha} + (\dot{\vec{\omega}} \times \vec{\mathbf{r}}) + \vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}) \tag{4.13}$$

where $\dot{\vec{\omega}}$ denotes the angular acceleration vector between the two frames of reference, and $\vec{\alpha}$ denotes the total linear acceleration of the vector.

4.1 MARVIN Inertial System Configuration

To describe the motion of the user's head, MARVIN utilizes six accelerometers in the geometric configuration shown in Figure 4.7. The accelerometers transduce the applied force along the axes specified, and each sensor has an associated frame of reference. The device is rigidly attached to the operator's head. To estimate the motion of the user's head it is necessary to describe the angular and linear acceleration of the entire device (not just the acceleration of the sensor positions). The acceleration required is of the origin of the device frame *O* positioned at the center of the device.

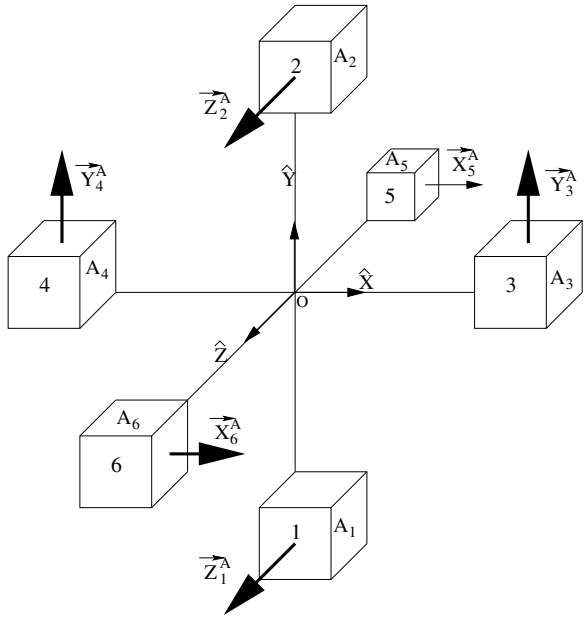


Figure 4.7: The Inertial System Labeled with Sensing Axes. The reference frame origin is denoted by O in the centre of the device with its associated orthogonal axes. Each accelerometer senses acceleration along the specified axis.

To determine the relationships between what is sensed by the accelerometers and the motion of the entire device, Equation 4.13 is applied to each accelerometer frame. Let A_i denote the reference frame of the i^{th} accelerometer.

For the following derivation, define the 3D position vector of an accelerometer as $\vec{\mathbf{r}} = [r_x, r_y, r_z]^T$, its 3D angular acceleration vector as $\vec{\omega} = [\omega_x, \omega_y, \omega_z]^T$, and its 3D linear acceleration vector as $\alpha = [\ddot{x}, \ddot{y}, \ddot{z}]^T$.

Expanding $\vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}})$ into its vector components results in:

$$\vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}) = \begin{bmatrix} \omega_y(\omega_x r_y - \omega_y r_x) - \omega_z(\omega_z r_x - \omega_x r_z) \\ \omega_z(\omega_y r_z - \omega_z r_y) - \omega_x(\omega_x r_y - \omega_y r_x) \\ \omega_x(\omega_z r_x - \omega_x r_z) - \omega_y(\omega_y r_z - \omega_z r_y) \end{bmatrix} \tag{4.14}$$

Expanding $\dot{\vec{\omega}} \times \vec{\mathbf{r}}$:

$$\dot{\vec{\omega}} \times \vec{\mathbf{r}} = \begin{bmatrix} \dot{\omega}_y r_z - \dot{\omega}_z r_y \\ \dot{\omega}_z r_x - \dot{\omega}_x r_z \\ \dot{\omega}_x r_y - \dot{\omega}_y r_x \end{bmatrix} \tag{4.15}$$

These equations can now be applied to each accelerometer in its associated frame. The accelerometer positions (from Figure 4.7) are:

Accelerometer	Position
A_1	$\mathbf{r_1} = [0, -\delta_1, 0]^T$
A_2	$\mathbf{r_2} = [0, \delta_2, 0]^T$
A_3	$\mathbf{r_3} = [\delta_3, 0, 0]^T$
A_4	$\mathbf{r_4} = [-\delta_4, 0, 0]^T$
A_5	$\mathbf{r_5} = [0, 0, -\delta_5]^T$
A_6	$\mathbf{r_6} = [0, 0, \delta_6]^T$

These positions are used in Equation 4.14 and Equation 4.15 which results in three equations per accelerometer, one for each axis of the accelerometer frame. After some

simplification, the acceleration of each accelerometer frame is described as:

$$A_1 = \begin{bmatrix} \ddot{x} - \omega_y\omega_x\delta_1 + \dot{\omega}_z\delta_1 \\ \ddot{y} + \omega_z^2\delta_1 + \omega_x^2\delta_1 \\ {}^{\dagger}\ddot{\mathbf{z}} - \omega_\mathbf{y}\omega_\mathbf{z}\delta_\mathbf{1} - \dot{\omega}_\mathbf{x}\delta_\mathbf{1} \end{bmatrix} \qquad A_2 = \begin{bmatrix} \ddot{x} + \omega_y\omega_x\delta_2 - \dot{\omega}_z\delta_2 \\ \ddot{y} - \omega_z^2\delta_2 - \omega_x^2\delta_2 \\ {}^{\dagger}\ddot{\mathbf{z}} + \omega_\mathbf{y}\omega_\mathbf{z}\delta_\mathbf{2} + \dot{\omega}_\mathbf{x}\delta_\mathbf{2} \end{bmatrix} \tag{4.16}$$

$$A_3 = \begin{bmatrix} \ddot{x} - \omega_y^2\delta_3 - \omega_z^2\delta_3 \\ {}^{\dagger}\ddot{\mathbf{y}} + \omega_\mathbf{x}\omega_\mathbf{y}\delta_\mathbf{3} + \dot{\omega}_\mathbf{z}\delta_\mathbf{3} \\ \ddot{z} + \omega_x\omega_z\delta_3 - \dot{\omega}_y\delta_3 \end{bmatrix} \qquad A_4 = \begin{bmatrix} \ddot{x} + \omega_y^2\delta_4 + \omega_z^2\delta_4 \\ {}^{\dagger}\ddot{\mathbf{y}} - \omega_\mathbf{x}\omega_\mathbf{y}\delta_\mathbf{4} - \dot{\omega}_\mathbf{z}\delta_\mathbf{4} \\ \ddot{z} - \omega_x\omega_z\delta_4 + \dot{\omega}_y\delta_4 \end{bmatrix} \tag{4.17}$$

$$A_5 = \begin{bmatrix} {}^{\dagger}\ddot{\mathbf{x}} - \omega_\mathbf{z}\omega_\mathbf{x}\delta_\mathbf{5} - \dot{\omega}_\mathbf{y}\delta_\mathbf{5} \\ \ddot{y} - \omega_z\omega_y\delta_5 + \dot{\omega}_x\delta_5 \\ \ddot{z} + \omega_x^2\delta_5 + \omega_y^2\delta_5 \end{bmatrix} \qquad A_6 = \begin{bmatrix} {}^{\dagger}\ddot{\mathbf{x}} + \omega_\mathbf{z}\omega_\mathbf{x}\delta_\mathbf{6} + \dot{\omega}_\mathbf{y}\delta_\mathbf{6} \\ \ddot{y} + \omega_z\omega_y\delta_6 - \dot{\omega}_x\delta_6 \\ \ddot{z} - \omega_x^2\delta_6 - \omega_y^2\delta_6 \end{bmatrix} \tag{4.18}$$

$$\tag{4.19}$$

However, the accelerometers sense acceleration along a single axis only (specified by †) in Equations 4.16 to 4.18 resulting in a single equation per accelerometer.

The equations simplify to:

$$A_1 = \ddot{\mathbf{z}} - \omega_{\mathbf{y}}\omega_{\mathbf{z}}\delta_{\mathbf{1}} - \dot{\omega}_{\mathbf{x}}\delta_{\mathbf{1}} \tag{4.20}$$

$$A_2 = \ddot{\mathbf{z}} + \omega_{\mathbf{y}}\omega_{\mathbf{z}}\delta_{\mathbf{2}} + \dot{\omega}_{\mathbf{x}}\delta_{\mathbf{2}} \tag{4.21}$$

$$A_3 = \ddot{\mathbf{y}} + \omega_{\mathbf{x}}\omega_{\mathbf{y}}\delta_{\mathbf{3}} + \dot{\omega}_{\mathbf{z}}\delta_{\mathbf{3}} \tag{4.22}$$

$$A_4 = \ddot{\mathbf{y}} - \omega_{\mathbf{x}}\omega_{\mathbf{y}}\delta_{\mathbf{4}} - \dot{\omega}_{\mathbf{z}}\delta_{\mathbf{4}} \tag{4.23}$$

$$A_5 = \ddot{\mathbf{x}} - \omega_{\mathbf{z}}\omega_{\mathbf{x}}\delta_{\mathbf{5}} - \dot{\omega}_{\mathbf{y}}\delta_{\mathbf{5}} \tag{4.24}$$

$$A_6 = \ddot{\mathbf{x}} + \omega_{\mathbf{z}}\omega_{\mathbf{x}}\delta_{\mathbf{6}} + \dot{\omega}_{\mathbf{y}}\delta_{\mathbf{6}} \tag{4.25}$$

The equations can be simplified further by ensuring that $\delta = \delta_1 = \delta_2 = \delta_3 = \delta_4 = \delta_5 = \delta_6$ through accurate manufacturing of the sensor housing. Simplifying these equations under the condition that each accelerometer is δ length away from the origin:

$$A_1 = \ddot{z} - \omega_y\omega_z\delta - \dot{\omega}_x\delta \tag{4.26}$$

$$A_2 = \ddot{z} + \omega_y\omega_z\delta + \dot{\omega}_x\delta \tag{4.27}$$

$$A_3 = \ddot{y} + \omega_x\omega_y\delta + \dot{\omega}_z\delta \tag{4.28}$$

$$A_4 = \ddot{y} - \omega_x\omega_y\delta - \dot{\omega}_z\delta \tag{4.29}$$

$$A_5 = \ddot{x} - \omega_z\omega_x\delta - \dot{\omega}_y\delta \tag{4.30}$$

$$A_6 = \ddot{x} + \omega_z\omega_x\delta + \dot{\omega}_y\delta \tag{4.31}$$

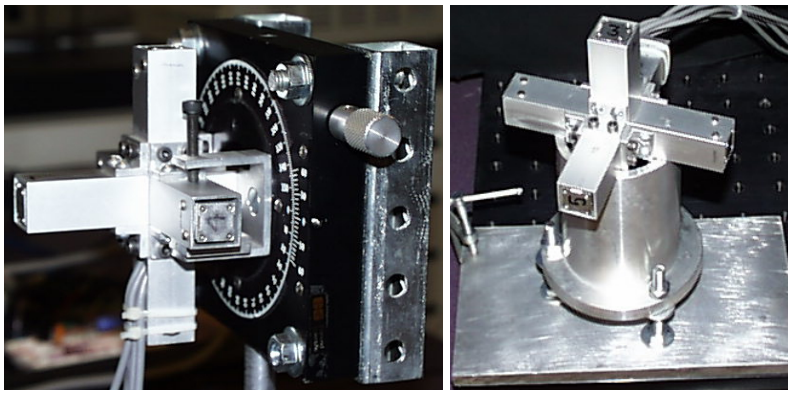
By adding pairs of accelerometers (from Equation 4.26 to 4.31) together, the linear acceleration of the sensor frame can be estimated as:

$$\begin{aligned} \frac{A_1 + A_2}{2} &= \ddot{z} \\ \frac{A_3 + A_4}{2} &= \ddot{y} \\ \frac{A_5 + A_6}{2} &= \ddot{x} \end{aligned} \tag{4.32}$$

Note that the linear acceleration estimated in Equation 4.32 is biased by gravity. It is impossible to distinguish between a linear acceleration applied to the device from the acceleration induced by the earth's gravitational field. In order to subtract the gravity vector from the linear acceleration measurements, an estimate of the orientation of the frame is needed.

Subtracting pairs of acceleration measurements (from Equation 4.26 through 4.31) results in:

$$\begin{aligned} \frac{A_2 - A_1}{2} &= \omega_y \omega_z \delta + \dot{\omega}_x \delta \\ \frac{A_3 - A_4}{2} &= \omega_x \omega_y \delta + \dot{\omega}_z \delta \\ \frac{A_6 - A_5}{2} &= \omega_z \omega_x \delta + \dot{\omega}_y \delta \end{aligned} \tag{4.33}$$



(a) (b)

Figure 4.8: The Inertial Hardware and Calibration equipment.

and solving for the angular accelerations:

$$\begin{aligned}
\dot{\omega}_x &= \frac{A_2 - A_1}{2\delta} - \omega_y\omega_z \\
\dot{\omega}_z &= \frac{A_3 - A_4}{2\delta} - \omega_x\omega_y \\
\dot{\omega}_y &= \frac{A_6 - A_5}{2\delta} - \omega_z\omega_x
\end{aligned} \tag{4.34}$$

As can be seen by Equation 4.34 the angular acceleration can be estimated by subtracting the outputs of the accelerometers as long as the angular velocity is estimated (or measured).

The six accelerometer configuration has been designed, developed and implemented. The hardware device is shown in Figure 4.8 and the specifications of the device is given in Appendix C.

4.2 Alternate Solution

The above calculations are correct only for the configuration of six accelerometers described above, however a more general approach can be developed. Given that the acceleration of a point is described by

$$\vec{A}_{\mathbf{r}} = \vec{\alpha} + (\dot{\vec{\omega}} \times \vec{\mathbf{r}}) + \vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}) \tag{4.35}$$

the j^{th} accelerometer senses only along a specified axis, $\vec{n}_j$. Thus the output of the accelerometer can be described as

$$\vec{a}_j = \vec{n}_j \cdot (\vec{\alpha} + (\dot{\vec{\omega}} \times \vec{\mathbf{r}}_j) + \vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}_j)) \tag{4.36}$$

Given that this equation has nine unknowns, $\vec{\alpha}, \vec{\omega}, \dot{\vec{\omega}}$ are 3×1 vectors, nine accelerometer estimates can be used (as in [50]) to solve this system of equations using standard techniques. More than nine would allow the use of least-squares methods[7] leading to a more robust solution. However, in the MARVIN configuration only six accelerometers are used which makes this problem underconstrained. Given that $\vec{\omega}$ and $\dot{\vec{\omega}}$ are related through time-integration, $\vec{\omega}$ can be estimated at each time step and "removed" from the system of equations leading to a system of six equations with six unknowns.

Expanding Equation 4.36

$$\vec{a}_j = \vec{n}_j \cdot \vec{\alpha} + \vec{n}_j \cdot (\dot{\vec{\omega}} \times \vec{\mathbf{r}}_j) + \vec{n}_j \cdot (\vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}_j)) \tag{4.37}$$

Now, by moving the terms containing $\vec{\omega}$ to the left hand side of the equation

$$\vec{a}_j - \vec{n}_j \cdot (\vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}_j)) = \vec{n}_j \cdot \vec{\alpha} + \vec{n}_j \cdot (\dot{\vec{\omega}} \times \vec{\mathbf{r}}_j) \tag{4.38}$$

Let Ω_j denote $\vec{n}_j \cdot (\vec{\omega} \times (\vec{\omega} \times \vec{\mathbf{r}}))$ for simplicity

$$\vec{a}_j - \Omega_j = \vec{n}_j \cdot \vec{\alpha} + \vec{n}_j \cdot (\dot{\vec{\omega}} \times \vec{\mathbf{r}}_j) \tag{4.39}$$

Given the relation

$$a \cdot (b \times c) = (c \times a) \cdot b \tag{4.40}$$

Equation 4.39 becomes

$$\vec{a}_j - \Omega_j = \vec{n}_j \cdot \vec{\alpha} + (\vec{\mathbf{r}}_j \times \vec{n}_j) \cdot \dot{\vec{\omega}} \tag{4.41}$$

which can be expressed in matrix form as

$$\begin{bmatrix} \vec{a}_j - \Omega_j \end{bmatrix} = \begin{bmatrix} \vec{n}_j{}^T & (\vec{\mathbf{r}}_j \times \vec{n}_j)^T \end{bmatrix} \begin{bmatrix} \vec{\alpha} \\ \dot{\vec{\omega}} \end{bmatrix} \tag{4.42}$$

Since Equation 4.42 is a linear system of equations, standard techniques for inverting matrices or solving systems of equations can be used. Note that for an appropriate choice of sensor positions $\vec{n}_j$ the matrix

$$\begin{bmatrix} \vec{n}_j{}^T & (\vec{\mathbf{r}} \times \vec{n}_j)^T \end{bmatrix} \tag{4.43}$$

is a 6×6 invertible matrix and the system is solvable provided that the measurements provide acceleration information on all orthogonal axes.

4.3 Determining Pose

Assuming that the calibration constants are known, the acceleration data is computed from the raw digitized sensor outputs. This is discussed later in Section 4.4. The following algorithm to determine the pose of the device assumes that the sensor outputs have already been transformed into raw acceleration measurements.

In the following, the pose is defined as having two components:

1. A quaternion, $\hat{q}$, describing the orientation of the sensor frame.

2. A vector, $\vec{p}$, describing the position of the sensor frame with respect to an inertial world frame.

As seen in Figure 4.9, the method is decomposed into several parts. Here, the orientation and position estimation will be dealt with separately.

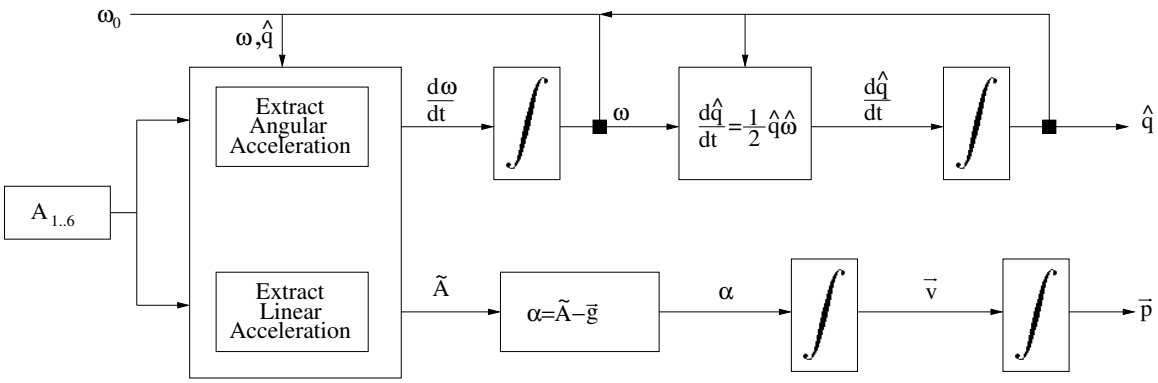


Figure 4.9: Determining pose from accelerometers. This illustrates an algorithm of how to estimate the pose of the sensor frame in MARVIN using inertial data. The pose is composed of a position vector $\vec{p}$ and a quaternion $\hat{q}$ representing the orientation. To estimate $\hat{q}$, first the angular acceleration values are extracted from the sensor data. Then this angular acceleration is integrated into the angular velocity of the sensor frame. This is then converted into a rate quaternion and integrated to obtain the final orientation. The position estimation is also straightforward, the linear acceleration of the sensor frame is extracted from the sensor data. This is biased by gravity. The real linear acceleration must be computed by subtracting the gravity vector expressed in the inertial frame. Next, the linear acceleration is integrated to obtain the velocity $\vec{v}$ which in turn is integrated to obtain the final position vector, $\vec{p}$.

4.3.1 Orientation Estimation

In order to estimate the orientation of the device, the following must be performed:

1. Angular Acceleration Extraction

2. Angular Velocity Estimation

3. Rate Quaternion Estimation

4. Final Quaternion Estimation

Given the raw acceleration data from the six accelerometers, it is possible to simply use Equation 4.34 and use differential pairs of the accelerometers to extract the angular

acceleration of the sensor frame.

Equation 4.34 requires an initial estimate of the angular velocity. A reasonable assumption would be to set the initial angular velocity to zero.

The subsequent angular velocity can then be estimated using the relationship:

$$\vec{\omega} = \int_0^t \dot{\vec{\omega}} dt \tag{4.44}$$

Since a quaternion representation is desired for the orientation, a rate quaternion can be computed using the relationship derived in Appendix A for the quaternion derivative, namely

$$\dot{\hat{q}} = \frac{1}{2} \hat{q} \otimes \hat{\omega} \tag{4.45}$$

Once the rate quaternion has been computed, it is then possible to integrate the quaternion into a final orientation estimate, $\hat{q}$.

Using this algorithm, the orientation of the sensor frame can be estimated assuming that the sensors are properly calibrated. Now, the second component of the pose estimate must be computed, namely the position vector.

4.3.2 Position Estimation

To estimate the position of the sensor frame, four steps are followed:

1. Linear Acceleration Extraction

2. Gravity Compensation

3. Linear Velocity Estimation

4. Final Position Estimation

The position vector of the sensor frame is entirely dependent on the linear acceleration of the sensor frame. However, the effect of gravity can not be decoupled from the accelerometer measurements. Thus the effects of gravity must be compensated for in the position estimation. Since the estimated linear acceleration vector is expressed in the inertial frame, the gravity vector may be subtracted from the acceleration estimate directly.

First, to compute the linear acceleration of the sensor frame, Equation 4.32 is applied to the pairs of accelerometer outputs. Then, the effect of gravity is compensated for by subtracting the gravity vector from the linear acceleration vector. The gravity compensation can be defined as

$$\alpha = \vec{\mathbf{A}} - \vec{g} \tag{4.46}$$

where $\vec{\mathbf{A}}$ is the output determined with Equation 4.32 and $\vec{g} = [0, -9.81, 0]$ is the constant gravity vector pointing along the negative Y-axis.

Once the linear acceleration of the sensor frame is estimated, the linear velocity, $\vec{v}$ can be computed using the relationship

$$\vec{v} = \int_0^t \alpha dt \tag{4.47}$$

The position vector, $\vec{p}$, can then be computed using a second integral and a second-order Runge-Kutta technique[59],

$$\vec{p} = \int_0^t \vec{v} dt \tag{4.48}$$

4.4 Calibration

The previous sections showed how to estimate the pose of the inertial device in theory. In practice, however, there are many other concerns that need to be addressed. In order for the above to work, the sensors should be positioned at exactly the same distance, δ, away from the origin. Each pair of accelerometer's sensitive axes must be aligned and orthogonal to the other pairs of accelerometers' sensitive axes, i.e. there must not be any misalignment between the accelerometers. Real-world sensors output their values in voltages which must be converted to the proper acceleration values using *biases* and *scale* factors. Any error in these biases and scales will quickly corrupt the displacement estimate after two numerical integrations. Noise characteristics of the sensors must be taken into account as well.

In order to accomplish the difficult task of compensating for misalignment and error, a calibration routine must be developed to estimate these parameters. Since multiple sensors are used and each have their own calibration constants, they must be calibrated separately followed calibration of the entire device with respect to the sensor frame origin.

4.4.1 Multiple Sensor Model

In [73], a method for calibrating an inertial pose estimation system for use in robotic systems is presented. The model presented uses 3 orthogonal rate gyroscopes and 3 orthogonal linear accelerometers to compute the pose of the robot.

Briefly, an accelerometer can be modelled as

$$A_i = B_i + (1 + S_i) \cdot < \vec{\alpha}(r_i) + \vec{g}, \vec{n}_i >$$

where A_i is the output of the i^{th} accelerometer, B_i, S_i are the bias and scales respectively, $\vec{n}_i$ is the axis the accelerometer senses upon, $\vec{\alpha}(r_i)$ is the linear acceleration at the accelerometer position r_i, and $\vec{g}$ is the gravity vector in the inertial frame.

In a static situation (when no motion is present), the linear acceleration term, $\alpha(r_i)$, becomes zero and the sensor output is modelled as:

$$A_i = B_i + (1 + S_i) \cdot < \vec{g}, \vec{n}_i >$$

Note that A_i provides a maximum output when $\vec{g}$ is aligned with $\vec{n}_i$ and is in the *same* direction. A_i is also at a minimum when $\vec{g}$ is aligned with $\vec{n}_i$ and is in the *opposite* direction. Let A_i^{j+} denote the acceleration on axis i when A_j is maximal, and A_i^{j-} as the acceleration

on axis i when A_j is minimal,

$$A_i^{j+} = B_i + (1+S_i)\cdot||\vec{g}||\cdot<\vec{n}_j,\vec{n}_i>$$

$$A_i^{j-} = B_i - (1+S_i)\cdot||\vec{g}||\cdot<\vec{n}_j,\vec{n}_i>$$

In the above model, the calibration parameters needed to be estimated are

1. Scale factors and biases of each accelerometer, S_i, B_i.

2. Inner products $<\vec{n}_j,\vec{n}_i>\, i \neq j$ which represent the misalignment of the accelerometers.

These are the *intrinsic parameters* of the sensor and can be estimated as:

$$B_i = \frac{A_i^{j+} + A_i^{j-}}{2} \text{ for } j = 1,2,3$$

$$1+S_i = \frac{A_i^{i+} - A_i^{i-}}{2\cdot||\vec{g}||}$$

$$<\vec{n}_j,\vec{n}_i> = <\vec{n}_i,\vec{n}_j> = \frac{A_i^{j+} - A_i^{j-}}{A_i^{i+} - A_i^{i-}} = \frac{A_j^{i+} - A_j^{i-}}{A_j^{j+} - A_j^{j-}}$$

Thus, it is possible to find the calibration parameters if one is able to precisely position each accelerometer in such a way to observe maximum and minimum values on each sensor. In [73], an autocalibration routine using a robotic arm is employed to iteratively re-estimate the calibration parameters converging on a final solution.

4.4.2 MARVIN Inertial Calibration

The calibration method described above and in [73] assumes that three orthogonal accelerometers are used and are co-located. This is not the case in the MARVIN system where six-accelerometers are used. The technique must be adapted for use in MARVIN, however much of the ideas presented in [73] can be directly applied on a per accelerometer basis.

The calibration technique developed for the MARVIN system needs to perform the following:

1. Find biases and scaling factors to convert voltage output into usable acceleration values.

2. Determine the direction of each accelerometer sensing axis.

These two steps can be solved for using four measurements (the minimum number of measurements needed). Due to physical constraints of the sensor housing, these four measurements can be described as

1. Place sensor housing so the +X axis is aligned with the gravity vector.

2. Place sensor housing so the -X axis is aligned with the gravity vector.

3. Place sensor housing so the +Y axis is aligned with the gravity vector.

4. Place sensor housing so the -Z axis is aligned with the gravity vector.

Measurements from all accelerometers are taken simultaneously for each orientation. The bias and scaling of each accelerometer can be found by taking four specific measurements from each accelerometer.

Let the output of the j^{th} accelerometer be a_j and let the i^{th} calibration measurement for this accelerometer be denoted as ${}^{i}a_j$. Also let B_j, S_j denote the bias and scale respectively of the j^{th} accelerometer and let the gravity vector be $\vec{g} = [g_x, g_y, g_z]^T$ and the sensitive axis of this accelerometer be $\vec{n}_j = [n_x, n_y, n_z]^T$. Each calibration orientation can be described by:

$$ {}^{1}a_j = B_j + S_j g_x n_x \tag{4.49} $$

$$ {}^{2}a_j = B_j - S_j g_x n_x \tag{4.50} $$

$$ {}^{3}a_j = B_j + S_j g_y n_y \tag{4.51} $$

$$ {}^{4}a_j = B_j - S_j g_z n_z \tag{4.52} $$

The bias for each accelerometer can be computed as

$$ B_j = \frac{{}^{1}a_j + {}^{2}a_j}{2} \tag{4.53} $$

Now, applying the computed bias to three independent measurements

$$^{2}a'_j = -(^{2}a_j - B_j) = S_j g_x n_x \tag{4.54}$$

$$^{3}a'_j = {}^{3}a_j - B_j = S_j g_y n_y \tag{4.55}$$

$$^{4}a'_j = -(^{4}a_j - B_j) = S_j g_z n_z \tag{4.56}$$

Note that this can be described simply as

$$\vec{a}'_j = S_j \vec{g} \cdot \vec{n} \tag{4.57}$$

and also note that the following is also true

$$|\vec{a}'_j| = -S_j |\vec{g}||\vec{n}| \tag{4.58}$$

Thus, the scaling can be computed as

$$S_j = -\frac{|\vec{a}'_j|}{|\vec{g}||\vec{n}|} \tag{4.59}$$

Note that $|\vec{n}| = 1$ since it is a unit vector, thus the scaling for this accelerometer can be determined by

$$S_j = -\frac{|\vec{a}'_j|}{|\vec{g}|} \tag{4.60}$$

Once the bias and scales are determined for each accelerometer, the sensing direction vector for each sensor must also be computed. This is accomplished by creating a system of equations and solving for the components of $\vec{n}_j$.

$$^{i}\vec{a}_j = B_j + S_j A \vec{n}_j \vec{g} \tag{4.61}$$

where A is defined as

$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.62}$$

and $^{i}\vec{a}_j = [^{1}a'_j, ^{2}a'_j, ^{3}a'_j, ^{4}a'_j]^T$ is the measurement vector.

This system can be solved by using the normal equations[7] of the matrix A and computing

$$\vec{n}_j = (A^T A)^{-1} A^T \frac{^{i}\vec{a}_j - B_j}{S_j |\vec{g}|} \tag{4.63}$$

This method is capable of computing the biases and scales of each accelerometer as well as the direction of the sensing axis by placing the sensor housing in four or more orientations with one of its axes aligned with the gravity vector. By modeling the directions of the sensing axes of each accelerometer, the misalignments are unneeded since this is inherent in the representation.

4.5 Inertial System Simulator

In order to verify that the above discussed method for determining position and orientation of the inertial device is correct, a software simulator was developed. A screenshot of the simulator can be seen in Figure 4.10. The simulator places six "virtual" accelerometers in the same configuration as the real device and positions it within a "virtual" IVY. The pose of the virtual device can be modified and moved interactively. The simulation timer callback is programmed to update the simulation at a known update rate and at each step computes the output of each accelerometer. The orientation of the virtual device is maintained as a quaternion, $\hat{q}$, and a position vector, $\vec{p}$, and each is updated appropriately. At each step of the simulation, the angular velocity, $\vec{\omega}$, is computed by

$$\frac{d\hat{q}}{dt} = \frac{1}{2}\hat{q} \otimes \hat{\omega} \tag{4.64}$$

$$\hat{\omega} = 2\hat{q}^{-1}\frac{d\hat{q}}{dt} \tag{4.65}$$

where $\hat{\omega} = [0, \vec{\omega}]^T$ and a first order approximation to the quaternion derivative is used

$$\frac{d\hat{q}}{dt} \approx \frac{\hat{q}_t - \hat{q}_{t-1}}{\Delta t} \tag{4.66}$$

Next, the angular acceleration, linear velocity, and finally linear acceleration is com-

puted using is computed using a first order approximation to the derivative, namely

$$\dot{\vec{\omega}} = \frac{d\vec{\omega}}{dt} \approx \frac{\vec{\omega}_t - \vec{\omega}_{t-1}}{\Delta t} \tag{4.67}$$

$$\vec{v} = \frac{d\vec{p}}{dt} \approx \frac{\vec{p}_t - \vec{p}_{t-1}}{\Delta t} \tag{4.68}$$

$$\vec{\alpha}_0 = \frac{d\vec{v}}{dt} \approx \frac{\vec{v}_t - \vec{v}_{t-1}}{\Delta t} \tag{4.69}$$

$$\tag{4.70}$$

The linear acceleration is biased by gravity to simulate real-world processes as

$$\vec{\alpha} = \vec{\alpha}_0 + \vec{g} \tag{4.71}$$

Now, to compute what each accelerometer senses,

$$a_j = \vec{n}_j \cdot (\vec{\alpha} + (\dot{\vec{\omega}} \times \vec{r}_j) + (\vec{\omega} \times (\vec{\omega} \times \vec{r}_j)) \tag{4.72}$$

is evaluated where a_j is the output of the j^{th} accelerometer, $\vec{n}_j$ is the direction of the sensitive axis of the j^{th} accelerometer, $\vec{r}_j$ is the position of the j^{th} accelerometer. Note that $\vec{n}_j$ and $\vec{r}_j$ are expressed in the inertial frame and change with the orientation of the device.

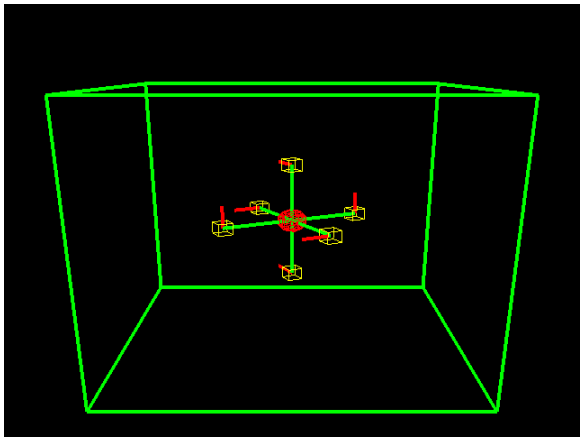


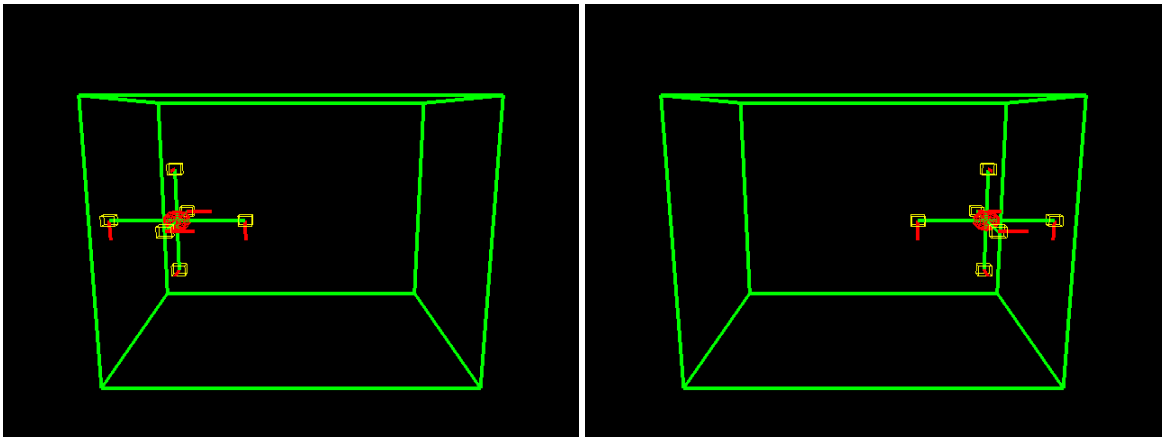
Figure 4.10: Screenshot of the Inertial Simulator. This shows the "virtual" inertial device with its six accelerometers at the appropriate positions. The red lines denote the direction of the sensitive axes of each accelerometer. The surrounding cube denotes the "virtual" IVY display and the red sphere in the centre of the device shows the position of the device.

4.6 Simulation Results

The inertial simulator is able to provide perfect data in order to verify that the method is correct. Also, the simulator is capable of simulating data from accelerometers that have misaligned axes as in the actual device. Several tests were performed which illustrate the correctness of the integration loop. For the following, results from purely translational motion and purely rotational motion will be shown. In each, the estimated data is shown with the ground truth data output from the simulation. Also, it is important to note that the Euler integration[59] method is used causing some numerical errors to occur in the resulting data.

4.6.1 Translation

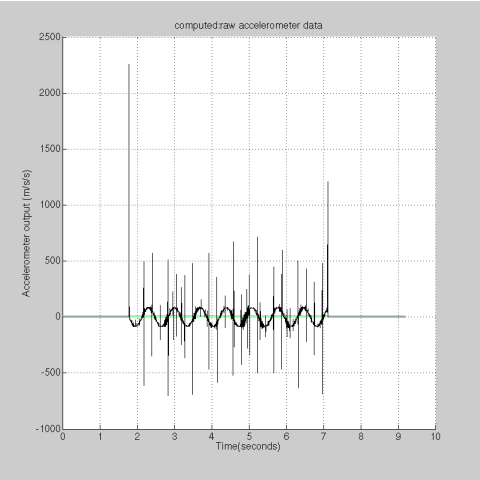
Two tests are shown that illustrate the correctness of the system during a purely translational movement. Data from ideal and perfectly aligned accelerometers are shown followed by a test with misaligned axes in order to illustrate the effects of accelerometer misalignment.



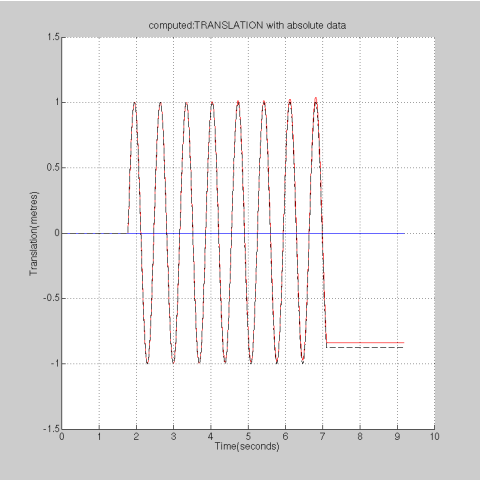
(a) Start Position (b) End Position

Figure 4.11: Inertial Simulator Translation. This illustrates the motion involved in the performed tests on translation. The device is moved from one side of the virtual IVY to the other side of the virtual IVY at constant velocity.

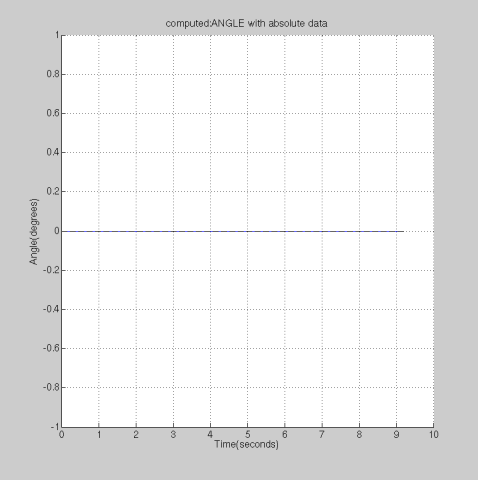
For this test, the device was oscillated sinusoidally along the X-axis. Figure 4.12(a) shows the raw data output from the inertial simulator. The integrated output shows a relative translation from the origin and no orientation change. The translation computed from the raw accelerometer data can be seen in Figure 4.12(b) plotted with ground truth data.



(a) Raw Accelerometer Data



(b) Estimated Translational Motion (dotted black line is ground truth, red line is estimated translation)



(c) Estimated Angular Motion

Figure 4.12: Simulation Results with Purely Translational Movement and Ideal Sensing Axes

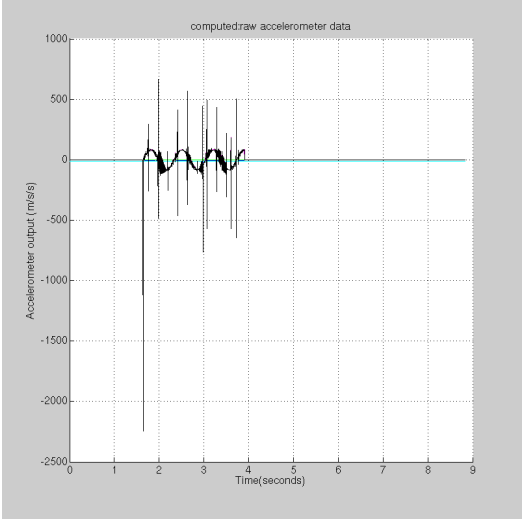
Ideal Translation with Misalignment

For this test, the accelerometer sensing axes were determined using the calibration results from the actual device. The sensing directions are defined in Table 4.1.

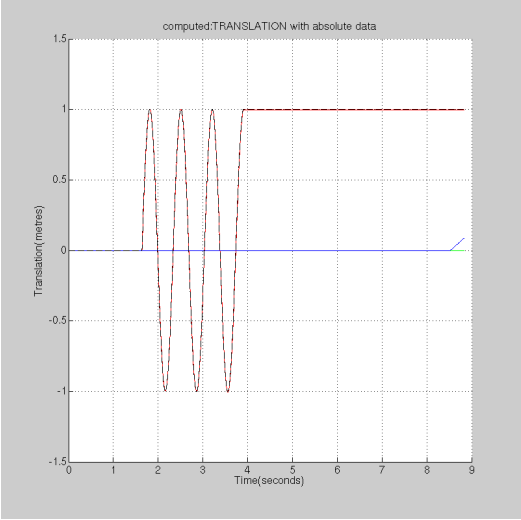
Accelerometer	X	Y	Z
1	0.004454	-0.025742	-0.999659
2	0.012620	0.000073	-0.999920
3	-0.016351	-0.999858	0.004054
4	0.000157	-0.999978	-0.006581
5	-0.999843	0.009090	-0.015183
6	-0.999651	-0.013739	-0.022575

Table 4.1: Accelerometer sensing directions with calibration from actual device ($\vec{\mathbf{n}}_j$).

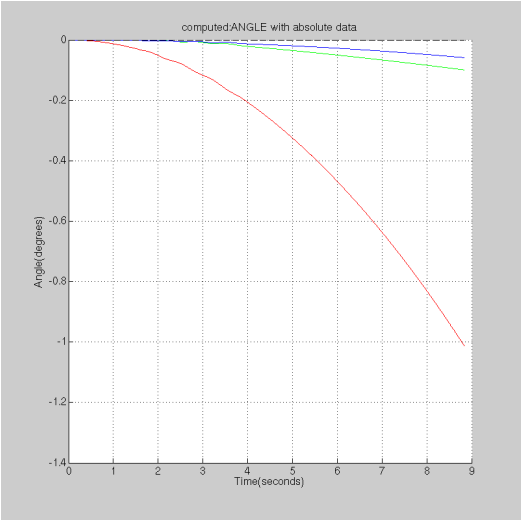
The previous test was repeated incorporating the misalignment of the accelerometers. The raw data produced from the simulator can be see in Figure 4.13(a) and computing the final pose from the raw data using the described method takes the misalignments into account to produce the proper result. The computed results can be seen in Figure 4.13(b). Numerical integration and roundoff errors occur and are accentuated by the use of Euler's method of integration. Since the system is tightly coupled, this error accumulates and biases the angular acceleration that is extracted resulting in a small error in angular motion over several seconds.



(a) Raw Accelerometer Data

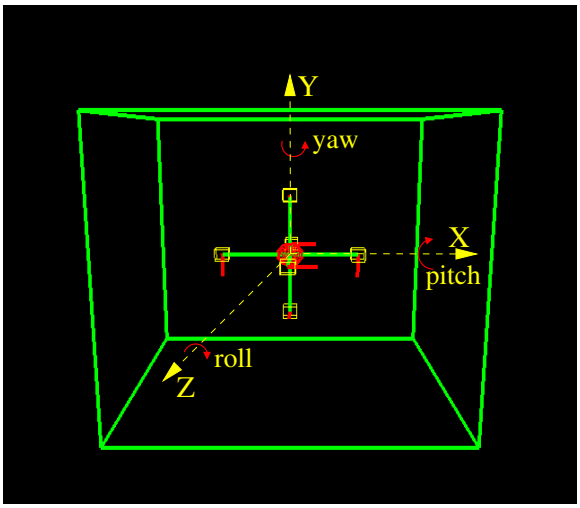


(b) Estimated Translational Motion (dotted black line is ground truth, red line is estimated translation)



(c) Estimated Angular Motion (dotted black line is ground truth, red, green, and blue correspond to yaw, pitch and roll angle respectively)

Figure 4.13: Simulation Results with Purely Translational Movement and Misaligned Sensing Axes



(a)

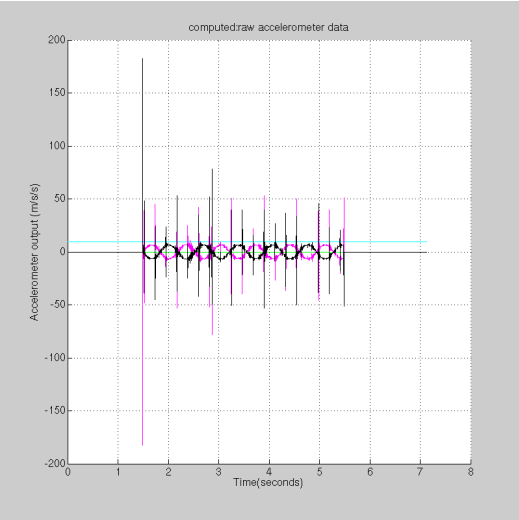
Figure 4.14: Inertial Simulator Rotation. This figure illustrates the axes of rotation performed with the inertial simulator.

4.6.2 Ideal Rotation

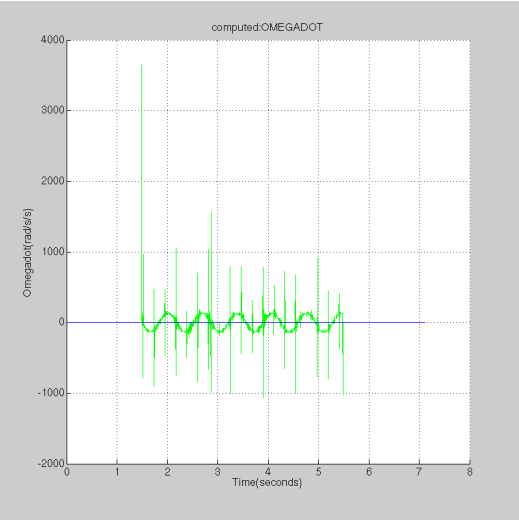
To test rotation, first the ideal sensing axes are simulated and the virtual device is rotated about each of its three axes; Yaw, Pitch and Roll. The device is rotated $\pm 90^o$ over several seconds on each axis and the accelerometer output is recorded.

As shown in Figure 4.14, the yaw rotation is about the world Y-axis of the inertial device. As can be seen during the movement, the computed angular motion follows the absolute motion curve precisely with small roundoff errors accumulating. The raw accelerometer data is shown in Figure 4.15(a) and the extracted angular acceleration can be seen in Figure 4.15(b). Through integration of the acceleration, the angular velocity is computed and plotted in Figure 4.15(c) which is also integrated to estimate the angular motion

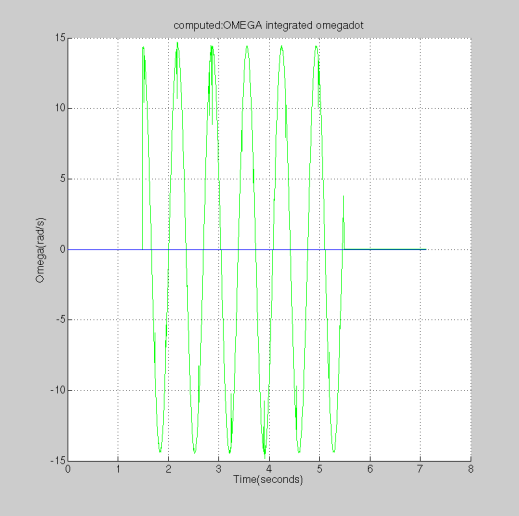
shown in Figure 4.15(d) and 4.16(c). For completeness the extracted linear acceleration is shown in Figure 4.16(a) which by double integration results in the translation shown in Figure 4.16(b). The same experiments were also conducted on the other two rotations (Pitch and Roll) and the results are summarized in Figure 4.17.



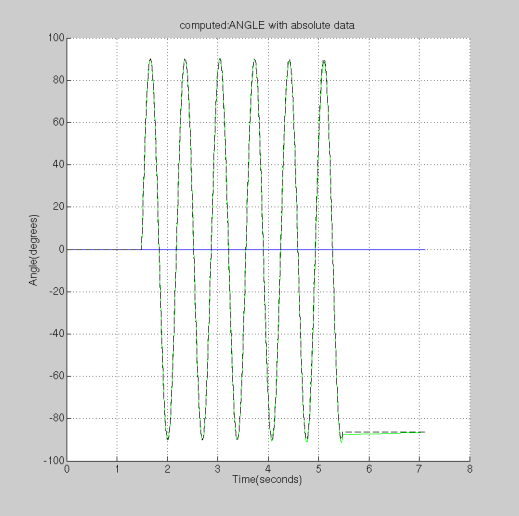
(a) Raw Accelerometer Data (all 6 accelerometers)



(b) Estimated Omegadot

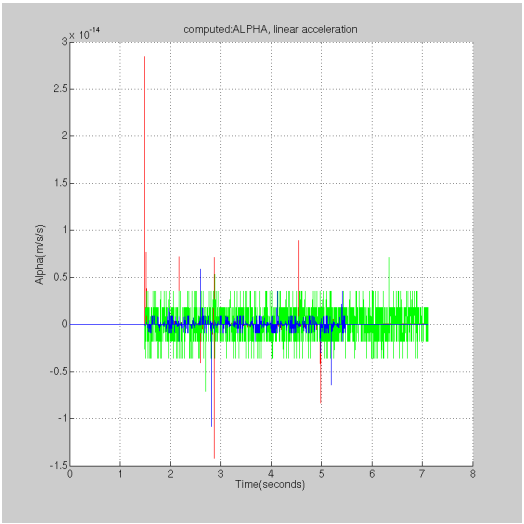


(c) Estimated Omega

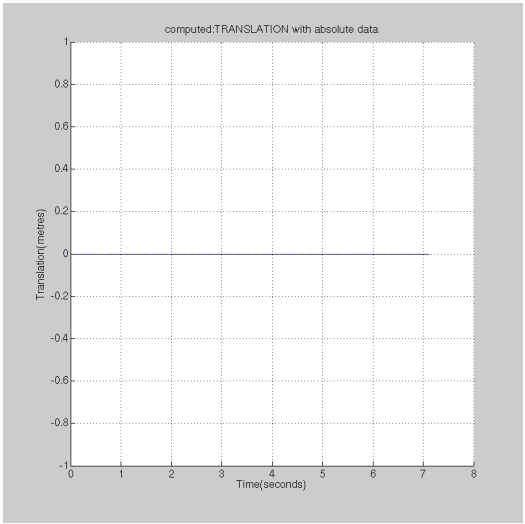


(d) Estimated Theta (black dotted line is ground truth, green line is estimated Yaw angle)

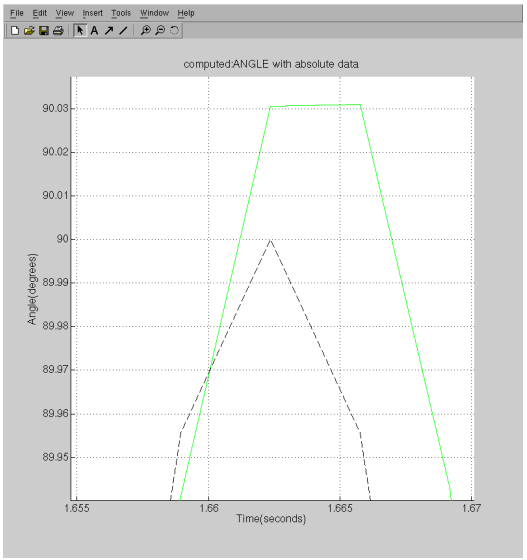
Figure 4.15: Simulation Results with Ideal Yaw Rotation



(a) Estimated Alpha

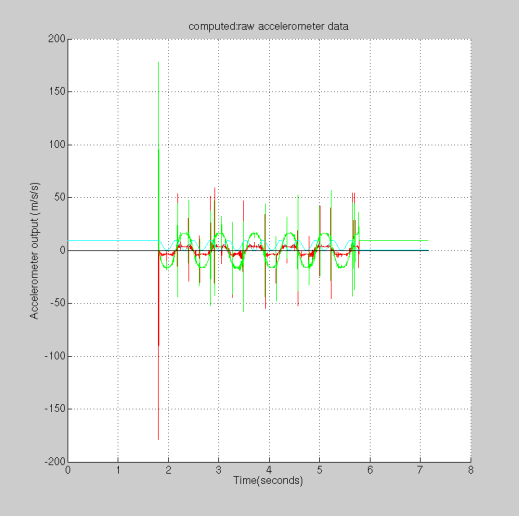


(b) Estimated Translational Movement

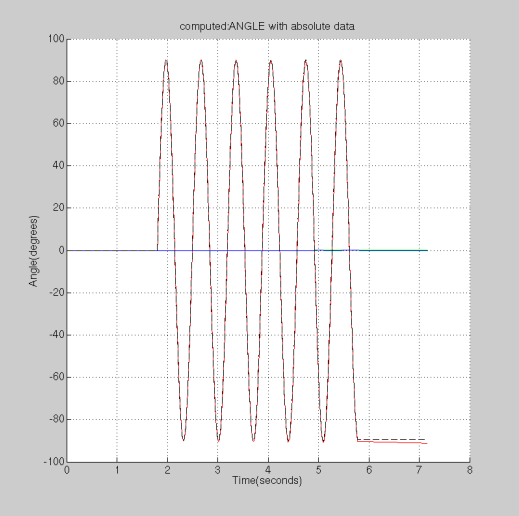


(c) Estimated Yaw Rotation Closeup (black dotted line is ground truth, green line is estimated Yaw angle)

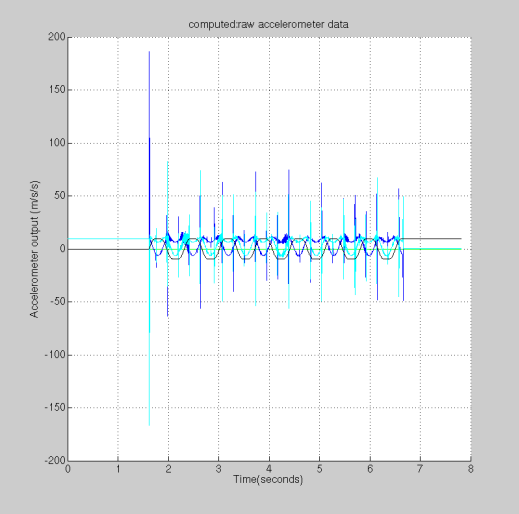
Figure 4.16: Simulation Results with Ideal Yaw Rotation (continued)



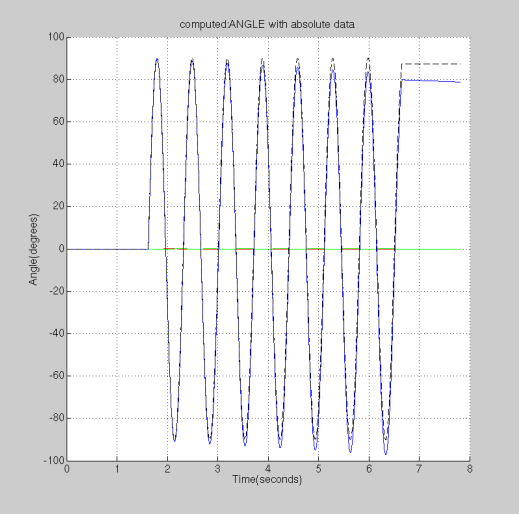
(a) Pitch:Raw Accelerometer Data (of all 6 accelerometers)



(b) Pitch:Estimated Theta (dotted black line is ground truth, red line is estimated Pitch angle)



(c) Roll:Raw Accelerometer Data (of all 6 accelerometers)



(d) Roll:Estimated Theta (dotted black line is ground truth, blue line is estimated Roll angle)

Figure 4.17: Simulation Results with Ideal Pitch and Roll Rotation Experiments

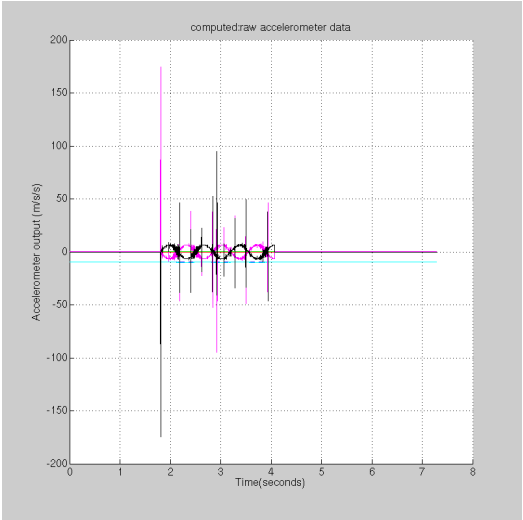
4.6.3 Ideal Rotation with Misalignment

The previously described rotation experiments were repeated with the calibration data from the physical device. The system is shown to work properly but with a slight increase in accumulative error due to the misalignments. The results for Yaw are shown in Figures 4.18 and 4.19 and the results from the Pitch and Roll experiments are summarized in Figure 4.20.

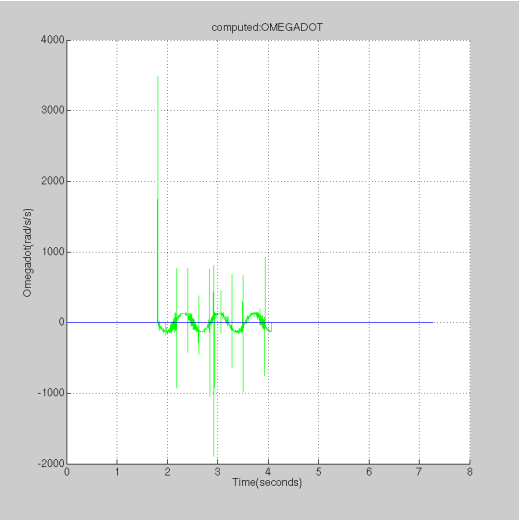
4.7 Summary

In this chapter, the theory behind an inertial pose estimation system was described that utilizes six accelerometers in a known geometry. The mathematics was developed for the acceleration of rigid bodies and was applied to the MARVIN inertial hardware. An inertial simulator was described which is an interactive program capable of simulating the acceleration that each accelerometer senses during a given motion of the virtual inertial device. It was shown that the system works in simulation with ideal data and that the misalignments of the accelerometer sensing directions can be compensated for through accurate calibration. The calibration method used in the MARVIN system was described and results were shown that illustrate the correctness of the pose estimation method. The math has shown to be extremely sensitive to noise in the raw data and misalignment of the accelerometer pairs. The system quickly diverges when noise is present, however with smoothing and filtering of the data the system can provide reliable pose estimates a small number of seconds. This indicates that a secondary system is needed that runs at an update rate of greater than

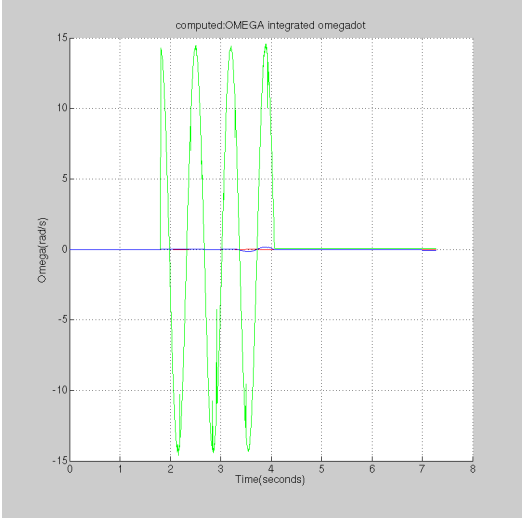
1Hz to reset the inertial system pose estimation loop. As discussed in Chapter 2, a hybrid tracking approach is desired.



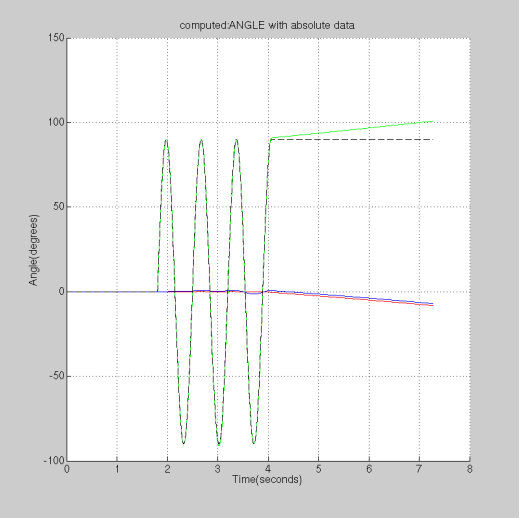
(a) Raw Accelerometer Data (of all 6 accelerometers)



(b) Estimated Omegadot

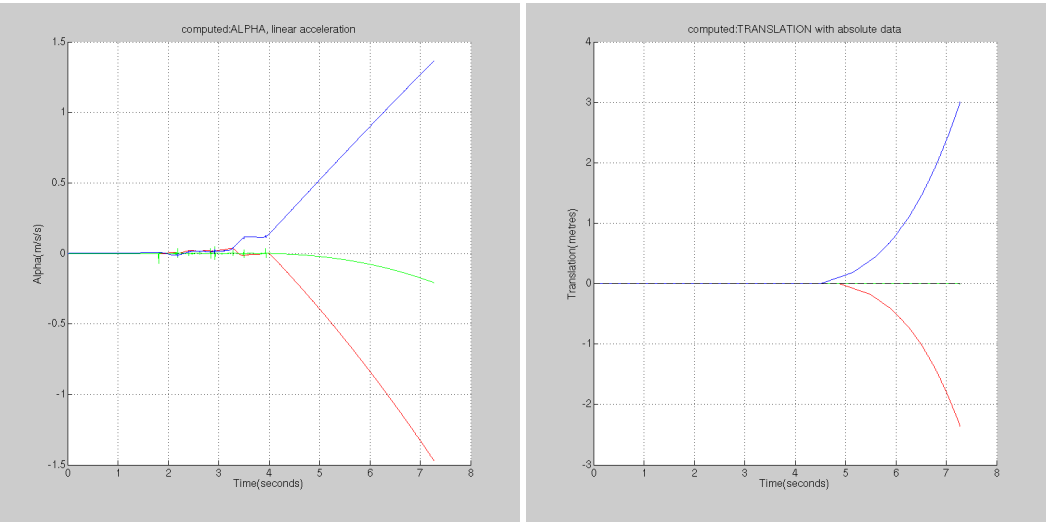


(c) Estimated Omega



(d) Estimated Theta (black dotted line is ground truth, green line is estimated Yaw angle)

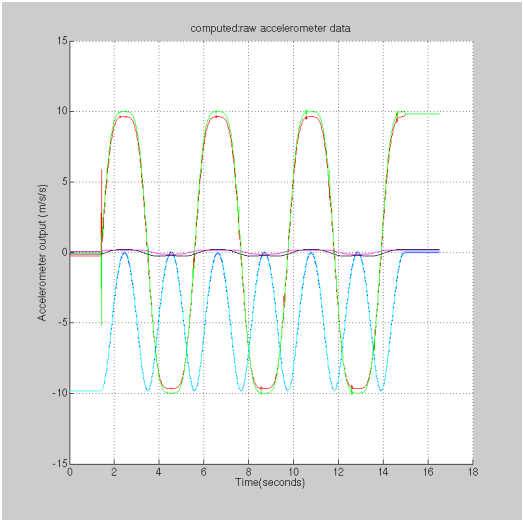
Figure 4.18: Simulation Results with Yaw Rotation and Misaligned Sensing Axes



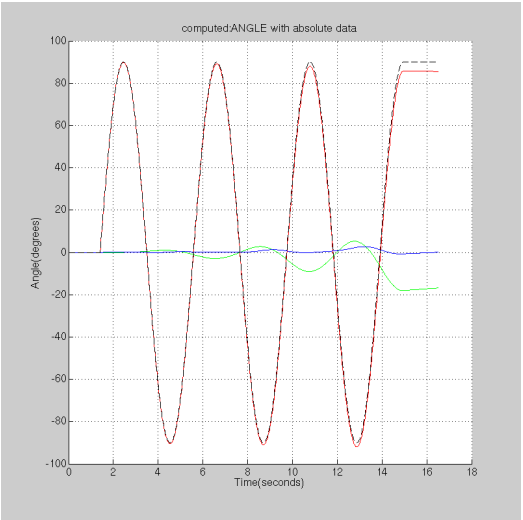
(a) Estimated Alpha (red, green, blue are linear accelerations allong the X,Y,Z axes respectively)

(b) Estimated Translational Movement (red, green, blue are linear translations allong the X,Y,Z axes respectively)

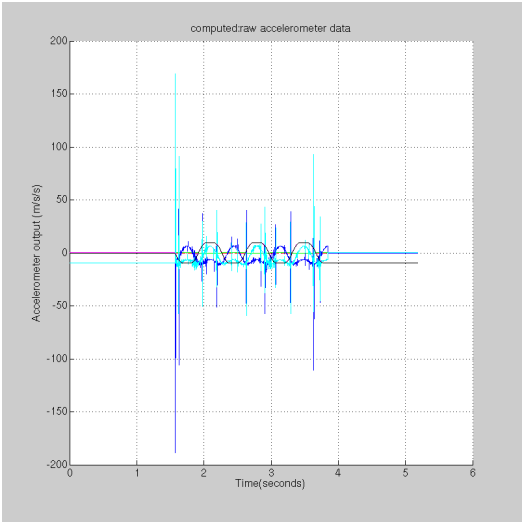
Figure 4.19: Simulation Results with Yaw Rotation and Misaligned Sensing Axes



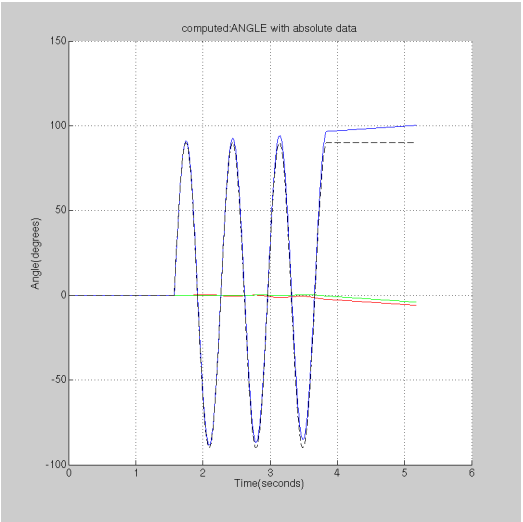
(a) Pitch:Raw Accelerometer Data (of all 6 accelerometers)



(b) Pitch:Estimated Theta (dotted black line is ground truth, red, green and blue are the Pitch, Yaw and Roll angles respectively)



(c) Roll:Raw Accelerometer Data (of all 6 accelerometers)



(d) Roll:Estimated Theta (dotted black line is ground truth, red, green and blue are the Pitch, Yaw and Roll angles respectively)

Figure 4.20: Simulation Results with Pitch and Roll Rotation and Misaligned Sensing Axes

Chapter 5

The Optical System

No current optical head-tracking solution is suitable for use in fully-enclosed immersive projective displays. Commercially available optical head tracking systems require a line-of-sight to the user or the placement of known landmarks in the scene. In fully-enclosed environments, known landmarks cannot be placed in the environment as they will interfere with the user's visual experience. The line-of-sight constraint for these trackers is violated since the user is within a fully-enclosed volume. The optical component of the MARVIN tracking system uses this line-of-sight restriction to its advantage. This chapter develops the theory and implementation behind the optical tracking component of MARVIN.

MARVIN utilizes digital video cameras placed outside of the immersive display, positioned to view the rear-projection screens. By attaching a group of low-power laser diodes to a helmet worn by the user, the user's head motion can be tracked. The lasers project onto the screen surfaces of the immersive display and since the screen surface is translucent, the laser light is visible from the projector/camera side of the display. By tracking the laser projections in camera space and positioning the lasers in a known geometry, it is possible to obtain strong constraints on the position and orientation of the tracking device. These constraints allow for the computation of the correct head pose from four tracked laser projections. Figure 5.1 illustrates the basic approach.

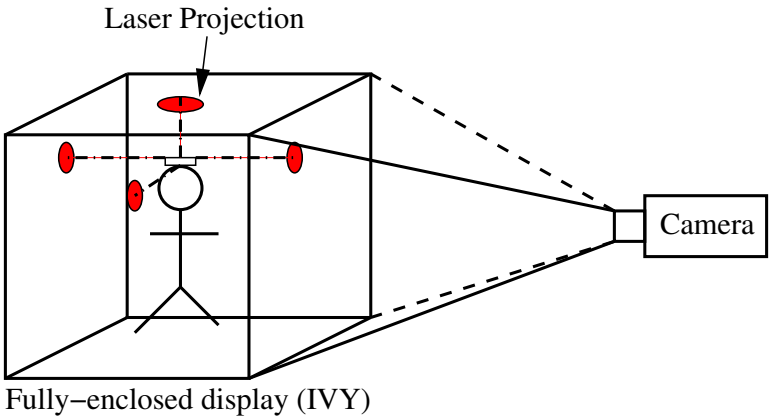


Figure 5.1: Optical tracking system basic approach. Cameras are located outside of the display and track the projections of laser emitters attached to the user's head on the display surfaces.

5.1 Basic Approach

MARVIN utilizes a novel *outside-in* optical technique to track the user's head motion. Digital FireWire® (IEEE 1394) cameras are placed outside of the fully-enclosed display and each is positioned to view one of the rear-projection screen surfaces that make up IVY. In the current implementation, eight cameras are used since the floor and ceiling of IVY required two projectors (and thus two cameras) for each surface due to physical space constraints. A group of low power laser diodes are attached to the user's head, positioned so that the emitted laser beams (and more importantly the projections of these beams onto the surfaces of IVY) cannot be seen by the user. Each camera is equipped with a wavelength notch filter with a peak response at the laser diode wavelength (650nm). This simplifies image processing and enables real-time performance. The image from each camera is analyzed to determine the centroid of the laser projections on the screen surfaces. When

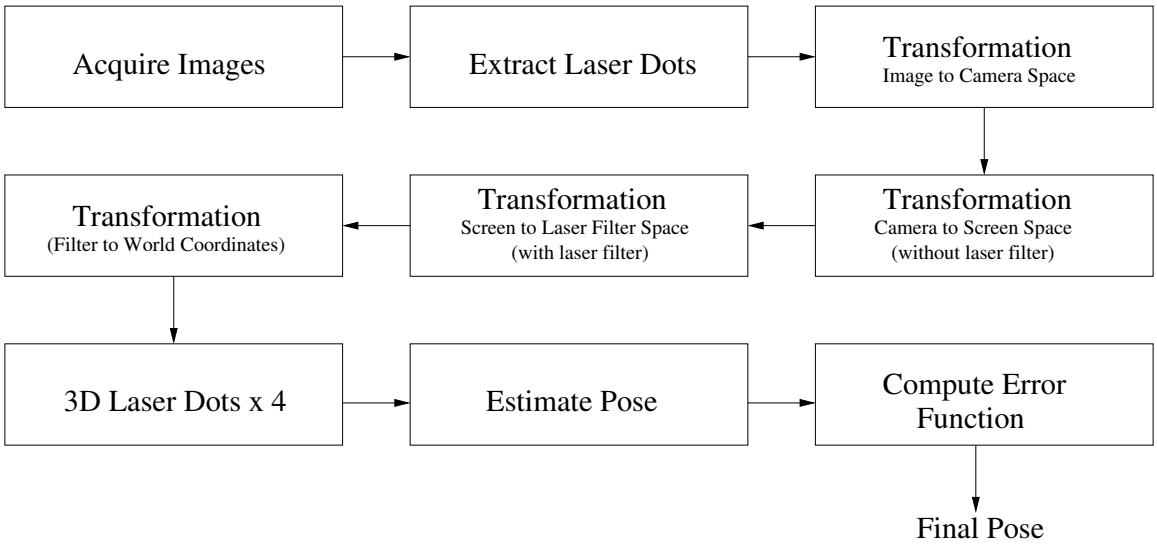


Figure 5.2: Optical tracker algorithm. Each step is explained in detail throughout this chapter.

four laser projections are found, they are transformed into world-coordinates (determined through an offline calibration routine) and are used to estimate the pose of the tracking device. By imposing constraints based on the known geometry of the laser diode housing, the position and orientation is computed from the tracked laser projections. A recursive least-squares (extended Kalman) filter is used to smooth, gate, and maintain the final pose of the user. The entire algorithm is summarized in Figure 5.2 and each component is explained in detail in this chapter.

The optical tracking method is decomposed into parts as

- Calibration of the system (performed once and discussed later this chapter).
- Laser dot localization in camera images. Note that more than one laser projection can exist on a single wall.

- Pose estimation using the tracked laser dots and geometrical constraints.
- Maintain pose estimate via temporal coherence.

5.2 Tracking Laser Projections

Tracking a laser projection within an image has been used before in the human-computer-interaction field[57, 79, 12, 56]. Techniques used to identify the lasers in the images include pattern recognition techniques and convolution filters, colour identification, brightest peak detection, and fitting mathematical functions to areas containing bright pixels to compute the subpixel coordinates of the centroid (as in [20]). In practice, convolution techniques can be optimized to run in real-time (or implemented in hardware) for a single image.

MARVIN requires the use of eight digital cameras. Thus, a computationally inexpensive image processing routine is needed. An algorithm that requires a single pass through each grayscale image is preferred over a colour-based approach for two reasons. First, colour images require three times the amount of data to be transferred over the computer's PCI bus quickly exhausting the bandwidth of standard PCs. A second issue is that the projected laser point is bright enough to saturate the CCD and washes out the colour of the imaged laser dot. In order to simplify the required image processing, a wavelength notch filter on each camera is used to ensure that only laser light (650nm wavelength) is visible in the camera image even in the presence of bright imagery being projected on the screens. This was employed for robustness, to ensure that no false positives due to bright light emit-

ted from the projectors are mistaken as laser dots, and to reduce the amount of data that needs to be acquired and transmitted by the camera. Since the cameras acquire 640x480 resolution images, the 2.29m screen is imaged at approximately 500 pixels, making 1 camera pixel correspond to approximately 0.5cm on the screen surface. This suggests that a highly accurate technique is required to find and track the laser projections throughout the images, obtaining subpixel precision whenever possible.

5.2.1 Grouping and Locating Laser Dots

First, a single-pass pseudo-propagation algorithm is performed on a per image basis to find and label pixels associated with potential laser dots. It is assumed that potential laser illuminated pixels cannot be closer than δx pixels from the illuminated pixels associated with a different laser emitter. This is a reasonable assumption due to the physical geometry of the laser diodes; two lasers are always a minimum of 30 pixels apart, thus setting $\delta x < 30$ is sufficient. For a laser illuminated pixel at image point (u,v), a neighbouring region of δx pixels in a labelling array is examined (i.e. the region $u-\delta x \ldots u+\delta x, v-\delta x \ldots v+\delta x$ is examined) to determine if the laser illuminated pixel should be associated with an already identified laser dot. If no such association is found, the pixel is associated with a new laser emitter identifier (id).

Given the id of a found laser dot, compute

$$u_{id} = \sum_{i-\delta x, j-\delta x}^{i+\delta x, j+\delta x} i * I(i,j) \tag{5.1}$$

$$v_{id} = \sum_{i-\delta x, j-\delta x}^{i+\delta x, j+\delta x} j * I(i,j) \tag{5.2}$$

$$I_{id} = \sum_{i-\delta x, j-\delta x}^{i+\delta x, j+\delta x} I(i,j) \tag{5.3}$$

Alternatively, this computation could be implemented as in [20] by fitting an exponential (Gaussian) function to the intensities to acquire the subpixel peak of the laser spot, however due to the size of the laser dot in the images (sometimes down to a very small number of pixels) it was found that an intensity-weighted geometric centroid computation is sufficient.

Performing this calculation for the entire image, an array of potential laser dots is produced and their respective (u,v) locations are weighted by their intensities. The estimate $(\hat{u}_{id}, \hat{v}_{id})$ is computed using the values in Equations5.1 to 5.3 as

$$\hat{u}_{id} = \frac{u_{id}}{I_{id}} \tag{5.4}$$

$$\hat{v}_{id} = \frac{v_{id}}{I_{id}} \tag{5.5}$$

Note that if the intensities are uniform throughout the analysed image region, the computed laser centre is the geometric centroid of the region. It is possible that spurious "novel" points will be identified by this process. To eliminate such false positives, the distance be-

Listing 5.1: Algorithm to find the laser dots in all 8 images

```
function findLasersInIVY(Image Images[8])
        /* returns LASER_DOTS array of 2D points */
        for ALL img← 1...8
                LASER_DOTS[img] = getLaserDots(Images(img));

        return LASER_DOTS;
end function
```

tween laser dot clusters are computed. If the squared distance is beneath some empirically found threshold, then the estimates are merged. This is reasonable since there are a small number of potential laser dots at any given time, typically a list of less than ten potential candidates. Listings 5.1 through 5.4 summarize the approach.

The labelling/merging algorithm finds multiple laser dots in each image with subpixel estimation. The method is quite robust and requires only one pass through the image. Figure 5.3 shows a screenshot of the algorithm tracking 3 laser dots.

Listing 5.2: Algorithm to find the laser dots in a single image

```
function getLaserDots(Image I, int Threshold, int Neighbourhood)
        /* returns LASERS array of 2D points */
        /* find all potential lasers in the image */
        for ALL u←1...I_width, v←1...I_height
                if I(u,v) > Threshold
                        /* we have a potential laser */
                        /* lets store its ID and info*/
                        ID = getLaserID(u,v,Neighbourhood);
                        ID = setLaserID(u,v,ID);

                        LASERS(ID).x += I(u,v)*u;
                        LASERS(ID).y += I(u,v)*v;
                        LASERS(ID).weight += I(u,v);

        /* use weighting to compute proper centroid */
        for ALL ID ←1...NUM_IDS
                LASERS(ID).x /= LASERS(ID).weight;
                LASERS(ID).y /= LASERS(ID).weight;

        /* merge all candidates within δ_x pixels of each laser */
        for ALL i←1...NUM_IDS,
                for each j←i+1... NUM_IDS
                        if (DISTANCE(LASERS(i),LASERS(j)) < δ_x)
                                LASERS(i).x += LASERS(j).x;
                                LASERS(i).y += LASERS(j).y;
                                LASERS(i).x /= 2;
                                LASERS(i).y /= 2;
                                REMOVE(LASERS(j));
        RESET_IDS();
        return LASERS;
end function
```

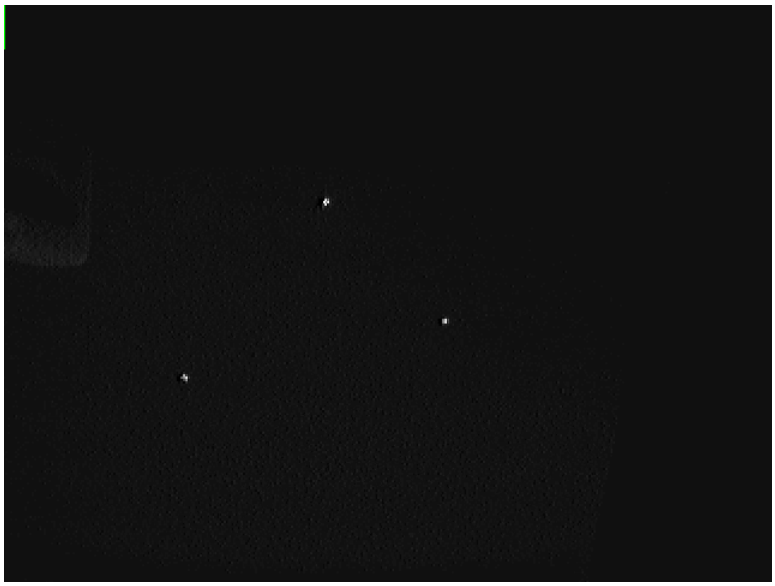
Listing 5.3: Algorithm to return a valid ID from the labelling array

```
function getLaserID(int u, int v, int N)
        /* returns MAXID */
        /* the maximum ID found in the neighbourhood */
        MAXID = -1;
        ID = 0;
        for ALL U ←u−N/2...u+N/2, V ←v−N/2...v+N/2
                if (IDS(U,V) != 0)
                        ID = IDS(U,V);
                        if (ID > MAXID)
                                ID = IDS(U,V)
        return MAXID;
end function
```

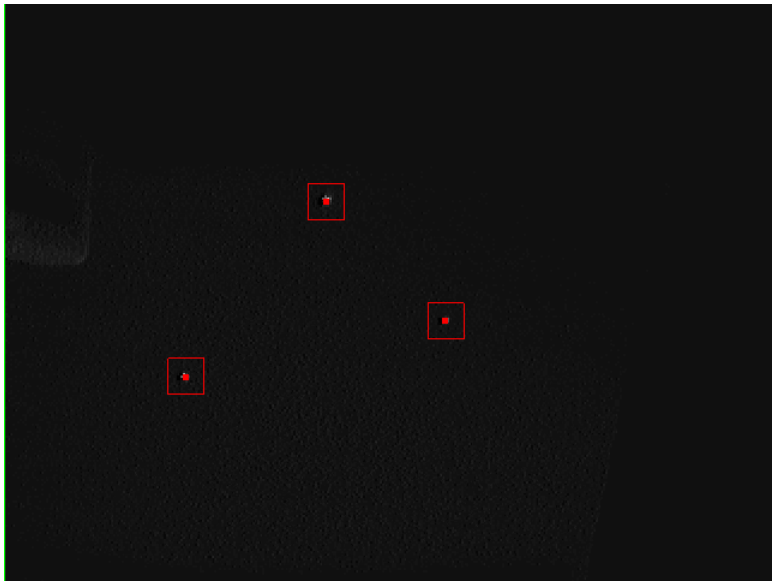
Listing 5.4: Algorithm to set the ID in the labelling array

```
function setLaserId(int u, int v, int ID)
        /* returns ID2 the current id */
        ID2 = ID;
        if (ID2 == -1)
                IDS(u,v) = NEXT_ID;
                ID2 = NEXT_ID;
                NEXT_ID++;
        else
                IDS(u,v) = ID2;

        return ID2;
end function
```



(a) Raw intensity image of 3 laser dots



(b) System tracking the lasers online

Figure 5.3: Multiple Laser Dot Tracking. (a) shows the raw image taken with 3 lasers in the image and (b) shows the system finding the centroid. The red square surrounding the tracked dot denotes the tracking region (enlarged for viewing purposes).

5.3 Estimating Position and Orientation

Given that all four lasers are tracked accurately on the screen surfaces, it is possible to compute the position and orientation of the tracking device. Each laser diode in the tracking device is identical in specifications (see Appendix C) and all are connected to a common constant battery power source. Since each laser dot is the same color, intensity, and shape it is not possible to distinguish between them (identify the laser diode that emitted the beam) with image processing techniques. It is possible to distinguish the laser dots using frequency modulation, wavelength differences, or using optical filters that create patterns from a laser dot, however, exploiting the known geometry of the lasers is strong enough to determine the pose of the device.

Various configurations of laser diodes could be used to localize the user. MARVIN uses a simple arrangement of four laser diodes in the geometric configuration shown in Figure 5.4. Two of the laser diodes are arranged so they project in opposite directions along a line, and the other two diodes are arranged so they project orthogonal to each other and orthogonal to this line. The projection directions of all four laser diodes intersect at a single point, P_0. Given the projections of the four laser diodes on the exterior walls of the environment it is possible to obtain strong constraints on P_0 and to define a 3D coordinate frame aligned with this point.

To demonstrate this, the problem is broken down into two parts. The first is to determine P_0 and the coordinate system with origin P_0 given four laser points and the second is to

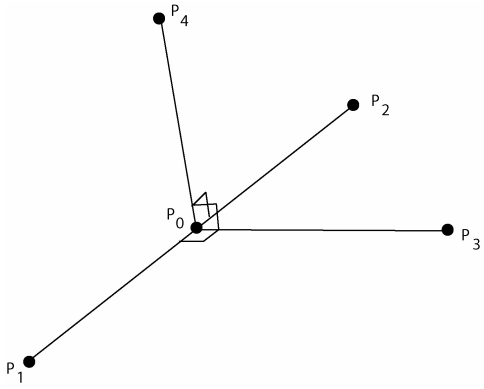


Figure 5.4: Basic laser geometry. The four lasers are established so that lines drawn through their beams would intersect at a common point P_0, and $\vec{P_3P_0} \cdot \vec{P_1P_2} = \vec{P_4P_0} \cdot \vec{P_1P_2} = \vec{P_3P_0} \cdot \vec{P_4P_0} = 0$.

determine a mapping between laser projections and laser emitters (i.e. a proper labelling of projections and emitters) to ensure that the computed pose is correct.

For the remainder of this discussion, let $P_1 ... P_4$ be the 3D positions at which the laser beams from the respective laser diodes strike the walls of the environment (i.e. P_1 is the location at which the beam from laser 1 struck the wall of IVY, P_2 is the location of laser 2, etc.). Also let $\overrightarrow{ab}$ denote a vector through points a and b determined by $\overrightarrow{ab} = \vec{b} - \vec{a}$. P_0 lies at the intersection of $\overrightarrow{P_1P_2}$ with a perpendicular line that passes through point P_3 (see Figure 5.4). This point can be found by noting that P_0 lies along the line defined by $P_1 + \lambda(P_2 - P_1)$ and $\overrightarrow{P_1P_2} \cdot \overrightarrow{P_0P_3} = 0$. Solving these equations for P_0 yields

$$P_0 = P_1 + \frac{(P_3 - P_1) \cdot (P_2 - P_1)}{||P_2 - P_1||^2}(P_2 - P_1)$$

This defines the origin of the frame, $\overrightarrow{P_0P_3}$ defines the direction vector for the frame, and

the normal of the plane defined by points $\{P_1, P_2, P_3\}$, $\vec{n} = \overrightarrow{P_0P_1} \times \overrightarrow{P_0P_3}$, determines the direction of the "up" vector. Although P_4 is not required in order to compute this frame (provided that the assignment of laser spots to diodes is known), P_4 will prove useful by providing an additional constraint for resolving pose ambiguities. In terms of the geometry it is important to note that $\overrightarrow{P_0P_4}$ is perpendicular to the plane defined by $\{P_1, P_2, P_3\}$ and that $\vec{n} \cdot (\overrightarrow{P_0P_1} \times \overrightarrow{P_0P_3}) > 0$.

These calculations require that the correspondence between each laser diode and each laser projection are known. In practice this may be accomplished using different wavelengths, or pulsing the lasers at known times. In the current implementation, the geometry is used to place constraints on the finite number of possible mappings between emitters and points. The mapping that minimizes an error function is chosen as the correct configuration. The appropriate labelings of the tracked laser projections P_i, P_j, P_k, and P_l with the actual laser points P_1, P_2, P_3, and P_4 must be determined. There are 24 (4!) possible assignments of the laser points to the emitters. Of all 24 possible assignments, only four are consistent with the geometry of the emitters. Figure 5.6 shows examples of the possible labelings and the impact this has on the pose computation.

Although there are four configurations that are consistent with the geometry of the laser diodes, two of these incorrect assignments only occur in extreme conditions and all are easily disambiguated using temporal coherence. If the correct assignment is $(P_i, P_j, P_k, P_l) \rightarrow (P_1, P_2, P_3, P_4)$, then the three incorrect assignments are

1. $(P_i, P_j, P_k, P_l) \rightarrow (P_2, P_1, P_4, P_3)$. This configuration has the same P_0 as the correct

configuration, but is a reflection of the correct configuration. With a 15Hz sampling rate, the user is unable to perform the required head rotation between the two configurations and temporal coherence constraints can be used.

2. $(P_i, P_j, P_k, P_l) \rightarrow (P_3, P_4, P_2, P_1)$. This incorrect assignment and the final remaining assignment have a different P_0, and an orientation change of at least 90 degrees. This configuration, like the following configuration, is extremely unstable and can only occur under extremely unusual conditions. With a 15Hz sampling rate, the user would have to rotate at roughly 675 deg/sec before this configuration can be confused with the correct one.

3. $(P_i, P_j, P_k, P_l) \rightarrow (P_4, P_3, P_1, P_2)$. This incorrect assignment is similar to the one above. It has a different P_0 as well as at least a 90 degree orientation change.

A simple temporal tracking system coupled with gating is used to discard these incorrect assignments. Although enforcing these constraints maintains a consistent pose, there is still an issue of estimating the initial pose. In practice, this is accomplished by having the user move to an approximately known pose during system initialization (knowing that the user is upright and looking towards a given wall is sufficient to disambiguate the correct pose from the other potential distractors).

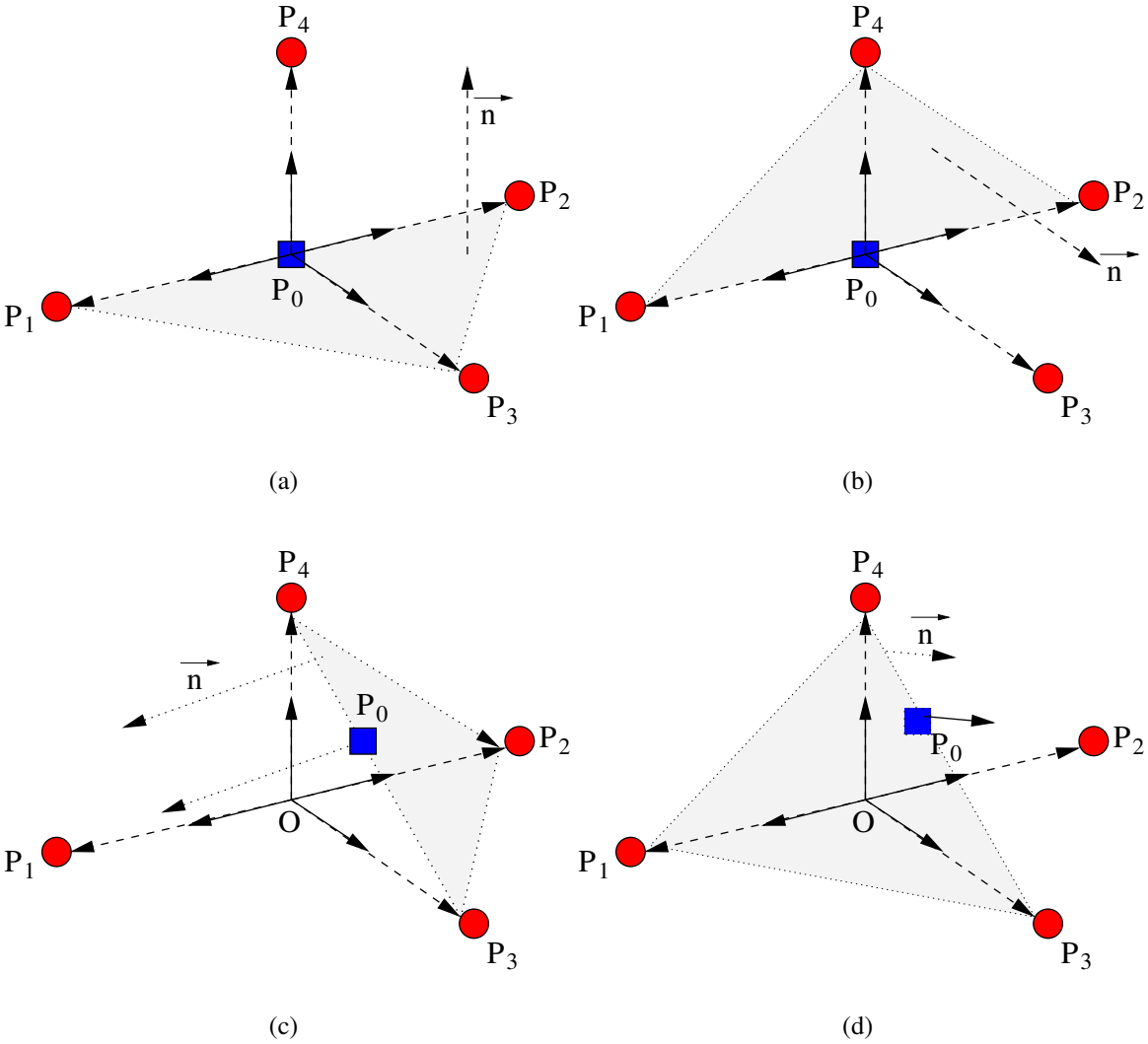


Figure 5.5: These are the four assignments that satisfy the static constraints. (a) is the correct assignment while (b),(c), and (d) show the pose computed from incorrectly assigned laser projections. The blue square labelled P_0 is the estimated position from the laser projections and the actual laser diode directions are shown using solid lines with arrows. The dotted lines with arrows illustrate the laser beams and the red circles are the intersections of the laser beams with the walls of the immersive display. The shaded polygon illustrates which points were used in this labelling to compute the plane normal, the computed $\vec{n}$ also shown here. The dotted line emanating from P_0 parallel to $\vec{n}$ is the computed $\vec{\text{Up}}$ vector. (c) and (d) are extremely unstable conditions and only occur when $|OP_4| = |OP_3|$ and $|OP_1| = |OP_2|$ which cannot occur due in practice due to numerical precision.

5.3.1 Discarding Invalid Configurations

Once the four tracked 2D laser points are available, it is possible to determine the pose of the person being tracked. To choose among the 24 possible labelings of the 4 laser dots, geometric constraints are imposed on the solution and an error function, $\varepsilon(i)$, is computed to determine the goodness-of-fit of the labeling. Using three constraints, the correct solution (up to a reflection, see Figure 5.6(b)) that corresponds properly to the pose of the device can be determined.

The error function is defined as

$$\varepsilon(i) = \varepsilon_{\perp}(i) + D(i) + \mathcal{F}(i) \tag{5.6}$$

where i is the current permutation of laser points, $\varepsilon_{\perp}(i)$ (the perpendicular error) is the sum of the dot products of vectors that should be perpendicular in this configuration (the absolute value of each dot product ensures an increasing function), and $D(i)$ is the shortest distance between the computed position P_0^i and $P_0^{i'}$ where $P_0^{i'}$ is computed using $P_4^i P_2^i P_1^i$ rather than $P_3^i P_2^i P_1^i$. This will eliminate many of the incorrect labelings since P_0^i and $P_0^{i'}$ will not coincide if the plane normal is computed with the incorrect points. $\mathcal{F}(i)$ is a binary function that returns 1 only when the computed plane normal is not in the same direction

as P_4^i. The perpendicular error, $\varepsilon_\perp(i)$ is defined as

$$\begin{aligned}\varepsilon_\perp(i) =& |(P_0^i P_4^i \cdot P_0^i P_1^i)| + |(P_0^i P_4^i \cdot P_0^i P_2^i)| + \\ & |(P_0^i P_4^i \cdot P_0^i P_3^i)| + |(P_0^i P_1^i \cdot P_0^i P_3^i)| + \\ & |(P_0^i P_2^i \cdot P_0^i P_3^i)|\end{aligned} \tag{5.7}$$

$D(i)$ is defined as

$$D(i) = ||P_0^i - P_0^{i'}||^2 \tag{5.8}$$

and

$$\mathcal{F}(i) = \begin{bmatrix} 0 & \vec{n} \cdot (P_4^i - P_0^i) > 0 \\ 1 & otherwise \end{bmatrix} \tag{5.9}$$

After evaluating $\varepsilon(i)$ for each possible labeling, the results are sorted in ascending order according to this error function and the first 2 solutions are taken as the correct pose and its reflection. Given a correct mapping at the previous time step, distinguishing between the two stable solutions is accomplished by noting that to confuse the correct mapping with the incorrect one, the user would have to rotate over 180^o. Given a 15Hz update rate, it is sufficient to check the angle between the normal in the previous computations and the normals computed in the two remaining labellings.

Since quaternions are used to represent the orientation of the user, an error quaternion can be computed (see Appendix A) between the previous time step and the two possible orientations at the current time step. Given the correct quaternion at the previous time step,

$\hat{q}_{t-1}$, and the two possible orientations $\hat{q}_1, \hat{q}_2$, the error quaternions can be computed by

$$\hat{q}_{\varepsilon 1} = \hat{q}_{t-1}\hat{q}_1^{-1} \tag{5.10}$$

$$\hat{q}_{\varepsilon 2} = \hat{q}_{t-1}\hat{q}_2^{-1} \tag{5.11}$$

$$\tag{5.12}$$

and the correct orientation is determined as the error quaternion with the smallest associated rotation angle, i.e. take the orientation with the smallest $cos^{-1}(\hat{q}_{\varepsilon i}[0])$.

5.3.2 Optical System Simulator

To verify that this method worked in practice, a simulation environment was developed that simulates the tracking device with known orientation inside the fully-enclosed display. The simulator intersects the simulated rays of each laser with each wall and draws the 2D projection point on the wall. The simulator then re-computes the orientation using only these points with the above calculation. A plane representing the device's orientation, the estimated position, the estimated up vector, and the estimated direction vector are displayed. Different techniques for computing the pose of the device could be easily tested without having to set up the entire physical system every time an adjustment is made. Several configurations of the laser diodes could be tested without building a new physical hardware prototype for every adjustment. The same code library was used in the simulator as in the tracking driver, allowing the software prototype to be ported to the working system easily.

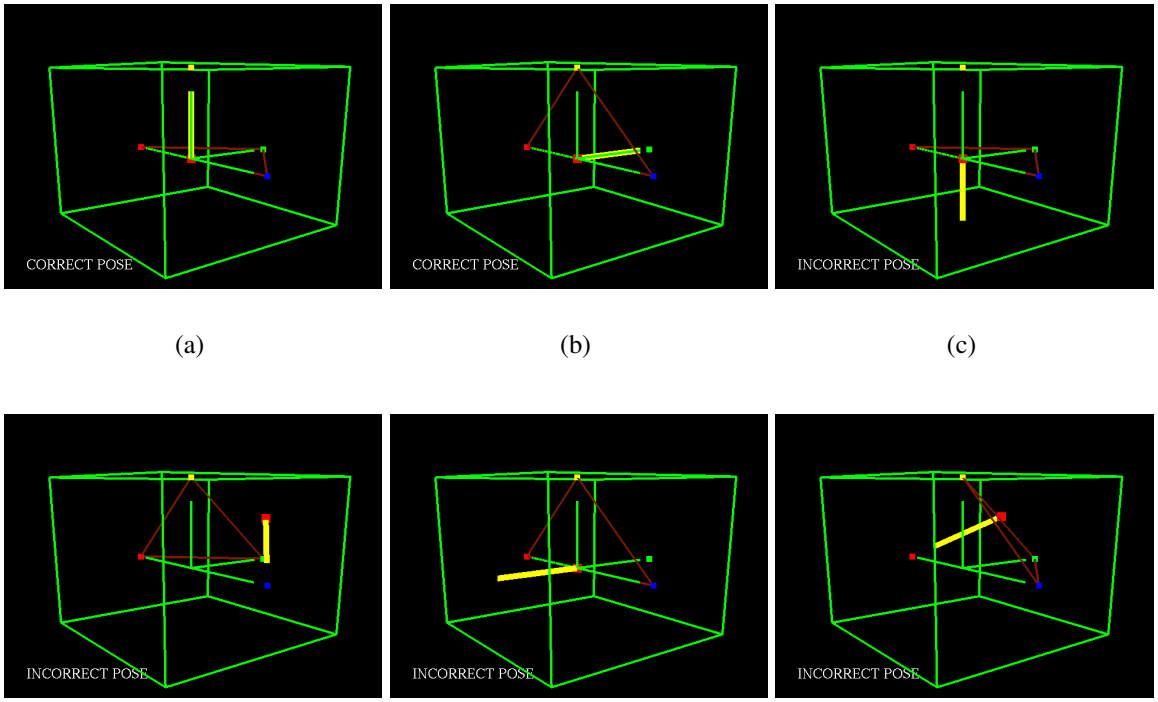


Figure 5.6: Examples of the 24 possible labelings and their associated computed pose. Shown here are screenshots from a simulator designed to test the configuration constraints on the laser geometry. The surrounding cube is an analogue of IVY while the smaller dots on the sides of the cube are the associated laser projections. The thick line (shown in yellow) is the computed *Up* vector, and the computed position is the large dot (shown in red). The connecting lines between laser points indicate which lasers were used to compute the plane for orientation. Each image is labeled with the text "CORRECT POSE" or "INCORRECT POSE" which is automatically computed using only static constraints. (a) is the correctly computed pose while (b) is incorrect but cannot be distinguished using only static constraints. A simple temporal mechanism is used to distinguish between these two solutions.

5.4 Calibration

In order to determine the user's head pose accurately, the transformations from camera space to world space must be determined. Due to the large physical area covered by each camera, it is absolutely essential to be as precise as possible.

The calibration method must:

1. be easy to perform and take a minimal amount of time to recalibrate the entire system in the event that the mirrors, screens, or cameras are moved.

2. be precise (within 1 pixel error).

5.4.1 Calibration Method

Calibrating the optical tracking system is decomposed into four steps:

1. Find the Intrinsic parameters: Camera Calibration.

2. Find the Extrinsic parameters 1: Camera to Screen Transformation.

3. Find the Extrinsic parameters 2: Screen to Laser Filter Transformation.

4. Find the Extrinsic parameters 3: Filter to World Transformation.

Intrinsic Calibration: Camera Calibration

A complete review of camera models and camera calibration is beyond the scope of this thesis, see [29, 19, 33, 32, 40] for more details. However, the camera models and calibration

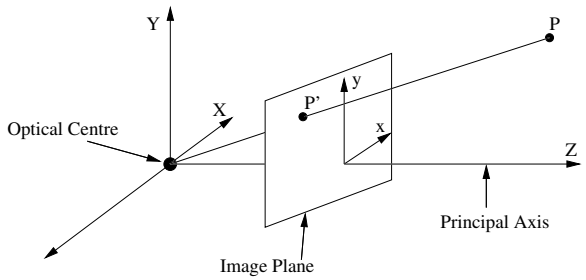


Figure 5.7: Basic Pinhole Camera Geometry.

procedures pertaining to this work will be discussed in this section. A camera is a device that performs a mapping from 3D points in space to 2D points on an image plane. There are many different ways this mapping can be modelled (see [29, 19]). The basic pinhole camera model (Figure 5.7) maps a point in space $X = [x, y, z]^T$ to a 2D point in the image, (u, v). The 2D point is defined as the intersection between a ray, determined by the center of projection of the camera and the scene point, and the plane defined by the focal length, i.e. plane $Z = f$.

$$(x, y, z)^T \rightarrow (\frac{fx}{z}, \frac{fy}{z})^T \tag{5.13}$$

The center of projection of the camera, where the rays emanate from, is called the optical center. The line through this point and perpendicular to the image plane is called the *principal axis* and the point at which the principal axis intersects the image plane is called the *principal point*.

In order to use this camera model as a tranformation, homogeneous coordinates are

often used and the relationship is defined as

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.14}$$

Now, it is possible to incorporate the translation in the image due to the principal point offset (with respect to the center of the image) by using

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} fx + zp_x \\ fy + zp_y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.15}$$

The matrix

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

is commonly referred to as the camera calibration matrix.

For CCD cameras, the pixels are usually not square and sometimes are not even rectangular. This can be accounted for in part by introducing a scaling on each of the focal axes

and a skew parameter as in

$$K = \begin{bmatrix} f_x & \alpha & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.17}$$

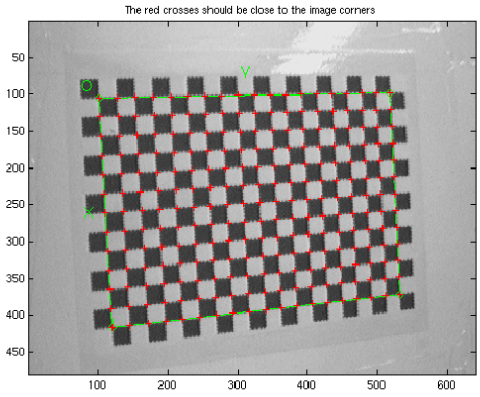
where f_x and f_y are the focal lengths on each of the axes and α is the skew parameter. The skew parameter is often set to zero or very insignificant due to advances in CCD manufacturing.

A general perspective camera can be represented by a 3×4 matrix, P, of rank 3, namely

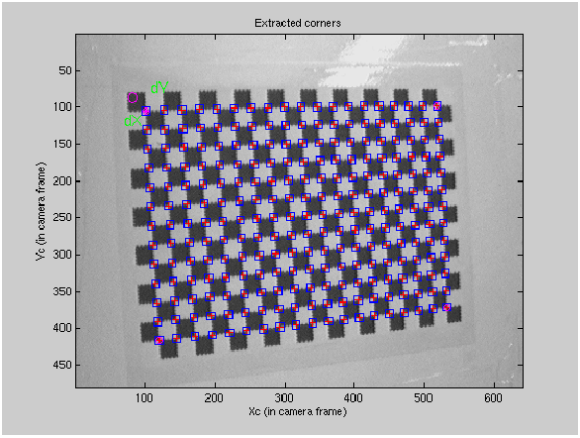
$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{5.18}$$

For most cameras, it is necessary to model the lens distortion in order to accurately determine points in camera coordinates by observing pixel locations. Thus, an offline camera calibration was performed by imaging a known target in multiple orientations (see Figure 5.8). A checkerboard pattern on a flat plane was used as a target. The Camera Calibration Toolbox in Matlab[8] provides a graphical interface to extract grid corners to approximately 0.1 pixel precision and outputs the camera calibration matrix, radial distortion coefficients and tangential distortion coefficients.

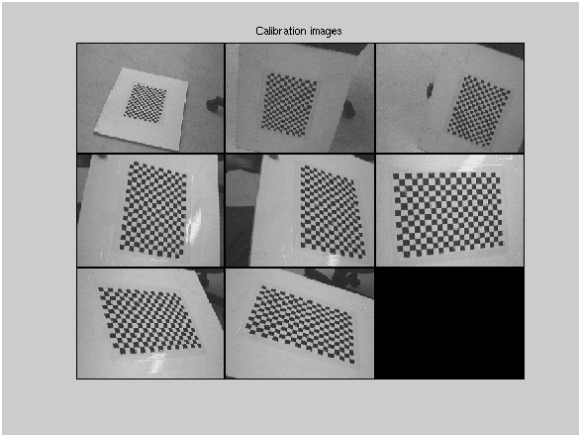
Although more sophisticated camera models exist[33, 32], the perspective camera model has proven sufficient for the optical tracking system discussed here.



(a)

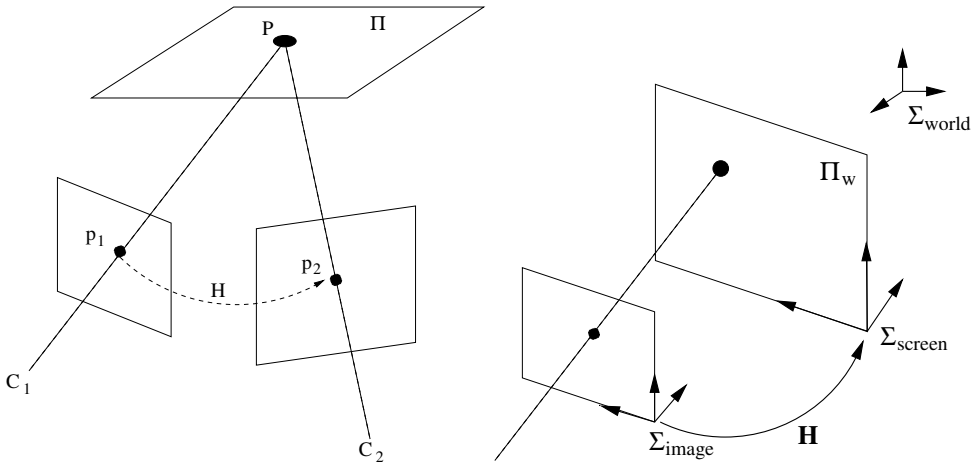


(b)



(c)

Figure 5.8: Intrinsic Camera Calibration Images taken with a Unibrain Fire-i400 camera. This illustrates the use of the calibration toolbox in Matlab. (a) shows the estimated corners while (b) shows the extracted subpixel corners. (c) is a mosaic showing all of the calibration images used for intrinsic calibration of this camera.



(a) Plane-to-Plane Homography (b) IVY Image to Screen Homography

Figure 5.9: (a) Given a point, P, in world coordinates which lies on a plane determined by Π, two image planes that image this point can be related through a transformation H. (b) This illustrates the use of a 2D homography in the MARVIN system for IVY. The image plane and screens are modeled as planes, with one of their axes aligned. Given measurements in the image, a transformation H must be determined to transform image points into screen coordinates. In the case of perfect alignment, H simply denotes a scaling factor. However since the cameras and screens are not perfectly aligned, in general a Homography, H, must be determined fully.

Extrinsic Calibration 1: Camera to Screen Transformation

The transformation from camera space to screen space is required to transform points on the image into points on the screen. Since both the screen and image are modeled as planes, a plane-to-plane 2D homography suffices. The only remaining issue is how to determine known points on the screen surface to within image resolution (i.e. less than 0.5 cm error). This is achieved by projecting a pattern onto the screen with the projectors for IVY and physically measuring 4 extremity points such as the corners of a rectangle. Another possibility is to ensure that the projectors are calibrated and project a known pattern onto

the screen to eliminate the need for manual measurement. A more attractive approach is to project a known pattern onto the screen using laser light.

Using the Direct Linear Transformation (DLT) algorithm as discussed in [29] at least four points must be registered to within image resolution (less than 0.5 cm) on the physical screen relative the screen origin (one of the four screen corners).

Given two sets of matching points, let P and P' denote these point clouds. First, the measured point clouds must be normalized by calculating transformations $\mathbf{T}$ and $\mathbf{T}'$ that ensure that the centroid of each point cloud is $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and the average distance of these transformed points is $\sqrt{2}$. Let $\mathbf{T}$ be the transformation of the first point cloud, ${}^iP = [{}^iP_x, {}^iP_y]$, with mean $M = [m_x, m_y]^T$ and $\mathbf{T}'$ be the transformation of the second point cloud, ${}^iP' = [{}^iP'_x, {}^iP'_y]$, with mean $M' = [m'_x, m'_y]^T$. It is extemely important to normalize the point clouds for numerical stability[30].

The normalization transformations are defined by

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{2}}{RMS\{^iP\}} & 0 & 0 \\ 0 & \frac{\sqrt{2}}{RMS\{^iP\}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -m_x \\ 0 & 1 & -m_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

$$\mathbf{T}' = \begin{bmatrix} \frac{\sqrt{2}}{RMS\{^iP'\}} & 0 & 0 \\ 0 & \frac{\sqrt{2}}{RMS\{^iP'\}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -m'_x \\ 0 & 1 & -m'_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

$$RMS\{^iP\} = \frac{\sum_i^n \sqrt{(^iP_x - m_x)^2 + (^iP_y - m_y)^2}}{n} \tag{5.21}$$

$$RMS\{^iP'\} = \frac{\sum_i^n \sqrt{(^iP'_x - m'_x)^2 + (^iP'_y - m'_y)^2}}{n} \tag{5.22}$$

Each point, $^iP = [^iP_x, ^iP_y]$ and $^iP' = [^iP'_x, ^iP'_y]$, is transformed by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} ^iP_x \\ \\ ^iP_y \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{T}' \begin{bmatrix} ^iP'_x \\ \\ ^iP'_y \end{bmatrix} \tag{5.23}$$

which ensures that the centroid of each point cloud is $[0,0,1]^T$ and that the average distance to the points is indeed $\sqrt{2}$.

Given any pair of n matching 2D homogeneous points,

$$P_{image} = P_i = \begin{bmatrix} x_i & y_i & w_i \end{bmatrix}^T \tag{5.24}$$

$$P_{screen} = P_i' = \begin{bmatrix} x_i' & y_i' & w_i' \end{bmatrix}^T \tag{5.25}$$

The linear transformation H (seen in Figure 5.9) can be defined as

$$P_i' = \mathbf{H}P_i \tag{5.26}$$

In projective geometry, there is a duality between points and lines. Thus each point can also be described as a line. Since P_i and P_i' represent the same 3D point on the world plane, the lines represented by P_i' and $\mathbf{H}P_i$ must be parallel in projective space. The cross product of parallel lines should be exactly zero giving rise to the following relationship:

$$P_i' \times \mathbf{H}P_i = 0 \tag{5.27}$$

Given two vectors $\vec{a} = [a_x, a_y, a_z]^T$ and $\vec{b} = [b_x, b_y, b_z]^T$, the cross product $\vec{a} \times \vec{b}$ can be expressed in matrix form as

$$\vec{a} \times \vec{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \tag{5.28}$$

Expanding and Equation 5.27 into its components using Equation 5.28 and denoting the transformation H as

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{5.29}$$

results in

$$P_i' \times \mathbf{H}P_i = \begin{bmatrix} y_i'(h_7x_i + h_8y_i + h_9w_i) - w_i'(h_4x_i + h_5y_i + h_6w_i) \\ w_i'(h_1x_i + h_2y_i + h_3w_i) - x_i'(h_7x_i + h_8y_i + h_9w_i) \\ x_i'(h_4x_i + h_5y_i + h_6w_i) - y_i'(h_1x_i + h_2y_i + h_3w_i) \end{bmatrix} \tag{5.30}$$

This can be simplified by denoting the transformation as a vector

$$\mathbf{h} = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T \tag{5.31}$$

and factoring its elements from Equation 5.30. A system of equations, $A_i\mathbf{h} = 0$ is then obtained where

$$A_i = \begin{bmatrix} 0 & 0 & 0 & -w_i'x_i & -w_i'y_i & -w_i'w_i & y_i'x_i & y_i'y_i & y_i'w_i \\ w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \\ -y_i'x_i & -y_i'y_i & -y_i'w_i & x_i'x_i & x_i'y_i & x_i'w_i & 0 & 0 & 0 \end{bmatrix} \tag{5.32}$$

or written more compactly as

$$A_i = \begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{P_i}^T & y_i'\mathbf{P_i}^T \\ w_i'\mathbf{P_i}^T & \mathbf{0}^T & -x_i'\mathbf{P_i}^T \\ -y_i'\mathbf{P_i}^T & x_i'\mathbf{P_i}^T & \mathbf{0}^T \end{bmatrix} \tag{5.33}$$

Note that only two of the three rows in A_i are linearly independent, the third row can be removed resulting in a 2×9 matrix

$$A_i = \begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{P_i}^T & y_i'\mathbf{P_i}^T \\ w_i'\mathbf{P_i}^T & \mathbf{0}^T & -x_i'\mathbf{P_i}^T \end{bmatrix}_{2\times 9} \tag{5.34}$$

Now, in order to solve the system of equations defined in Equation 5.34, at least four matching points must be found in the two images. This can be generalized to n points where $n \geq 4$. For n points, create a $2n \times 9$ matrix

$$A = A_{2n\times 9} = \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_n \end{bmatrix} \tag{5.35}$$

and solve $A\mathbf{h} = 0$ through a singular value decomposition (SVD), $A = UDV^T$. $\mathbf{h}$ can be extracted as the last column of V or rather the column of V that corresponds to the smallest

singular value.

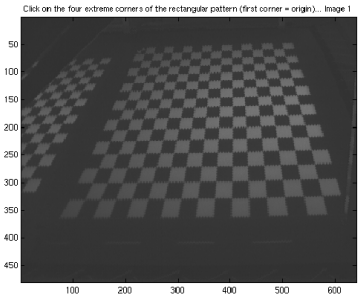
Now, the solution **h** represents the elements of the desired transformation, namely

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{5.36}$$

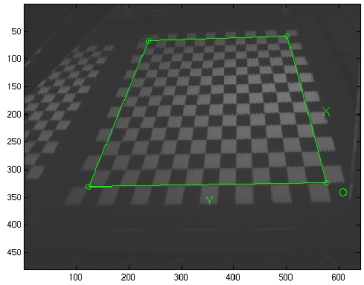
This corresponds to the transformation between *normalized* point clouds. The transformation must then be de-normalized using the previously computed normalizing transformations $\mathbf{T}$ and $\mathbf{T}'$. This is accomplished by

$$H_{final} = T'^{T} H T \tag{5.37}$$

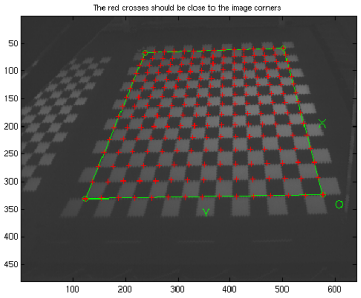
H_{final} is the final 3×3 homography which relates the two images and can be used to transform points in image (or camera) coordinates to points in IVY screen coordinates. An illustration of the calibration method is shown in Figure 5.10 and the results of applying the calibration homographies to overlapping images is shown in Figure 5.11. Typical calibration images for each screen surface are also shown in Figure 5.12.



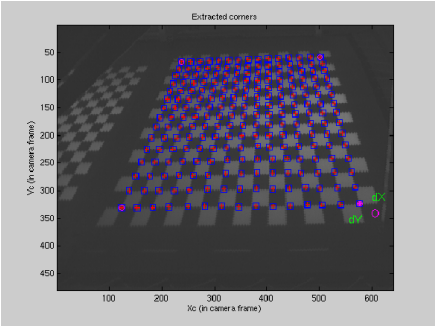
(a) Original Image



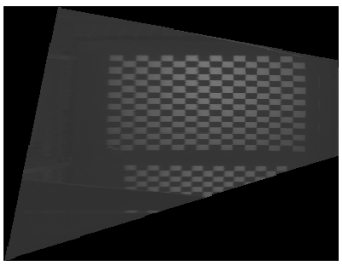
(b) Four Corners Selected



(c) Grid Corners Estimated

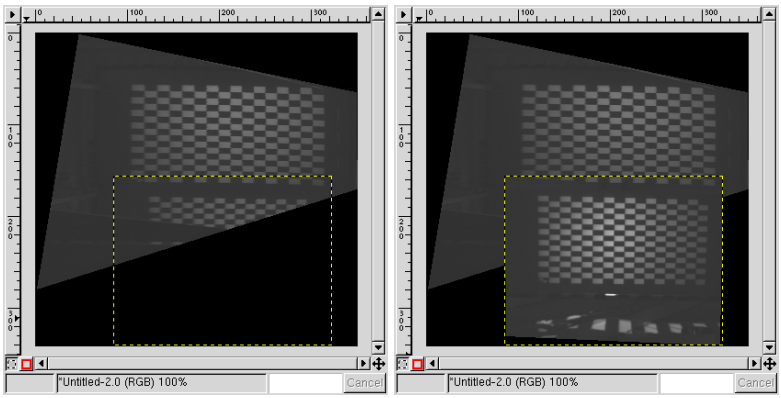


(d) Subpixel Corners Extracted



(e) Image warped using computed Homography

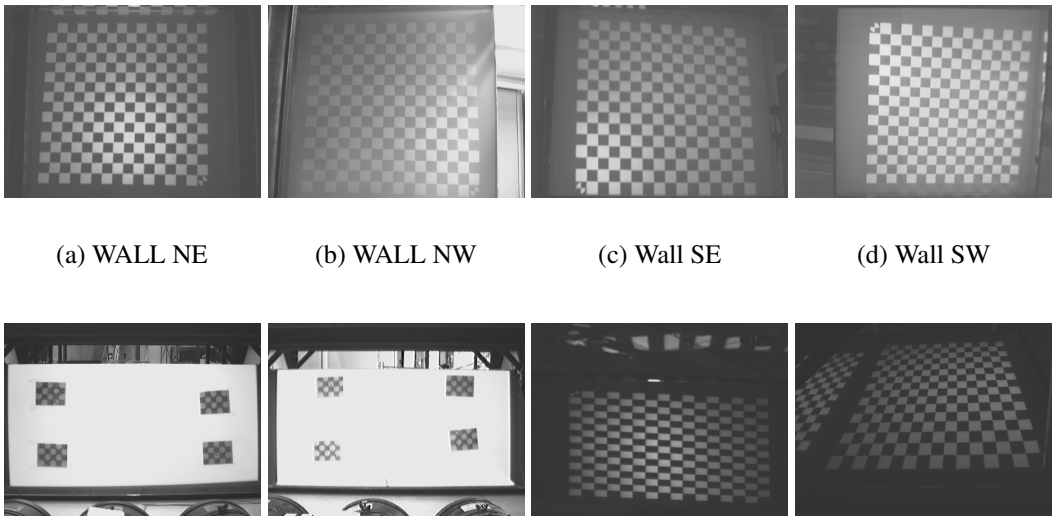
Figure 5.10: Screen Calibration Method. This illustrates the calibration method. (a) first an image is taken, (b) then four extreme corners are selected which are measured manually in centimetres. (c) the other grid corners are estimated and (d) extracted. (e) shows the original image warped using the computed homography for sanity checking.



(a) Half of ceiling showing location where image from second ceiling camera should be overlaid

(b) Overlay of calibrated ceiling images

Figure 5.11: An image mosaic of the two ceiling camera images after warping by their associated homographies. Since both cameras were calibrated to a common reference point, the overlapping regions are registered properly.



(a) WALL NE (b) WALL NW (c) Wall SE (d) Wall SW

(e) Floor 1 (f) Floor 2 (g) Ceiling 1 (h) Ceiling 2

Figure 5.12: Typical images used for calibrating the image to screen transformation. Four corners of the grid are extracted and manually measured to generate correspondences.

5.4.2 Extrinsic Calibration 2: the Laser Wavelength Filter Transformation

After finding the linear homography that transforms the camera image coordinates into screen coordinates, the laser wavelength filter must be placed on the camera lenses. This requires a small physical force to be applied to the camera which could possibly move or rotate the camera thus invalidating the previously computed transformation. To account for this, a second transformation is determined that corrects this possible error. A second planar homography suffices to model the error since the image plane suffers a possible rotation/translation and this can be modeled as a projective distortion of the image.

In order to compute this homography, at least four points (see [29, 19]) must be registered between an image acquired *before* the filter was positioned on the lens and an image acquired *after* the lens is in place. Since the wavelength filter suppresses all light outside of the 650nm range, at least four laser spots must be used to determine the correspondences. Thus, at least four lasers are positioned in a stationary position within the immersive display with their beams intersecting the screen for which the transformation is to be determined. The ambient light is reduced in the room and a *before* image (I_{before}) is captured. The filter is then very gently placed on the lens to be calibrated and the *after* image (I_{after}) is captured. Now, the corresponding points must be extracted from the images. This is performed manually per image with subpixel resolution using Matlab and the image zoom functionality. It is extremely important to ensure that the laser diodes do not move between the acquisi-

tion of the two images otherwise the computed homography will be invalid and it will be necessary to restart the calibration procedure.

Once the correspondences are known, the previously discussed DLT algorithm is used to determine the homgraphy between I_{after} and I_{before}. Note that the order here is extremely important. If the wavelength filter is attached gently enough, the camera will not move and the computed homography will be of zero magnitude. These must be discarded since multiplying the previous transformation by a zero matrix will result in a transformation that maps all points to zero. Otherwise the final transformation is computed by:

$$\mathbf{H_{final}} = \mathbf{H_{nofilter}}\mathbf{H_{filter}} \tag{5.38}$$

where $H_{nofilter}$ is the homography obtained previously without the wavelength filter in place and H_{filter} is the homography obtained between I_{after} and I_{before}.

Finally, to transform camera coordinates $\begin{bmatrix} u & v \end{bmatrix}^T$ to screen coordinates $\begin{bmatrix} X & Y \end{bmatrix}^T$, the following calculation is performed:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H_{final}} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.39}$$

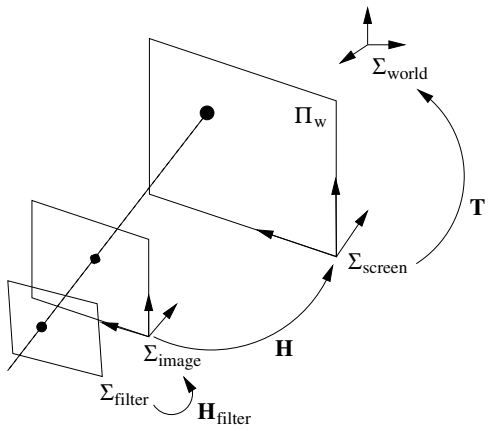


Figure 5.13: Screen-to-World Transformation. Given that the transformation H is already known which transforms image points to screen coordinates, a transformation T must be determined to transform screen points to world coordinates which lie on the plane Π_w.

5.4.3 Extrinsic Calibration 3: Screen to World Transformation

After computing the transformation from Camera coordinates to Screen coordinates, the transformation from screen coordinates to a common World coordinate system must be determined (see Figure 5.13). This is a rigid-body transformation composed of a rotation, **R**, and a translation, **T** per wall of the display. The origin of the world coordinate system is the centre of the fully-enclosed display volume.

To transform points on the screen to 3D world coordinates, the following relationship is defined:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R_{wall}} & -\mathbf{T_{wall}} \\ \hat{\mathbf{0}} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ \Pi_{wall} \\ 1 \end{bmatrix} \tag{5.40}$$

Where **R** and **T** are defined on a per wall basis as in Table 5.1. The translations were obtained by manually measuring the distance from the World coordinate origin to each wall. The rotations are defined by the screen coordinate system and its orientation with respect to the world coordinate system. Π_{wall} is the distance between the origin and the plane of the wall which is measured manually.

5.5 Evaluation

To evaluate the accuracy of the optical system, both the errors in the reported position and orientation must be estimated. Several tests were performed by placing the device in known locations (measured manually) and the reported positions from the tracking system were recorded. Also, the device was placed in a static position and rotated on all three of its axes to determine the accuracy of the reported orientation. This section describes the method for determining the accuracy of both the orientation and position and the results of each test performed is reported.

5.5.1 Orientation

In order to evaluate the accuracy of the orientation computed by the optical system, several tests were performed. The device was placed at a known location and rotated 360^o at 5^o intervals for the Yaw axis rotation. Then another test was performed per axis where the device was rotated at 1^o intervals for a smaller range and data was recorded at each

Wall	Rotation R_{wall}	Translation T_{wall}	Plane Π_{wall}
NW	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$	$114.3 * \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$	$114.5cm$
NE	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -114.3 & 121.92 & 0 \end{bmatrix}^T$	$115.5cm$
SW	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	$114.3 * \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$	$115.4cm$
SE	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$114.3 * \begin{bmatrix} 0 & 1 & -1 \end{bmatrix}^T$	$113.9cm$
FLOOR1	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -122.2375 & 0 & -121.92 \end{bmatrix}^T$	$114.3cm$
FLOOR2	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -122.2375 & 0 & -121.92 \end{bmatrix}^T$	$114.3cm$
CEIL1	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -122.2375 & 0 & -121.92 \end{bmatrix}^T$	$108.585cm$
CEIL2	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -122.2375 & 0 & -121.92 \end{bmatrix}^T$	$108.585cm$

Table 5.1: Screen to World Coordinate Transformations for IVY's eight projection surfaces.

orientation. Taking the first measurement as the reference orientation, the angles between the reference orientation and every other orientation was computed to show the error in orientation. A rotational stage with markings at 1^o intervals was used to perform both of these tests.

Yaw

The laser diode housing was placed in the center of IVY roughly 4' above the floor on a rotational stage that allowed the device to rotate at 1^o intervals. Figure 5.14 shows the raw data points for a full 360^o rotation on the azimuth at 5^o intervals. For each direction vector, points on the unit circle are drawn at both the measured and correct orientation in the same colour (note that due to the accuracy of the measurement these points appear almost coincident). In a cube-shaped immersive display the corners present problems with tracking: when the lasers shine into the corners, no data can be collected and tracking is lost until the lasers shine onto the screen. Note that in normal operation these gaps can be filled in using inertial data.

In a second orientation experiment, rotational data was collected over a 10 degree range at 1 degree intervals separately on the Yaw, Pitch, and Roll axes. The relative angles, shown in Table 5.2, were computed between the direction vector and the first reported direction vector for the Yaw axis, and between the up-vectors for Pitch and Roll. The mean error of each experiment was 0.1^o per axis and the maximum errors were 0.3^o, 0.25^o, 0.35^o for the Yaw, Pitch, Roll axes respectively.

Rotational Stage	Yaw Angle	Pitch Angle	Roll Angle
0^o	0.0000^o	0.0000^o	0.0000^o
1^o	0.9229^o	1.1329^o	0.8459^o
2^o	1.9101^o	2.1728^o	2.1036^o
3^o	3.2703^o	3.1580^o	2.9920^o
4^o	4.1654^o	4.0918^o	4.1093^o
5^o	5.0992^o	5.0284^o	5.1583^o
6^o	6.2851^o	6.0500^o	6.1239^o
7^o	7.0167^o	7.1388^o	7.0007^o
8^o	8.3210^o	8.0294^o	8.2022^o
9^o	9.1814^o	9.0891^o	9.0719^o
10^o	9.8664^o	10.2454^o	10.3525^o

Table 5.2: Computed angles between the reported direction vectors at 1^o increments. Yaw, Pitch, and Roll data were collected independently in separate experiments.

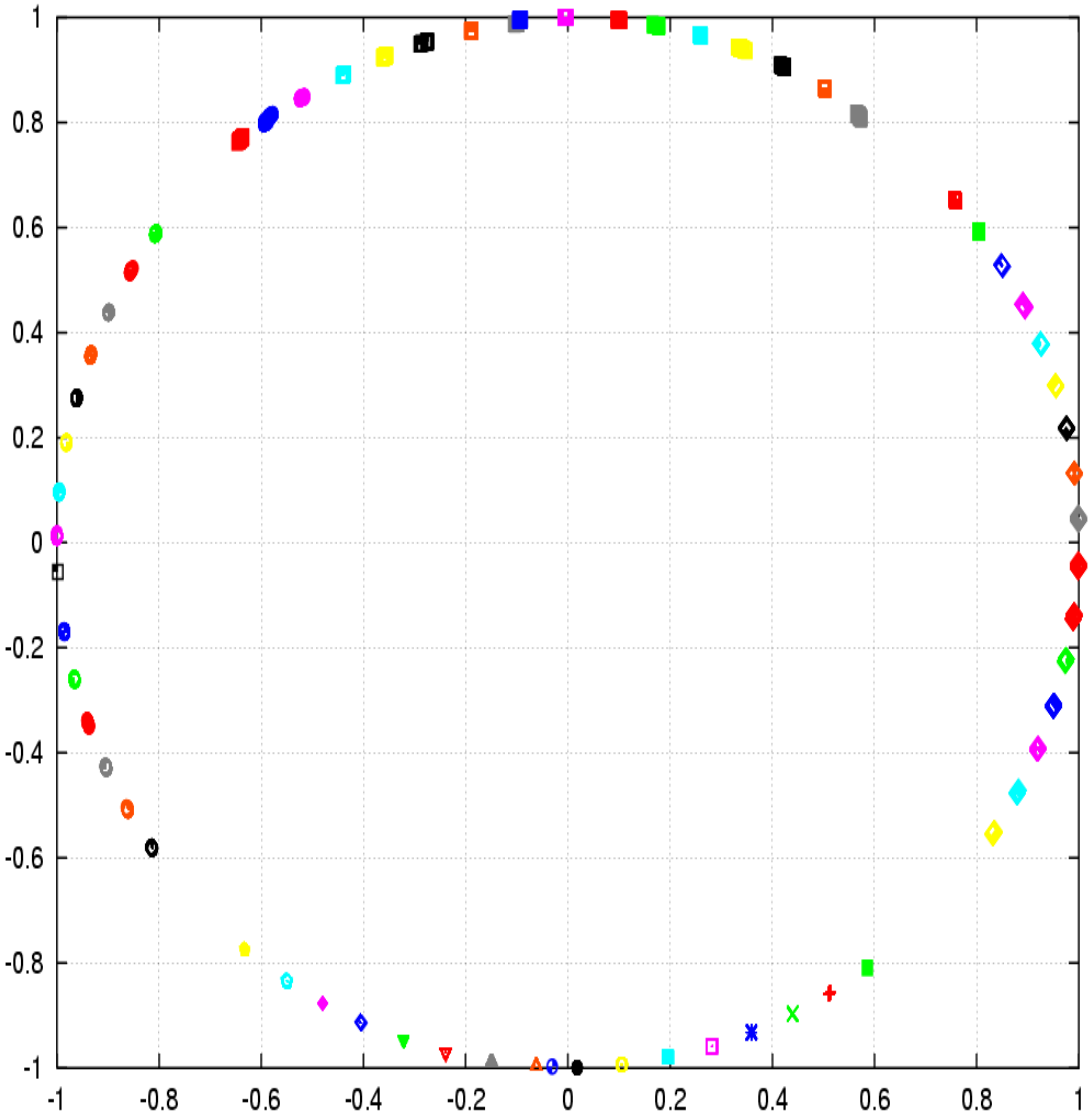


Figure 5.14: 360^o Raw Orientation Data (taken at 5^o intervals). Unit vectors are plotted, with the same symbol, in the recovered and measured directions. The plotted X-axis is the X-coordinate of the unit vector and the plotted Y-axis is the Z-coordinate of the unit vector. Note: the four large holes indicate positions where the lasers were shining into the corners of IVY and thus could not be tracked.

Absolute Position (X-Z) (metres)	Reported Position (X-Z) (metres)	Error X (metres)	Error Z (metres)
(0.00, 0.65)	(0.0072, 0.6511)	0.0072	0.0011
(-0.81, -0.65)	(-0.8167, -0.6527)	0.0067	0.0027
(0.81, 0.30)	(0.8504, 0.3016)	0.0404	0.0016
(-0.81, 0.65)	(-0.8175, 0.6559)	0.0075	0.0059
(-0.81, 0.30)	(-0.8175, 0.3059)	0.0075	0.0059
(0.00, 0.30)	(0.0046, 0.3066)	0.0046	0.0066
(0.81, 0.65)	(0.8587, 0.6584)	0.0487	0.0084
(0.81, -0.65)	0.8586, -0.6546)	0.0486	0.0046
(0.00, -0.65)	(0.0048, -0.6556)	0.0048	0.0056
(0.50, 0.65)	(0.5072, 0.6484)	0.0072	0.0016
(0.50, -0.65)	(0.5070, -0.6531)	0.0070	0.0031
(0.81, -0.30)	(0.8126, -0.3013)	0.0026	0.0013
(-0.81, -0.30)	(-0.8120, -0.3038)	0.0020	0.0038
(0.00, -0.30)	(-0.0056, -0.3045)	0.0056	0.0045
(0.81, -0.30)	(0.8154, -0.3070)	0.0054	0.0070
(-0.50, -0.30)	(-0.5045, -0.3080)	0.0045	0.0080
(-0.50, -0.65)	(-0.5014, -0.6509)	0.0014	0.0009
(-0.50, 0.30)	(-0.5056, 0.3007)	0.0056	0.0007
(-0.50, 0.65)	(-0.5041, 0.6542)	0.0041	0.0042
(0.50, -0.65)	(0.5051, -0.6534)	0.0051	0.0034
(0.50, -0.30)	(0.5040, -0.3026)	0.0040	0.0026
(0.50, 0.30)	(0.5063, 0.3042)	0.0063	0.0042
(0.50, 0.65)	(0.5070, 0.6512)	0.0070	0.0012

Table 5.3: Error associated with measured and reported tracker positons. All data points were taken at the same height (Y-coord) of 1.35m.

5.5.2 X-Z Position

To estimate the accuracy of the position estimates, the device was placed at 20 known locations within IVY (See Table 5.3) and recorded the tracker output. The raw data in this test are illustrated in Figure 5.15. The mean absolute position error was modest at 1.13cm but there were several cases where the error was nearly 5cm. All errors greater than 1.0cm occured when X = 81cm. This is due to the placement of one of the ceiling cameras. Due to space constraints on the physical layout, one camera is placed largely off-axis creating a large perspective distortion in the image.

The noise covariance of each position estimate was also computed and a typical example can be seen in Figure 5.16(a) using a Linear Kalman filter with variance of $1cm^2$ on position. The small covariance (approximately 0.5cm) in the position is attributable to the noise in each laser position estimate due to the limited resolution of the cameras. Since we are acquiring 640x480 resolution images from the cameras, the 2.29m screen is imaged at approximately 500 pixels, making 1 camera pixel correspond to approximately 0.5cm on the screen surface. Using higher resolution images would increase the precision of the tracking system since it would allow us to make better estimates of the laser positions. Since the walls of IVY are fabric walls, and thus vibrate and move slightly when in the presence of large motion within the display, we were concerned how this would affect the position estimate. We placed the device in a stationary position and recorded data while violently moving the screen fabric on all walls. The covariance of the estimate can be seen

in Figure 5.16(b). The system reacts well with a spread of approximately 1.5cm even in the presence of large motion of the screen surfaces.

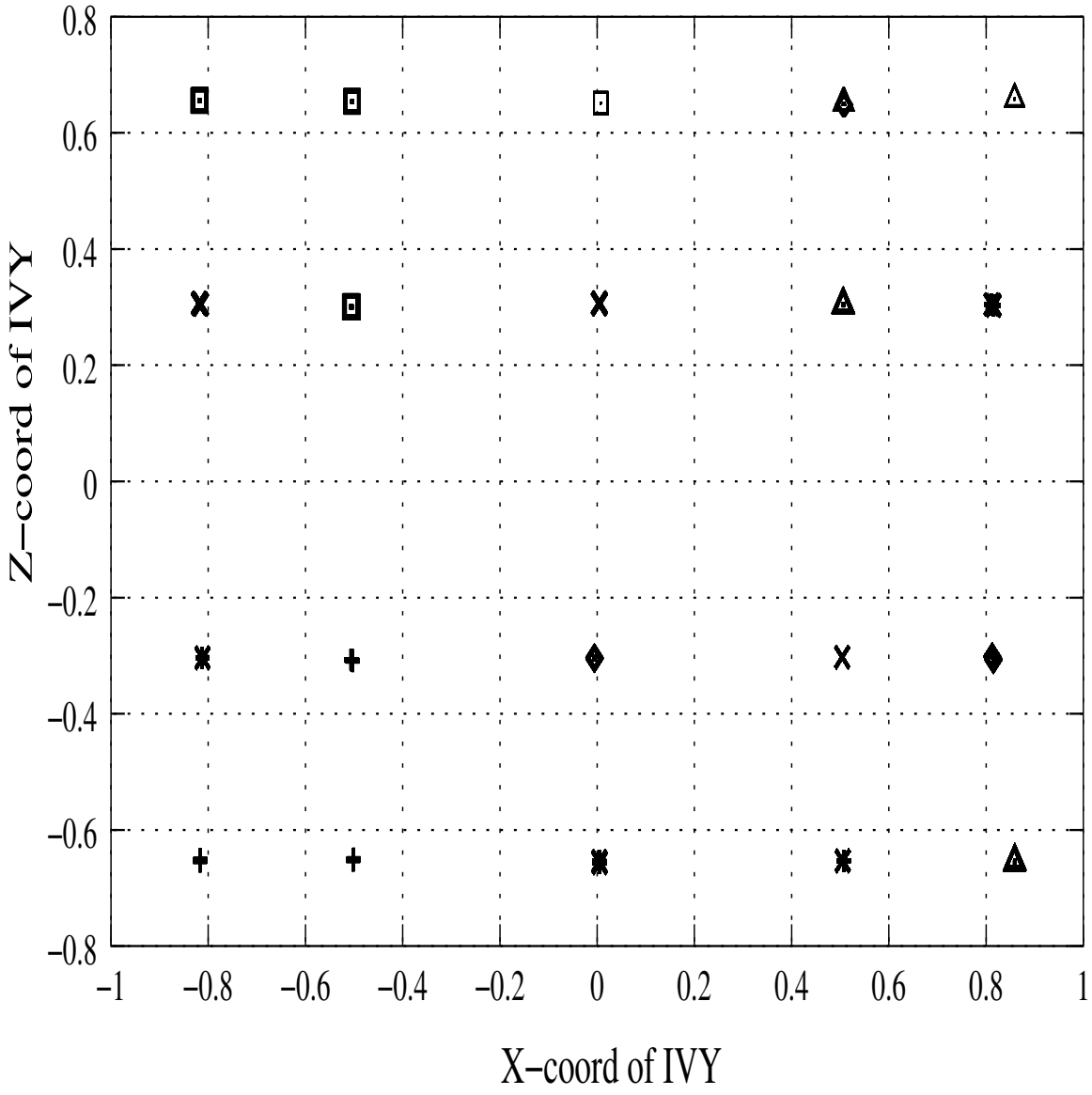
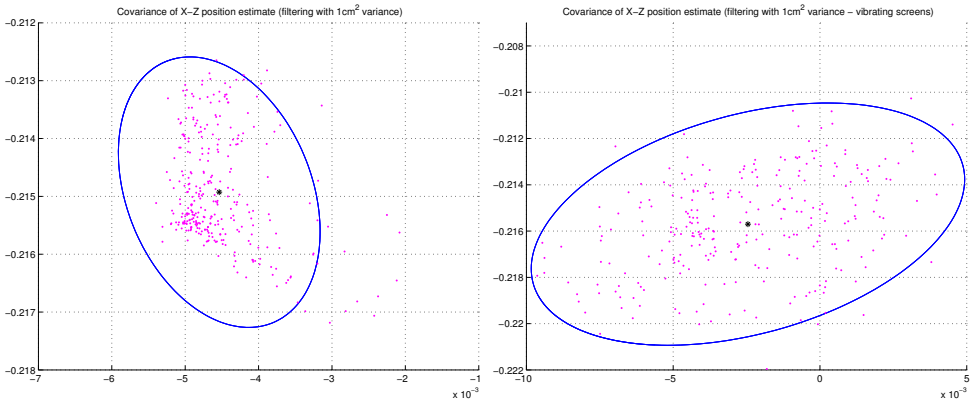


Figure 5.15: Reported Position of Tracker at known locations in metres. Different symbols are used only to distinguish different measurements.



(a) Typical Noise Covariance of Stationary X-Z position in metres (filtering with 1cm^2 variance). This shows a spread of 6mm on the Z-axis and 3mm on the X-axis.

(b) Typical Noise Covariance of Stationary X-Z position in metres (filtering with 1cm^2 variance). This shows that even in the presence of large motion of the screen surfaces (screen movement due to fast motion within IVY), the spread of the measurement is 1.5cm on the X-Axis and 1cm on the Z-axis.

Figure 5.16: Results from collected data. (a) shows the typical noise covariance of a stationary position, and (b) shows how the noise increases when the fabric screens vibrate due to fast motion.

Absolute Translation(cm)	Error in Reported Measurement(cm)
0.5	0.2737
1.0	0.4648
1.5	0.8857
2.0	0.5881
2.5	0.3169
3.0	0.3250
3.5	0.7313
4.0	0.7701
4.5	0.3009
5.0	0.3114
5.5	0.6956
6.0	0.2003
6.5	0.3395
7.0	0.5343
7.5	0.6190
8.0	0.2418
8.5	0.6824
9.0	0.6757
9.5	0.4169
10.0	0.2038
10.5	0.6852
11.0	0.4477

Table 5.4: Y Translation Error.

5.5.3 Y Position

Here the accuracy of the Y component of the reported position is discussed. The tracking device was positioned at different heights with a 0.5cm increment at the same location within IVY and the Y position was recorded. The total translation was 23cm. Tables 5.5 and 5.5 summarize the error. The mean error of this experiment was 0.48cm and the maximum error is 0.88cm

Absolute Translation(cm)	Error in Reported Measurement(cm)
11.5	0.5435
12.0	0.7359
12.5	0.3968
13.0	0.5874
13.5	0.1282
14.0	0.4066
14.5	0.3701
15.0	0.4174
15.5	0.3279
16.0	0.3273
16.5	0.4519
17.0	0.7914
17.5	0.5649
18.0	0.0767
18.5	0.7637
19.0	0.4203
19.5	0.6381
20.0	0.5306
20.5	0.5124
21.0	0.6278
21.5	0.4617
22.0	0.6405
22.5	0.6466
23.0	0.3964

Table 5.5: Y Translation Error (continued).

5.6 Summary

This chapter has introduced the optical component of the MARVIN tracking system. The tracking system discussed uses laser diodes attached to a helmet worn by the user. The projections of the lasers are tracked from outside of the display. The pose of the user is estimated by imposing constraints on the geometry of the laser diode housing and the tracked projections of the lasers. Imposing constraints on the geometry is sufficient to estimate the

pose of the user even though a unique solution is unavailable from the four-laser geometry. Using a temporal coherence and dynamic gating, the correct pose is maintained once the initial correct pose is chosen. This system has many advantages over the existing tracking technologies discussed in Chapter 2: the accuracy achieved is not a function of the distance of the user from a base station; the system performance is not degraded by metallic objects or other interference; and the user is untethered and is not required to wear a large encumbering device which could compromise their immersive experience. Since the laser diodes are aimed behind the user, their projections do not interfere with the user's visual experience. Also, using off-the-shelf FireWire® digital video cameras allows the tracking system to evolve with the commercial market making it possible to increase the resolution and framerate as new camera technology becomes available.

Chapter 6

System Integration

In Chapter 4, an inertial pose estimation system was described that is capable of an update rate of 300Hz. However, the drawback to this system is that due to calibration error, noise levels, and numerical precision concerns, the estimate is only reliable for approximately one second. The optical system described in Chapter 5 runs at a slower but reliable update rate of 15Hz when all of the laser projections are visible in the camera images. The optical system can provide highly accurate absolute pose estimates while the inertial system provides fast relative motion information. Using a hybrid data fusion technique, the advantages of these two systems can be combined to compensate for the disadvantages of each individual system.

This chapter discusses the system integration of the inertial pose estimation system with the optical tracking system described in previous chapters. First, the hardware to physically integrate these two systems is discussed followed by a discussion on data fusion algorithms and an explanation of the algorithm used in the MARVIN tracking system.

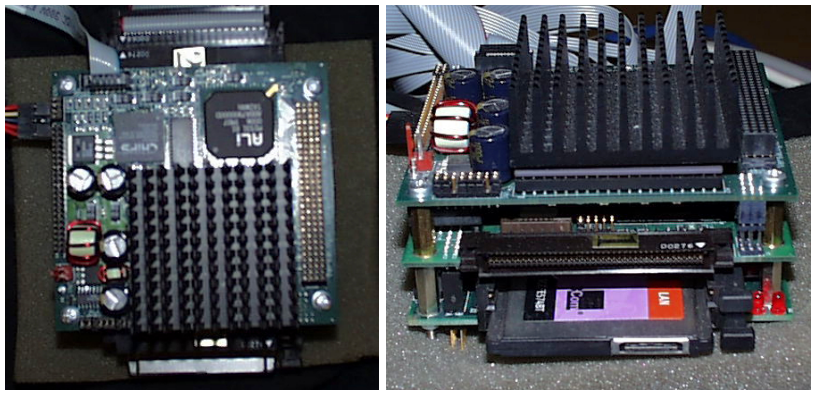


Figure 6.1: The Panther PC/104 Wearable Computer from Versalogic. The dimensions of the module is 9cm x 9.5cm x 8cm.

6.1 The Wearable System

In general, current tracking systems (magnetic and otherwise) are incapable of tracking a user within a fully-enclosed display without tethering them to the tracking equipment. As discussed previously, this imposes constraints on the user's motion and is undesirable. Thus the MARVIN tracking system should provide a "tetherless" solution. This is accomplished by using 802.11b wireless ethernet technology and a small lightweight wearable computer. The specifications of the wearable computer can be found in Appendix C but essentially it is a small computer with a standard connection interface for stacking other modules onto the computer. An image of the computer is shown in Figure 6.1.

In order to convert analog signals produced from the accelerometers into a useable digital signal, an analog-to-digital converter is needed. The VIPS 10 A/D converter used in the MARVIN system (shown in Figure 6.2) is a small module connected through the parallel port of the wearable computer. Using this allows the form factor of the wearable

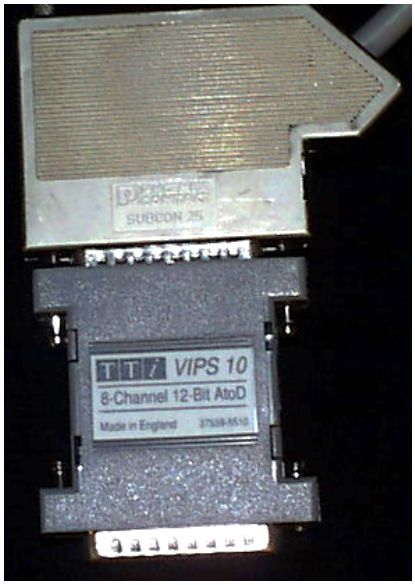


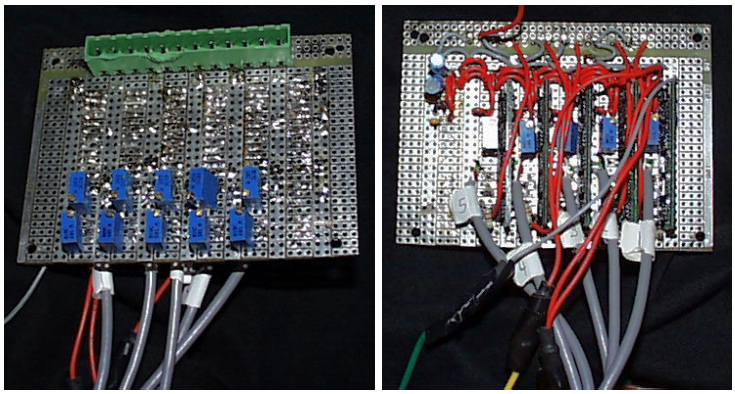
Figure 6.2: The VIPS 10 Analog-to-Digital Converter from TTi. The dimensions of the module is 8cm x 11cm x 1.5cm.

computer to be reduced since another A/D PC/104 module is unnecessary. Its specifications are given in Appendix C.

The laser tracking device is completely independent and no external wires are needed. The only requirement of the laser device is the need for a battery. Lithium batteries are used for power since the discharge rate is preferable for powering lasers. The laser tracking device is shown in Figure 6.4.

The inertial device must be connected directly to the VIPS 10 A/D converter in order for the signal to be acquired. Thus a wire per accelerometer is needed. Moreover, a power and amplification circuit (shown in Appendix C) is required for the sensors and is shown in Figure 6.3.

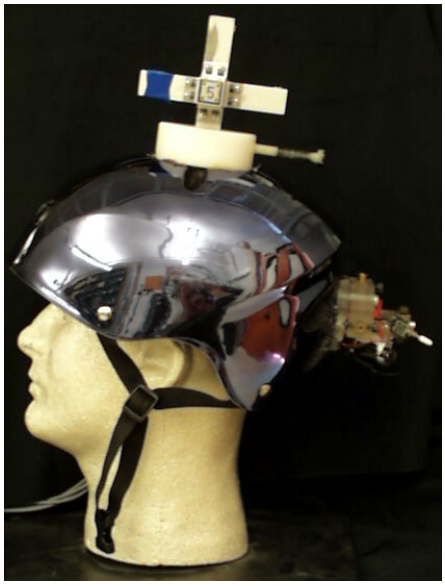
Since MARVIN is a head tracking system, the sensors must be attached to the user's



(a) (b)

Figure 6.3: The amplification circuit for the inertial device. The dimensions of the circuit are 10cm x 7cm x 3cm.

head. Thus the tracking equipment is placed on a helmet worn by the user at all times while tracking them. For this prototype, the optical laser tracking device is placed behind the user so that the lasers shine away from the visible field of view of the user. The inertial system is placed on top of the helmet and the fully assembled prototype can be seen in Figure 6.4.



(a) The helmet with tracking system

Figure 6.4: The Tracking System Helmet Prototype. This shows the helmet alone and with both the inertial and optical tracking systems attached to it.

6.2 Data Fusion

Since different types of sensors with differing characteristics are used to generate pose estimates for the MARVIN system, a data fusion technique must be employed that merges and smooths the sensor data into the "best" estimate. The Kalman filter (see Appendix B) is an optimal, recursive, linear least-squares estimator that uses the noise characteristics of each type of sensor and the error covariance of the estimated system to update the state to reflect the change reported by the sensor. An extended Kalman filter is a technique that sacrifices the "optimality" of the linear Kalman filter, however it works well in many instances to estimate nonlinear dynamic systems and is used in robotics, computer vision, and signal and control theory(see [43, 42, 48, 75, 4, 6, 17]).

A typical approach to data fusion is to use a complimentary filter[48, 22]. This type of filter uses complimentary sensors to maintain the pose of the object to be tracked. Typically GPS and inertial sensors are used concurrently for vehicle navigation. In this scenario, the inertial sensors (typically gyroscopes for orientation and a triad of linear accelerometers for displacement) are integrated into the heading and position of the vehicle. These are assumed to provide reliable data for extended periods of time, i.e. several minutes, without accumulating too much error. When the GPS position data is available, the error is estimated between the current inertial estimate and the GPS absolute position data with a Kalman filter. The Kalman filter employed estimates only the error dynamics of the system and appropriately updates the inertial system with these errors. This is illustrated in Fig-

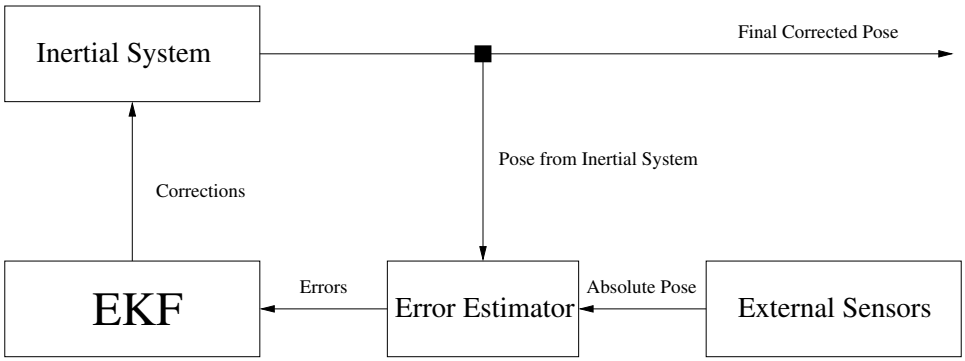


Figure 6.5: Typical Complimentary Indirect Data Fusion Filter for Navigation. The output of the inertial system is said to be reliable for several minutes (if not hours) and is continually output as valid pose data. At slower update intervals, an external system provides data to an error estimator that computes the error between the current inertial pose estimate and the absolute pose estimate provided. The error dynamics of the system are encapsulated in the Kalman filter and only the errors are estimated at each time step. The state at each update is sent to the inertial system which corrects the appropriate terms. This provides stability and ensures that the inertial system does not drift wildly.

ure 6.5. Foxlin[22] also developed a separate-bias complimentary Kalman filter for a head tracking scenario where the absolute pose data was provided by acoustic sensors and the relative motion was provided by gyroscopes and accelerometers.

6.2.1 Kalman Filter Devlopment for MARVIN

In the MARVIN system, the inertial pose estimator is reliable for short periods of approximately one second. The optical system is capable of providing estimates nominally at 15Hz. A complimentary filter framework for this tracking system can be developed that fuses the optical and inertial estimates appropriately. A Kalman filter estimates the pose from the inertial system since the integration can be performed within the Kalman framework and the optical pose estimates can be used as a control input to this filter. The control

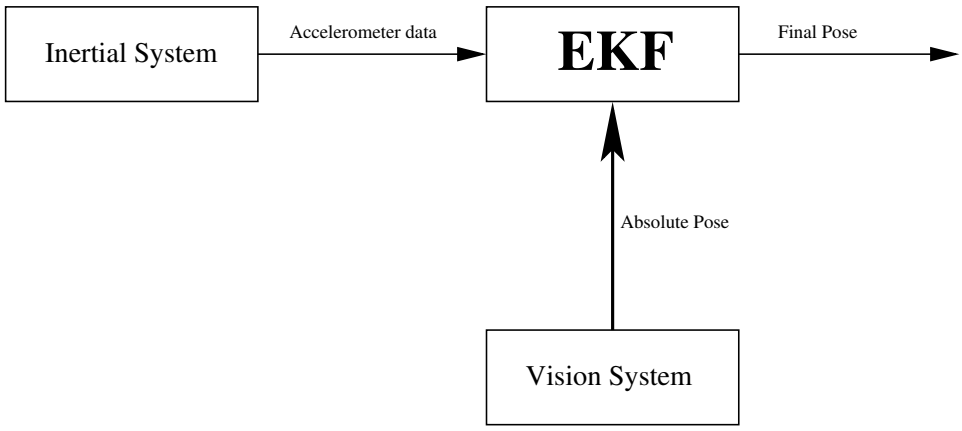


Figure 6.6: Complimentary Data Fusion Filter. An extended Kalman filter (EKF) is used to fuse the accelerometer and vision data into a single pose estimate. The EKF estimates the pose using the mathematics described in Chapter 4 and the vision system provides the absolute pose data at a lower update rate as a control input to the filter, effectively "resetting" the pose estimate at each update.

input acts as a reset switch that "resets" the state whenever an estimate becomes available. This allows the inertial system to provide motion data relative to the last pose update of the optical system. This is illustrated in Figure 6.6. In [4] an extended Kalman filter is developed that enables estimation of a quaternion for orientation. They utilize a separate linear Kalman filter to estimate the position of the tracked object. The extended Kalman filter developed in this chapter uses a quaternion representation for orientation but also estimates the position of the tracked object.

The state to be estimated

The first issue in developing a Kalman filter for the MARVIN tracking system is to determine what variables are being estimated in the state. For the purposes of this thesis, the

following state was sufficient:

$$\hat{x}_k = \begin{bmatrix} \hat{q} & \vec{\omega} & \dot{\vec{\omega}} & \vec{p} & \vec{v} & \vec{\alpha} \end{bmatrix}^T \tag{6.1}$$

For estimating the orientation, a quaternion representation, $\hat{q}$, is used, and the angular velocity and angular acceleration ($\vec{\omega}, \dot{\vec{\omega}}$) are estimated in the state. The position, linear velocity, and linear acceleration ($\vec{p}, \vec{v}, \vec{\alpha}$) are also estimated.

The next step in developing a Kalman filter is to determine the time update equations and the measurement update equations.

Time update

The time update equations perform the integration of the state from time $k-1$ to time k using a (possibly nonlinear) dynamical model. The time update step predicts the state and the covariance as

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{6.2}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{6.3}$$

where $f(\hat{x}_{k-1}, u_k, 0)$ is a (possibly nonlinear) function that integrates the state using the control input and the previous state. Q_{k-1} is the process noise covariance (possibly changing with time) and W_k is the jacobian matrix of $f(\cdot)$ with respect to $\hat{x}_k$. P_k^- is the predicted

covariance and A_k is the jacobian matrix of $f(\cdot)$ with respect to $\hat{x}_k$, namely

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{6.4}$$

In MARVIN, $f(\cdot)$ integrates the quaternion by noting that the derivative of a quaternion is defined as

$$\frac{d\hat{q}}{dt} = \frac{1}{2}\hat{q}\hat{\omega} \tag{6.5}$$

Any suitable numerical technique can be used to integrate the quaternion, i.e. a fourth-order Runge-kutta technique[59]. For the rest of the variables, the derivative of velocity is the acceleration (angular and linear) and the derivative of acceleration is set to zero.

Measurement update

When a measurement becomes available, the Kalman filter state must be updated accordingly. This is the measurement update step and the following is performed:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{6.6}$$

$$\hat{x}_k = \hat{x}_{k-1}^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{6.7}$$

$$P_k = (I - K_k H_k) P_k^- \tag{6.8}$$

where K_k is the Kalman gain for this time step, z_k is the measurement vector, R_k is the measurement noise covariance (possibly changing with time), and V_k is the jacobian matrix

of partial derivatives of $h(\cdot)$ with respect to the measurement noise. $h(\cdot)$ is the (possibly nonlinear) function that relates the predicted state to the measurement and H_k is the jacobian matrix of partial derivatives of $h(\cdot)$ with respect to $\hat{x}$ namely

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_k, 0) \tag{6.9}$$

In MARVIN, the measurement is defined as

$$z_k = \begin{bmatrix} \dot{\vec{\omega}} & \vec{\alpha} \end{bmatrix}^T \tag{6.10}$$

Since these are also estimated, the function $h(\hat{x}_k^-, 0)$ simply extracts the current estimate of angular acceleration and linear acceleration. Also, the jacobian matrix simply relates the state to the measurement and is defined as

$$H = \begin{bmatrix} \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{6.11}$$

Fusing with the Optical Data

The described Kalman filter estimates the state using MARVIN's inertial subsystem. The optical component must also now be incorporated. There are several ways to incorporate optical pose data into this framework. Possibly the simplest technique is to use the optical pose data as ground truth since it provides absolute data. This ground truth data can then

be used as a control input to the state, namely as u_k in the time update step. Whenever the optical data provides a pose estimate, the appropriate parameters in the state can be *overwritten* or *reset*. This creates a situation where the inertial data is used only during the period of time when the optical data is unavailable, i.e. between the 15Hz updates, allowing the system to keep the state estimated at the inertial system's update rate of approximately 300Hz. When the optical data is unavailable for slightly longer periods of time, the inertial data is solely used, i.e. when the lasers are not visible in the images at the corners of the screens.

The control input is thus defined as

$$u_k = \left[\begin{array}{c} \hat{q}, \vec{\omega}, \dot{\vec{\omega}}, \vec{p}, \vec{v}, \vec{\alpha} \end{array} \right] \tag{6.12}$$

where the velocities and accelerations are estimated from the optical system using a first-order approximation.

A second approach would be to use the optical data as another measurement to the Kalman filter and define a second jacobian matrix to relate the state to this new measurement. Unfortunately, in the MARVIN system since the inertial data drifts rapidly after one or two seconds, this approach might possibly diverge unrecoverably. If more reliable inertial data is used, this would be a possibility.

A third approach would be to derive a SCAAT filter (see Appendix B) for the data fusion. This would take a single accelerometer measurement as the measurement from the

inertial system and a single laser position as the measurement from the optical system. The filter would use the inertial formula

$$a_j = \vec{n}_j \cdot (\vec{\alpha} + \vec{g} + \dot{\vec{\omega}} \times \vec{r}_j + \vec{\omega} \times \vec{\omega} \times \vec{r}_j) \tag{6.13}$$

using the current state estimates of $\vec{\alpha}, \dot{\vec{\omega}}, \vec{\omega}$ and q (to rotate $\vec{n}_j, \vec{r}_j$) to predict the measurement from the accelerometer. For the optical system the SCAAT, filter poses a problem since it is impossible to distinguish the laser spots from one another in the current implementation. However, if the lasers are distinguishable, the SCAAT filter would use the current estimate of position and orientation and intersect the laser beam with the walls of IVY to determine the measurement prediction.

Chapter 7

Conclusions and Future Improvements

7.1 Conclusion

The optical tracking system developed in this thesis has been shown to be very effective at tracking the user within a fully-enclosed projective immersive display. The inertial system developed shows promising results in simulation and the method has been shown to estimate the pose of the device for an interval of approximately one second.

The system is fully mobile due to the use of the PC/104 and wireless ethernet technology. The optical system provides pose estimates at a reliable rate of 15-20Hz and the inertial data can be streamed at approximately 300Hz allowing real-time tracking performance. The tracking system framework presented is flexible and additional functionality could be implemented in the future.

7.2 Future Work

Several improvements could be made to the MARVIN tracking system to improve robustness. These improvements are beyond the scope of this thesis to implement but could further improve the performance of the tracking system in the future.

The full implementation of the data fusion algorithm remains to be implemented in the final system and is left for future work. As discussed in Chapter 6 the "best" way to fuse the inertial data with the optical data is to reset the inertial system parameters at every optical update. This will have the effect that the inertial system provides relative motion data between the optical measurements.

7.2.1 Improving the Optical System

The optical component of MARVIN has the capability for becoming a standalone tracking system in the future. The current limitation is the framerate of the cameras. Cameras with a higher framerate than the current 30fps would increase the responsiveness of the optical system making it more reliable as a standalone tracking unit. Higher imaging resolution would increase the accuracy of the pose estimate as a whole. In an efficient commercial implementation of this tracker, dedicated image warping boards could be used to eliminate the radial distortion of the cameras, the laser dot extraction could be performed in hardware using FPGA's (field programmable gate arrays). By implementing the entire algorithm in hardware, it would be possible to achieve extremely high update rates ensuring that the user is tracked consistently and smoothly.

Further work is needed to develop geometric constraints that would present unique solutions to the pose estimation and eliminate the need for temporal coherence. Another solution is to build a device that pulses the lasers rapidly as in [56]. By synchronizing the laser dot acquisition with the lasers, it is possible to distinguish which laser diode produced

each spot. Using this information in the SCAAT framework presents a very interesting opportunity to track the user even more reliably. This method would also allow the use of laser diodes in the infrared range where pulsed illumination of the invisible lasers is desirable to minimize the emitted power and lower the risk of retinal damage[13]. Using infrared lasers would also allow the lasers to be pointed in any direction since the human eye does not perceive light energy in the infrared range. This would resolve the possiblity of inhibiting the user's immersive experience. Another benefit to using this type of method is the possibility of enabling multiple users to be tracked simultaneously allowing collaboration within a fully-immersive display setting.

Other Laser Configurations

Alternative configurations of the laser diodes can be used that allow the user to be tracked within a fully-enclosed environment. During the development of this system, an alternate configuration in which the laser diode that defines the $\vec{up}$ vector was placed at a 45^o angle to the plane was considered. This configuration was tested in simulation and found that less constraints needed to be enforced on the solution, however it still did not provide a unique static solution and thus was not implemented in hardware.

Other methods could be used to identify the laser diodes, including

- Laser diodes with differing wavelengths.
- Time multiplexing of the laser diodes. By pulsing the lasers at known times, the

system could measure each laser projection separately. However, the drawbacks to this method include the need for synchronization between the laser system and the cameras and the lack of synchronization between measurements (needed for the pose computation discussed). In the current implementation this solution is not feasible as the digital cameras used are not equipped with external synchronization capability.

7.2.2 Improving the Inertial System

The inertial system could be augmented with sensors that provide different types of data. Using gyroscopes would allow the system to measure the rotational velocity allowing a single integration for rotational displacement. A different approach would involve the use of nine accelerometers instead of gyroscopes. By using nine accelerometers in total, the rotational acceleration is directly solvable[50] and could provide more stability in the pose estimate. An autocalibration routine could be developed to refine the estimates of the calibration parameters.

7.2.3 Other Improvements

An autocalibration method for MARVIN would be an interesting extension to this framework. The homographies from image space to screen space could be calibrated online allowing higher accuracy for the optical component. This would also reduce the time it takes to calibrate the system. An initial estimate of the transformation could be provided

manually and by tracking the device in multiple known positions and orientation would allow an iterative convergence on the proper transformations. Another extension is to employ a prediction mechanism to predict the total end-to-end system latency of the tracker. The predicted states could be sent over the network at pre-specified intervals diminishing the latency and increasing the responsiveness of the entire tracking system. Another interesting extension to this framework would be to develop an Unscented Kalman Filter that uses the nonlinear system dynamics directly as in [42].

Appendix A

Quaternions and Rotation Sequences

This appendix provides some necessary background on representing rigid body rotations in 3D. The focus here is on quaternions since the development of this thesis is dependent on this representation. The material covered here was accumulated from a selection of excellent discussions on 3D rotations (see [5, 49, 46, 14]), and the reader is strongly urged to read these for more in-depth coverage.

A.1 Euler Angles

The Euler angle representation uses three rotations about specfic orthonormal axes of the current working frame to report orientation. The axes are typically orthogonal body-fixed, orthogonal earth-fixed, or gimbal axes. These angles are commonly known as *Roll-Elevation-Azimuth* or *Roll-Pitch-Yaw* angles since the order of rotation in an earth-fixed frame is first about an axis pointing 'north' (Roll), followed by an axis pointing 'east' (Elevation), then finally by an axis pointing 'down' (Azimuth) respectively. These angles have symbols ϕ(Roll),θ(Elevation),ψ(Azimuth)

In the Euler angle representation, singularities exist when the rigid body points directly up or directly down. In this orientation, the roll and azimuth angles are not defined uniquely

and only the sum or difference of these angles are uniquely defined. This is commonly known as a gimbal lock situation (a degree of freedom is lost in this orientation). This is an unwanted side effect of the representation when dealing with rigid bodies that are capable of assuming a vertical orientation.

A.2 Rotation Matrices

A rotation in 3D can be defined as a 3×3 matrix, $R_{3\times3}$. Rotation matrices around the X,Y, and Z axes can be determined and are summarized below:

$$R_X(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) \\ 0 & -\sin(\psi) & \cos(\psi) \end{bmatrix} \tag{A.1}$$

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{A.2}$$

$$R_Z(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

In order to rotate a vector $P = [x, y, z]^T$ by these angles, the following calculation is

performed:

$$P' = R_X(\psi)R_Y(\theta)R_Z(\phi)P \tag{A.4}$$

where $P' = [x', y', z']^T$ is the new rotated vector. Typically $R = R_x R_y R_z$ is different than the euler angle representation; euler angles depend on the body frame axes of the object to be tracked while in general, rotation matrices use world-fixed axes to define the rotation.

A.3 Quaternions

Using quaternions to represent rotation has been a common practice in the computer graphics, robotics, and aerospace realms. The advantage of using this representation is that all orientations can be represented without singularities. Here, the basics of quaternions and their operations will be reviewed.

Quaternions were discovered in 1843 by Sir William R. Hamilton[27]. He described a set of operations on four-dimensional complex vectors which he named *quaternions*. The quaternion was defined as having a real scalar component and an imaginary vector component. This representation turned out to have deep impact in three-dimensional geometry by being able to describe a 3D rotation and operations upon them. There are several different notations for quaternions that are used extensively, namely

$\hat{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$	Linear combination
$\hat{q} = (w, x, y, z)$	4D vector
$\hat{q} = (w, \vec{v})$	Scalar with imaginary vector

where q is a quaternion, w is the real scalar part, and $\vec{v} = (x, y, z)$ is the imaginary vector part.

The important relationship defined by Hamilton was particularly

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{A.5}$$

and thus the following is also true

$$\begin{aligned} \mathbf{ij} &= \mathbf{k} &&= -\mathbf{ji} \\ \mathbf{jk} &= \mathbf{i} &&= -\mathbf{kj} \\ \mathbf{ki} &= \mathbf{j} &&= -\mathbf{ik} \end{aligned} \tag{A.6}$$

A.3.1 Quaternion Algebra

Operations such as equality, addition, subtraction, scalar multiplication, and quaternion multiplication can be defined for the quaternion algebra. The following useful forms of quaternions can also be defined: Norm of the quaternion, normalized unit quaternion, quaternion conjugate, and quaternion inverse.

For the following definitions, let $\hat{q}_1$ and $\hat{q}_2$ be two quaternions such that

$$\hat{q}_1 = w_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k} \tag{A.7}$$

$$\hat{q}_2 = w_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k} \tag{A.8}$$

Equality

Two quaternions are said to be equal if and only if all components are identical. Thus,

$$w_1 = w_2 \tag{A.9}$$

$$x_1 = x_2 \tag{A.10}$$

$$y_1 = y_2 \tag{A.11}$$

$$z_1 = z_2 \tag{A.12}$$

Addition

Quaternion addition (and subtraction) is defined simply as the sum(or difference) of the individual elements of each quaternion. Let $\hat{q}_3 = w_3 + x_3\mathbf{i} + y_3\mathbf{j} + z_3\mathbf{k}$ be the resulting

quaternion of the addition of $\hat{q}_1$ and $\hat{q}_2$,

$$w_3 = w_1 + w_2 \tag{A.13}$$

$$x_3 = x_1 + x_2 \tag{A.14}$$

$$y_3 = y_1 + y_2 \tag{A.15}$$

$$z_3 = z_1 + z_2 \tag{A.16}$$

$$\tag{A.17}$$

or more compactly

$$\hat{q}_3 = \hat{q}_1 + \hat{q}_2 = (w_1 + w_2) + (x_1 + x_2)\mathbf{i} + (y_1 + y_2)\mathbf{j} + (z_1 + z_2)\mathbf{k} \tag{A.18}$$

Scalar Multiplication

Multiplying a quaternion by a scalar, s, is simply the process of multiplying each element of the quaternion by s, namely

$$s\hat{q}_1 = sw_1 + sx_1\mathbf{i} + sy_1\mathbf{j} + sz_1\mathbf{k} \tag{A.19}$$

Quaternion Multiplication

This is a more complicated operator than multiplying the quaternion by a scalar. The multiplication of two quaternions is itself a quaternion. Also, in order to define quaternion

multiplication the above relationships in Equation A.5 and Equation A.6 must be used to simplify the equation. For the following let $\otimes$ denote quaternion multiplication. Thus,

$$\begin{aligned}
\hat{q}_1 \otimes \hat{q}_2 &= (w_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k})(w_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}) \\
&= w_1w_2 + \mathbf{i}x_1w_2 + \mathbf{j}y_1w_2 + \mathbf{k}z_1w_2 \\
&\quad + \mathbf{i}w_1x_2 + \mathbf{i}^2x_1x_2 + \mathbf{ij}y_1x_2 + \mathbf{ik}z_1x_2 \\
&\quad + \mathbf{j}w_1y_2 + \mathbf{ji}x_1y_2 + \mathbf{j}^2y_1y_2 + \mathbf{jk}z_1y_2 \\
&\quad + \mathbf{k}w_1z_2 + \mathbf{ki}x_1z_2 + \mathbf{kj}y_1z_2 + \mathbf{k}^2z_1z_2 \\
&= (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2) \\
&\quad + \mathbf{i}(x_1w_1 + w_1x_2 - z_1y_2 + y_1z_2) \\
&\quad + \mathbf{j}(y_1w_2 + z_1x_2 + w_1y_2 - x_1z_2) \\
&\quad + \mathbf{k}(z_1w_2 - y_1x_2 + x_1y_2 + w_1z_2)
\end{aligned} \tag{A.20}$$

which can be written more compactly as

$$\hat{q}_1 \otimes \hat{q}_2 = (w_1, \vec{v_1})(w_2, \vec{v_2}) = (w_1w_2 - v_1 \cdot v_2, w_1v_2 + w_2v_1 + v_1 \times v_2) \tag{A.21}$$

It is noted that the cross-product of the two vectors makes this operation non-commutative.

Quaternion Conjugate

The quaternion conjugate or complex conjugate of a quaternion $\hat{q}_1$ is denoted as $\hat{q}_1^*$ and is defined as

$$\hat{q}_1^* = (w_1, -\vec{v}) = w_1 - x_1\mathbf{i} - y_1\mathbf{j} - z_1\mathbf{k} \tag{A.22}$$

Quaternion Norm

It is important to note that the norm of a quaternion, $N(\hat{q})$ or $|\hat{q}|$ is a scalar value defined as

$$N(\hat{q}) = \sqrt{\hat{q}^*\hat{q}} = \sqrt{w^2 + x^2 + y^2 + z^2} \tag{A.23}$$

Normalized Unit Quaternion

An important notion is the normalized quaternion which is simply defined as

$$\hat{q}_{norm} = \frac{\hat{q}}{N(\hat{q})} \tag{A.24}$$

where $\hat{q}_{norm}$ is the quaternion $\hat{q}$ normalized to unit length.

Quaternion Inverse

The inverse of quaternion is defined as

$$\hat{q}^{-1} = \frac{\hat{q}^*}{\hat{q}^*\hat{q}} = \frac{\hat{q}^*}{N(\hat{q})} \tag{A.25}$$

For unit quaternions however, it can be seen that the inverse is identical to the conjugate namely that $\hat{q}^{-1} = \hat{q}^*$.

A.3.2 Quaternions as Rotations

It can be shown (see [46] for a beautiful and intuitive derivation) that there exists a special quaternion that represents a rotation in three-dimensional space. Also, there exists a special rotation operator that when used will rotate a vector by the amount represented by the quaternion.

This representation is sometimes called an axis-angle representation. The following special quaternion is defined for a rotation of angle θ about a unit vector representing the axis of rotation $\vec{v} = (v_x, v_y, v_z)$:

$$\hat{q}_\theta = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\vec{v}) \tag{A.26}$$

Note however that this is only true if $N^2(\hat{q}_\theta) = |\hat{q}_\theta|^2 = 1$, thus the quaternion must be normalized prior to its usage.

Quaternion Rotation Operator

In order to define a quaternion rotation operator that rotates vectors in 3D, one needs to define the following notation. A vector in 3D $\vec{v} = (v_x, v_y.v_z)$ can be expressed as a *pure*

quaternion, $\hat{\vec{v}}$, where the scalar part is set to zero. Thus,

$$\hat{\vec{v}} = (0, \vec{v}) = 0 + v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k} \tag{A.27}$$

The rotation operator to rotate a vector, $\vec{v}$, by the axis and angle represented by a quaternion, $\hat{q}$, can now be defined as :

$$\widehat{\vec{v}_r} = \hat{q} \otimes \hat{\vec{v}} \otimes \hat{q}^* = \hat{q}\hat{\vec{v}}\hat{q}^* = (0, \vec{v}_r) \tag{A.28}$$

where v_r is the rotated vector.

A.3.3 Quaternion Rates

It is possible to find the derivative of a quaternion, $\hat{q}$, which is necessary in kinematics and pose estimation methods. First, it is known that two quaternions, $\hat{q}(t)$ and $\hat{q}(t+\Delta t)$ can be related by an incremental transition quaternion. Therefore,

$$\hat{q}(t+\Delta t) = \hat{q}(t)\Delta r(t) \tag{A.29}$$

where $\Delta r(t)$ is the incremental quaternion representing angle $\frac{\alpha}{2}$ about an axis of rotation $\vec{v}$. Namely,

$$\Delta r(t) = \cos(\frac{\alpha}{2}) + \vec{v}\sin(\frac{\alpha}{2}) \tag{A.30}$$

Substituting Equation A.30 into Equation A.29 results in

$$\hat{q}(t+\Delta t) = \hat{q}(t)(\cos(\frac{\alpha}{2}) + \vec{v}\sin(\frac{\alpha}{2})) \tag{A.31}$$

And using the small angle assumption that $\cos(\Delta\theta) = 1$ and $\sin(\Delta\theta) = \theta$

$$\hat{q}(t+\Delta t) = \hat{q}(t)(1 + \vec{v}\frac{\alpha}{2}) \tag{A.32}$$

Simplifying and dividing by Δt on both sides results in

$$\hat{q}(t+\Delta t) - \hat{q}(t) = \hat{q}(t)\vec{v}\frac{\alpha}{2} \tag{A.33}$$

$$\frac{d\hat{q}}{dt} = \lim_{\Delta t \to 0} \frac{\hat{q}(t+\Delta t) - \hat{q}(t)}{\Delta t} \tag{A.34}$$

$$= \lim_{\Delta t \to 0} \frac{\hat{q}(t)\vec{v}\frac{\alpha}{2}}{\Delta t} \tag{A.35}$$

$$= \hat{q}(t)\vec{v}\frac{\omega(t)}{2} \tag{A.36}$$

$$= \frac{1}{2}\hat{q}(t)\hat{\omega}(t) \tag{A.37}$$

where $\omega(t)$ is the scalar angular rate, and $\hat{\omega}(t)$ is the angular rate pure quaternion of $\Delta r(t)$, namely the quaternion $\hat{\omega}(t) = [0, \omega(t)]$.

This is an extremely useful relation since it allows rigid bodies to be tracked without evaluating trigonometric functions, the only caveat is that the angular rate must be available.

A.3.4 Quaternion Integration

Integrating a quaternion in time simply follows the normal rules of integration. An important step however after the integration is to enforce the length of the resulting quaternion to be 1 through normalization. Thus, the Taylor series expansion for integrating a quaternion is

$$\hat{q}(t+\delta t) = \hat{q}(t) + dt\dot{\hat{q}}(t) + \frac{dt^2}{2}\ddot{\hat{q}}(t) + \mathcal{O}(dt^3) \tag{A.38}$$

In [26], a leap-frog integrator is used to perform the integration and this is defined as

$$\hat{q}(t+dt) = \hat{q}(t) + dt\dot{\hat{q}}(t+dt/2) + \mathcal{O}(dt^3) \tag{A.39}$$

$$\text{where } \hat{q}(t+dt/2) = \hat{q}(t) + \frac{dt}{2}\dot{\hat{q}}(t) \tag{A.40}$$

A.3.5 Computing the angular velocity between two frames

Given two frames representing a rotation in time by the quaternions, $\hat{q}(t)$ and $\hat{q}(t-\delta t)$, it is possible to determine the angular velocity between these two frames which may be required in many instances. This uses the previously discussed quaternion derivative formula and is

determined by

$$\dot{\hat{q}}(t) = \frac{1}{2}\hat{q}(t)\hat{\omega}(t) \tag{A.41}$$

$$\hat{q}(t)^{-1}\dot{\hat{q}}(t) = \frac{1}{2}\hat{q}(t)^{-1}\hat{q}(t)\hat{\omega}(t) \tag{A.42}$$

$$2\hat{q}(t)^{-1}\dot{\hat{q}}(t) = \hat{\omega}(t) \tag{A.43}$$

Now, in order to compute the above, it is necessary to have an estimate of $\hat{\dot{q}}(t)$. Given that in a tracking scenario, the two quaternions to be compared are within a very small time interval, a first order approximation to the derivative is sufficient. The relationship

$$\dot{\hat{q}}(t) = \frac{\hat{q}(t) - \hat{q}(t-\delta t)}{\delta t} \tag{A.44}$$

is sufficient to compute the angular velocity using $\hat{\omega}(t) = 2\hat{q}(t)^{-1}\hat{\dot{q}}(t)$.

A.3.6 Error Quaternion or Estimating the Rotation between Two Frames

In order to estimate the rotation, $\hat{q}_{12}$, between two frames represented by quaternions, $\hat{q}_1$ and $\hat{q}_2$, one computes

$$\hat{q}_{12} = \hat{q}_2\hat{q}_1^{-1} \tag{A.45}$$

where $\hat{q}_{12}$ is a quaternion that represents a rotation that rotates vectors expressed in frame 1 to vectors expressed in frame 2.

A good application of this in a tracking system is to evaluate the error in an estimated

pose given knowledge of the ground truth pose. Thus,

$$\hat{q}_{\varepsilon} = \hat{q}_{actual}\hat{q}^{-1}_{estimated} \tag{A.46}$$

A.3.7 Quaternions and other representations

Since all of the discussed representations fundamentally represent the same rotation, it is possible to convert a rotation described by a quaternion into other representations. It is also possible to perform the reverse and convert other representations to quaternions if it is needed. Here, algorithms will be presented to convert between quaternions, euler angles, and rotation matrices. These are useful if it is desired to use quaternions with efficient implementations of matrix algebra or in systems where the internal representation is based on Euler angles but where quaternions would be advantageous to be used, as in graphics or robotics applications.

Quaternions to Rotation Matrices

It is possible to convert a quaternion, $\hat{q} = (w,x,y,z)$ into a 3×3 rotation matrix, R_q, using the following:

$$R_q = \begin{bmatrix} 2w^2+2x^2-1 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 2w^2+2y^2-1 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 2w^2+2z^2-1 \end{bmatrix} \tag{A.47}$$

Quaternion to Euler Angles

Since a rotation sequence using Euler angles is defined by

$$
\begin{aligned}
M &= M(\phi)M(\theta)M(\psi) \\
&= \begin{bmatrix}
\cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\
\cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) \\
-\sin(\psi)\cos(\phi) & +\cos(\psi)\cos(\phi) & \\
\cos(\psi)\sin(\theta)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) \\
+\sin(\psi)\sin(\phi) & -\cos(\psi)\sin(\phi) &
\end{bmatrix} \qquad \text{(A.48)} \\
&= \begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33}
\end{bmatrix} \\
&= R_q \qquad \text{(A.49)}
\end{aligned}
$$

It is possible to derive the angles from R_q from the following:

$$\tan(\psi) = \frac{m_{12}}{m_{11}} \tag{A.50}$$

$$\sin(\theta) = -m_{13} \tag{A.51}$$

$$\tan(\phi) = \frac{m_{23}}{m_{33}} \tag{A.52}$$

where $\hat{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ and

$$m_{11} = 2w^2 + 2x^2 - 1 \tag{A.53}$$

$$m_{12} = 2xy - 2wz \tag{A.54}$$

$$m_{13} = 2xz + 2wy \tag{A.55}$$

$$m_{23} = 2yz - 2wx \tag{A.56}$$

$$m_{33} = 2w^2 + 2z^2 - 1 \tag{A.57}$$

Rotation Matrix to Quaternion

To convert a rotation matrix, M, to a quaternion, $\hat{q}$, it is possible to use Equation A.49 to write

$$4wx = m_{23} - m_{32} \tag{A.58}$$

$$4wy = m_{31} - m_{13} \tag{A.59}$$

$$4wz = m_{12} - m_{21} \tag{A.60}$$

and $trace(M) = 4w^2 - 1$.

It follows that

$$w = \frac{\sqrt{m_{11} + m_{22} + m_{33} + 1}}{2} \tag{A.61}$$

and finally

$$x = \frac{m_{23} - m_{32}}{4w} \tag{A.62}$$

$$y = \frac{m_{31} - m_{13}}{4w} \tag{A.63}$$

$$z = \frac{m_{12} - m_{21}}{4w} \tag{A.64}$$

Euler Angles to Quaternion

It is also possible to convert the Euler angle representation to a quaternion.

Given that ψ is the Azimuth angle, θ is the Elevation angle, and ϕ is the Roll angle, the

quaternion is defined as:

$$\hat{q}_e = \hat{q}_z \hat{q}_y \hat{q}_x = \hat{q}_z \otimes \hat{q}_y \otimes \hat{q}_x \tag{A.65}$$

and

$$\hat{q}_z = \cos\left(\frac{\Psi}{2}\right) + \cos\left(\frac{\Psi}{2}\right)\mathbf{k} \tag{A.66}$$

$$\hat{q}_y = \cos\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)\mathbf{j} \tag{A.67}$$

$$\hat{q}_x = \cos\left(\frac{\phi}{2}\right) + \cos\left(\frac{\phi}{2}\right)\mathbf{i} \tag{A.68}$$

Appendix B

The Kalman Filter

This appendix reviews the Kalman filter. For more detail please see the extensive literature on this topic, in particular [43, 76, 10, 48, 17] provide a solid review of the Kalman filter.

The Kalman filter is essentially a set of mathematical equations that estimate a linear process with known dynamics. The computer vision, graphics, and robotics fields have used this formulation to estimate many processes including target tracking, dead reckoning, robot pose maintenance, and the fusion of information from multiple sensors with differing characteristics. Even though the *true* version of the Kalman filter introduced in [43] was proven to be an "optimal" estimation filter for linear dynamics, there have been several extensions to the framework including the extended Kalman filter (EKF) which is a *sub-optimal* filter linearizing the non-linear system dynamics. However, the EKF is used extensively since it performs well in many instances. Another extension to this framework is the Unscented Kalman filter[42] which develops a Kalman filter framework that uses the non-linear system dynamics directly.

This appendix introduces the discrete linear Kalman filter and extended Kalman filter followed by a brief overview of other interesting extensions to the Kalman framework.

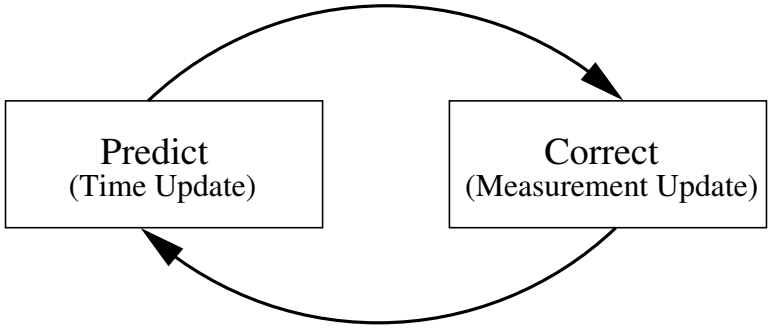


Figure B.1: The Predictor-Corrector Feedback mechanism of the Kalman filter.

In general, a linear stochastic process can be described in state vector notation as

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{B.1}$$

and a measurement of the system is described as

$$z_k = Hx_k + v_k \tag{B.2}$$

where w_k and v_k represent the noise characteristics of the system and measurement processes respectively. Kalman assumed that the noise processes of the system and measurement are random, independent, zero-mean and with normal probability distibutions. x_k is the $n \times 1$ state vector of n variables to be estimated, A is the $n \times n$ system dynamics matrix relating the system at time step $k-1$ to the next time step k, u_k is the optional control input to the system and B relates the control input to the state. z_k denotes the $m \times 1$ measurement vector and H is a $m \times n$ matrix that relates the state to the measurement.

B.1 The Discrete Linear Kalman Filter

The Kalman filter estimates the process of n variables using a recursive predictor-corrector feedback loop formulation (see Figure B.1). Essentially, the filter first predicts what the state should look like at a given time step, it then obtains a *noisy* measurement from some sensor. The measurement is then incorporated into the state using the residual of the prediction and the measurement weighted by a gain factor that is computed with the noise characteristics of the process and measurement. The notation that is used here follows [76] and is summarized in Table B.1.

Notation	Size	Description
$\hat{x}_k$	$n \times 1$	State vector of variables, estimated at time step k
z_k	$m \times 1$	Measurement vector at time step k
A	$n \times n$	System dynamics matrix
u_k	$l \times 1$	Optional control input at time step k
B	$n \times l$	Matrix that relates control input to the state
P_k	$n \times n$	State covariance matrix at time step k
Q	$n \times n$	Process noise covariance matrix
R	$m \times m$	Measurement noise covariance matrix
K	$n \times m$	Kalman gain matrix
H	$m \times n$	Matrix relating state to measurement
I	$n \times n$	Identity matrix
$\hat{x}_k^-$	$n \times 1$	Predicted state
P_k^-	$n \times n$	Predicted covariance

Table B.1: The notation used in the form of the Kalman Filter discussed here.

In order to accomplish the task of recursively estimating the state variables, the Kalman filter performs a *time-update* step where it predicts the state vector and the process covari-

ance without any additional measurements by

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{B.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{B.4}$$

After the time-update step, a measurement is obtained and the state must be updated appropriately. This is called the *measurement-update* step. First, the Kalman gain factor is computed using the predicted state covariance matrix and the sensor noise characteristics. The residual between the predicted state and the measurement is computed and weighted by the gain. The state is then updated by incorporating the predicted state and the weighted residual. The final step is to update the state covariance matrix using the computed gain and the predicted state covariance.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{B.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{B.6}$$

$$P_k = (I - K_k H)P_k^- \tag{B.7}$$

B.1.1 An Example

As an example of using the Kalman filter, a simple process of estimating the position of a particle in 2D is presented. To demonstrate the use of a linear Kalman filter, imagine

Time Update Step
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update Step
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
$P_k = (I - K_k H)P_k^-$

Table B.2: Summary of the Discrete Linear Kalman Filter Equations

a particle that can be modeled to move at constant velocity in a straight line. Thus, the differential equation governing this motion would be

$$x_k = x_{k-1} + \dot{x}_{k-1}\Delta t \tag{B.8}$$

$$\dot{x}_k = \dot{x}_{k-1} \tag{B.9}$$

Using this model, a particle $P = [p_x, p_y]$ moves in a straight line with constant velocity $\vec{v} = [v_x, v_y]$. Thus the state vector will be

$$x_k = [p_x, p_y, v_x, v_y]^T \tag{B.10}$$

The 4×4 system dynamics matrix, A, would then be defined as

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.11}$$

Now, imagine that a device exists to measure the position of the particle at regular time intervals and each measurement is independent and has a 2.45cm RMS white noise covariance on both the X and Y axes. Thus, the measurement noise covariance matrix is

$$R = \begin{bmatrix} 2.45^2 & 0 \\ 0 & 2.45^2 \end{bmatrix} \tag{B.12}$$

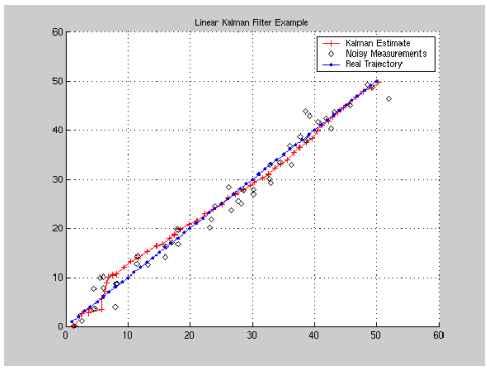
In order to relate the 4×1 state vector with the 2×1 measurement vector, H needs to be defined as

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{B.13}$$

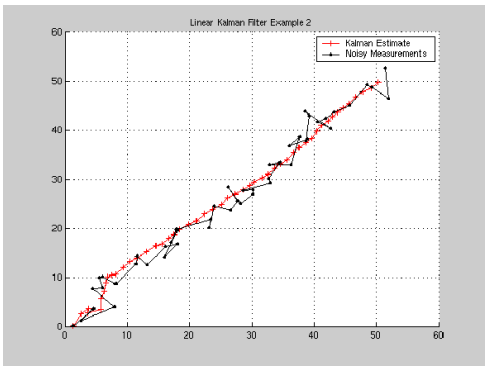
For simplicity, the process noise covariance will be set as $Q = 1e^{-5}$ and $\Delta t = 1$.

Finally, to run some experiments on simulated data, Matlab was used to generate ground-truth data which was then perturbed by a white noise of 2.45cm variance to simulate the measurements from the fictional device.

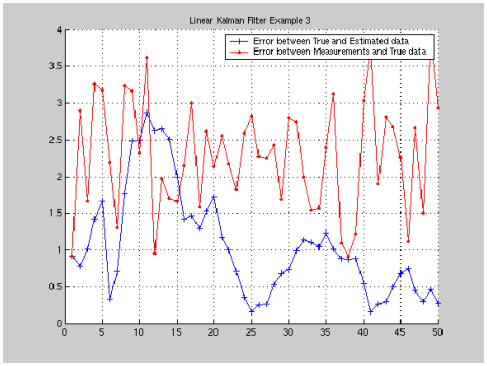
The noisy data was then run through the Kalman filter equations recording the state



(a)



(b)



(c)

Figure B.2: Results of the linear Kalman filter with a position-velocity model.

estimate at each time step (see Figure B.2(a)). Figure B.2(b) shows the state estimation data with the noisy data alone for clarity and illustration purposes. The error was then computed between the estimated state and the ground-truth data (shown in Figure B.2(c)) with the error computed between the ground-truth data and the noisy measurement data. It can be seen that the filter error decreases as time increases. The Matlab code that implements this Kalman filter can be seen in Listings B.1 and B.2

Listing B.1: Time Update Step of the Kalman Filter

```
function [Xp, Pp] = time_update(A,X,P,B,Control,Q)
%% time update step of the kalman filter
% returns the predicted state X, and covariance Pp
Xp = A*X + B*Control;
Pp = A*P*A' + Q;
```

Listing B.2: Measurement Update Step of the Kalman Filter

```
function [X,P] = measurement_update(z,Xp,Pp,H,I,R)
%% measurement_update step of the kalman filter
% returns the updated state, X and covariance P

% compute the Kalman Gain
temp = H*Pp*H' + R;
K = Pp*H'*inv(temp);

% Compute the residual of the measurement and
% the predicted state
Residual = z - H*Xp;

% update the state with the weighted residual
X = Xp + K*Residual;

% update the Covariance
P = (I - K*H)*Pp;
```

B.2 The Extended Kalman Filter

The formulation that Kalman derived was proven to be an optimal estimation filter for discrete linear systems. It is rare in computer vision, graphics and robotics, that the system dynamics of the process to be estimated is in fact linear. The process or measurement relationship typically is governed by a non-linear system of equations. In order to use the Kalman filter framework for these non-linear systems, an extension named the *extended* Kalman filter is used which linearizes the system about the current mean and covariance.

The process to be estimated is now governed by a non-linear function, namely

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{B.14}$$

and the measurement now becomes a function as well,

$$z_k = h(x_k, v_k) \tag{B.15}$$

The notation used in the extended Kalman filter is summarized in Table B.3.

Notation	Size	Description
$\hat{x}_k$	$n \times 1$	State vector of variables, estimated at time step k
z_k	$m \times 1$	Measurement vector at time step k
u_k	$l \times 1$	Optional control input at time step k
P_k	$n \times n$	State covariance matrix at time step k
Q	$n \times n$	Process noise covariance matrix
R	$m \times m$	Measurement noise covariance matrix
K	$n \times m$	Kalman gain matrix
I	$n \times n$	Identity matrix
A	$n \times n$	Jacobian Matrix of Partial derivatives of $f(\cdot)$ with respect to x $A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0)$
W	$n \times n$	Jacobian Matrix of Partial derivatives of $f(\cdot)$ with respect to noise w $W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_k, 0)$
H	$m \times n$	Jacobian Matrix of Partial derivatives of $h(\cdot)$ with respect to x $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_k, 0)$
V	$m \times m$	Jacobian Matrix of Partial derivatives of $h(\cdot)$ with respect to noise v $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{x}_k, 0)$
$\hat{x}_k^-$	$n \times 1$	Predicted state
P_k^-	$n \times n$	Predicted Covariance

Table B.3: The notation used in the form of the extended Kalman filter discussed here.

The *time-update* equations in this framework now become

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{B.16}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{B.17}$$

and the *measurement-update* equations are now

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{B.18}$$

$$\hat{x}_k = \hat{x}_{k-1}^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{B.19}$$

$$P_k = (I - K_k H_k) P_k^- \tag{B.20}$$

Note that the Jacobians A, W, H, V are different at each timestep and must be recalculated. The Jacobian H_k is important since it propagates only the relevant component of the measurement information. It magnifies only the portion of the residual that affects the state.

Time Update Step
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0)$
$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
Measurement Update Step
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$
$\hat{x}_k = \hat{x}_{k-1}^- + K_k(z_k - h(\hat{x}_k^-, 0))$
$P_k = (I - K_k H_k) P_k^-$

Table B.4: Summary of the Extended Kalman Filter Equations

B.3 The SCAAT Kalman Filter

SCAAT[75], or the *single-constraint-at-a-time* algorithm, is another important extension to the Kalman filter framework. It was designed to be used in systems where the sensors are not synchronized and provide incomplete information about the estimated process. The basic approach is to change the idea of predicting the **state** and computing the residual between the predicted state and the measurement, to predicting the **measurement** using the current state estimate and then computing the residual between the predicted measurement and the actual sensor measurement.

This was implemented for the HiBall®[75] tracking system which saw a noticeable decrease in latency and an increase in accuracy. The algorithm also provides a way to perform autocalibration of the system or sensors which can be very valuable in practice.

The algorithm assumes that the measurement is an incomplete representation of the world, i.e. use the position of one LED, or output of one accelerometer. This leads to a *locally unobservable* system but over time the measurements are able to provide a *globally observable* system. Each sensor type, σ, uses a measurement prediction function $h(\cdot)$ that must be defined relating the state to the measurement obtained by the sensor. This function takes the current estimate of the state, the sensor-source pair, and computes the measurement that the sensor should report at this time step. The SCAAT algorithm is summarized below, however the reader is urged to see [75] for more detail, the derivation of the filter, and experimental results.

The process to be estimated is a constant velocity model and the orientation is represented as three small changes in angles. It is assumed that the measurement update rate is fast enough and a linear prediction is used for simplicity. The state to be estimated is thus $[x, y, z, \dot{x}, \dot{y}, \dot{z}, \Delta\phi, \Delta\theta, \Delta\psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$. Also, to avoid gimbal lock scenarios, the global orientation is stored outside of the filter as a quaternion, $\hat{q}$, and is updated at each step using the incremental angles estimated in the filter state. Also, an important step is to zero the incremental angles at the end of the measurement-update.

The time-update step is defined as in the linear Kalman filter, namely

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{B.21}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{B.22}$$

The measurement-update is then performed in several steps discussed below. The next step is to predict the measurement using the current state, sensor and source information. The sensor is defined as the device that obtains the measurement (i.e. camera) and the source is defined as the device that emits the information to be sensed (i.e. a single LED).

$$\hat{z}_k = h_{k,\sigma}(\hat{x}_k^-, b_k, c_k) \tag{B.23}$$

$$H_k = H_{k,\sigma}(\hat{x}_k^-, b_k, c_k) \tag{B.24}$$

where b_k is the sensor parameter vector, c_k is the source parameter vector, $H_{k,\sigma}$ is a jacobian

matrix of partial derivatives of $h_k(\cdot)$ with respect to x, namely

$$H_{k,\sigma}(\hat{x}_k^-, b_k, c_k)_{[i,j]} = \frac{\partial}{\partial x_{[j]}} h_{k,\sigma}(\hat{x}_k^-, b_k, c_k)_{[i]} \tag{B.25}$$

After predicting the measurement, the Kalman gain is computed

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_{k,\sigma})^{-1} \tag{B.26}$$

Then, the residual between the predicted measurement and the obtained measurement from the sensor-source pair is computed,

$$\Delta z_k = z_{k,\sigma} - \hat{z}_k \tag{B.27}$$

The state and covariance are then updated properly,

$$\hat{x}_k = \hat{x}_k^- + K_k \Delta z_k \tag{B.28}$$

$$P_k = (I - K_k H_k) P_k^- \tag{B.29}$$

Finally, an important update must take place. The external quaternion must be updated using the estimated small angle changes and then the state vector components of these angles must be set to zero.

$$\Delta\hat{q} = \text{quaternion}(\hat{x}_k(\Delta\phi), \hat{x}_k(\Delta\theta), \hat{x}_k(\Delta\psi)) \tag{B.30}$$

$$\hat{q} = \hat{q} \otimes \Delta\hat{q} \tag{B.31}$$

$$\hat{x}_k(\Delta\phi) = \hat{x}_k(\Delta\theta) = \hat{x}_k(\Delta\psi) = 0 \tag{B.32}$$

Time Update Step

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update Step

1. Predict the measurement and compute H_k

$$\hat{z}_k = h_{k,\sigma}(\hat{x}_k^-, b_k, c_k)$$

$$H_k = H_{k,\sigma}(\hat{x}_k^-, b_k, c_k)$$

2. Compute the Kalman gain factor

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_{k,\sigma})^{-1}$$

3. Compute the residual

$$\Delta z_k = z_{k,\sigma} - \hat{z}_k$$

4. Update the state and covariance

$$\hat{x}_k = \hat{x}_k^- + K_k \Delta z_k$$

$$P_k = (I - K_k H_k) P_k^-$$

5. Update external quaternion and incremental angles

$$\Delta\hat{q} = \text{quaternion}(\hat{x}_k(\Delta\phi), \hat{x}_k(\Delta\theta), \hat{x}_k(\Delta\psi))$$

$$\hat{q} = \hat{q} \otimes \Delta\hat{q}$$

$$\hat{x}_k(\Delta\phi) = \hat{x}_k(\Delta\theta) = \hat{x}_k(\Delta\psi) = 0$$

Table B.5: Summary of the SCAAT algorithm

Appendix C

Hardware Details

C.1 Inertial Device

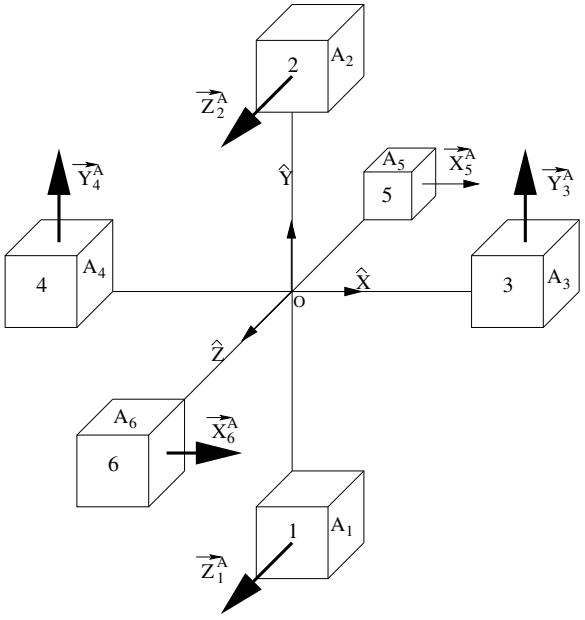
The Inertial device is composed of six accelerometers arranged in pairs such that each pair of accelerometers are orthogonal to the other pairs. Figure C.1 shows the configuration of the device and the reference frames attached to each accelerometer.

C.1.1 Acceleromter Details

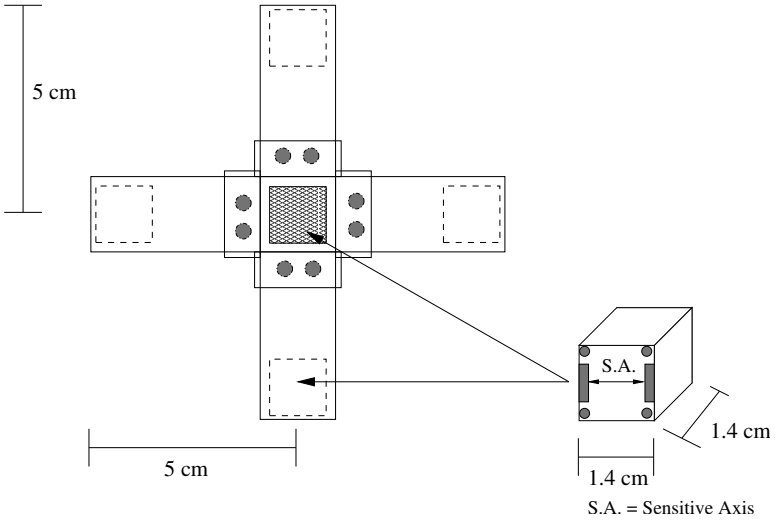
The AS-5GA accelerometers were purchased from Kyowa Electronic Intstruments Co., Ltd. in Tokyo, Japan. The sensors are rated at $5G(49.03m/s^2)$ typically and have characteristics as shown in Table C.1.

Frequency Response:	$0 \sim 110Hz \pm 5\%$
Rated Output:	$\frac{640\mu V/V}{1280\times10^{-6}}$
Nonlinearity:	$1.00\%RO$
Calibration Constant:	$\frac{0.07660m/s^2(0.007812G)/1\mu V/V}{0.03830m/s^2(0.003906G)/1\times10-6}$
Input & Output resistance:	Input: 122.8Ω
	Output: 122.8Ω

Table C.1: Typical Characteristics of Accelerometer AS-5GA S/N:EK4970005



(a) Ideal Inertial device with labeled axes



(b) CAD drawing of inertial device

Figure C.1: Accelerometer Configuration

Amplification Circuit

The amplification circuit (Figure C.2) was designed by RHC & Associates in Mississauga, Ontario. If the circuit is provided with an excitation voltage, it will regulate and amplify the voltage to range between $\pm 2.0V$. Thus, $1G = 400mV$.

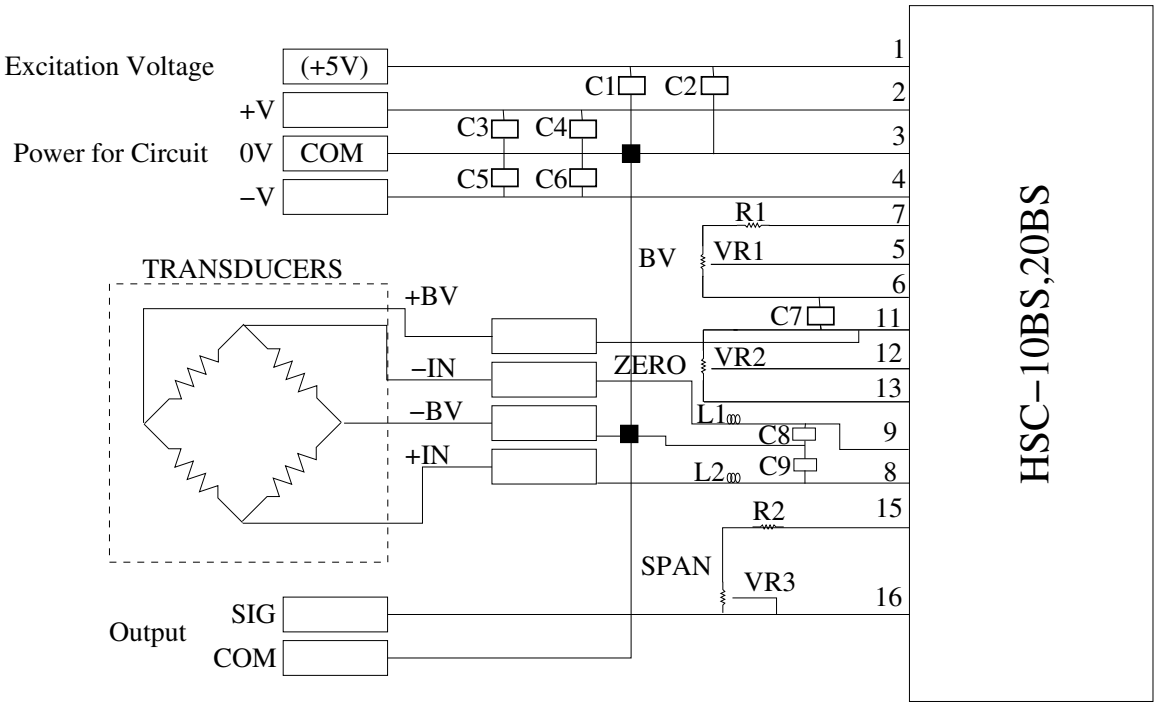


Figure C.2: Amplification Circuit from RHC & Associates for Accelerometers. Capacitors C1,3,5: 25V 100μF. Capacitors C2,4,6,7,8,9: 0.01μF. L1,L2 are 10μH coils. Fixed Resistor R1 is 33K ohm and the variable resistor VR1 is 5K ohm. with adjustment range 3.9V to 5.8V. The Excitation voltage used for the transducer is 5V. The power for the circuit is $\pm$15V when using the standalone powersupply and $\pm$5V when using camcorder batteries.

C.1.2 A/D Converter Details

In order to convert the analog signal provided by the accelerometer signal conditioning circuit to a usable digital signal, a VIPS (Virtual Instrument Pod System) Analog to Digital (A/D) converter was purchased that connects to the standard parallel port of the wearable computer. See Table C.2 for technical specifications.

Number of Channels:	8 single ended or 4 differential or mixture of both
Resolution:	12 bits
Conversion Time:	$10\mu s$
Sampling Rate:	$20\mu s$ per channel
Input Voltage Range:	$0 \sim 4.096V$ (unipolar) $\pm 2.047V$(bipolar)
Internal Reference:	$4.096V \pm 3mV$ (trimmed)
Internal Ref Temp Coefficient:	$\pm 50ppm/^oC$max.
External Reference Range:	$1.5V \sim 5.0V$
Relative Accuracy:	$\pm 1LSB$ max.
Offset Error:	$\pm 3LSB$ max.
Max. Permissible Input Voltage :	$\pm 6V$

Table C.2: Hardware Specifications for VIPS 10 A/D Convertor

C.2 Laser Device

The Laser system device is comprised of four lasers, three of which are orthogonal to each other and define the orientation plane. The fourth laser lies on the plane and points in the opposite direction of one of the other lasers for determining position in the plane. A CAD drawing of the laser device is shown in Figure C.3.

C.2.1 Laser Details

The lasers were purchased from *The Laser Guy* wholesaler (http://www.thelaserguy.com/) for $10 USD plus shipping, handling, and importation costs.

Operating Current:	60 mA
Operating Voltage:	2.6 VDC to 4.0 VDC
Spot size:	At 5 meters is 6mm diameter
Operating Temperature:	-10 C to 40 C
Range:	2600 Feet
Diode:	645 nm, 5 mW industrial grade
Life:	rated at 12 000 to 15 000 hours on constant operation

Table C.3: Laser Diode Characteristics

C.2.2 Laser Optical Filter

The optical filters were purchased from Coherent (ph:(530-889-5365)). The filters (id:35-3961-000) are 25.4mm diameter and have a peak response at 650nm.

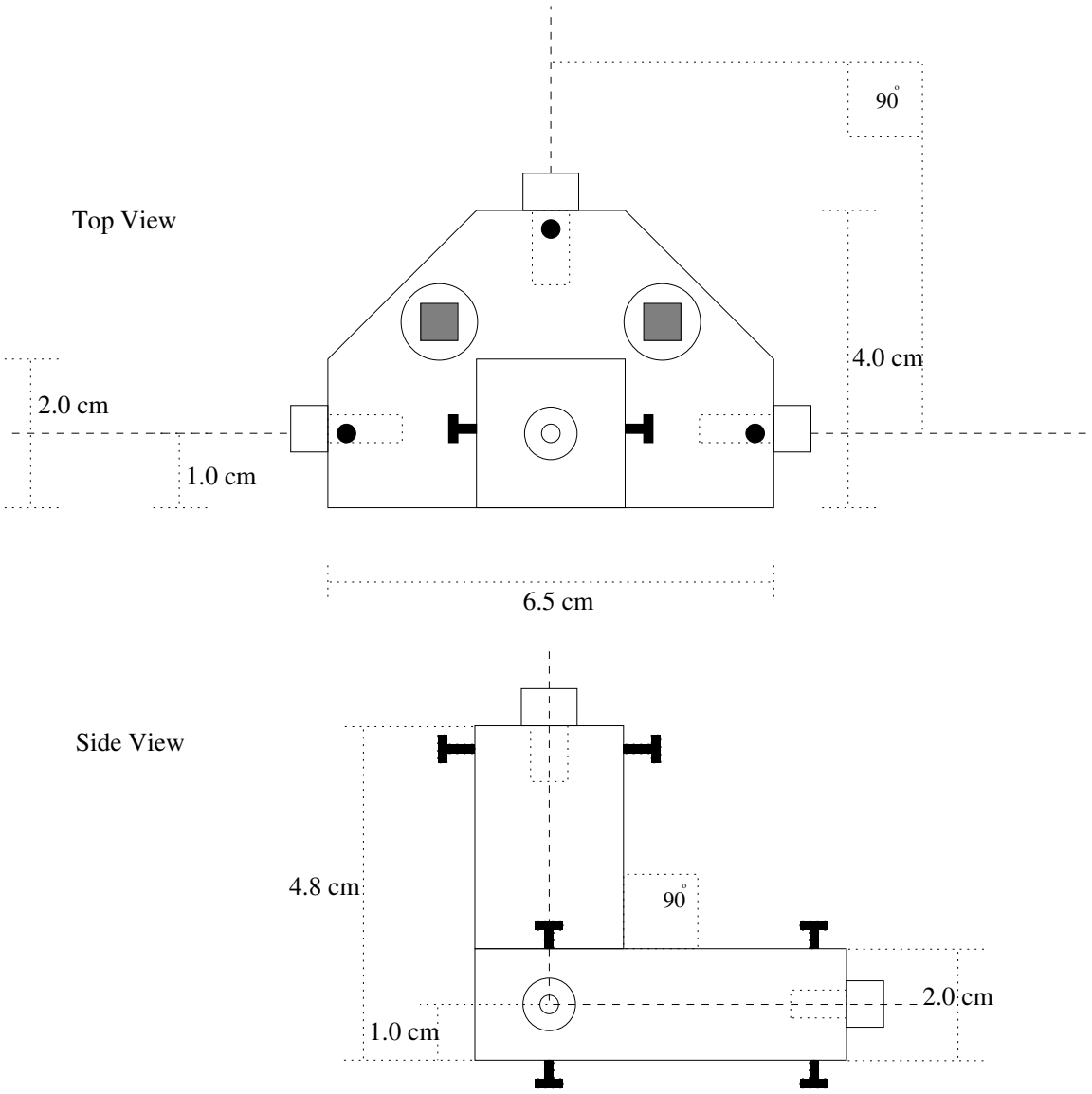


Figure C.3: The Laser System Hardware

C.3 Camera Details

The Firewire® *Fire-i400*™ industrial cameras (Figure C.4) were purchased from Unibrain. See Figure C.5 for the shutter speeds that these cameras support.

Figure C.4: Unibrain Fire-i400 industrial camera

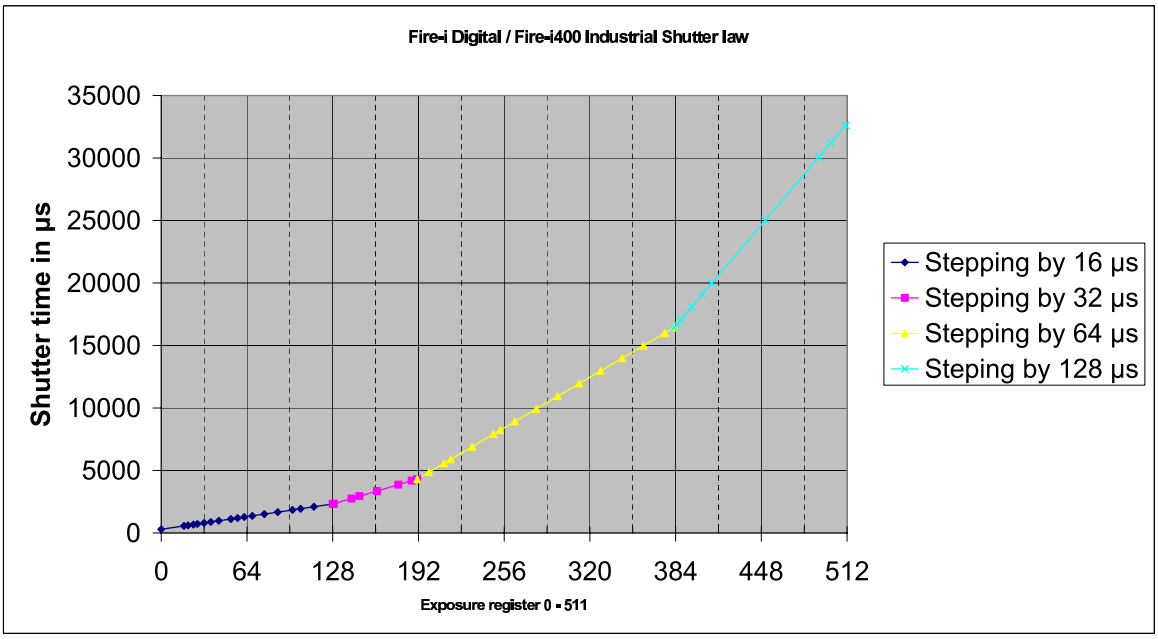


Figure C.5: Unibrain Fire-i400 Shutter Speeds and associated register values.

Interface:	IEEE-1394a 400 Mbps, 2ports (6pins)
Camera Type:	IIDC-1394 Digital Camera, V1.04 Specification compliant
Sensor Type:	Sony Wfine* 1/4" CCD Color, progressive *Wfine CCD is a trademark of Sony Corporation
Max Resolution:	VGA 640x480
Optics:	C-Mount
Video Modes:	YUV, RGB-24bit, Mono-8bit
Frame Rates:	30,15,7.5,3.75 frames per second
Gain:	Auto or Manual 0-30 dB
Shutter:	Auto or Manual 1/3400s - 1/31s
Gamma:	ON/OFF
White Balance:	Automatic or Manual Control
Color Saturation:	Adjustable
Backlight Compensation:	6 modes + OFF
Sharpness:	Adjustable
Power Supply:	8 to 30 VDC, by 1394 bus Consumption 1W max, 0.9W typical

Table C.4: Unibrain Fire-i400 Digital Camera Specifications

C.4 Wearable Computer Details

The PC/104 computer was purchased from VersaLogic Inc. Note that since the time of purchasing, AMD has discontinued the CPU line and this EPM-CPU-7 module has been updated to a Pentium CPU rather than the AMD K6 that is used here.

The PC has an AMD K6 CPU module, and a PCMCIA Module also purchased from VersaLogic. The specifications of the wearable computer are found in Table C.5 and an image of the actual hardware is shown in Figure C.6.

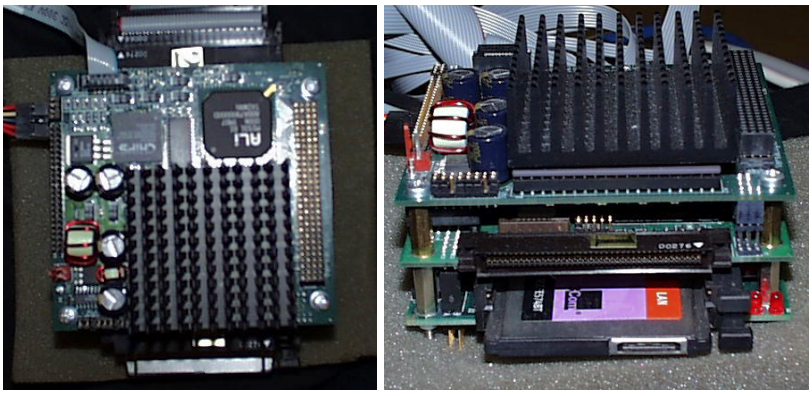


Figure C.6: The Panther PC/104 Wearable Computer from Versalogic.

Board Size:	3.55" x 3.775" (PC/104 standard). 4.23" x 3.775" including connectors. Two board set. Height: 1.9"
Storage Temperature:	-40^o C to $+85^o$ C
Operating Temperature:	Standard versions: 0^o C to $+60^o$ C Extended temperature version: -20^o C to $+85^o$ C
Power requirements:	$+5V \pm 5\%$ @ 3.50A typ. 17.5W (EPM-CPU-6/7c) $+5V \pm 5\%$ @ 4.25A typ. 21.3W (EPM-CPU-6g) $+5V \pm 5\%$ @ 1.95A typ. 9.7W (EPM-CPU-7s) $+5V \pm 5\%$ @ 2.05A typ. 10.3W (EPM-CPU-7t) 3.3V or ś12V required by some expansion modules
Bus Speed:	CPU External: 66 MHz PCI, PC/104-Plus: 33 MHz PC/104: 8 MHz
System Reset:	Vcc sensing, resets below 4.70V typ. Watchdog timeout
Humidity:	Less than 95%, noncondensing
DRAM Interface:	One 144-pin SODIMM socket, 8 to 256 MB, EDO (60 ns) or 3.3V SDRAM (66 MHz or PC-100).
Flash/BBSRAM Interface:	One 32-pin JEDEC DIP socket. Accepts battery-backed static RAM chip or M-Systems DiskOnChip.
Video Interface:	Based on Intel/C&T 69000 chip, 2 MB VRAM standard. 69030 chip with 4 MB VRAM optional. 44-pin flat panel display interface.
IDE Interface:	One channel PCI-based. Supports up to two IDE devices. Supports high speed Mode 4 and Ultra DMA drives.
Floppy Disk Interface:	Supports two floppy drives.
Ethernet Interface:	Autodetect 10BaseT / 100BaseTX based on AMD 79C973.
COM 1 Interface:	RS-232, 16C550 compatible, 115K baud max.
COM 2 Interface:	RS-232/422/485, 16C550 compatible, 460K baud max.
LPT Interface:	Bi-directional/EPP/ECP compatible.
Connectors:	I/O: Two high-density 80-pin Video: 10-pin .1" CRT connector 44-pin 2mm FPD connector Power: 10-pin .1"
Compatibility:	PC/104 - Full compliance PC/104-Plus - Full compliance, 3.3V or 5V modules

Table C.5: Specifications of the EPM-CPU-7 PC/104 Computer.

Appendix D

Company Information

Table D.1 is a list of companies that manufacture and sell tracking equipment for virtual reality applications. Table D.2 is a list of companies that manufacture hardware components that was used in the development of this thesis.

Company	Products	Address
Intersense	IS-900 InertiaCube^2	1 North Avenue Burlington, MA 01803, USA ph: (781)270-0090 fax: (781)229-8995 http://www.isense.com/
Ascension Technology Corporation	Flock of Birds® MotionStar®Wireless 3D-Bird ® LaserBIRD®	P.O. Box 527 Burlington, VT 05402, USA ph: (802)893-6657 fax: (802)893-6659 http://www.ascension-tech.com/
Polhemus	FASTRAK®	40 Hercules Dr. Colchester, VT 05446, USA ph: (802)655-3159 fax: (802)655-1439 http://www.polhemus.com/
3rdTech	HiBall®	119 E. Franlin St., 3rd Floor Chapel Hill, NC 27514-3620, USA ph: (919)929-1903 fax: (919)929-2098 http://www.3rdtech.com/
Meta Motion	Gypsy^{TM}Jr.	258 Bush St. #1 San Fransisco, CA 94104, USA ph: (415)-550-META fax: (415)-550-6384 http://www.metamotion.com/

Table D.1: Tracking Equipment Companies. These companies sell tracking equipment and other products mentioned throughout this thesis.

Company	Products	Address
TTi	VIPS10 A/D Converter	Thurlby Thandar Instruments Glebe Road, Huntingdon Cambridgeshire, PE18 7DX England http://www.tti-test.com
RHC & Associates	Transducer Amplification Circuit	RHC & Associates 2180 Dunwin Drive, Unit 4 Mississauga, ON, L5L 5M8 ph:(905) 828-6221 fax:(905)828-6408 Contact: Michel Therrien http://www.rhctest.com
Unibrain	Fire-i400^{TM}Digital Cameras	Unibrain Inc. One Annabel Lane, Suite 109 San Ramon, CA, 94583 ph:(925)-866-3000 fax:(925)-866-3520 http://www.unibrain.com
Coherent	Optical Filters	Coherent Inc. 2303 Lindbergh Street Auburn, CA, 95602 ph:(530)889-5365 fax:(530)889-5366 http://www.coherentinc.com
The Laser Guy	Laser diodes	The Laser Guy P.O. Box 849 1502 Second Street Seabrook, Texas, 77586 ph:(281)455-8008 (local) fax:(603)590-5267 http://www.thelaserguy.com
VersaLogic	PC/104 Wearable Computer EPM-7 (Panther)	VersaLogic Corp. 3888 Stewart Rd. Eugene, Oregon, 97402 ph:(541)485-8575 fax:(541)485-5712 http://www.versalogic.com

Table D.2: Hardware Component Companies. Specific hardware components used in this thesis were purchased from these companies.

Bibliography

[1] R. S. Allison, L. R. Harris, A. Hogue, U. Jasiobedzka, H. Jenkin, M. Jenkin, P. Jaekl, J. Laurence, G. Pentile, F. Redlick, J. Zacher, and D. Zikovitz. Simulating self motion ii: A virtual reality tricycle. *Virtual Reality*, 6:86–95, 2002.

[2] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, and J. E. Zacher. Tolerance of temporal delay in virtual environments. In *IEEE Int. Conference on Virtual Reality*, volume 3, 2001.

[3] R. S. Allison, I. P. Howard, and J. E. Zacher. Effect of field size, head motion and rotational velocity on roll vection and illusory self-tilt in a tumbling room. In *Perception*, volume 28, pages 299–306, 1999.

[4] R. Azuma. Predictive tracking for augmented reality. Technical Report TR95-007, UNC Chapel Hill, Dept. of Computer Science, 1995.

[5] E. Bachmann. *Inertial and Magnetic Tracking of Limb Segment Orientation for Inserting Humans into Synthetic Environments*. PhD thesis, Naval Postgraduate Institute, Monterey, CA., 2000.

[6] E. R Bachmann, R. B. McGhee, X. Yun, and M. J. Zyda. Inertial and magnetic posture tracking for inserting humans into networked virtual environments. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 9–16, Banff, Alberta, Canada, November, 15-17 1985. (slides).

[7] A. Bjorck. *Numerical Methods for Least Squares Problems*. S.I.A.M. Society for Industrial and Applied Mathematics, 1996.

[8] J. Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguet/calib_doc.

[9] K. R. Britting. *Inertial Navigation Systems Analysis*. Wiley-Interscience, New York, 1971.

[10] E. Brookner. *Tracking and Kalman Filtering Made Easy*. John Wiley & Sons Inc., 1998.

[11] C. Broxmeyer. *Inertial Navigation Systems*. McGraw-Hill, New York, 1964.

[12] X. Chen and J. Davis. Lumipoint, multi-user laser-based interaction on large tiled displays. Technical Report CS-2000-06, Stanford University, 2000. http://graphics.stanford.edu/papers/multiuser/.

[13] Ascension Technology Corporation. The laserBIRD. http://www.ascension-tech.com/products/laserbird.php.

[14] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1989.

[15] J. L. Crowley and Y. Demazeau. Principles and techniques for sensor data fusion. *Signal Processing*, 32(1-2):5–27, 1993.

[16] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection based virtual reality: The design and implementation of the cave. In *Proc. SIGGRAPH '93*, pages 135–142, 1993.

[17] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[18] J. L Farrell. *Integrated Aircraft Navigation*. Academic Press, New York, 1976.

[19] O. Faugeras and Q. T. Luong. *The Geometry of Multiple Images*. The MIT Press, Cambridge, MA, 2001.

[20] R. B. Fisher and D. K. Naidu. *A Comparison of Algorithms for Subpixel Peak Detection*. Springer-Verlag, Heidelberg, 1996. http://citeseer.nj.nec.com/482699.html.

[21] Center for Parallel Computing. Primeur: Advancing European Technology Frontiers, World's first fully immersive VR-CUBE installed at PDC in Sweden, 1998.

[22] E. Foxlin. Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 184–194, 1996.

[23] E. Foxlin. *Motion Tracking Requirements and Technologies*, chapter 8, pages 163–210. Lawrence Erlbaum Associates, Mahwah, NJ., 2002. Ed. Kay M. Stanney.

[24] Eric Foxlin. Inertial head-tracking. Master's thesis, Department of Computer Science, Massachusetts Intitute of Technology, August 1993.

[25] K. Fujii, Y. Asano, N. Kubota, and H. Tanahashi. User interface device for the immersive 6-screens display "cosmos". In *Proc. VSMM'00*, 2000.

[26] G. Garberoglio. *Dynamical properties of H-bonded liquids: A theoretical and computer simulation study.* PhD thesis, Universita degli Studi di Trento, Facolta di Scienze Matematiche Fisiche e Naturali, 2001.

[27] Sir W. R. Hamilton. On a new species of imaginary quantities connected with a theory of quaternions. In *Proceedings of the Royal Irish Academy*, pages 424–434. Royal Irish Academy, 1843. Vol. 2 (1844).

[28] L. R. Harris, M. Jenkin, and D. C. Zikovitz. Visual and non-visual cues in the perception of linear self motion. *Experimental Brain Research*, 135:12–21, 1999.

[29] R. Hartley and A. Zissserman. *Multiple View Geometry*. Cambridge University Press; ISBN:0521623049, 2000.

[30] R. I. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6), June 1997.

[31] D. Hearn and M. P Baker. *Computer Graphics - C version*. Prentice Hall Inc., 2nd edition, 1997.

[32] J. Heikkila. Geometric camera calibration using circular control points. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2000. vol. 22, no. 10, pp.1066-1077.

[33] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1106–1112, 1997. San Juan, Puerto Rico.

[34] F. Hetmann, R. Herpers, and W. Heiden. The Immersion Square – immersive vr with standard components. In *Proc. Virtual Environment on a PC Cluster Workshop, Protvino, Russia*, 2002.

[35] A. Hogue, M. Robinson, M. R. Jenkin, and R. S. Allison. A vision-based head tracking system for fully immersive displays. In J. Deisinger and A. Kunz, editors, *7th International Immersive Projection Technologies Workshop in conjunction with the 9th Eurographics Workshop on Virtual Environments*, May 22-23 2003. to appear.

[36] I. P. Howard. *Human Visual Orientation*. John Wiley and Sons, New York, 1982.

[37] I. P. Howard and W. B. Templeton. *Human Spatial Orientation*. Wiley, London, 1966.

[38] Fraunhofer Institute IAO. http://vr.iao.fhg.de/6-Side-Cave/index.en.html.

[39] Virtual Reality Applications Center Iowa State University. http://www.vrac.iastate.edu/about/labs/c6.

[40] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill Inc., New York, 1995.

[41] H. L. Jenkin, R. T. Dyde, M. R. Jenkin, and L. R. Harris. Judging the direction of up in a tilted room. In *Perception*, 2002.

[42] S. J. Julier and J. K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, University of Oxford, Oxford, UK, 1996. Robotics Research Group, Department of Engineering Science.

[43] R. E. Kalman. A new approach to linear filtering and prediction problems. In *Transaction of the ASME - Journal of Basic Engineering*, pages 35–45, 1960. 82(Series D).

[44] V. Kindratenko. A survey of electromagnetic position tracker calibration techniques. In *Virtual Reality: Research, Development, and Applications*, 2000. vol.5, no.3, pp. 169-182.

[45] V. Kindratenko. A comparison of the accuracy of an electromagnetic and hybrid ultrasound-inertia position tracking system. In *Presence: Teloperators and Virtual Environments*, 2001. vol.10, no.6, pp. 657-663.

[46] J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1999.

[47] A. Lawrence. *Modern Inertial Technology*. Springer-Verlag, 2nd edition, 1998.

[48] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[49] R. B. McGhee, E. R. Bachmann, and M. J. Zyda. Rigid body dynamics, inertial reference frames, and graphics coordinate systems: A resolution of conflicting conventions and terminology. Technical Report NPS-MV-01-002, Naval Postgraduate Institute, Monterey, CA, November 2000.

[50] N. K. Mital and A. I. King. Computation of rigid-body rotation in three-dimensional space from body-fixed linear acceleration measurements. In *Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers, Bioengineering Division*, San Fransisco, CA, 1978. Paper 78-WA/Bio-5.

[51] J. R. W. Morris. Accelerometry – a technique for the measurement of human body movements. *Journal of Biomechanics*, 6:729–736, 1973.

[52] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *IEEE International Symposium on Mixed and Augmented Reality*, Sept 30-Oct.2 2002.

[53] I. Newton. *Philisophiae Naturalis Principia Mathematica*. 1687.

[54] University of Minnesota. Powerwall. http://www.lcse.umn.edu/research/powerwall/ powerwall.html.

[55] University of Tokyo. http://www.iml.u-tokyo.ac.jp/facilities/index_e.html.

[56] Ji-Young Oh and Wolfgang Stuerzlinger. Laser Pointers as Collaborative Pointing Devices. In *Proc. Graphics Interface*, pages 141–150, May 2002.

[57] D. R. Olsen Jr. and T. Nielsen. Laser pointer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22. ACM Press, 2001.

[58] R. H. Parvin. *Inertial Navigation*. Van Nostrand, Princeton, New Jersey, 1962.

[59] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[60] F. Raab. Remote object position locator. *US Patent 4054881*, October 1977.

[61] F. Raab, E. Bood, O. Steiner, and H. Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronics Systems*, 15(5):709–717, 1979.

[62] I. Rötzer. Fraunhofer Magazine, Synthetic worlds within six walls 2:2001.

[63] F. P. Redlick, M. Jenkin, and L. R. Harris. Humans can use optic flow to estimate distance of travel. *Vis. Res.*, 41:213–219, 2001.

[64] Wearable Computing Systems Research. http://www.wear-it.net/.

[65] M. Robinson, J. Laurence, J. Zacher, A. Hogue, R. Allison, L. R. Harris, M. Jenkin, and W. Stuerzlinger. Ivy: The immersive visual environment at york. In *6th International Immersive Projection Technology Symposium, March 24-25, 2002, Orlando, Fl.*, 2002.

[66] J. P. Rolland, L. Davis, and Y. Baillot. A survey of tracking technology for virtual environments. In Barfield and NJ. Caudell, Mahwah, editors, *Augmented Reality and Wearable Computers*, 2001.

[67] D. H. Sutherland. The evolution of clinical gait analysis - Part II Kinematics. *Gait & Posture*, 16(2):159–179, October 2002.

[68] I. E. Sutherland. A head-mounted three-dimensional display. In *1968 Fall Joint Computer Conference, AFIPS Conference Proceedings*, 1968. 33,757-764.

[69] D. H. Titterton and J. L. Weston. *Strapdown inertial navigation technology*. Peter Peregrinus Ltd., London, 1997.

[70] Aalborg University. The VRMedialab. http://www.vrmedialab.dk/.

[71] Beckman Institute University of Illinois, Integrated Systems Laboratory. A laboratory for immersive conitive experiments. http://www.isl.uiuc.edu/Virtual%20Tour /Tour-Pages/meet_alice.htm.

[72] VICON. Vicon motion capture. http://www.vicon.com/entertainment/applications/film.html, 2003.

[73] T. Vieville and O. Faugeras. Computation of inertial information on a robot. In *Fifth International Symposium on Robotics Research*, pages 57–65. MIT-Press, 1989.

[74] Joseph J. La Viola and Jr. A discussion of cybersickness in virtual environments. In *SIGCHI Bulletin*, January 2000. vol.32 no.1, pp.47-56.

[75] G. Welch. *SCAAT: Incremental Tracking with Incomplete Information*. PhD thesis, Chapel Hill, NC, 1996.

[76] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR95-041, University of North Carolina at Chapel Hill, 1995.

[77] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. pages 1–10.

[78] W. Wiesel. *Spaceflight Dynamics*. Irwin McGraw-Hill, 2nd edition, 1997.

[79] T. Winograd and F. Guimbretière. Visual instruments for an interactive mural. In *Conference extended abstracts on Human factors in computer systems*, pages 234–235. ACM Press, 1999.

[80] M. J. Wurpts, T. L. Swanson, and R. Tapia. Tracking technologies for virtual reality training applications: A case study. In *22nd I/ITSEC Conference Proceedings*, Orlando, Fl., November 2000.

[81] T. Yamada, M. Hirose, and Y. Isda. Development of a complete immersive display: Cosmos. In *Proc. VSMM'98*, pages 522–527, 1998.

[82] Y. Yokokohji, Y. Sugawara, and T. Yoshikawa. Accurate image overlay on video see-through hmds using vision and accelerometers. In *Proceedings of IEEE Virtual Reality*, March 2001.

[83] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Proceedings of IEEE Virtual Reality*, pages 71–78, 2001.

[84] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proceedings of IEEE Virtual Reality*, pages 260–267, March 1999.