2009-01-16

An Inertial-Optical Tracking System for Quantitative, Freehand, 3D Ultrasound

Abraham Myron Goldsmith
Worcester Polytechnic Institute

An Inertial-Optical Tracking System for Quantitative, Freehand, 3D Ultrasound

by

Abraham Myron Goldsmith

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical Engineering

December 2008

APPROVED:

Dr. Peder C. Pedersen _____

Dr. Cosme Furlong _____

Dr. Edward Clancy _____

Dr. Thomas L. Szabo _____

Dr. Fred J. Looft _____

Department of Electrical and Computer Engineering

Abstract

Three dimensional (3D) ultrasound has become an increasingly popular medical imaging tool over the last decade. It offers significant advantages over Two Dimensional (2D) ultrasound, such as improved accuracy, the ability to display image planes that are physically impossible with 2D ultrasound, and reduced dependence on the skill of the sonographer. Among 3D medical imaging techniques, ultrasound is the only one portable enough to be used by first responders, on the battlefield, and in rural areas.

There are three basic methods of acquiring 3D ultrasound images. In the first method, a 2D array transducer is used to capture a 3D volume directly, using electronic beam steering. This method is mainly used for echocardiography. In the second method, a linear array transducer is mechanically actuated, giving a slower and less expensive alternative to the 2D array. The third method uses a linear array transducer that is moved by hand. This method is known as freehand 3D ultrasound.

Whether using a 2D array or a mechanically actuated linear array transducer, the position and orientation of each image are known ahead of time. This is not the case for freehand scanning. To reconstruct a 3D volume from a series of 2D ultrasound images, assumptions must be made about the position and orientation of each image, or a mechanism for detecting the position and orientation of each image must be employed. The most widely used method for freehand 3D imaging relies on the assumption that the probe moves along a straight path with constant orientation and speed. This method requires considerable skill on the part of the sonographer. Another technique uses features within the images themselves to form an estimate of each image's relative location. However, these techniques

are not well accepted for diagnostic use because they are not always reliable. The final method for acquiring position and orientation information is to use a six Degree-of-Freedom (6 DoF) tracking system. Commercially available 6 DoF tracking systems use magnetic fields, ultrasonic ranging, or optical tracking to measure the position and orientation of a target. Although accurate, all of these systems have fundamental limitations in that they are relatively expensive and they all require sensors or transmitters to be placed in fixed locations to provide a fixed frame of reference.

The goal of the work presented here is to create a probe tracking system for freehand 3D ultrasound that does not rely on any fixed frame of reference. This system tracks the ultrasound probe using only sensors integrated into the probe itself. The advantages of such a system are that it requires no setup before it can be used, it is more portable because no extra equipment is required, it is immune from environmental interference, and it is less expensive than external tracking systems.

An ideal tracking system for freehand 3D ultrasound would track in all 6 DoF. However, current sensor technology limits this system to five. Linear transducer motion along the skin surface is tracked optically and transducer orientation is tracked using MEMS gyroscopes.

An optical tracking system was developed around an optical mouse sensor to provide linear position information by tracking the skin surface. Two versions were evaluated. One included an optical fiber bundle and the other did not. The purpose of the optical fiber is to allow the system to integrate more easily into existing probes by allowing the sensor and electronics to be mounted away from the scanning end of the probe. Each version was optimized to track features on the skin surface while providing adequate Depth Of Field (DOF) to accept variation in the height of the skin surface.

Orientation information is acquired using a 3 axis MEMS gyroscope. The sensor was thoroughly characterized to quantify performance in terms of accuracy and drift. This data provided a basis for estimating the achievable 3D reconstruction accuracy of the complete system. Electrical and mechanical components were designed to attach the sensor to the ultrasound probe in such a way as to simulate its being embedded in the probe itself.

An embedded system was developed to perform the processing necessary to translate

the sensor data into probe position and orientation estimates in real time. The system utilizes a Microblaze soft core microprocessor and a set of peripheral devices implemented in a Xilinx Spartan 3E field programmable gate array. The Xilinx Microkernel real time operating system performs essential system management tasks and provides a stable software platform for implementation of the inertial tracking algorithm.

Stradwin 3D ultrasound software was used to provide a user interface and perform the actual 3D volume reconstruction. Stradwin retrieves 2D ultrasound images from the Terason t3000 portable ultrasound system and communicates with the tracking system to gather position and orientation data. The 3D reconstruction is generated and displayed on the screen of the PC in real time. Stradwin also provides essential system features such as storage and retrieval of data, 3D data interaction, reslicing, manual 3D segmentation, and volume calculation for segmented regions.

The 3D reconstruction performance of the system was evaluated by freehand scanning a cylindrical inclusion in a CIRS model 044 ultrasound phantom. Five different motion profiles were used and each profile was repeated 10 times. This entire test regimen was performed twice, once with the optical tracking system using the optical fiber bundle, and once with the optical tracking system without the optical fiber bundle. 3D reconstructions were performed with and without the position and orientation data to provide a basis for comparison.

Volume error and surface error were used as the performance metrics. Volume error ranged from 1.3% to 5.3% with tracking information versus 15.6% to 21.9% without for the version of the system without the optical fiber bundle. Volume error ranged from 3.7% to 7.6% with tracking information versus 8.7% to 13.7% without for the version of the system with the optical fiber bundle. Surface error ranged from 0.319 mm RMS to 0.462 mm RMS with tracking information versus 0.678 mm RMS to 1.261 mm RMS without for the version of the system without the optical fiber bundle. Surface error ranged from 0.326 mm RMS to 0.774 mm RMS with tracking information versus 0.538 mm RMS to 1.657 mm RMS without for the version of the system with the optical fiber bundle.

The prototype tracking system successfully demonstrated that accurate 3D ultrasound

volumes can be generated from 2D freehand data using only sensors integrated into the ultrasound probe. One serious shortcoming of this system is that it only tracks 5 of the 6 degrees of freedom required to perform complete 3D reconstructions. The optical system provides information about linear movement but because it tracks a surface, it cannot measure vertical displacement. Overcoming this limitation is the most obvious candidate for future research using this system. The overall tracking platform, meaning the embedded tracking computer and the PC software, developed and integrated in this work, is ready to take advantage of vertical displacement data, should a method be developed for sensing it.

Acknowledgements

I would like to thank professor Peder C. Pedersen for his financial, academic, and personal support. His guidance played a crucial role in my success. I would also like to thank professor Cosme Furlong for his assistance in the development of the optical tracking system and professor Thomas Szabo for his myriad contributions to this effort.

For developing and making available the Stradwin 3D ultrasound software, I would like to thank Dr. Andrew Gee, Dr. Graham Treece, and Dr. Richard Prager, of the University of Cambridge in England. A special thanks to Dr. Treece for providing personal support and building a special version of Stradiwn to aide my experiments.

The ultrasound phantom used for the performance testing was loaned to us, free of charge, by CIRS Inc.

Last, but not least, I would like to thank Pat Morrison from the Atwater Kent shop and Neil Whitehouse from Higgens Laboratories for teaching me how to use machine tools. I could never have built this system without them.

Contents

List of Figures

List of Tables

List of Acronyms

2D	Two Dimensional
3D	Three Dimensional
ADC	Analog to Digital Converter
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CCD	Charge-Coupled Device
CCP	Capture and Compare
CK	Clock
CMOS	Complementary MetalâĂŞOxideâĂŞSemiconductor
CS	Chip Select
CT	Computed Tomography
CVG	Coriolis Vibratory Gyroscope
DCM	Direction Cosine Matrix
DLL	Dynamic Link Library
DoF	Degree of Freedom
DOF	Depth Of Field
DSP	Digital Signal Processor
DT	Data
DUT	Device Under Test
EEPROM	Electronically Erasable and Programmable Read Only Memory
EP	USB End Point
FFC	Flat Flex Cable
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FIT	Fixed Interval Timer
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FW	Firmware
GPIO	General Purpose Input Output
HID	USB Human Interface Device
HP-GL	Hewlett-Packard Graphics Language
HW	Hardware
I/O	Input / Output
I^2C	Inter-Integrated Circuit

IC	Integrated Circuit
ICP	Iterative Closest Point
ID	Image Distance
IDE	Integrated Development Environment
INS	Intertial Navigation System
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LMB	Local Memory Bus
LPCR	Line Profile Contrast Ratio
LPF	Low Pass Filter
LSI	Linear Shift Invariant
LVDS	Low Voltage Differential Signaling
MEMS	Microelectromechanical Systems
MISO	Master In Slave Out
MMU	Memory Management Unit
MOSI	Master Out Slave In
MRI	Magnetic Resonance Imaging
MSSP	Master Synchronous Serial Port
MT	Modulation Transfer
MTF	Modulation Transfer Function
NA	Numerical Aperture
OD	Object Distance
PC	Personal Computer
PCB	Printed Circuit Board
PLB	Processor Local Bus
PLL	Phase Locked Loop
POSIX	Portable Operating System Interface
PROM	Programmable Read Only Memory
PWM	Pulse Width Modulator
RAM	Random-Access Memory
RISC	Reduced Instruction Set Computing
RMS	Root Mean Square
ROM	Read Only Memory
RTV	Room Temperature Vulcanizing
RX	Receive
SCK	Serial Clock
SCL	Serial Clock
SDA	Serial Data
SDIO	Serial Data Input Output
SDO	Serial Data Out
SPI	Serial Peripheral Interface
SQUAL	Surface Quality
SS	Slave Select
SSE	Synchronous Serial Engine
SW	Software
TPI	Threads Per Inch

TTL	Transistor-Transistor Logic
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
USB-IF	Universal Serial Bus Implementers Forum
VHDL	VHSIC Hardware Description Language
XMK	Xilinx Micro Kernel
XPS	Xilinx Platform Studio

Chapter 1

Introduction

Conventional two dimensional (2D) ultrasound imaging has been an important medical imaging tool for over three decades. It has several advantages over other medical imaging modalities such as X-Ray, Computed Tomography (CT), and Magnetic Resonance Imaging (MRI). First, ultrasound produces a real time image. Second, ultrasound does not utilize ionizing radiation or strong magnetic fields. This makes ultrasound safer that MRI and CT. Third, ultrasound is much more portable than other medical imaging techniques. Finally, ultrasound is much less expensive.

Two dimensional ultrasound also has several disadvantages when compared to other medical imaging modalities. First, the image quality is generally poorer than other imaging techniques. This is because acoustic signals are strongly attenuated by human tissue and because of the complex interference between the acoustic wave and the soft tissue. Second, the types of images that can be captured with 2D ultrasound are limited. In most cases the transducer is held on the skin surface, which makes it impossible to image in planes that are parallel to the skin surface. X-Rays have a similar limitation in some cases, but CT and MRI can generate a 2D image on an arbitrary plane. Lastly, ultrasound has limited penetration into the body. It cannot be used to image the brain or lungs because bone and air-filled tissues severely attenuate the acoustic signal.

During a typical 2D ultrasound examination, the sonographer captures a set of 2D

images. These images are later reviewed by a doctor. Doctors often want to measure features in the images, such as the dimensions of tumors or cysts. Making good measurements of three dimensional (3D) structures using 2D ultrasound images is difficult. The relative position and orientation of each 2D image is not known, which means that quantitative measurements outside of each image plane are generally not possible. In practice, it is the skill of the sonographer that bridges this gap. He or she must take great care to follow the correct procedure for a particular examination and to acquire good images. The repeatability of measurements made this way is therefore limited because, despite the best effort of the sonographer, no two scans are exactly identical.

1.1 3D Ultrasound Acquisition Methods

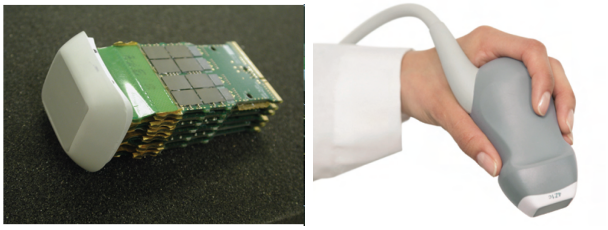
Three dimensional ultrasound was proposed as a way to address some of the shortcomings of 2D ultrasound, listed previously. 3D volumes can be resliced to create 2D images in an arbitrary plane, which eliminates the restrictions imposed by the need to image through the skin surface. 3D ultrasound can also be used to make accurate measurements. Assuming that the 3D image is generated accurately, the measurement process is less dependent on the skills of the sonographer. Measurements can be made directly in the 3D image, which eliminates the ambiguity caused by the unknown relationship between individual 2D images from a standard 2D examination. Repeatability increases for the same reason.

There are three basic techniques for acquiring 3D ultrasound images. They are described in the following sections.

1.1.1 2D Array Transducers

The most direct method for acquiring a 3D ultrasound image is to use a 2D array transducer. 2D array transducers generally produce the highest resolution 3D images and can do so at the highest rate. A state of the art 2D array transducer manufactured by Siemens is pictured in Figure 1.1. These types of transducers are used primarily in echocardiography for functional analysis of the heart. It is very difficult to manufacture large arrays because

of the large number of electrical connections and interference between the elements. The largest commercially available 2D arrays have approximately 3000 elements. This limits the size of the 3D image that they are capable of capturing. Because of the high level of electrical integration required and the difficulty of manufacturing them, 2D array transducers are also very expensive when compared to traditional 1D linear array transducers.



(a) (b)

Figure 1.1: Internal (a) and external (b) views of a Siemens 4Z1c 2D array transducer. [1]

1.1.2 Mechanically Actuated 1D Array Transducers

The second method of creating 3D ultrasound images is mechanical actuation of a 1D linear array transducer. Like traditional 2D ultrasound scanning, this method collects a sequence of 2D images. The mechanical actuator forces the images to be in fixed orientations and positions, which allows the individual 2D images to be accurately reassembled into a single 3D image. In the earliest mechanical scanning implementations, a 1D linear array transducer was mounted in an external actuator. An example of an external mechanical actuator is illustrated in Figure 1.2. The probe motion and the shape of the resulting 3D image is illustrated in Figure 1.4 (b). These mechanisms tended to be quite bulky and were never well accepted in the clinical environment.

More recently, probes have been developed with integral mechanical actuators, as illustrated in Figure 1.3. The probe is held stationary while the transducer moves through an arc inside the housing. The probe motion and the shape of the resulting 3D image is illustrated in Figure 1.4 (a). This approach solves the problem of bulkiness but the size of the 3D image that can be acquired is limited by the travel of the mechanical actuator in

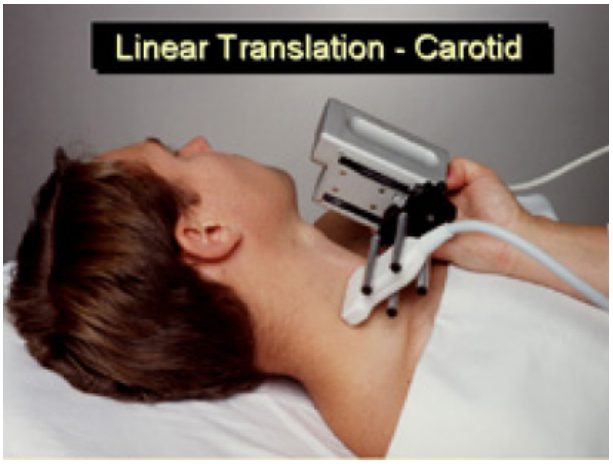


Figure 1.2: External type mechanical actuator for 3D ultrasound imaging

both cases.

1.1.3 Manually Actuated 1D Array Transducers

The third method of generating 3D ultrasound images is similar to the second technique, but with the mechanical actuator replaced by a human. A 1D linear array transducer is used to capture a sequence of 2D images, while a sonographer moves the probe over the region of interest. This acquisition technique is commonly referred to as freehand scanning. One advantage of this approach is that it does not place any constraints on the size of the 3D image. Another advantage is the low cost, as only software is required. Ideally, a freehand 3D ultrasound system should be able to create 3D images of arbitrary size and shape. The challenge of freehand 3D acquisition is that the relative position and orientation of the individual 2D images is no longer fixed, as illustrated in Figure 1.5. The images are no longer guaranteed to be parallel, equally spaced, and aligned, the way they were in Figure 1.4 (b). This variation makes 3D image reconstruction in freehand 3D ultrasound much challenging than it is for the other methods of 3D image acquisition.

Accurate 3D image reconstruction for freehand acquisition requires information about the position and orientation of each 2D image in the data set. The simplest way to do this is to assume that the sonographer keeps the probe orientation fixed and moved in a

Figure 1.3: A Siemens 7CF2 ultrasound transducer with internal mechanical actuator [2]

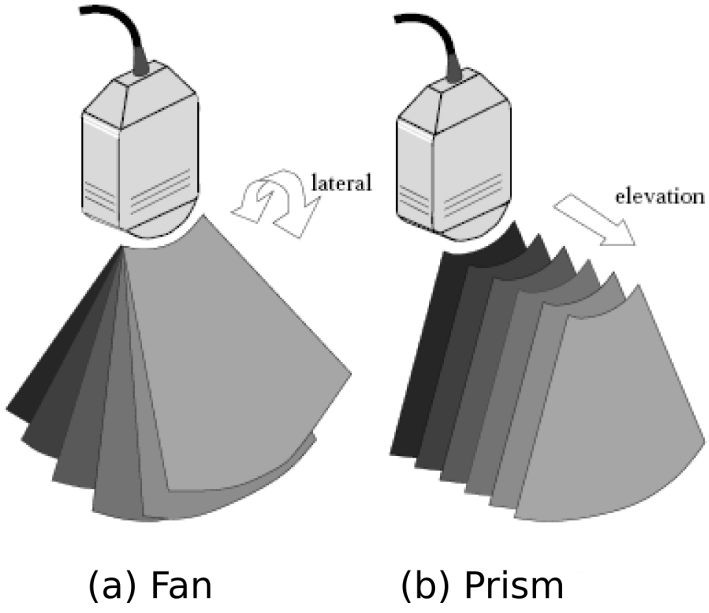


Figure 1.4: Common types of motion generated by mechanical actuation and corresponding 3D image volumes

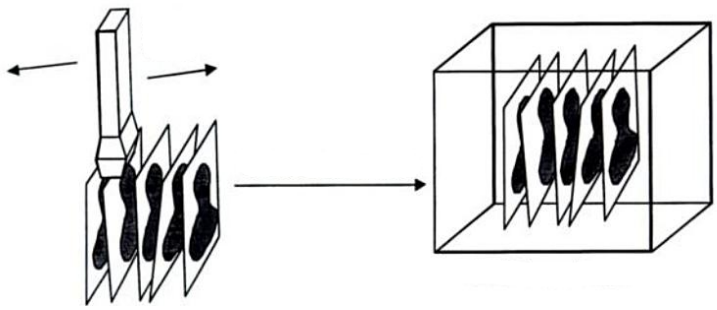


Figure 1.5: Illustration of a freehand 3D ultrasound scan [3]

straight path, at a fixed rate. The 3D image is then assembled by stacking the images one on top of another. Obtaining undistorted 3D images is thus very dependent on the skill of the sonogrpaher. Furthermore, the requirement that the probe's orientation remain fixed during the scan is a serious drawback of this approach. It limits the types of scans that can be performed to those that are relatively straight and flat.

Tracking systems can be used to overcome these limitations by recording the position and orientation of the probe during the scan. The 2D images are then inserted into the 3D image using that data, in order to generate an accurate image. There are several commercially available 6 Degree of Freedom (6 DoF) tracking systems that are suitable for this purpose. Most use either a magnetic field, such as the Ascension Flock of Birds, or optical emitters and reflectors, such as the Northern Digital Polaris, for tracking. Both of these technologies are quite accurate but both can be difficult to use in the clinical setting. Tracking systems based on magnetic fields are sensitive to metals and optical systems require line of sight. All of these tracking systems also require that the transmitter or receiver (whichever is not attached to the probe) remain at a fixed location relative to the probe.

The last method used to build a 3D image from a sequence of 2D images acquired by freehand scanning uses speckle decorrelation. Speckle is a type of noise present in ultrasound images, caused by interference from scattering among unresolved scatterers in the tissue. As long as successive images are close together, much of the tissue scanned to generate each image will be common to both. As a result, the correlation between the speckle patterns in successive images tends to move from high to low as the physical distance between the

images increases. This technique can be used to estimate the position and pose of two image planes relative to one another using only image processing. Although this technique does overcome the need for a tracking system, it has several important drawbacks. The scan direction must remain consistent throughout the scan because the algorithm can only estimate the spacing between two images. The algorithm also cannot account for angular variation between images in the sequence. Another problem is that the characteristics of speckle vary for different types of tissue.

1.2 Motivation for Development of a Novel Tracking System for Freehand 3D Ultrasound

The goal of this work is to create a tracking system for freehand 3D ultrasound imaging that is completely contained within the ultrasound probe. Such a system should be able to generate accurate 3D ultrasound images without the constraints of other 3D ultrasound imaging techniques and without the limitations and cost associated with external tracking systems. One of the Ultrasound Research Laboratory's ongoing projects is the development of a ruggedized, mobile ultrasound system for use in remote or rural areas, by first responders, and on the battlefield. The tracking system developed in this work will allow the capabilities of the portable system to be extended to include 3D imaging.

1.3 Overview of a Freehand 3D Ultrasound System

A freehand 3D ultrasound system is composed of 3 basic parts. There is a 2D ultrasound imaging system, a 5 DoF tracking system, and 3D image reconstruction software. In our prototype system, a Personal Computer (PC) is used as a host for the reconstruction software. Ultrasound images are generated with a Terason 3000 ultrasound system. It consists of the ultrasound transducer and a beamformer, which is connected to the PC using an IEEE 1394 bus, otherwise known as firewire. Software from Terason runs in the background on the PC to control the transducer and beamformer electronics and to make image frames available

to other applications. The tracking system consists of sensors mounted on the ultrasound transducer and an embedded computer system. The embedded computer system and the firmware running on it form the navigation computer that processes the raw sensor output to generate estimates of the transducer position and orientation. Linear motion along the skin surface is tracked optically. Microelectromechanical System (MEMS) gyroscopes are used to measure angular rate of change along three orthogonal axes. The navigation computer transmits the tracking information to the PC using a serial port. The 3D reconstruction software combines the 2D ultrasound images with the position information to reconstruct 3D images in real time, as well as providing a user interface for the system. In this system, Stradwin 3D ultrasound software running on the PC performs this function. Figure 1.6 is a high level block diagram of the system.

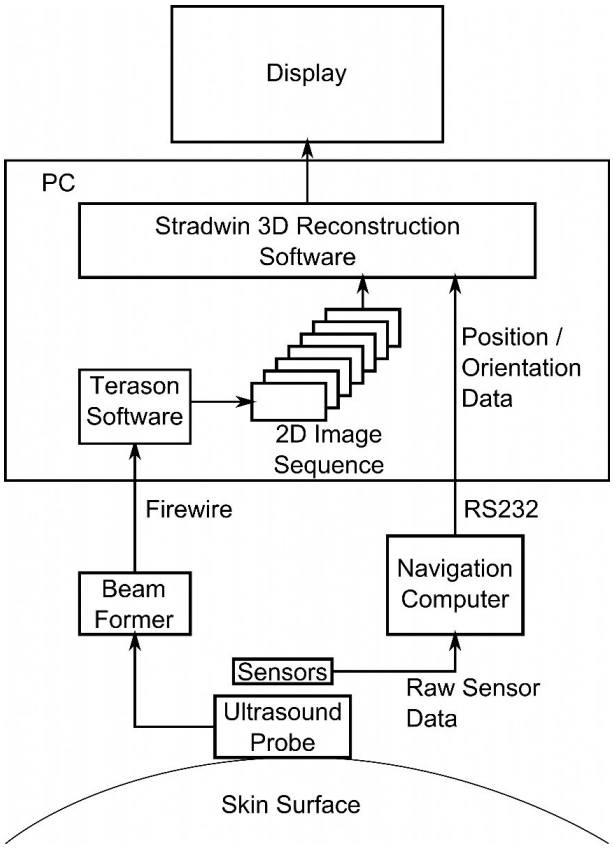


Figure 1.6: Block Diagram of the prototype freehand 3D ultrasound system

1.4 Thesis Outline

Chapter 2 describes two visualization tools that were evaluated for use in 3D reconstruction and system control. The basic functionality of the programs is covered as well as evaluation of reconstruction performance.

Chapter 3 is an overview of major components of the system. It is intended to serve as a quick reference that allows the reader to find basic information quickly without searching the entire document.

Chapter 4 describes the test systems that were developed for sensor evaluation. The hardware, firmware, motion platforms, and PC software used for testing is explained.

Chapter 5 describes experiments performed to evaluate MEMS gyroscopes and the results of those experiments.

Chapter 6 describes experiments performed on MEMS accelerometers and the results of those experiments.

Chapter 7 describes the experiments performed and apparatus developed to optimize the optical system and the prototype optical tracking devices that were constructed.

Chapter 8 is a basic introduction to inertial navigation. The basic terminology of inertial navigation is presented along with the necessary mathematical concepts. A detailed explanation of the navigation algorithm is also presented.

Chapter 9 describes the prototype freehand 3D ultrasound system that was constructed. The first half of the chapter is dedicated to the sensor interface electronics and the embedded computer system. The second half of the chapter describes the navigation computer firmware.

Chapter 10 covers performance evaluation results. The first part of chapter describes the test procedure. Then the error metrics are introduced and data analysis methods are described. Finally, the results are presented.

Chapter 11 is the conclusion. The results of the work are discussed along with avenues for further research.

Chapter 2

3D Visualization Platforms

A 3D visualization platform is a computer program that can display 3D data graphically. In our application this was ultrasound data. In addition to just displaying the ultrasound data, the visualization platform in our application served several other purposes. It was responsible for integrating image data from ultrasound software, position and orientation data from an external tracking system, and implementing the user interface.

The prototype uses the Terason t3000 imaging system to generate the ultrasound image data. The data is made available as a series of bitmaps, at a rate of between 5 and 30 frames per second. The data is accessed via ActiveX controls. The Terason executable runs in the background as a server, controlling the t3000 hardware. Client programs make requests for ultrasound data through the ActiveX interface. Position information is generated by an inertial-optical tracking system, which is described in greater detail in Chapter 9. User interface data is provided by the visualization platform itself. Examples of this type of data include controls for starting and stopping acquisition, ultrasound probe settings, view angle, etc.

To create a prototype system, only a minimal subset of the features typically found in commercial ultrasound systems are required. First, the visualization platform must be capable of storing a sequence of 2D images as a single 3D data set. This just means that the program must support 3D data. Second, the visualization platform must be capable

of rendering views of 3D data. Ideally, real-time visualization should be supported, such that the 3D volume is displayed on the screen as it is being acquired. However, this is not a strict requirement for a prototype. Third, the visualization platform must support a basic user interface that provides acquisition control, probe settings, and a means of saving and retrieving data. Lastly, the visualization platform must support position data input for the purpose of creating physically accurate 3D volumes. The program must have an algorithm that allows it to insert 2D image data into a 3D volume in such a way that the data is in the correct physical location with respect to the rest of the data in the volume. This particular requirement is the most difficult one to meet. Although there are several commercial 3D ultrasound imaging systems in existence today, they use proprietary software that only works with specific equipment and is not available to be licensed for outside use.

Over the course of this work two different visualization platforms were evaluated *CustusX* and *Stradwin*. CustusX was created by SINTEF, a Norwegian research group working in the field of imaging and visualization. They were contracted to produce a version of their Custus software to meet the requirements of the prototype system. This program is known as CustusX. Stradwin is a software package produced by researchers in the medical imaging group, within the machine intelligence laboratory, at Cambridge University in England. It was created as a research tool, specifically for 3D ultrasound applications. Best of all, it is free.

2.1 CustusX

CustusX is a 3D visualization tool developed by SINTEF, an independent research group headquartered in Trondheim, Norway. It is based on the ITK and VTK toolkits from Insight. CustusX renders 2D views of 3D data such as that generated by ultrasound scanners. Dr. Pedersen contracted SINTEF to build a custom version of CustusX to run on WindowsXP (it was originally developed for platforms running XFree86 and Xorg). This version was to serve as the visualization front end for a portable 3D ultrasound system.

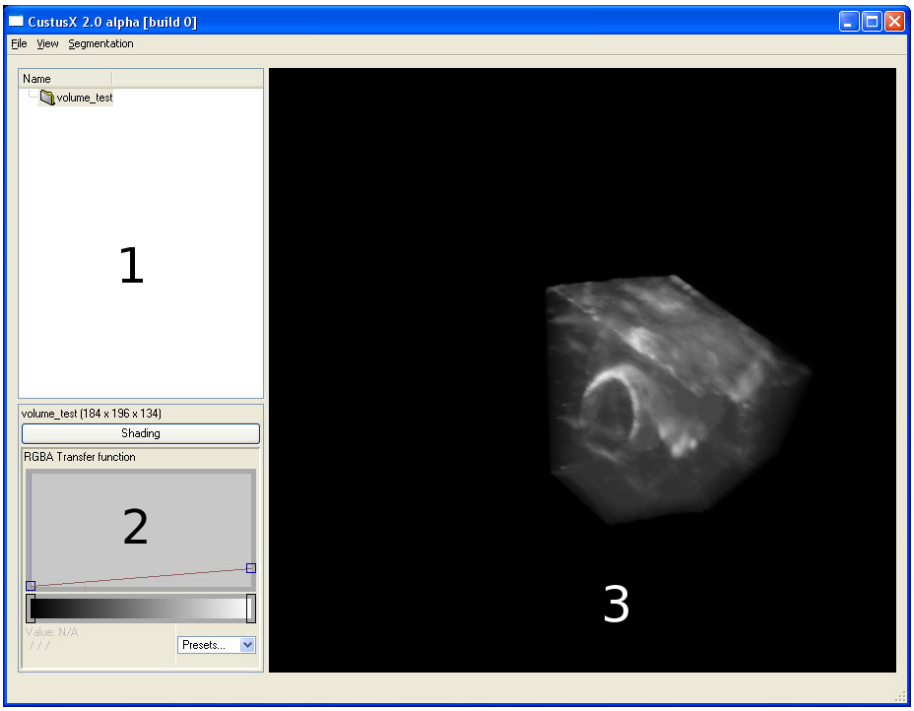


Figure 2.1: Layout of the CustusX user interface. (1) is the Volume Browser, (2) is Transfer Function, and (3) is the Volume Display

2.1.1 User Interface

This section briefly describes the CustusX interface. Please see the CustusX User's Guide, which is included in Appendix B, for a complete description.

The screen is divided into 3 sections, as illustrated in Figure 2.1:

1. Volume Browser

The Volume Browser displays a list of volumes that are currently loaded into memory. New volumes can be loaded by clicking on **File**, **Open**, **Load Volume Data** or by right clicking within the Volume Browser and selecting **Load Volume Data**. Volumes can also be copied and deleted using these two menus.

2. Transfer Function

The Transfer Function display allows the user to adjust how the data is displayed. By adjusting the curve the user can change the relative intensities of the discrete data values within the volume. The color that is assigned to a particular intensity value can be modified as well as the alpha, or transparency, value. By altering the alpha value assigned to different voxel intensities the user can make unimportant regions of the volume, that might otherwise obscure important features, less visible or invisible.

3. Volume Display

Once loaded, the dataset appears in the Volume Browser. The user may right click on the name of the dataset in the Volume Browser and select 'Show Volume in Scene,' in order to view the rendered volume in the Volume Display. The mouse can then be used to manipulate the position and orientation of the rendered data in the Volume Display.

2.1.2 Data Acquisition

CustusX's biggest weakness is that is does not support a direct interface to either an ultrasound transducer or a 6 DoF position sensing device. It was the intent of the original collaboration agreement that these features be developed jointly between WPI and SINTEF.

WPI and SINTEF agreed on an Application Programming Interface (API) that would allow the data collection routines to be implemented outside of the CustusX application. WPI would provide Windows Dynamic Link Library (DLL) that implemented the functions in the API. CustusX would then handle the reconstruction, rendering, and user interface functionality. For evaluation purposes, the first version of the DLL was designed to read ultrasound data from a series of bitmap images and position data from a series of text files. CustusX would request a new piece of ultrasound image data using the *Get_Image()* API call and new position data using the *Get_Position()* API call. Once called, *Get_Image()* would open the next in a series of sequentially named bitmap images, retrieve the intensity values, and return a pointer to the data to CustusX. *Get_Position()* would open the next in

a series of sequentially named text files containing the position and pose information. The data was then returned to CustusX in a structure defined in the API.

The requirement that the user be able to control the beginning and end of data acquisition was overlooked in the original specification. Although it may have been possible to implement this feature almost entirely in the DLL, the API specification made this very difficult. Because the API only provided *Get_Image()* and *Get_Position()* functions to access data, the application required prior knowledge about the total number of data frames to acquire and the frequency with which to acquire them. This, in turn, would have required modifications to the CustusX user interface. In order to support WPI's need to evaluate CustusX's reconstruction performance, SINTEF produced two alpha versions that were hard coded to acquire 200 and 400 data frames, respectively.

It turned out that CustusX already supported a far more powerful method for evaluating reconstruction performance. The portion of ITK that deals with format specific file Input / Output (I/O) is known as MetaIO. MetaIO supports an expansive array of medical imaging file formats for multidimensional data. It also supports importing custom data types via metadata provided in either the header of the data file or in a separate text file. SINTEF implemented an extension to the MetaIO library that enabled it to process position data along with the image data. Since the acquisition method implemented in the DLL was really just a very limited subset of the features already included in CustusX, the DLL was abandoned as a means of evaluation in favor of the MetaIO approach.

2.1.3 Evaluation

In order to verify that CustusX was correctly reconstructing volumes, it was necessary to devise a method for evaluation. A number of Matlab scripts were written that could generate sequences of bitmap images and text files containing artificial position information. Each image contained a single 2D slice of a 3D volume. As illustrated in Figure 2.3, the evaluation volume contained a 3D rectangle. Each pixel in each 2D slice is represented by 8 bits. The background of each 2D slice has an intensity value of 0 and the object has an intensity value of 255. Between 2D images in the sequence, the position of the object relative to the

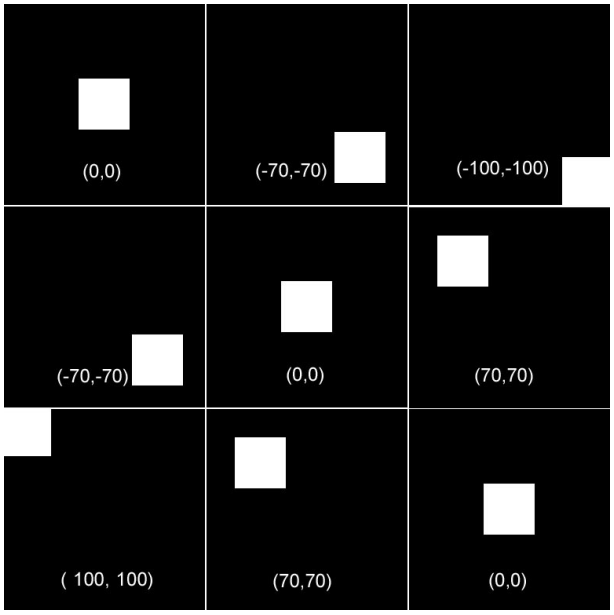


Figure 2.2: Artificial volume data

background was varied. The magnitude of the variation was then written to a corresponding text file. These artificial datasets were generated in groups of 200 and 400 sequential images to work with the two versions of CustusX.

Figure 2.2 illustrates the raw image sequence from the artificial volume data. The x and y coordinates are displayed in each image frame. The z coordinate is the equivalent of the sequence number. The sequence begins in the upper-left hand corner and proceeds row-wise from left to right, terminating in the lower right hand corner. CustusX's volume reconstruction with and without position information is shown in Figures 2.3 and 2.4 respectively.

The testing demonstrated that CustusX was able to correctly reconstruct 3D volumes using a combination of 2D images and position information. The evaluation also provided a lot of information about the overall maturity and usability of the program. CustusX was buggy and crash prone. Certain actions would cause the program to crash predictably while other crashes seemed more random in nature. For example, CustusX would crash without any error message when it was not able to allocate memory. Given that the original Unix

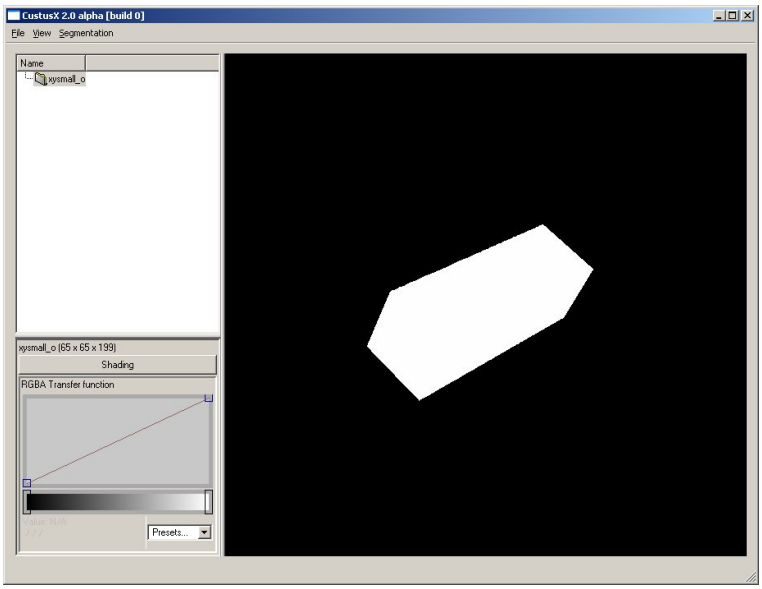


Figure 2.3: CustusX volume reconstruction with position information

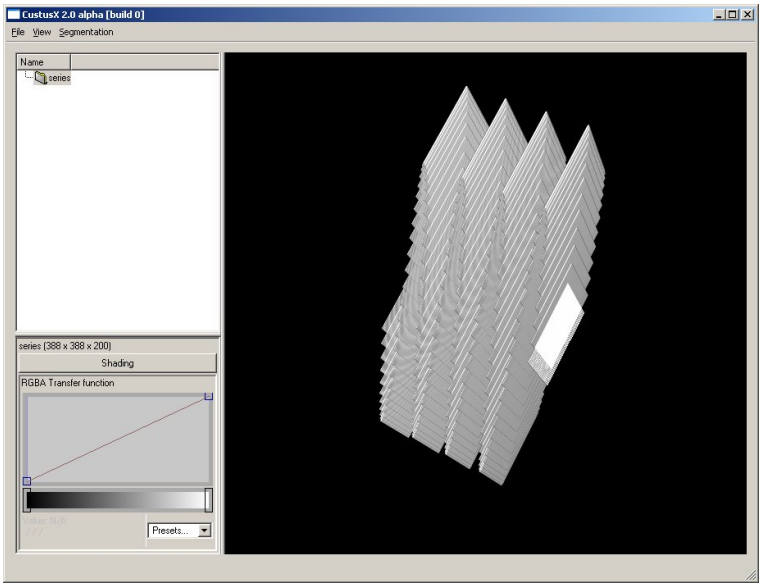


Figure 2.4: CustusX volume reconstruction without position information

version of Custus was stable enough to be used in the operating room, it seems reasonable to ask what makes CustusX so unstable. Since the volume reconstruction algorithms are the same, the problems most likely lie in the way CustusX interacts with the Windows memory manager and the graphics hardware. Both of these are substantially different between Windows and Unix. The lack of documentation was also a serious issue. Many of the features detailed in the user's guide were discovered by trial and error. Other features were only revealed after multiple emails and phone calls with SINTEF.

2.2 Stradwin

Stradwin is an experimental software tool developed specifically for freehand 3D ultrasound. It was developed by the Medical Imaging Group at the University of Cambridge in Cambridge, England. Stradwin is capable of interfacing directly with a Terason ultrasound and imaging system via Terason's ActiveX controls. Stradwin also interfaces directly with many commercial 6 DoF sensor systems, such as the Polhemus Patriot and Fastrak, Northern Digital Polaris, and the Ascension Flock of Birds. In addition, Stradwin incorporates a calibration facility, real-time volume data display, manual segmentation, volume measurement, volume reslice, uses and open file format, and has complete documentation. It is freely available for download from the Medical Imaging Group's website[1].

2.2.1 Stradwin User Interface

The Stradwin user interface is comprised of several windows containing controls and visualization tools. There are 3 main windows, illustrated in Figure 2.5:

1. The Tasks Window

The Tasks Window is used to configure the 3D reconstruction software. It has multiple tabs that provide the following functions: template save and restore, recording settings, probe calibration, isocenter configuration, pointer configuration, pointer calibration, probe pressure correction, drawing options, and landmarks.

2. The Image Window

The Image Window is where the volume data is displayed. It is separated into 3 parts. The top left portion displays individual ultrasound images, the top right portion shows the location of all features (individual ultrasound images, landmarks, segmentations, etc) in 3D space. The bottom portion of the Image Window can either be divided into two to show two orthogonal reslices, or used as a single large window displaying a reslice plane at any orientation.

[1] `http://mi.eng.cam.ac.uk/~rwp/stradwin/`

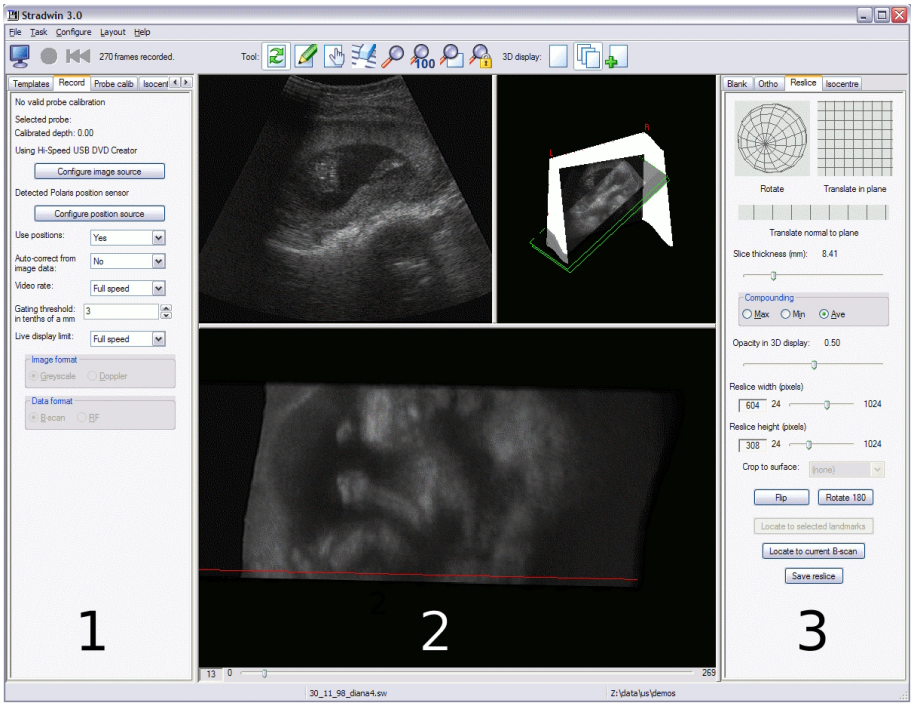


Figure 2.5: Layout of the Stradwin user interface. (1) is the Tasks Window, (2) is the Image Window, and (3) is the Visualization Window

3. The Visualization Window

The Visualization Window controls what is displayed in the lower portion of the Image Window. It has multiple tabs that provide the following functions: orthogonal display, reslice display, isocenter display, panorama display, and pressure display.

2.2.2 Data Acquisition

Stradwin acquires data directly from the ultrasound system and the tracking system. Ultrasound data can be acquired directly from a Terason t2000 or t3000 ultrasound imaging system or through a video capture device. Stradwin also supports acquisition of raw RF data using a Gage CompuScope analog data acquisition card and any ultrasound system with accessible analog RF lines, using the Terason t2000 or t3000 with the appropriate ActiveX control, or using a Diasus ultrasound system from Dynamic Imaging Ltd. Strad-

win supports the following position systems: the Polhemus Fastrak and Patriot, Northern Digital Polaris, and the Ascension MiniBird, LaserBird and Flock of Birds sensors.

These features made Stradwin ideally suited for the proof of concept system. However, there was one small complication. For the prototype tracking system to communicate with Stradwin, it would have to emulate the RS232 communication protocol used by the Polhemus Fastrak. Luckily, the developers were very supportive of this approach. All of the supported position sensing systems communicate with the PC via RS232; the developers suggested using the Fastrak protocol and supplied all of the necessary documentation.

2.2.3 Stradwin Evaluation

Stradwin has been the subject of numerous journal articles. In [18], the creators describe how Stradwin works internally. It discusses the image reconstruction algorithm, the acquisition process, and the visualization techniques employed. The segmentation and surface reconstruction algorithm is documented in [19],[20], and [21]. These articles fully document the reconstruction algorithms and their performance. In addition to inspiring confidence, this fact also relieved us of having to design our own experiments to show that it works. Chapter 10 deals with the use of Stradwin as an experimental tool. This includes capturing data and also manual segmentation of that data. Appendix A is a user's manual for the prototype, which contains a more detailed description of Stradwin and its use. The Stradwin help file is also an excellent reference. It is available from the help menu within the program and also online[2].

[2] `http://mi.eng.cam.ac.uk/~rwp/stradwin/docs/intro.htm`

Chapter 3

Electrical and Optical System Components

During the course of this work two fundamentally different approaches to probe tracking were investigated: a fully inertial implementation and a hybrid, inertial-optical implementation. The two approaches require different sets of sensors, but also have some components in common. The following section briefly describes all of the important sensors and components that were utilized, so that the reader can find this information without having to search through the entire paper.

There are two types of quantities that must be measured by the prototype the tracking system. These are linear displacement and angular displacement. There are many types of sensors for measuring linear displacement. Linear accelerometers are one type that was evaluated. Optical mouse sensors are another.

There are fewer available options for measuring angle. The simplest method is to use a set of two or three linear accelerometers to measure the force of gravity and then derive the angle using basic trigonometry. Unfortunately, this method requires that the only force acting on the accelerometers is gravity. They are therefore unsuitable for use in any application where the sensor will be moved during measurement. The most promising devices for this

application are MEMS angular rate gyroscopes. Angular rate gyroscopes measure angular rate of change, which is then integrated to give angular position. These devices posses the required accuracy, are small enough to be embedded in a transducer, and are inexpensive enough to be practical.

3.1 Gyroscopes

Many MEMS gyroscopes belong to a class of angular rate sensors known as Coriolis Vibratory Gyroscopes (CVGs). Although the physical structure of the device varies between manufacturers, all MEMS gyroscopes rely on a phenomenon know as Coriolis acceleration to measure rotational rate. Both of the gyroscopes evaluated in this work utilize an etched silicon structure.

The Coriolis force is a fictitious force, which arises from observing an object's motion from within a rotating reference frame. It is called fictitious because it is not part of Newton's laws of motion. Real forces are always caused by the interaction of two objects. The second object is nonexistent for fictitious forces. The Coriolis force exists only to explain the apparent motion of objects due to acceleration of the reference frame. Coriolis acceleration is caused by moving an object radially in a rotating reference frame [22].

In a CVG, a proof mass is tethered to a reference frame by silicon springs, as illustrated in Figure 3.1. The reference frame is tethered to the substrate by silicon springs as well, but at 90°with respect to the proof mass springs. The proof mass is driven to resonate at a fixed frequency. During rotation, the Coriolis acceleration experienced by the proof mass is translated into a displacement of the reference frame with respect to the substrate, as illustrated in Figure 3.2. The displacement is measured capacitively. The rotational position of the sensor is derived by integrating the output of the gyroscopes with respect to time. By using three orthogonal gyroscopes, transducer orientation can be tracked in all three rotational degrees of freedom.

MEMS angular rate gyroscopes are a relatively new sensor technology. Most of the first generation devices provided an analog output that required signal conditioning and digitizing

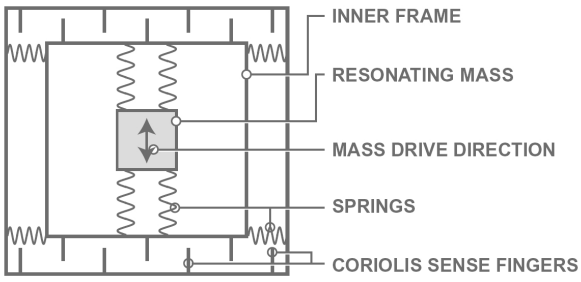


Figure 3.1: Basic structure of a Coriolis vibratory gyroscope [4]

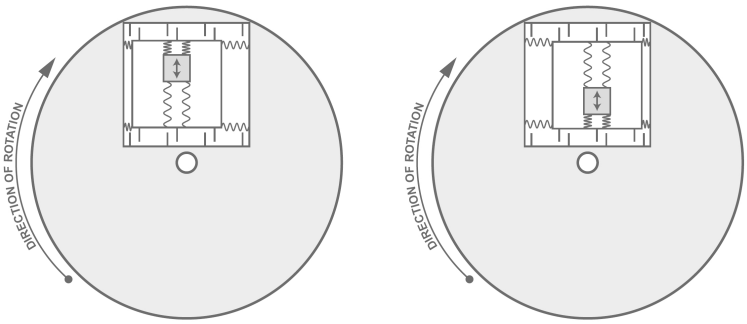


Figure 3.2: Illustration of displacement caused by Coriolis acceleration [4]

before use in a digital tracking system. The second generation, which have become available in the last year or two, often have more advanced features such as a digital interface, integrated signal processing, and others that make system integration much simpler. This class of sensor was highly desirable because they greatly simplify the system design.

Two CVGs were selected for evaluation: the Gyration MG1101a and the Analog Devices ADIS16255. The MG1101a is a 2-axis MEMS CVG. The Analog Devices ADIS16255 is a single-axis MEMS CVG. These two sensors are described in the following sections.

While the ADIS16255 and MG1101a evaluations were in progress, Analog Devices released the preliminary specification for the ADIS16350. This device offers a 3-axis gyroscope and a 3-axis linear accelerometer in a single package. The high level of integration promised to greatly simplify the overall system. Unfortunately, it would only be available near the end of this research. Communication with Analog Devices confirmed that the sensor ele-

Figure 3.3: Gyration MG1101a dual-axis MEMS CVG [5]

ment in the ADIS16350 and the ADIS16255 is identical. Based on this information, the ADIS16255 was evaluated and its performance was assumed to be representative of that of the ADIS16350.

3.1.1 Gyration MG1101a

The MG1101a, pictured in Figure 3.3, is a dual-axis angular rate gyroscope manufactured by Gyration, which was recently acquired by *Movea SA*. It features a dynamic range of $\pm 500^\circ/s$, analog bandwidth of 14.5 Hz, Inter-Integrated Circuit (I^2C) digital interface, 29 Hz digital sampling rate, integrated voltage and temperature sensors, and internal EEPROM for calibration data. Two key figures of merit are the rate noise of $0.18^\circ/s$ and the drift rate of $2^\circ/s$. The complete specifications can be found in Appendix C.

3.1.2 Analog Devices ADIS16255

The ADIS16255, pictured in Figure 3.4, is a single-axis angular rate gyroscope manufactured by Analog Devices. It features user selectable dynamic ranges of $\pm 320^\circ/s$, $\pm 160^\circ/s$ and $\pm 80^\circ/s$, 50 Hz analog bandwidth, selectable digital sampling rate between 0.129 Hz and 256 Hz, temperature compensation, integrated voltage and temperature sensing, optional

Figure 3.4: Analog Devices ADIS16255 single-axis MEMS CVG evaluation module [6]

digital filtering, a Serial Peripheral Interface (SPI) digital interface, bias calibration, and an integrator for angle estimation. As with the MG1101a, the critical Figures of merit are the rate noise of 0.48°/s Root Mean Square (RMS) and drift of $3.6°/\sqrt{hr}$. The complete specifications can be found in Appendix D.

3.1.3 Analog Devices ADIS16350

The ADIS16350, pictured in Figure 3.5, is a highly integrated inertial sensor. It includes three orthogonal angular rate gyroscopes, three orthogonal linear accelerometers, three embedded temperature sensors, and supply voltage monitoring. The sensor elements are fully integrated with the Analog to Digital Conversion (ADC) hardware, Digital Signal Processor (DSP), and a SPI serial port, as illustrated in Figure 3.6.

The ADC block has three dynamic range settings: ±75°/s, ±150°/s, and ±300°/s. The sampling rate is also adjustable over a wide range from 0.413 Hz to 819.2 Hz. The analog signal conditioning electronics have an fixed bandwidth of 350 Hz for all sensor elements. The DSP implements several advanced features that make this device particularly attractive. Offset and scale factor correction registers allow error compensation to be applied in the device. The DSP also implements a Finite Impulse Response (FIR) filter that is adjustable from 2 to 64 taps in power of two step sizes, which correspond to a -3 dB cutoff point between the full 350 Hz analog bandwidth and approximately 8 Hz. One of the most innovative features is the linear acceleration compensation, which uses the signals from the linear accelerometers to correct the output of the gyroscopes. The noise in the rate output

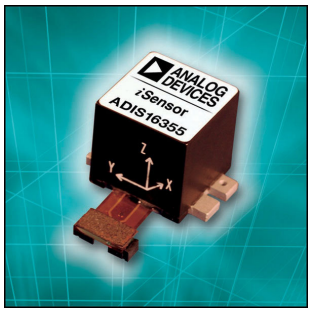


Figure 3.5: Analog Devices ADIS16350 [7]

signal (rate noise) is specified as $0.6°/s$ and the drift is specified as $4.2°/\sqrt{hr}$, with the largest dynamic range setting and no digital filtering. The complete specifications can be found in Appendix E.

3.2 Accelerometers

3.2.1 Linear Accelerometers

MEMS linear accelerometers have been available for much longer than MEMS gyroscopes. As expected, there are a much wider range of these devices available, and they are tailored specifically for a number of applications. There are many different physical structures employed, but most of them rely on the same basic principle. A free floating beam is etched into the surface of the surface of a silicon chip and tethered to the substrate by springs, which are also etched. Fingers are attached to the beam perpendicular to the sensitive axis. The free-floating fingers are interleaved with fixed fingers, which are attached to the substrate, as illustrated in Figure 3.7. Figure 3.8 is a high magnification image of an actual MEMS linear accelerometer.

Each set of fingers forms a differential capacitor. When acceleration is applied along the sensitive axis the spacing between the fingers changes. The resulting change in capacitance is then converted into an electrical signal and in some cases, a digital quantity. The spring constant of the silicon springs and the mass of the beam determine the sensitivity of the

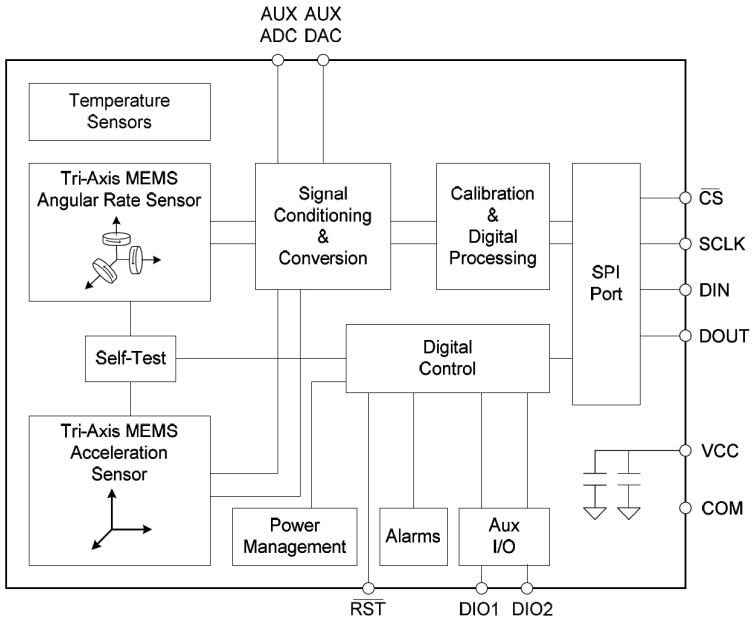


Figure 3.6: Analog Devices ADIS16350 Tri-Axis MEMS CVG block diagram [8]

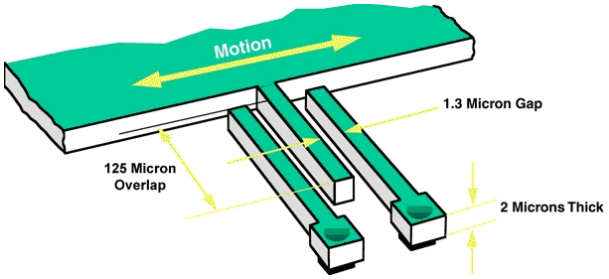


Figure 3.7: Basic structure of a linear MEMS accelerometer [9]

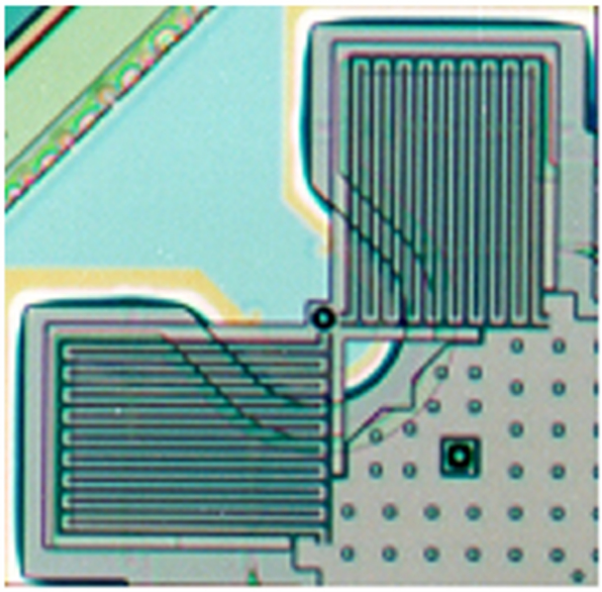


Figure 3.8: Actual structure of a MEMS accelerometer [9]

device and also govern the ruggedness and physical size of the sense element.

This application favors sensitivity over a large dynamic range. Also, the accelerometers need to work in the presence of gravity which requires a dynamic range larger than gravitational acceleration. Like the CVGs that were selected for evaluation, accelerometers that offered high levels of integration were highly desirable.

3.2.2 ST Microelectronics LIS3LV02DL 3-Axis Linear Accelerometer

The LIS3LV02DL is a 3-axis linear accelerometer manufactured by ST Microelectronics. It is a 3-axis device featuring a SPI compatible digital serial interface, ±2 g and ±6 g dynamic range and corresponding sensitivities of 0.00957 $\frac{m/s^2}{LSB}$ and 0.02882 $\frac{m/s^2}{LSB}$, respectively. It also integrates signal processing functions like digital high and low pass filters and angle estimation. The EK3LV02DQ evaluation kit, pictured in Figure 3.9, provides a simple Hardware (HW) and Software (SW) platform for evaluating the device's capabilities. The complete specifications can be found in Appendix G.

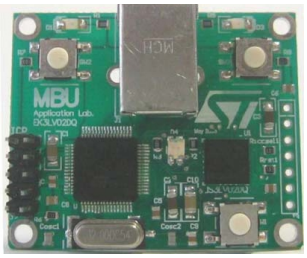


Figure 3.9: LIS3LV02DL Tri-Axis linear accelerometer evaluation module

3.3 Optical Linear Motion Sensors, Lenses, and Optical Fiber Bundles

Optical sensing was also evaluated as method for measuring linear displacement. The only sensors that satisfy the requirements of the prototype are those used in optical mice. These sensors are very inexpensive while supporting high sensitivities. They use a Light Emitting Diode (LED) or a laser diode to illuminate the tracking surface and a small Charge Coupled Device (CCD) array to capture low resolution images at a very high rate. Lenses are used to focus light onto the surface and then onto image sensor. Figure 3.10 illustrates the conceptual arrangement of the components.

The optical mouse sensor continuously captures images of the tracking surface. Surface features are extracted and tracked through the image sequence. Because there is a fixed relationship between the area of the image sensor and area on the tracking surface, the distance traveled in the x and y dimensions can be determined. The exact algorithms used for tracking are trade secrets, but the literature suggests that they are based on edge detection and phase correlation [23]. The ADNS-2610 was evaluated extensively in [24] and was shown to meet the accuracy requirements of ultrasound image registration. It was also integrated into a demonstration freehand 3D ultrasound system that was able to generate accurate reconstructions in the presence variable scan rate, in the x linear DoF, and deviation from a straight line scan path, in the y linear DoF.

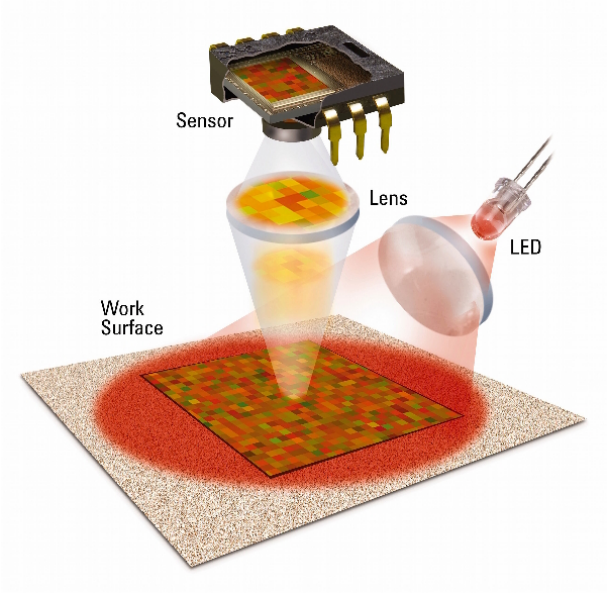


Figure 3.10: Conceptual arrangement of optical mouse system components [10]

3.3.1 Avago ADNS-2610

The Avago ADNS-2610, pictured in Figure 3.11, is a low-cost, small form factor, optical mouse sensor. It provides a simple digital interface via a two wire, SPI serial port. It has a resolution of 400 counts/in and can sense rates of motion up to 12 in/s. The device also supports image capture. Image capture is important in this application because it provides a mechanism for calibrating of the overall optical system. Although its performance specifications are some of the lowest in Avago's product lineup, the ADNS-2610 has been proven adequate for this application. Carsten Poulsen's evaluation demonstrated accuracy within 1% of the expected values. The complete specifications can be found in Appendix F.

3.3.2 Avago HDNS-2100 Optical Mouse Lens

The Avago HDNS-2100 optical mouse lens is designed specifically for use with the ADNS-2610 optical mouse sensor. It is made from injection molded plastic and it includes locating features that ensure proper alignment between the lens and the sensor, the lens and the mouse housing, and spacing between the sensor and the tracking surface. The ADNS-2610

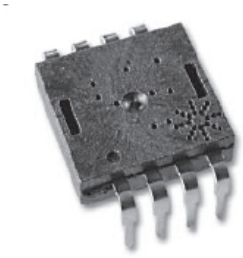


Figure 3.11: Avago Technology ADNS-2610 optical motion sensor

has a very small aperture through which light can enter. In order to direct light into the device the HDNS-2100 has a very low Numerical Aperture (NA). When assembled as specified by Avago, it has a nominal magnification of 1.0 and depth of field of ±.5mm.

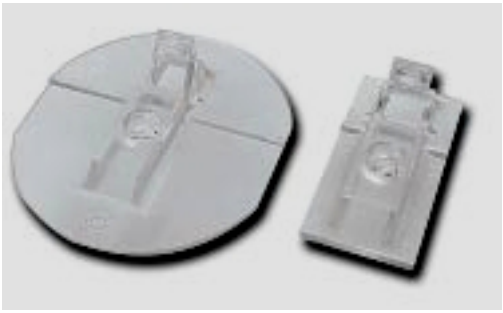


Figure 3.12: Avago HDNS-2100 optical mouse lens [11]

3.3.3 Elmo QT288 Objective Lens

The Elmo QT288 micro lens, pictured in Figure 3.13, is intended to be used in conjunction with Elmo's $\frac{1}{4}$ in diameter CCD video camera. It features an 8 mm focal length, an aperture ratio of 1:2.8, and a 90 mm - ∞ Depth Of Field (DOF). The QT288 is composed of multiple aspheric lenses combined in such as way as to minimize chromatic and monochromatic aberrations. Prior work indicated that simple lenses, composed of just one or two pieces of glass, were inadequate when coupled with an optical fiber. This lens has high enough optical quality to be used with digital cameras in the mega-pixel range.

Figure 3.13: Elmo QT288 micro lens

3.3.4 Schott Acid Leached Fiber Bundle

The Scott acid leached fiber bundle possesses a number of unique features. One of its primary uses is in endoscopes and as such, it is extremely flexible, as illustrated in Figure 3.14. It is composed of 18,000 individual glass fibers, each of which is 7.6 μm in diameter, and it has a NA of 0.38. Irene Gouverneur did a preliminary investigation that showed that this fiber has excellent imaging performance when coupled with the Elmo lens [17].

3.3.5 Edmund's Optical Fiber Optic Rod

Fiber optic rods, also know as image conduits, are composed of many individual fibers embedded in a rigid medium such as glass or epoxy. Several examples are pictured in Figure 3.15. Three types offered by Edmund's Optical as standard products were evaluated: a 3,012 element 0.062 in diameter rod, a 3,012 element 0.125 in diameter rod, and a 50,419 element 0.125 in diameter rod. All samples were one inch in length. The complete specifications can be found in Appendix H. These image conduits were investigated in [24] and discounted, but as shown in Chapter 7, they can be made to work.

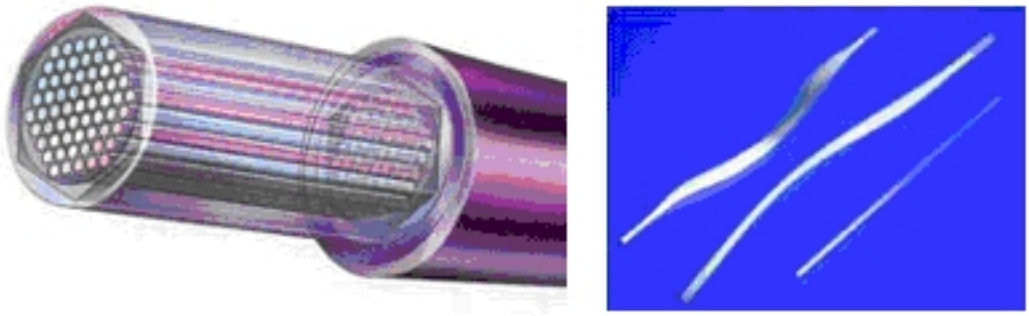


Figure 3.14: Schott acid leached fiber bundle

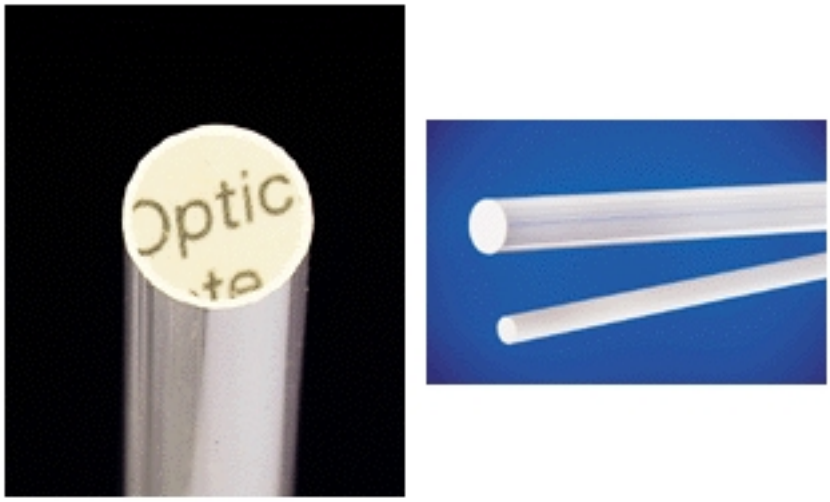


Figure 3.15: Edmund's Optical rigid optical conduit

Chapter 4

Test Platforms

Three types of sensors were evaluated for inclusion in the tracking system: MEMS gyroscopes, MEMS accelerometers, and optical trackers. To test them, it was necessary to develop a test platform that could communicate with each of the sensors, generate test stimuli, collect and store data, and analyze that data. Each class of sensor required a different type of stimulus: rotational motion for the gyroscopes and linear motion for the accelerometers and optical trackers. Furthermore, the individual sensors had different electrical specifications and communications interfaces. These competing requirements made it difficult to find a single test platform that could support all of the sensors.

The high level structure of the system is illustrated in Figure 4.1. A Windows PC was used to control the individual system components. Matlab was utilized extensively to automate the test process. A PICDEM FS-USB demonstration board was connected to the PC over Universal Serial Bus (USB) and interfaced with each sensor using either a SPI or I^2C serial port. Test data was transferred to and from the sensors over the serial interfaces and then forwarded to the PC over the USB link. The rotary and linear motion platforms also connected to the PC. As indicated by the dotted line in Figure 4.1, the SoloistCP servo drive (Aerotech Corp.) was configured via USB, but did not require real-time control during testing. The solid line connecting the EK3LV02DQ linear accelerometer evaluation module and the PC indicates that USB was used to transfer real-time test data.

The solid line connecting the HP7255A XY plotter also indicates that it required real-time communication during tests. However, in this case, the communications channel was implemented using RS232.

Because the optical tracking sensors needed to be evaluated as part of a larger optical system, the details of those tests and test apparatus are presented separately in Chapter 7. However, the data acquisition system described here was used for that testing. For that reason, the details of the electrical interface and the firmware required to control it are presented here.

4.1 Data Acquisition Hardware

The core of the sensor evaluation system was a PICDEM FS-USB demonstration board from Microchip Technology Inc. The board is pictured in Figure 4.2. It features a PIC18F4550 microcontroller with a full-speed USB interface and a broad feature set, suitable for interfacing with all of the sensors we planned to evaluate. Each of the sensors required different serial port hardware for communication, and none of them could be connected directly to a PC's serial port. The PICDEM FS-USB demonstration board was used as the common tool for communicating with the individual sensors and then relaying that data to the host PC. The following sections will describe the PICDEM FS-USB demonstration board in greater detail and then document the electrical interfaces between it and the individual sensors that were evaluated.

4.1.1 PICDEM FS-USB Demonstration Board

The PICDEM FS-USB demonstration board provides an impressive feature set given its $60 price. It features a full-speed, USB 2.0 compliant interface, RS232, high General Purpose I/O (GPIO) pin count, multiple power supply options, and a PIC18F4550 microcontroller operating at a maximum clock frequency of 48 MHz. The microcontroller itself has many useful features including a Universal Synchronous-Asynchronous Receiver-Transmitter (USART), SPI serial port, I^2C serial port, a multi-channel ADC, and multiple Capture and

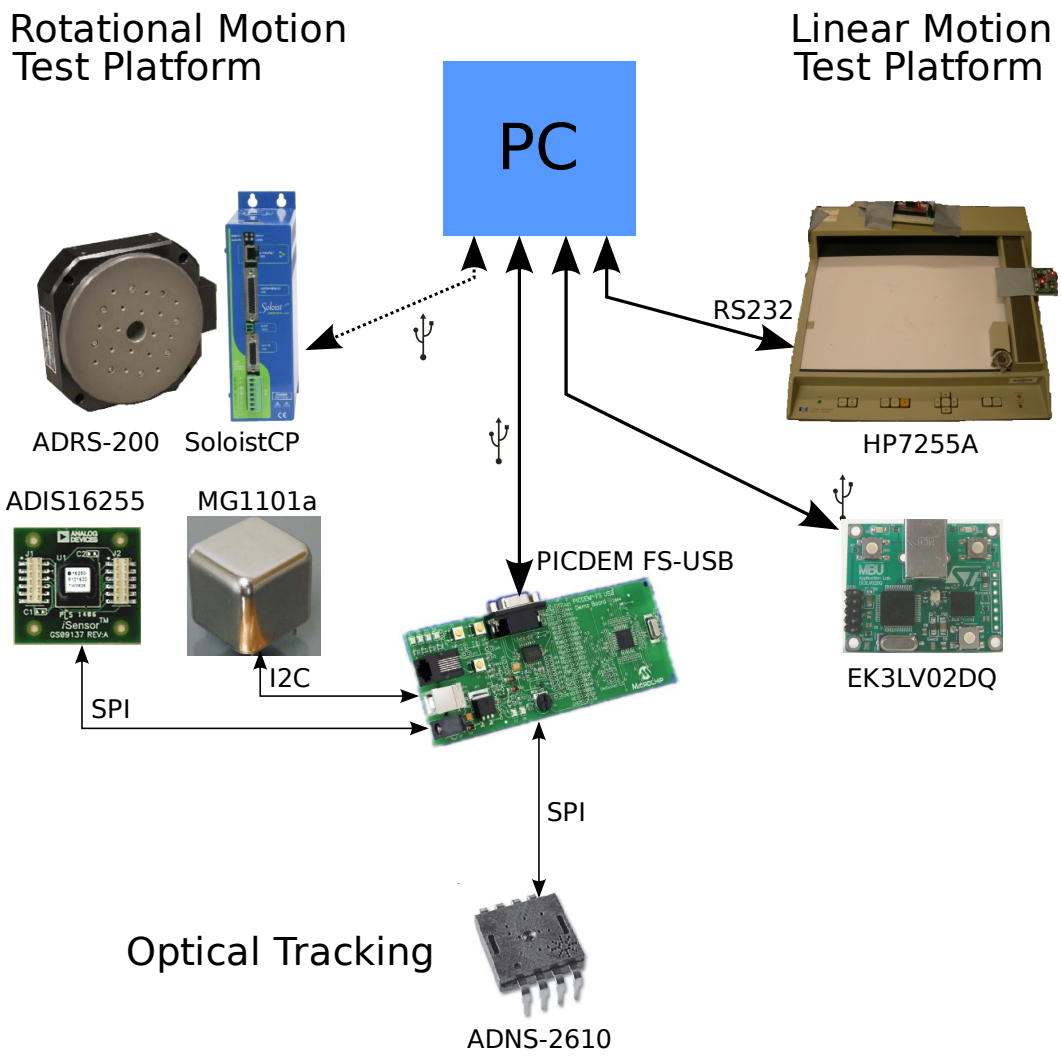


Figure 4.1: High level block diagram of the test system

Figure 4.2: PICDEM FS-USB demonstration board [12]

Compare (CCP), Pulse Width Modulation (PWM), and timer modules. This particular tool was selected for several reasons. First, purchasing a demonstration board saved a lot of time when compared to building one from scratch. Support for all major serial interface standards was also very important because all of the sensors identified for evaluation had digital front-ends utilizing some type of serial interface. In addition, early in the course of this work it appeared that the proof-of-concept system would use CustusX for 3D visualization and reconstruction. Had that been the case, the acquisition system would not have been required to do any signal processing, and the PICDEM board would have been useful in the prototype system as well.

4.1.2 MG1101a Gyroscope Physical Interface

The MG1101a gyroscope uses an I^2C serial interface, which is a two-wire, multi-master serial bus. The MG1101a evaluation module provided access to the I^2C Receive (RX) and Transmit (TX) signals as well as power and ground on a 4 x 0.100 in header. The MG1101a evaluation module is pictured in Figure 4.3. A four conductor ribbon cable was constructed to connect the demonstration board to the PICDEM board.

The I^2C implementation used in this system is very simple. The PIC18F4550 acts as the master and the MG1101a acts as the slave. The PIC's RB0 and RB1 pins are connected to

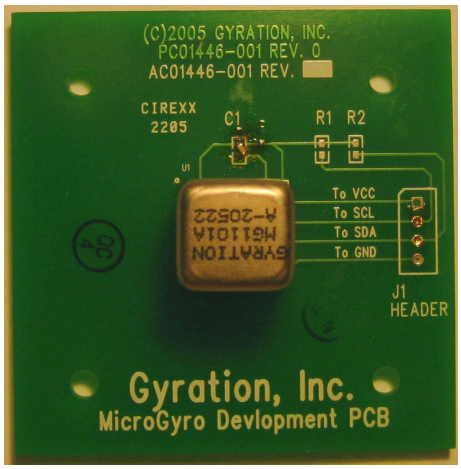


Figure 4.3: MG1101a evaluation module

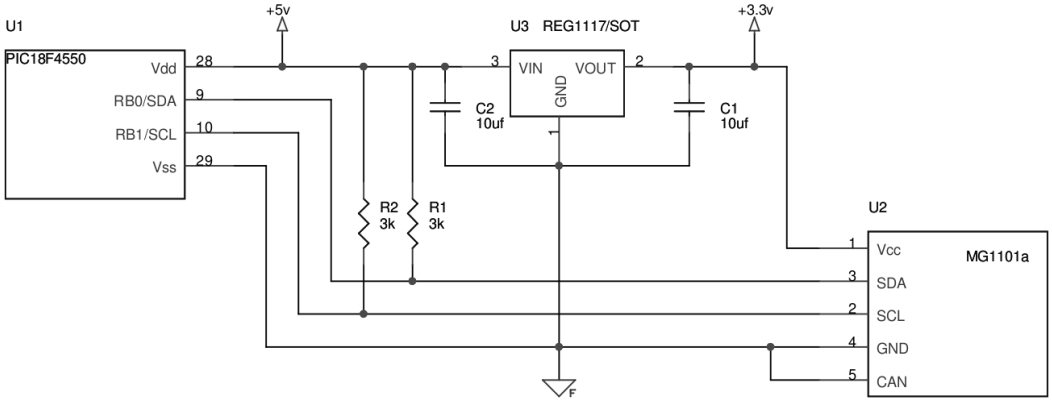


Figure 4.4: MG1101a to PIC18F4550 interface schematic

the Serial Data (SDA) and Serial Clock (SCL) signals of the I^2C bus, as are the SDA and SCL pins of the MG1101a. These two signals are pulled up to the +5 v supply of the PIC with 3 kΩ resistors. A linear voltage regulator converts +5 v from the PICDEM board to the +3.3 v required by the MG1101a. Even though the MG1101a uses a +3.3v supply, its SDA and SCL pins are +5 v tolerant, which simplifies the interface by eliminating the need for a level translator. The circuit schematic is presented in Figure 4.4.

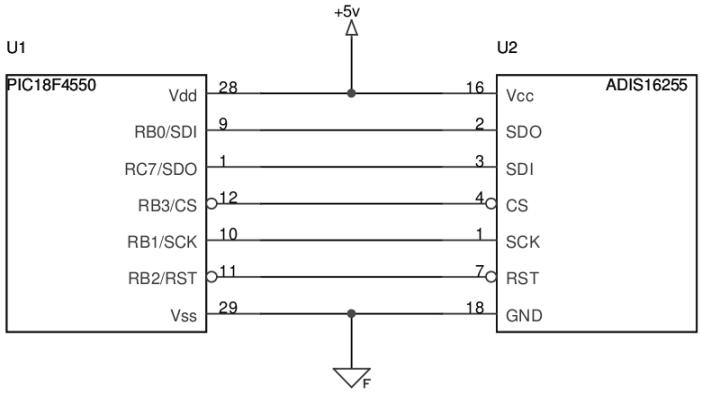


Figure 4.5: ADIS16255 to PIC18F4550 interface schematic

4.1.3 ADIS16255 Gyroscope Physical Interface

The ADIS16255 gyroscope has a SPI serial interface. SPI is four-wire, full duplex, synchronous serial bus. The PIC18F4550 acts as the bus master and the ADIS16255 acts as a slave. The schematic for the interface between them is illustrated in Figure 4.5. The PIC's RB0, RB1, RB3, and RC7 are connected to the Serial Data In (SDI), Serial Clock (SCK), Chip Select (CS), and Serial Data Out (SDO) SPI bus signals, respectively. On the ADIS16255 side of the bus the SDO signal is connected to the SDI pin and the SDI signal is connected to the SDO pin. The ADIS16255 requires a +5 v supply which eliminates the regulator that was required for the MG1101a. The SPI electrical specification does not call for pull-up resistors and they are also absent in Figure 4.5.

4.1.4 ADNS-2610 Optical Mouse Sensor Physical Interface

Unlike the other sensors, the ADNS-2610 optical mouse sensor requires a number of external components to function. It is possible to modify the Printed Circuit Board (PCB) of an optical mouse for use with the PICDEM based acquisition system, but the physical dimensions of the mouse PCB make it difficult to use in conjunction with the rest of the optical test apparatus. A small PCB, pictured in Figure 4.6, was designed to allow the sensor and its auxiliary components to fit in the available space. It also has features that allow it to

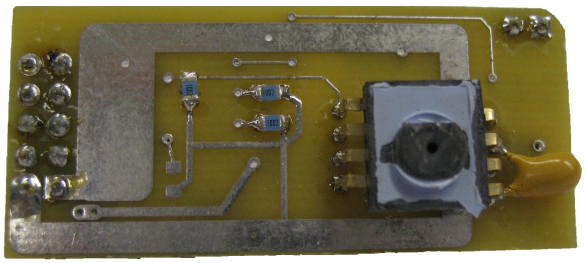


Figure 4.6: ADNS-2610 carrier PCB

interface with devices other than the PICDEM board. Components are populated based on the intended application. Figure 4.7 is a schematic of the interface between the ADNS-2610 and the PIC18F4550. U2 is an open-drain, non-inverting buffer Integrated Circuit (IC) that is compatible with a wide range of I/O voltage standards, such as Transistor-Trasistor Logic (TTL) and Complimentary Metal Oxide Semiconductor (CMOS). It allows the ADNS-2610, which requires a +5 v supply, to interface with systems using an equal or lower signaling voltage. In Figure 4.7, that voltage is specified as +3.3 v, however, U2 is actually compatible with voltages down to +1.8 v. That can be accomplished by applying the lower signaling voltage to J2-3 instead of +3.3 v. For instance, to connect to the PICDEM board, +5 v should be applied to J2-3. The Field Programmable Gate Array (FPGA) board that the final system is implemented on has a maximum I/O voltage of +3.3 v, in which case +3.3 v must be supplied on J2-3, as illustrated in Figure 4.7.

The ADNS-2610 optical mouse sensor uses a half-duplex synchronous serial interface. The serial interface does not conform to a specific standard, but it is generic enough to be connected to most microcontroller serial ports. The PIC18F4550 acts as the master and the ADNS-2610 acts as the slave. The PIC's RC7 and RC6 pins are connected to the Data (DT) and Clock (CK) bus signals, respectively. The DT signal is connected to the ADNS-2610's Serial Data I/O (SDIO) pin and the CK signal to the SCK pin.

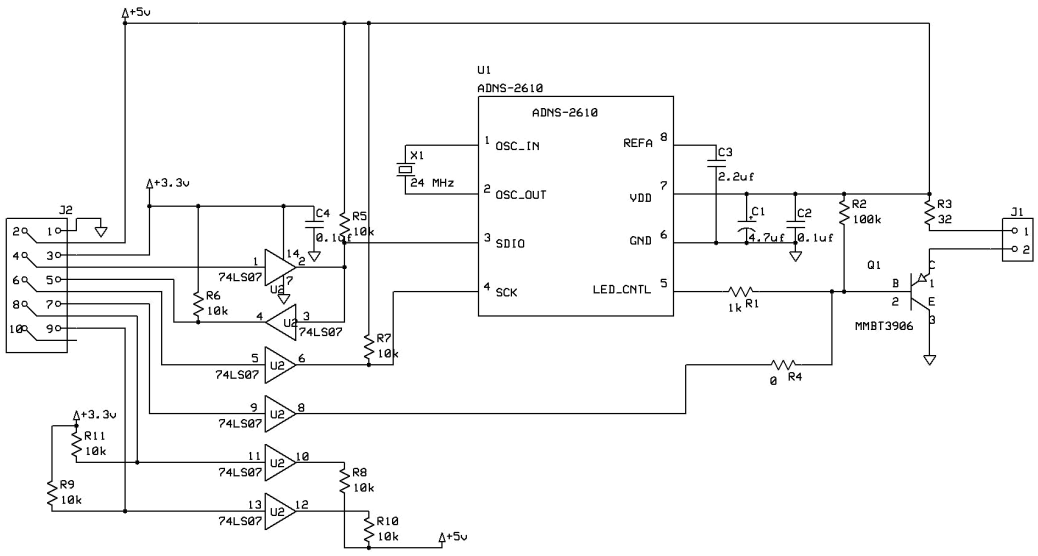


Figure 4.7: Schematic for the ADNS-2610 to PIC18F4550 interface board

4.1.5 EK3LV02DQ Linear Accelerometer Evaluation Module

The LIS3LV02DL 3-axis linear accelerometer was tested using the EK3LV02DQ evaluation module. The evaluation module has a USB interface and came with software that allowed data to be collected directly. No additional hardware or software was required.

4.2 Firmware

The acquisition system's firmware is written in C for the Microchip C18 compiler. The base Firmware (FW) was provided by Microchip, including the USB stack. The system was used to interface with 6 different sensors, 3 of which (the ADI iMEMS devices) were the same with respect to the PIC interface. Each different sensor interface required different serial port configurations and different serial communications routines. The rest of the FW is common. Because only one sensor would be attached to the acquisition system at any given time, three separate versions of the code were developed rather than attempt to make the sensor interface configurable during execution.

The following sections document the firmware that was developed for the PICDEM FS-

USB board. The FW for communicating with each sensor will be treated individually. The FW that handles the communication between the PICDEM board and the PC will be discussed in Section 4.5. Please note that the LIS3LV02DL linear accelerometer was obtained as part of the EK3LV02DQ USB based evaluation module that was already fully programmed. Therefore, it will not be covered in this section.

4.2.1 MG1101a Gyroscope Communication Firmware

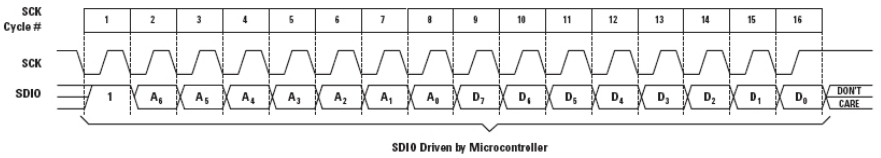
The FW that handled communication with the MG1101a Gyroscope was the simplest. This was due largely to the fact that most of timing in the I^2C protocol is either handled in hardware or defined explicitly in the protocol. All of the low level communications routines were provided in the PIC C18 API, which also simplified the programming task.

The MG1101a is logically separated into two distinct devices: the gyroscope and an Electronically Erasable Programmable Read Only Memory (EEPROM). The EEPROM is 1024 bytes long and the first 16 bytes are used to store calibration constants that are loaded at the factory. The rest of EEPROM is available for user data. When the MG1101 is first powered up, the calibration data must be read out of the EEPROM and then loaded into the gyroscope. The I^2C protocol embeds the address of the target slave device in the first byte of the serial transaction. The address of the EEPROM is 0x50 and the address of the gyroscope is 0x57. To load the calibration constants into the gyroscope, the PIC FW performs 16 sequential reads to memory addresses 0x00 through 0x0F of the EEPROM. The FW then performs 16 sequential writes to addresses 0x00 through 0x0F of the gyroscope.

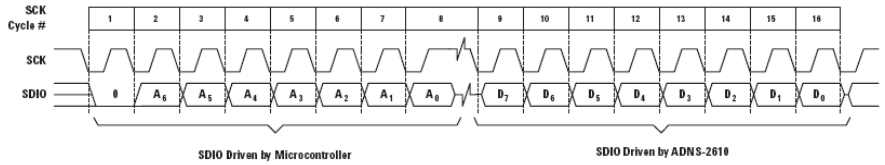
During normal operation, the PIC's TMR0 is used to provide an interrupt every 33ms. The Interrupt Service Routine (ISR) performs a 2 byte read to retrieve the angular rate data and then stores the result until the host PC requests it.

4.2.2 ADNS-2610 Optical Mouse Sensor Communication Firmware

The ADNS-2610 communication routines are also relatively simple. It uses a half-duplex serial interface and it does not need to be polled frequently. Each serial transfer consists of a command phase and a data phase. A single byte is transferred in each phase. Bit 0 of



(a) Write



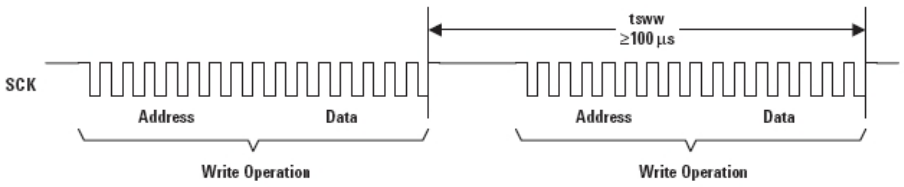
(b) Read

Figure 4.8: ADNS-2610 command formats

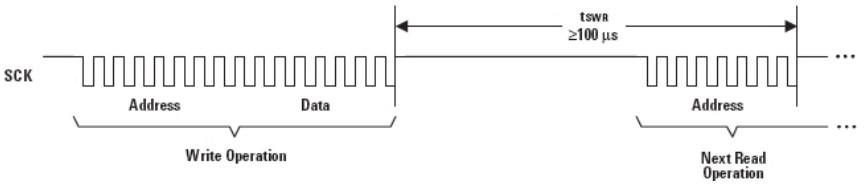
the command byte indicates whether the command is a read or a write, and bits 1-7 are the register address, as illustrated in Figure 4.8. Write commands can be followed immediately by the data byte. However, read commands require a pause of at least 100 μs between the command and data phases. There are also additional timing requirements for sequential writes, writes followed by reads, and reads followed by anything, as illustrated in Figures 4.9(a), 4.9(b), and 4.9(c), respectively.

Polled serial communication is implemented using a timer to generate synchronous interrupts. The PIC's TMR0 is configured to interrupt every 100 ms. The TMR0 ISR then performs a read from the `DELTA_X` and `DELTA_Y` registers of the ADNS-2610. The required intra and inter-command timing is generated using the *delay()* functions provided by the Microchip C18 API [25]. Write commands are only utilized for device control and do not occur during normal operation. Figure 4.10(a) illustrates the write process and Figure 4.10(b) illustrates the read process.

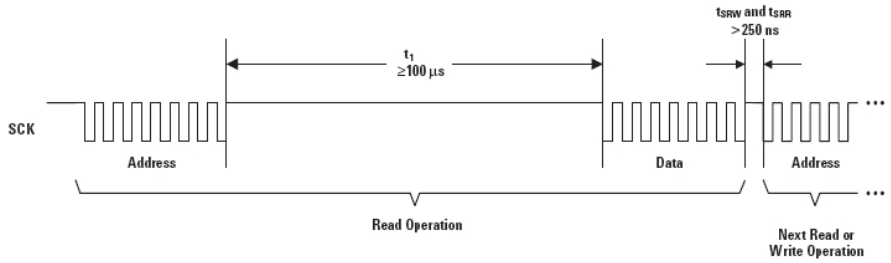
Another important feature of the ADNS-2610 is its ability to capture an image of the surface it is tracking. This feature was used extensively in the design and testing of the optical system. The image capture process and the normal tracking functions cannot take place simultaneously. In order to start an image acquisition, a write command is issued



(a) Write after write

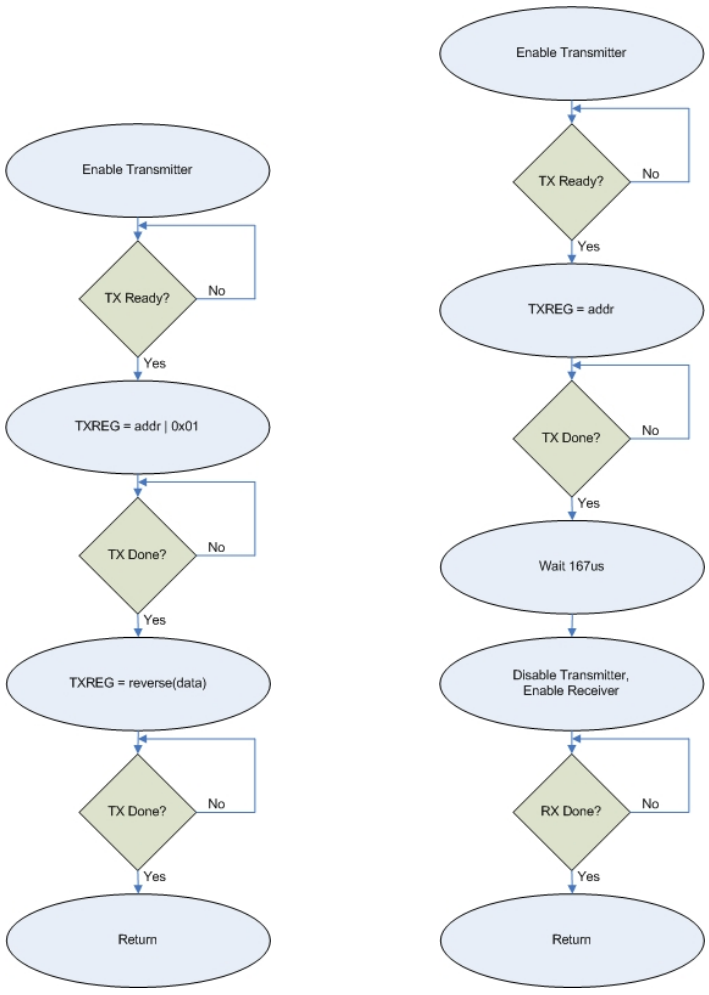


(b) Read after write



(c) Anything after read

Figure 4.9: ADNS-2610 inter-command timing



(a) Write (b) Read

Figure 4.10: Flowcharts for the basic ADNS-2610 serial communication routines

to the `PIXEL_DATA` register. This is followed by 324 consecutive reads of the `PIXEL_DATA` register to retrieve the 18x18 pixels. It is important to note that the ADNS-2610 requires a significant period of time between these successive reads, more than is suggested by the minimum sequential read timing specified in Figure 4.9. The exact delay period was never quantified experimentally, but a delay of 833 μs works. The process is further complicated by the fact that 324 bytes is more than can be accommodated in a single transfer to the USB host and is also larger than the Random Access Memory (RAM) available in the PIC at runtime. The solution is illustrated in Figure 4.11, where a static counter is used to break the process into multiple parts.

4.2.3 ADIS16255 Gyroscope Communication Firmware

The ADIS16255 communication routines are more complicated than those of the ADNS-2610. This is due to the complicated serial protocol employed. The ADIS16255 uses a full duplex SPI serial link with a clock frequency of 2 MHz. The PIC's Master Synchronous Serial Port (MSSP) is configured as a SPI master. None of the module's internal clock generation methods yield an acceptable transmission frequency so TMR2 is used as the clock source with a period of 2 cycles of the master clock. The PIC's RB3 I/O pin is used as a CS, also known as a Slave Select (SS) signal.

The ADIS16255 uses a 16 bit serial data format as illustrated in Figure 4.12. The first half of the word is a command byte and the second is a data byte. Bit 7 of the command byte defines whether the command is a read or a write. Bits 0-6 define which register the command will access. For read commands, the value of the data byte is not important because it is not used by the device. For write commands, the data byte contains the data to be written. The ADIS16255 has 16 bit registers that are byte addressable. This means that two write commands are required to change the state of an entire register. Because the interface is full duplex, each time a byte is transmitted a byte is also received. That byte is dependent on the previous command and may or may not be useful data. If the previous command was a read command, then the two bytes clocked in when the subsequent command is issued contain the data from the addressed register. Regardless of which byte

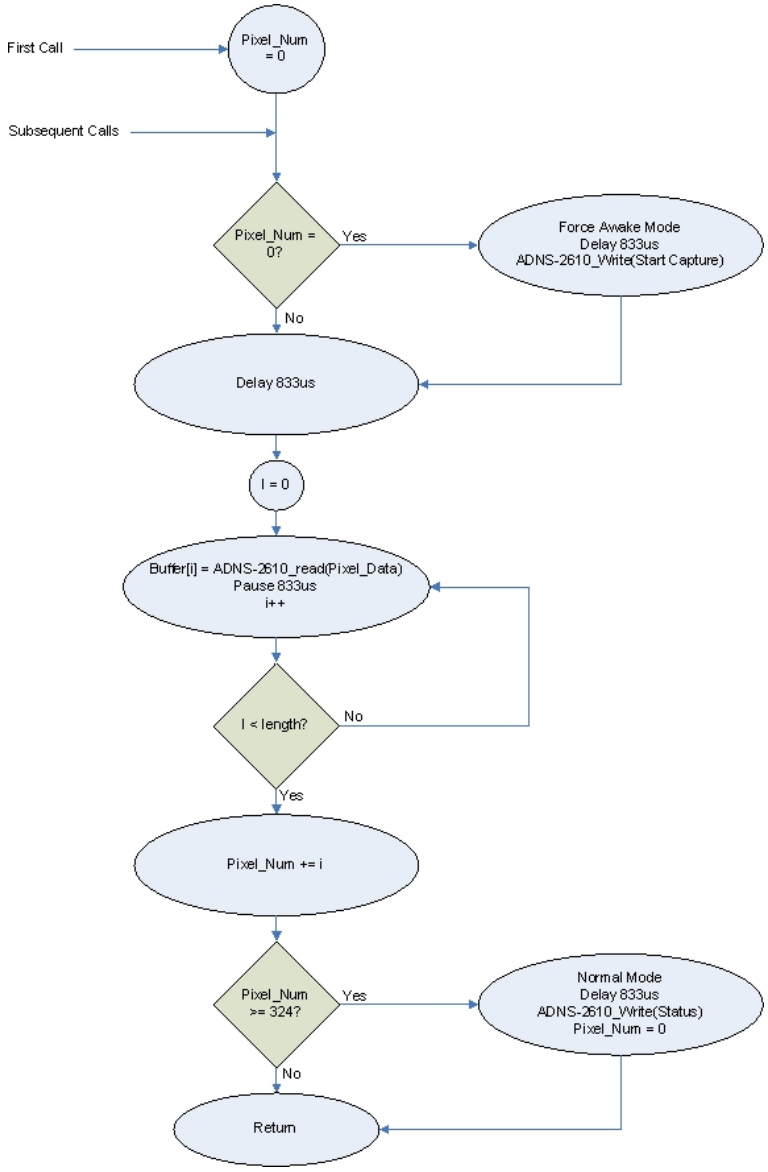


Figure 4.11: ADNS-2610 image retrieval flow chart

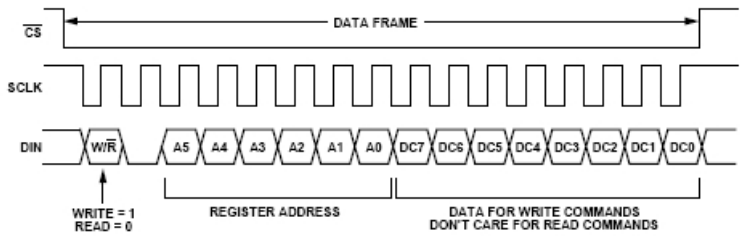


Figure 4.12: ADIS16255 DIN sequence [13]

of a register is addressed by a read command, the entire 16 bit contents of the register will be transferred during the subsequent command. If the previous command was a write, the data returned during the current command has no meaning. The net result is that two 16 bit transfers are required to complete all commands other than a single-byte write.

There is an additional complication in the interface between the ADIS16255 and PIC. The PICs MSSP, like most SPI serial ports, treats each transfer as a single 8 bit transaction. The ADIS16255 SPI serial port utilizes 16 bit transactions and does not directly support single byte transfers. The waveform in Figure 4.12 makes no provision for a pause between the first and second bytes of the transfer, but the PIC is not capable of transmitting two bytes back-to-back without inserting at least one idle CS cycle in between. As a result, the automatic features of the PICs MSSP must be bypassed, and many of the functions controlled directly by the FW. The PIC's MSSP would normally drive CS automatically but in this situation that would result in CS going high after only a byte. Therefore, the CS signal is connected to one of the PICs GPIO pins and controlled directly by the FW. Luckily, the ADIS16255 serial port does not care about the relative timing of each bit in a given transfer. Once activated by a falling edge on CS, all that is required are 16 rising edges of SCK and a rising edge on CS to complete the transfer. This fact was demonstrated experimentally but it was also confirmed by an Analog Devices application engineer.

Both the ADIS16255 and the PIC clock data out on the falling edge of SCK, and clock data in on the rising edge of SCK. When combined with the requirement that SCK idle high, as illustrated in Figure 4.13, this means that none of the PIC's MSSP operating modes are directly compatible with the ADIS16255. To circumvent the problem, the PIC FW changes

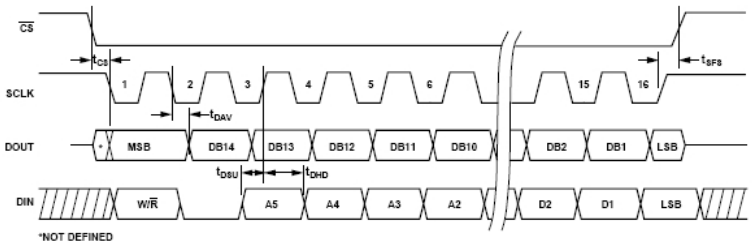


Figure 4.13: ADIS16255 low level timing diagram [13]

the operating mode of the MSSP at the beginning and end of each 16 bit transfer. Prior to the start of a transfer, the MSSP is set so that SCK idles high. To begin a transfer, CS is driven low, SCK is set to idle low, and data is written to the MSSP transmit register. The next SCK falling edge clocks out the first data bit. The FW polls the MSSP and when the first byte transfer is complete, the second byte is immediately written. Since the PIC is executing instructions six times faster than the SCK frequency, there is no interruption in the 16 bit frame, as seen by the ADIS16255. The second byte is transferred the same way as the first and when the transfer is complete, the MSSP mode is again switched so that SCK idles high. The FW then drives CS high to complete the transfer. The process is illustrated in Figure 4.14.

There are two communication modes implemented in the FW. The first is used when configuring the ADIS16255 and essentially treats the MSSP as if it were an asynchronous interface. Register writes consist of a single 16 bit transaction, as illustrated in Figure 4.14(a). Register reads consist of two 16 bit transactions, as illustrated in Figure 4.14(b). The first transfer issues the read command and the second is a dummy command whose only purpose is to clock in the data from the previous read. This is inefficient but necessary. The second mode is utilized when the ADIS16255's angular rate output is being actively sampled. In this mode, TMR0 is configured to interrupt every 3.906 ms. The ISR then performs a single 16 bit read transaction. Because the read command is always the same in this mode, the FW takes advantage of the full-duplex nature of the interface. The data that is clocked in by the PIC is the data from the previous command. In this way, read commands can be performed effectively with only a single read transaction. The only caveat

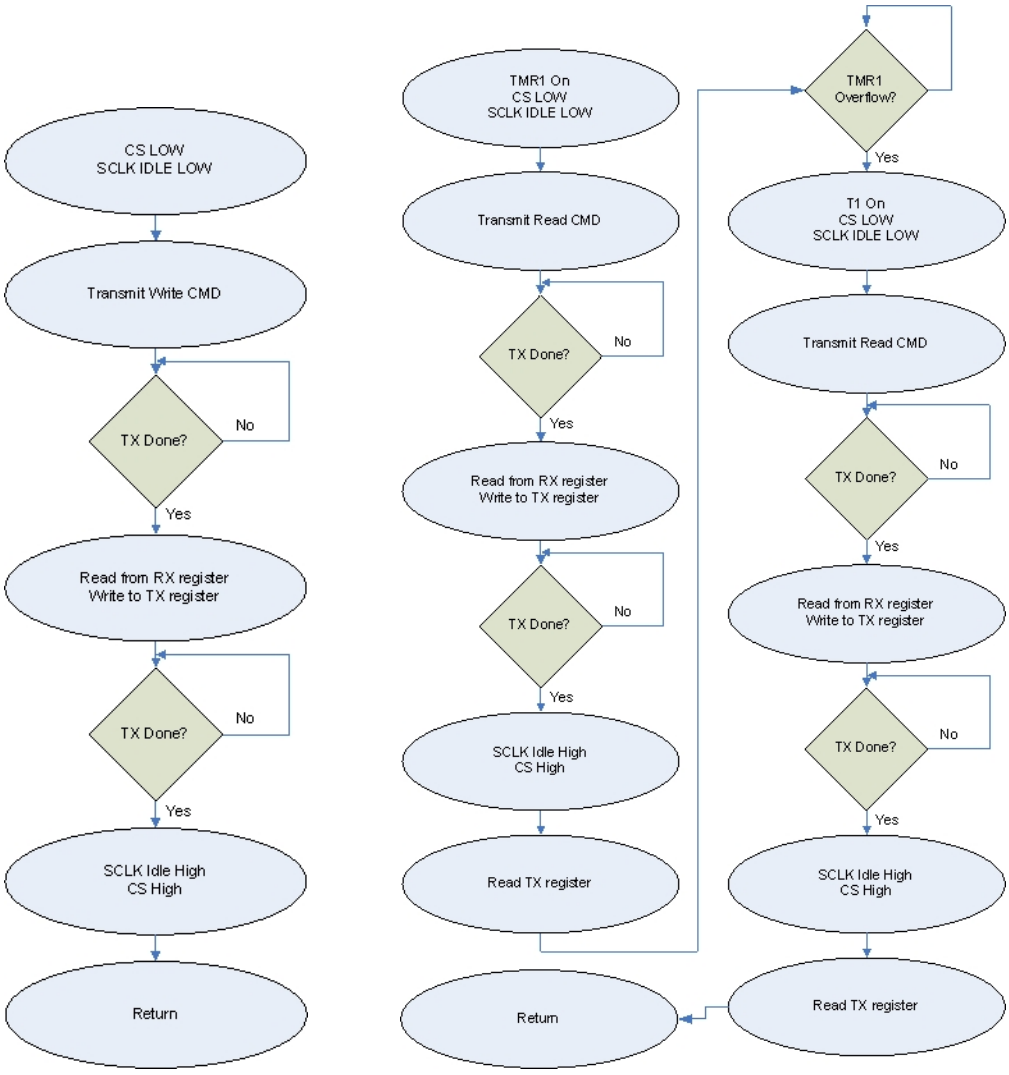
is that a read command must be issued before activating the TMR0 interrupt, so that the first data read is valid.

4.3 Aerotech ADRS-200 Rotation Table and Soloist CP Servo Controller

An Aerotech ADRS-200 rotation table, in conjunction with an Aerotech SoloistCP servo controller, were used to provide test stimulus for the MEMS gyroscopes. The ADRS-200 is pictured in Figure 4.15(a) and the SoloistCP in Figure 4.15(b). The ADRS-200 is a very high quality machine incorporating a direct-drive, slotless, brushless servomotor and a 10,000 cnt/rev encoder. The SoloistCP is a combination servo controller and amplifier. This combination allows the ADRS-200 to achieve an accuracy of greater than 300 μrad and a repeatability of 20 μrad. The slotless motor design allows the table to rotate at very low angular velocities with very little variation. This was a critical feature for our tests because we were primarily interested evaluating gyroscope performance below 10°/s.

The SoloistCP servo controller came with a PC based Integrated Development Environment (IDE) that was used to develop the motion programs used for testing. A detailed explanation of the specific motion profiles can be found in Chapter 5. Once the programs were complete they were downloaded to the controller and stored in its non-volatile memory.

In order to eliminate the need for operator participation during the tests, an automatic triggering mechanism was developed. The SoloistCP has a number of digital and analog inputs. The status of these inputs can be queried during program execution. An output port on the PIC was connected to one of the inputs on the SoloistCP. The state of the PIC output can be changed by sending it the appropriate command over USB. Each motion program starts by polling the value of the trigger input. When the voltage on the input changes from 5 v to 0 v, the program proceeds. When the program is complete, it goes back to the beginning and waits for another trigger. This allows a Matlab script to synchronize the motion program with the data acquisition system. In this way, a test can be looped indefinitely without human interaction.



(a) Write (b) Read

Figure 4.14: Flowcharts for the ADIS16255 single read and write transactions

(a) (b)

Figure 4.15: The ADRS-200 rotation table and SoloistCP servo controller

4.4 HP7255A XY Plotter

The HP7255A XY plotter, pictured in Figure 4.16, was used as the linear motion platform for testing the MEMS linear accelerometers. The Device Under Test (DUT) was mounted to the plotter arm so that it would experience the same motions that the plotter pen went through. Commands were sent to the plotter over an RS-232 interface. The plotter accepts HP Graphics Language (HP-GL), which is a simple, American Standard Code for Information Interchange (ASCII) based protocol. Matlab scripts were used to send the command sequence necessary to produce a particular motion profile. More information regarding the actual profiles used during testing can be found in Chapter 6.

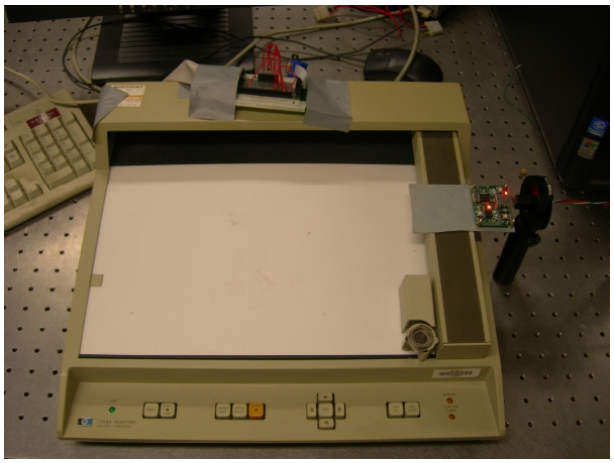


Figure 4.16: The HP7255A XY plotter

4.5 Data Collection and Analysis Software

A WindowsXP PC was used to interface with the system components, automate the test process, and record and store the data. Matlab was used to collect data from the data acquisition system as well as to control the motion platforms. A DLL and kernel mode device driver were developed to enable Matlab to communicate with the data acquisition system over USB. The HP7255A plotter was controlled directly using Matlab's built in serial

I/O functionality.

The hierarchy of host PC software is illustrated in Figure 4.17. At the lowest level, kernel mode drivers interface directly with the PC hardware. For the RS232 serial port this functionality is included in the Windows operating system. Matlab can then access the serial port directly as illustrated by the arrows on the right hand side of Figure 4.17. The USB dataflow is illustrated on the left hand side of Figure 4.17. The `mchpusb.sys` device driver, provided by Microchip, implements the kernel mode code that actually interacts with the USB hardware. The code in `mchpusbapi.dll` implements the user mode routines that are called by user applications. In this case, the user mode application is another DLL, `adis_usb_ctrl.dll`, that implements the high level USB communications protocol that is used to control the test system and transfer data. At the top level, a Matlab script provides the overall test automation functionality.

In order to understand how the interface between the PC and the PICDEM board was implemented, some familiarity with USB is required. The following section provides a brief introduction to the USB protocol.

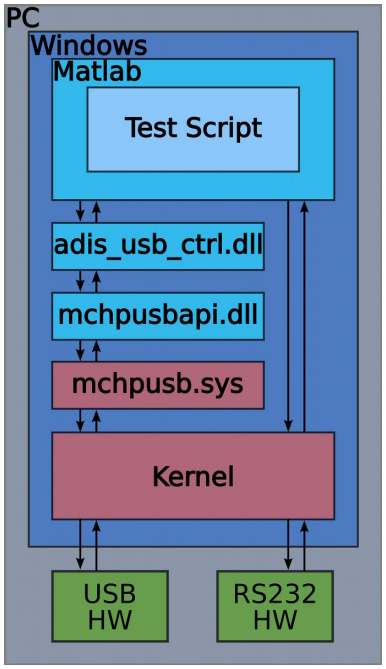


Figure 4.17: Major host software components

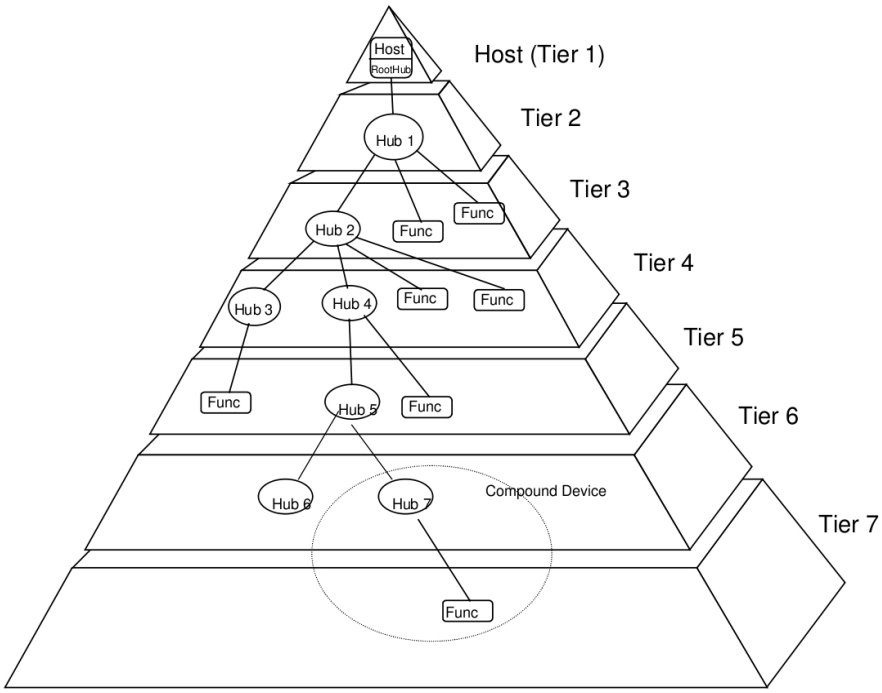


Figure 4.18: USB topology [14]

4.5.1 Universal Serial Bus

The data acquisition system communicates with a PC using the USB. A large portion of the effort that went into the development of this system was concentrated on the microcontroller firmware and Windows device drivers that were required to link the two. The brief introduction to USB presented here is intended to provide some context for the firmware and software descriptions that follow.

USB is a high speed serial interface specification for computers and computer peripherals. The USB Implementers Forum, Inc. (USB-IF)[1] is responsible for maintaining the USB standard. The USB specification covers all aspects of the interface including mechanical requirements, electrical specifications, the protocol layer, bus topology, device framework, and host hardware and software. The specification is presently at revision 2.0.

The USB connects USB devices with the USB host [14]. In any USB system there

[1] `http://www.usb.org/`

can be only one host, also known as the host controller. The host controller is composed of hardware, firmware, and/or software. USB devices are further broken down into classes, such as hub, Human Interface Device (HID), printer, mass storage device, etc. Fundamentally, however, there are only two device classes: Hubs and Functions. The USB host contains a hub, known as the root hub, from which all other connections originate. The overall bus topology is illustrated in Figure 4.18.

The physical layer of USB is very simple. It consists of a single cable composed of four conductors. Two of these are for +5 v and GND. The specification guarantees that at least 100 mA and at most 500 mA of current can be supplied over the cable. The other two conductors form a differential pair that is used for the actual data transmission, as in Figure 4.19.

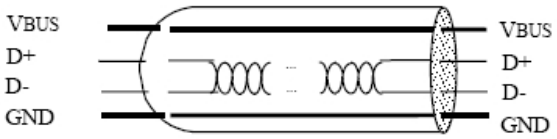


Figure 4.19: The USB physical layer [14]

USB is a polled architecture. All data transfers are host initiated. A connection between a host and a device is known as a pipe. The device side of a host-device pipe is called an End Point (EP). EPs are enumerated sequentially and are composed of an In EP and an Out EP. The In EP is for data that is to be transferred from the device to the host. The Out EP is for data that is to be transferred from the host to the device. All devices are required to implement at least one endpoint (EP0). The host-device connection to EP0 is known as the Default Control Pipe, which is used to initialize the device and for device control. Other EPs may be defined for a particular device, but may never exceed 15 in number, in addition to EP0. This arrangement is illustrated in Figure 4.20.

The USB specification describes four types of data transfers. The bus itself is capable of supporting a wide variety of communication regimes, from streams, which do not even guarantee data delivery, to transfers with guaranteed delivery and latency. Control transfers are used to configure a device when it is attached and for other device specific purposes.

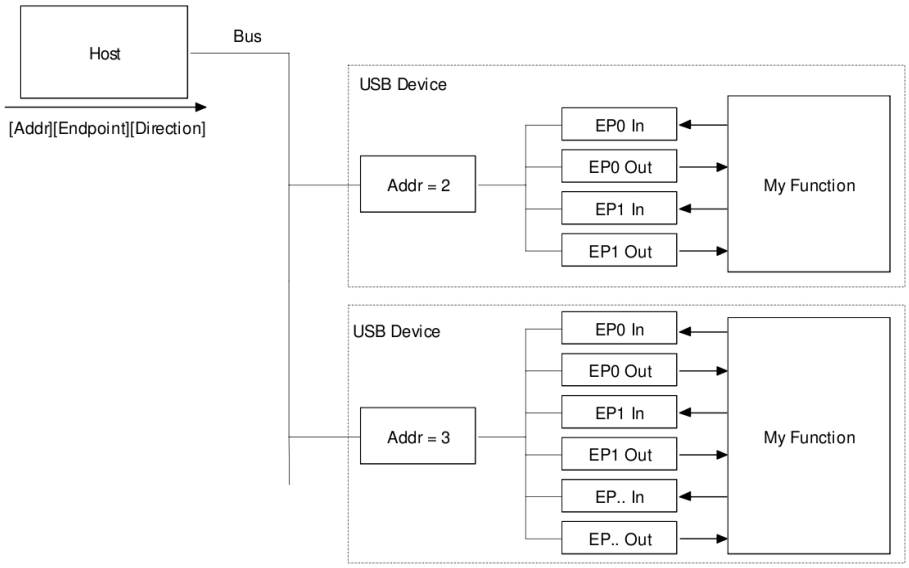


Figure 4.20: USB device architecture[15]

Bulk data transfers are used to transfer relatively large chunks of data that do not have a temporal pattern. Interrupt data transfers are used for time sensitive data that must be delivered reliably. Isochronous transfers use a pre-negotiated amount of USB bandwidth with a pre-negotiated delivery latency.

4.5.2 User Level USB Protocol

The FW for host communication is common to all variants of the PIC FW. This allowed a common set of communication tools to be utilized on the PC side of the USB bus. As previously stated, all of the low-level FW for managing the USB HW was provided by Microchip. A custom user-level protocol was built on top of the existing USB driver stack.

The user-level protocol is based on 64 byte packets. Each packet is equivalent to a single USB interrupt transfer. The first byte of each packet contains a command code and the second byte contains the length of the payload. Even though the packets are always 64 bytes long, the amount of meaningful data can vary. The content of the remaining 62 bytes is dependent on the command. The commands, along with their packet formats, are listed in Table 4.1. The host PC is always the initiator of a transaction. The PC sends a command

to the PIC and the PIC responds according to the command issued.

Table 4.1: User level USB protocol packet formats

		CMD	LEN	D[0]	...	...	...	...	...	D[61]
CAL_DATA	OUT	0x38	0x01	XX	XX	XX	XX	XX	XX	XX
	IN	0x38	0x40	D[0]	D[1]	D[2]	...	D[15]	XX	XX
RD_GYRO	OUT	0x39	0x01	XX	XX	XX	XX	XX	XX	XX
	IN	0x39	0x40	PKT	CNT	D[0]_H	D[0]_L	...	D[CNT-1]_H	D[CNT-1]_L
GET_IMAGE	OUT	0x40	0x01	XX	XX	XX	XX	XX	XX	XX
	IN	0x40	0x40	D[0]	...	...	...	...	...	D[61]
GYRO_CMD	OUT	0x41	0x01	CMD	DATA	XX	XX	XX	XX	XX
	IN	0x41	0x40	RES_H	RES_L	XX	XX	XX	XX	XX
ADIS_CFG	OUT	0x42	0x01	CMD	DATA	XX	XX	XX	XX	XX
	IN	0x42	0x40	VOLT_H	VOLT_L	TEMP_H	TEMP_L	XX	XX	XX
ADNS_DATA	OUT	0x43	0x01	CMD	DATA	XX	XX	XX	XX	XX
	IN	0x43	0x40	STATUS	DY	DX	SQUAL	PMAX	PMIN	PSUM
TRIGGER	OUT	0x44	0x01	STATE	XX	XX	XX	XX	XX	XX
	IN	0x44	0x40	STATE	XX	XX	XX	XX	XX	XX
XX: Don't Care ...: Sequential Enumeration										

The PIC's USB engine has a block of shared RAM that is used as a buffer for sending and receiving data. The data packet structures are implemented as 64 byte long arrays that exist in this shared memory. The arrays must not be accessed by the FW while the USB engine is operating on them. Because the host communication is not synchronized with the sensor communication, contention can occur between the USB engine and an ISR trying to write sensor data into the buffer. A double buffering mechanism is employed to avoid this. In this scheme, two 64 byte arrays are declared in the shared memory. The FW and the USB engine each access these buffers through a separate pointer. From the perspective of either, there appears to be only one buffer. In reality, the two pointers are never pointing to the same block of data, so the USB engine is working on one buffer while the FW is working on the other. When the USB engine completes a transfer, it swaps the contents of each pointer. This allows the system to tolerate variation in the host polling rate, as well as managing potential contention between the USB engine and the FW.

4.5.3 Device Driver

A custom device driver allowed Matlab to communicate with the test electronics over the USB interface. The device driver has three parts, a DLL and .sys file that provide an API for transferring data over USB, and a DLL that implements the high level protocol for configuring and transferring data from the test system. The low-level USB API was provided by Microchip and used without modification. The protocol level DLL was based

Table 4.2: Microchip Low-Level USB Communication API implemented in `mpusbapi.dll`.

DWORD MPUSBGetDLLVersion(void)	Returns the version # of the DLL
*DWORD MPUSBGetDeviceCount(CHAR *pVID_PID)*	Returns the number of devices with matching VID&PID a string containing the PID&VID value of the target device
HANDLE MPUSBOpen(	Returns the handle to the endpoint pipe with matching VID&PID
DWORD instance,	Instance number of the device to open
*CHAR *pVID_PID,*	A string containing the PID&VID value of the target device
*CHAR *pEP,*	A string of the target endpoint to open
DWORD dwDir,	Specifies the direction of the endpoint
DWORD dwReserved)	Future use
DWORD MPUSBRead(	Read data from the specified pipe and return status code
HANDLE handle,	Identifies the endpoint pipe to be read
*VOID *pData,*	Points to the buffer that receives the data read from the pipe
DWORD dwLen,	Specifies the number of bytes to be read from the pipe
*DWORD *pLength,*	Points to the number of bytes read
DWORD dwMilliseconds)	Specifies the time-out interval, in milliseconds
DWORD MPUSBWrite(	Write data to the specified pipe and return status code
HANDLE handle,	Identifies the endpoint pipe to be read
*VOID *pData,*	Points to the buffer that receives the data read from the pipe
DWORD dwLen,	Specifies the number of bytes to be read from the pipe
*DWORD *pLength,*	Points to the number of bytes read
DWORD dwMilliseconds)	Specifies the time-out interval, in milliseconds
DWORD MPUSBReadInt(	Read data from the specified pipe using the interrupt transfer mode and return status code
HANDLE handle,	Identifies the endpoint pipe to be read
*VOID *pData,*	Points to the buffer that receives the data read from the pipe
DWORD dwLen,	Specifies the number of bytes to be read from the pipe
*DWORD *pLength,*	Points to the number of bytes read
DWORD dwMilliseconds)	Specifies the time-out interval, in milliseconds
BOOL MPUSBClose(HANDLE handle)	Closes a given handle Identifies the endpoint pipe to be read

on Microchip code but was almost entirely custom. It was developed in Microsoft Visual Studio.

The Microchip USB API handles all of the low-level Windows system calls necessary to actually control the USB hardware. The mchpusb.sys file is registered with Windows using the supplied `mchpusb.inf` file and implements the kernel mode functions necessary to directly control the PC hardware. `Mchpusbapi.dll` implements the API functions that a user mode application can call to setup the hardware and transfer data. The member functions and their parameters are listed in Table 4.2. The file `Mchpusb.h` is included in the user application, in this case another DLL, to define the API functions that are imported when the DLL is loaded.

`Adis_usb_ctrl.dll` implements the user level control and transfer protocol that allows Matlab to communicate with the test system. An important feature of `Adis_usb_ctrl.dll`

Table 4.3: High-Level USB Communication API implemented in `adis_usb_ctrl.dll`.

VOID ADIS_Init_USB_Connection(VOID)	Initializes the driver and loads `mchpusbapi.dll`
VOID ADIS_Close_USB_Connection(VOID)	unloads `mchppusbapi.dll`
UNSIGNED INT ADIS_Generic_Xfer(	Transmits a generic 64 byte packet to the target and returns the response
*UNSIGNED CHAR *rx_buffer,*	Pointer to a pre-allocated buffer for the response packet
*UNSIGNED CHAR *tx_buffer,*	Pointer to buffer containing the packet to transmit
DWORD tx_len)	Number of bytes within the packet that contain useful data
UNSIGNED INT ADIS_Config(	Configures the target to begin acquiring data continuously
UNSIGNED CHAR CMD,	Device register address that will be polled during the test
UNSIGNED CHAR DATA,	Device register address that will be polled during the test
UNSIGNED CHAR cnt_MSB,	Unused
UNSIGNED CHAR cnt_LSB)	Unused
VOID ADIS_Test(	Executes a test run. Function will stream data from the target, format it, cast it to float, and return it in the buffer supplied by the caller
*FLOAT *data_buffer,*	Pointer to a pre-allocated buffer to hold the result data
DWORD buffer_length,	Number of samples to aquire
DWORD sps,	Specifies the sampling rate. Used for logging purposes only
FLOAT scale_factor,	Specifies the conversion factor between the integer sample value and the FP rate
INT offset)	Unused

is that the functions it exports adhere to the C calling convention, which is necessary for Matlab to load it. The API is illustrated in Table 4.3. *ADIS_Init_USB_Connection()* must be called before any communication can occur. This function loads `Mchpusbapi.dll` so that the low-level communications routines are available. The *ADIS_Generic_Xfer()* function is used to send and receive arbitrary USB packets. This function allows the caller to build a 64 byte packet, send it to the target, and receive the raw response. In this way, the high level protocol can be implemented at the Matlab script level. The advantage of this is that the high-level protocol can be modified without having to also modify and recompile `adis_usb_ctrl.dll`.

Implementation of the user protocol at the script level worked fine for most tasks. However, during real-time data acquisition, Matlab was not fast enough to parse the packets without dropping some of them. *ADIS_Test()* is used to receive data from the target during an actual test run. Matlab allocates a block of memory large enough to accommodate the entire test record and passes *ADIS_Test()* a pointer to it. *ADIS_Test()* then manages the transfers, formats the individual data samples, converts them to real-world floating point values, and returns control the Matlab script. The advantage of this is that the data formatting and type casting operations occur at the speed of native C++ and the overhead of

calling into the DLL for every packet is eliminated.

4.5.4 Matlab Scripts

A library of Matlab scripts was developed to aide in the test process. The scripts were developed using a layered approach. At the lowest level, basic scripts handled specific tasks like sending a generic packet to the data acquisition system or initializing the USB connection. Scripts at the middle level aggregated multiple, low-level scripts to perform more complex tasks like capturing an image from the ADNS-2610 tracking sensor or triggering the rotation table. The highest level Matlab scripts were used to automate repetitive tasks like performing the same test many times in a row, storing all of the data, and analyzing it.

Chapter 5

Gyroscope Experiments

The purpose of these tests was to characterize the MG1101a and ADIS16255 MEMS angular rate gyroscopes. The tests are broken down into two types: static tests and dynamic tests. The static tests were designed to evaluate the gyroscopes output noise while the sensor is at rest as well as the bias present in the output signal. The dynamic tests evaluate the same parameters in the presence of angular motion. The static and dynamic test data were then compared to test for the presence of motion dependent effects.

5.1 Procedure

5.1.1 Static Performance Tests

The sensor under test is mounted to the rotation table, described previously in Section 4.3, and held stationary. A series of 100 tests were performed and each test had a duration of 60s. At the beginning of each trial the sensor output was recorded for 5s. The mean of that data was calculated using (5.1), where N is the number of samples and ω_i, $i \in [1, N]$ are the individual angular rate samples. The mean is then used as an estimate of the bias. The bias, b , is then subtracted from each subsequent sample, ω_{ri}, of the 60s test run, as given in (5.2). The bias corrected angular velocity data, ω_{ci}, is then integrated using (5.3)

to calculate angular position.

$$b = \frac{1}{N}\sum_{i=1}^{N}\omega_i \tag{5.1}$$

$$\omega_{ci} = \omega_{ri} - b \tag{5.2}$$

5.1.2 Dynamic Performance Tests

The sensor under test is mounted to the rotation table. A series of 100 tests were performed and each test had a duration of 21 s, where the first 5 s are used to determine the bias and the subsequent 16 s are recorded as the actual data. Before any testing was performed, the motion controller was initialized with the appropriate motion program. At the beginning of each trial the sensor output was recorded for 5 s. The mean of that data was calculated using (5.1). The mean value, b, was then used to correct subsequent angular rate samples, ω_{ri}, yielding corrected angular rate values, ω_{ci}, as shown in (5.2).

The angular displacement after the Nth sample, θ_N, was calculated from the corrected angular rate samples, ω_{ci}, where f_s is the sampling rate, using (5.3).

$$\theta_N = \frac{1}{f_s}\sum_{i=1}^{N}\omega_{ci} \tag{5.3}$$

Equations 5.2 and 5.3 can also be combined into a more compact form as in 5.4.

$$\theta_N = \frac{1}{f_s}\sum_{i=1}^{N}(\omega_{ri} - b) \tag{5.4}$$

5.2 Results and Discussion

5.2.1 Static Test Results

Angular Velocity Measurements

Figures 5.1 and 5.2 summarize the angular velocity data obtained by recording the gyroscope output while the gyroscope was stationary. 100 trials were conducted, each of which had a duration of 60 s. Since the gyroscopes were stationary, noise was the only component present in the output signal.

RMS noise is a measure of the magnitude of the AC noise in the angular velocity. The RMS noise for each trial, indicated by the "+" in Figures 5.1 and 5.2, was calculated by summing the squared difference between each corrected angular velocity sample, ω_{ci}, and the mean of all the corrected angular velocity samples, $\overline{\omega_c}$, dividing by the number of samples, N, and then taking the square root, as shown in (5.5).

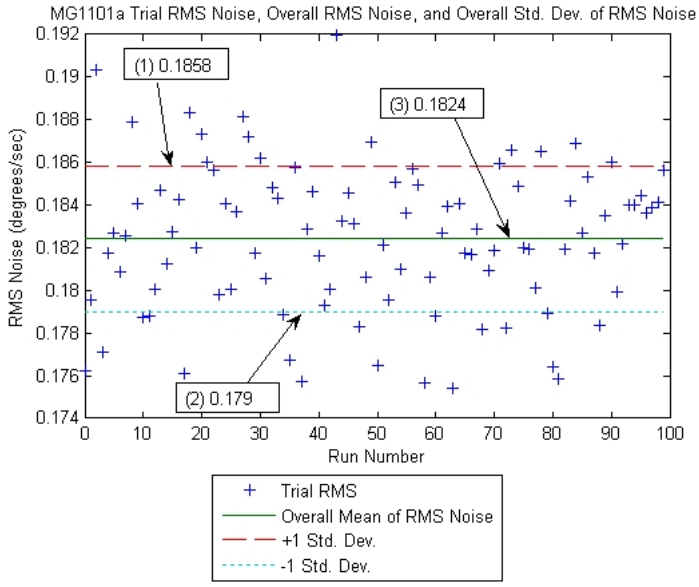


Figure 5.1: MG1101a static test velocity data summary

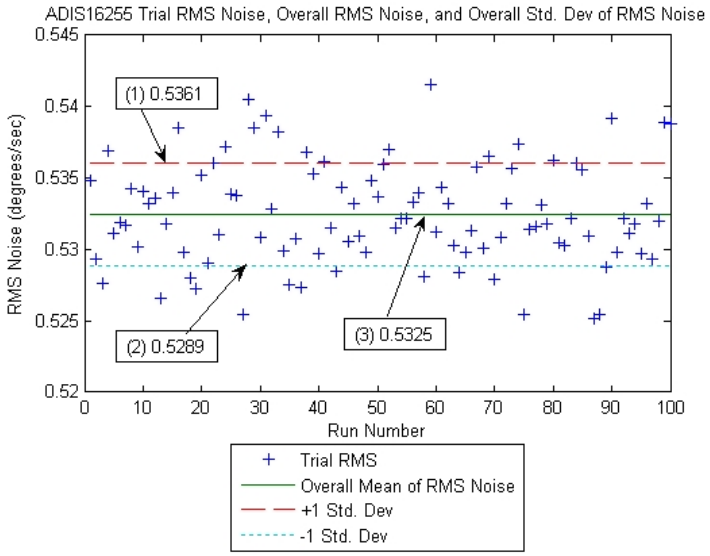


Figure 5.2: ADIS16255 static test velocity data summary

$$RMS\ Noise = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\omega_{ci} - \overline{\omega_c})^2} \tag{5.5}$$

The standard deviation, σ, is calculated using the same equation form as was used to compute the RMS noise. However, the standard deviation is a more general term and is defined separately in (5.6).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X})^2} \tag{5.6}$$

The ω_{ci} and $\overline{\omega_c}$ terms in (5.5) are equivalent to the X_i and $\overline{X}$ terms, respectively, in (5.6). The overall mean of the RMS noise, $\overline{\sigma}$, is the average RMS noise for all trials. It was calculated using equation 5.7, where N is the number of trials and RMS_i is the RMS noise value for an individual trial.

$$\overline{\sigma} = \frac{1}{N}\sum_{i=1}^{N} RMS_i \tag{5.7}$$

In Figures 5.1 and 5.2, the lines pointed out by (1), (2), and (3) represent the $+1\sigma$, -1σ,

Table 5.1: Comparison of static velocity test results

	MG1101a	ADIS16255
RMS Noise	0.1824	0.5325
Mean (°/s)	0.0013	-0.0041
Range of RMS Noise (°/s)	0.0165	0.0164

and mean value of each experiment, respectively. The MG1101a exhibited better noise performance than the ADIS16255 did. Neither sensor exhibited any significant variation in RMS noise from run to run. The velocity noise of the ADIS16255 could be reduced by using the built-in filtering function. However, the MG1101a does not have that feature, so only a comparison of the basic features was performed.

The RMS noise of the MG1101a is specified as 0.18°/s nominally. This corresponds exactly to the experimental results. The ADIS16255 specifies a typical RMS noise value of 0.48°/s which is in line with the observed value of 0.5325°/s.

Angular Position

The static angular position data illustrated in Figure 5.3 for the MG1101a and Figure 5.4 for the ADIS16255 was derived from the static angular velocity data using (5.3).

Analysis of the position data provides insight into the accuracy that will be achievable in the prototype tracking system. The noise sources present in the postition signal are known collectively as drift rate. They are represented graphically in Figure 5.5.

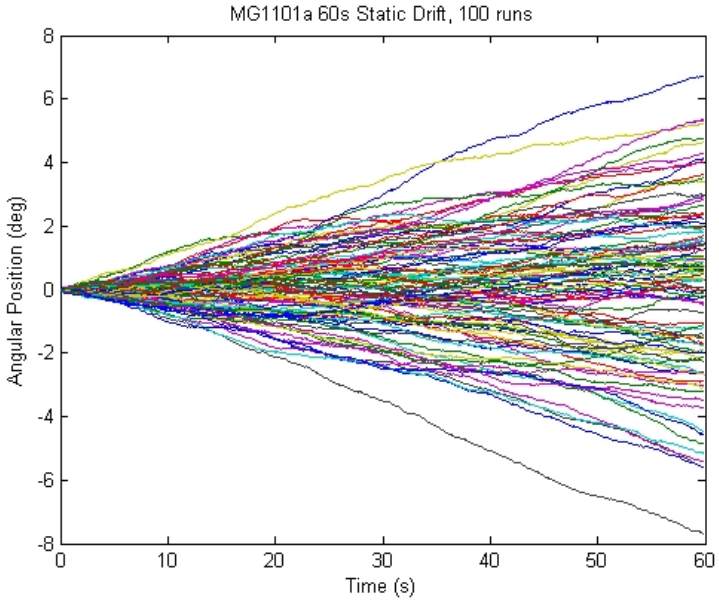


Figure 5.3: MG1101a static test position data

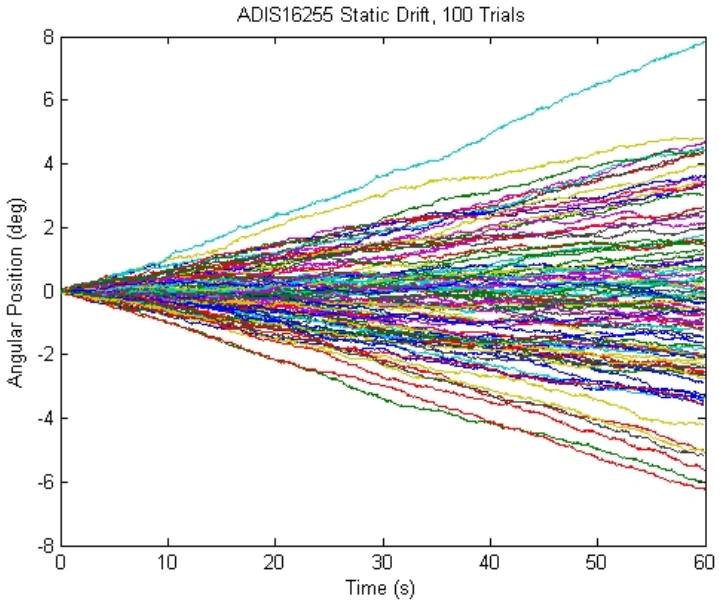


Figure 5.4: ADIS16255 static test position data

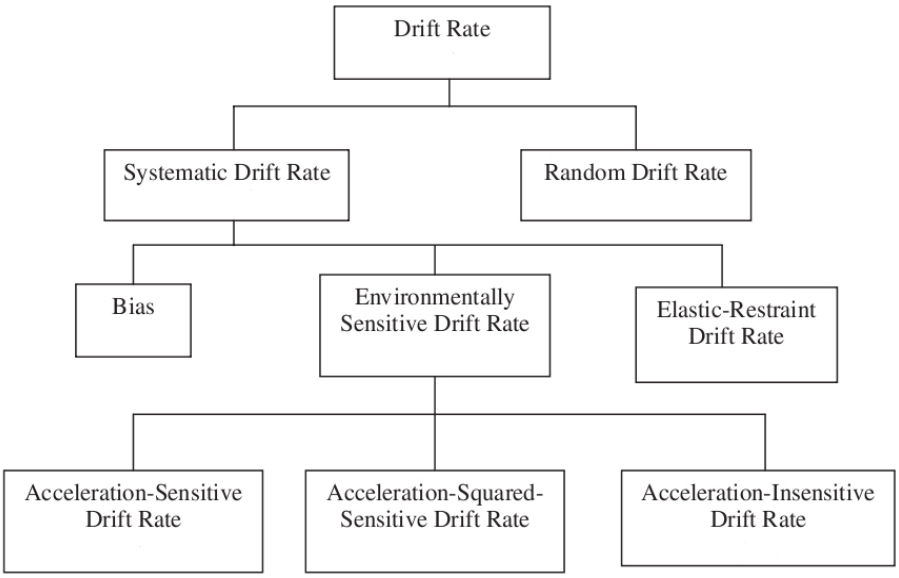


Figure 5.5: Components of gyroscope drift rate [16]

Drift rate is the component of gyroscope output that is functionally independent of input rotation. It is expressed as an angular rate in °/s [16]. Drift rate has many components. Of the components listed, we are primarily interested in bias and random drift. Elastic restraint drift only occurs in mechanical gyroscopes. Other environmental effects, such as supply voltage and temperature, were held constant during testing and are therefore ignored. Acceleration-sensitive drift is a component of the overall drift rate that is correlated with linear acceleration. MEMS gyroscopes are only sensitive to linear acceleration in plane with the sensitive axis. In our testing the only linear acceleration present was gravity and it was oriented perpendicular to the sensitive axis.

Equation (5.8) models overall drift rate at sample i (d_i) as the sum of the systematic drift rate at sample i (d_{si}) and the random drift rate at sample i (d_{ri}), for the static test series. Based on the assumptions made in the previous paragraph, systematic drift rate is equivalent to bias for these tests. This allows (5.8) to be restated with the bias at sample i (b_i) substituted for systematic drift rate, as in (5.9).

$$d_i = d_{si} + d_{ri} \tag{5.8}$$

$$d_i = b_i + d_{ri} \tag{5.9}$$

Bias is the average, over a specified period of time, of gyroscope output that has no correlation with input rotation or acceleration. It is expressed here in °/s. Bias instability is the random variation in bias over a specified period of time and it is also expressed in °/s [16]. The bias (b_i) has both a systematic (b_s) and a random component (b_{ri}). The systematic component is sensitive to supply voltage and temperature. Because these two factors were held constant during testing, the systematic bias is represented as a constant in the bias model presented in (5.10) below.

$$b_i = b_s + b_{ri} \tag{5.10}$$

Bias and bias instability are both important factors influencing the accuracy of the overall system.

Random drift rate is the random, time-varying component of the overall drift rate. When integrated, random drift results in angular random walk [16], which is the other major source of inaccuracy in this system. Random walks are zero-mean Gaussian stochastic processes with a standard deviation that grows as the square root of time. The 1σ value of the angular random walk is typically used to quantify the random drift rate of a sensor in $^\circ/\sqrt{hr}$. If the bias estimate is completely correct and the bias does not change over the course of the trial, the mean of the angular velocity should be completely attributable to random drift. In reality, however, the bias does change over the course of each trial. This makes it difficult to quantify the magnitude of the bias instability and drift rate as two separate error terms. Going by the strict definition of bias, it will be presented here as the mean of the gyroscope's output over the entire test run, calculated using (5.1) where N is the sample rate in samples per second multiplied by the test duration (60 s). The variation in bias between trials can then be represented as the standard deviation of all the bias values calculated for the individual trials. The standard deviation is calculated using (5.6), again with N equal to the total number of samples in the trial. The resulting values are listed in

Table 5.2: Bias and angular random walk statistics

	MG1101a	ADIS16255
Bias Instability (°/s, 1σ)	0.0458	0.0446
Std. Dev. of Angular Random Walk (°/s)	0.0037	0.0024

row one of Table 5.2.

As stated previously, quantifying the random drift rate by direct inspection of the gyroscope's output is difficult because there is no clear way to distinguish between random variations in bias and the overall drift. Angular random walk provides an alternative. The first step in quantifying angular random walk is subtracting the bias from the gyroscope output on a sample by sample basis. This is done using (5.2), where the bias, b, now represents the mean of the entire time series. The time series is then integrated using (5.3). The resulting data contains only the position variations that are the result of random drift in the gyroscope's output.

Angular random walk is a stochastic process. Therefore, its standard deviation can be utilized as a measure. By calculating the standard deviation of each position series (derived in the paragraph above) and then taking the mean over all 100 trials, a statistical representation of the random variation in position that is due only to random drift in the gyroscope's output can be derived. Finally, this result is expressed as a rate by dividing by the test duration. The resulting values are listed in row two of Table 5.2.

Ultimately, the overall accuracy of the system is set by the aggregate of all the error sources. A simple way of quantifying this is to look at the std. dev. of the static position data. The standard deviation of the entire data set at time i is given by (5.6), where N is the number of individual trials (100). For each time sample time i, the standard deviation encompasses approximately 68% of the data. Put another way, if we repeat this test, there is a 68% chance that the estimated position will fall within the $\pm 1\sigma$ value at sample time i. It is also informative to look at the maximum and minimum values of the data as absolute bounds on the error. The maximum, minimum, mean and standard deviation are represented graphically in Figure 5.6 for the MG1101a and Figure 5.7 for the ADIS16255.

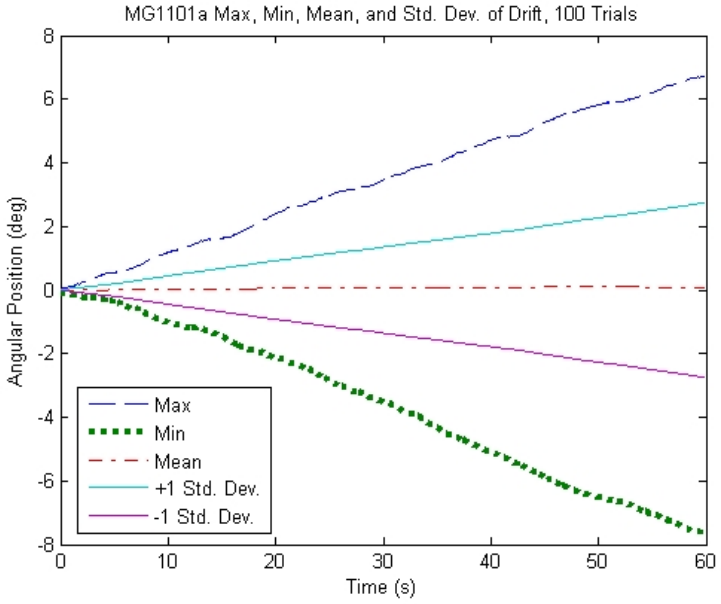


Figure 5.6: MG1101a static test position data summary

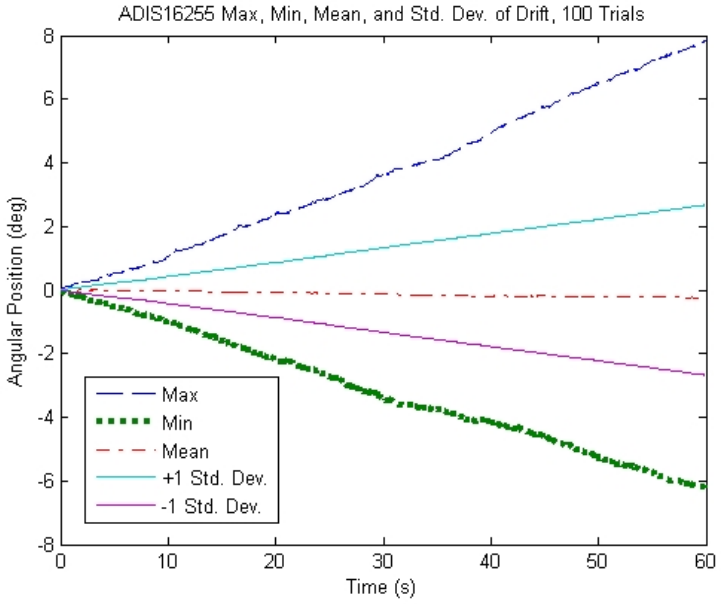


Figure 5.7: ADIS16255 static test position data summary

Table 5.3: Position data summary statistics, 100 trials, 60s per trail

	MG1101a	ADIS16255
Maximum Ending Position (°)	6.7468	7.8600
Maximum Error Growth Rate (°/s)	0.1146	0.1330
Minimum Ending Position (°)	-7.7074	-6.2486
Minimum Error Growth Rate (°/s)	-0.1356	-0.1043
Std. Dev. of Ending Positions (°, 1σ, 68% Confidence)	2.7506	2.6738
Std. Dev. of Ending Positions (°, 2σ, 95% Confidence)	5.5012	5.3477
Overall Drift Rate (°/s, 1σ, 68% Confidence)	0.0454	0.0448
Overall Drift Rate (°/s, 2σ, 95% Confidence)	0.0909	0.0895
Mean of Ending Positions	0.0800	-0.2470

For the metrics identified above, it is also useful to convert them to rates such that the values of the maximum, minimum, and standard deviation can be estimated for any arbitrary test duration. This is done by fitting a 1st order polynomial to the data of the form:

$$p(x) = p_1 x + p_0 \tag{5.11}$$

The slope of that line, p_1, is the rate of growth corresponding to that particular parameter. The values of the maximum, minimum, std. dev., and their corresponding growth rates are summarized in Table 5.3 for both the ADIS16255 and MG1101a. Overall conclusions as to which gyroscope is more suitable for this application will be presented at the end of this chapter, after the results of the dynamic tests are presented and discussed.

5.2.2 Dynamic Test Results

Dynamic tests were performed at ±1°/s, ±3°/s, ±5°/s, and ±10°/s. Each trial had a duration of 16 s and 100 trials were performed at each angular velocity. Each test began with a 5 s initial bias calibration. Then the Aerotech ADRS-200 rotation table remained idle for 1 s, moved in the negative direction at the specified rate for 1 s, remained idle for 1 s, moved in the positive direction at the specified rate for 2 s, remained idle for 1 s, moved in the negative direction at the specified rate for 2 s, remained idle for 1 s, moved in the positive direction at the specified rate for 2 s, remained idle for 1 s, moved in the negative direction at the specified rate for 1 s, and then idled for the remainder of the test.

The same profile was used for all of the tests, the only thing that changed was the angular velocity. A rate-ramp test was also performed. The rate-ramp test began with a 5 s initial bias calibration. Then the table remained idle for 1 s, moved in the negative direction at 1°/s for 2 s, remained idle for 1 s, moved in the negative direction at 2°/s for 2 s, remained idle for 1 s, moved in the negative direction at 3°/s for 2 s, remained idle for 1 s, moved in the negative direction at 4°/s for 2 s, remained idle for 1 s, moved in the positive direction at 10°/s for 2 s, and, finally, remained idle for 2 s.

Test Results for $\omega = 1°/s$

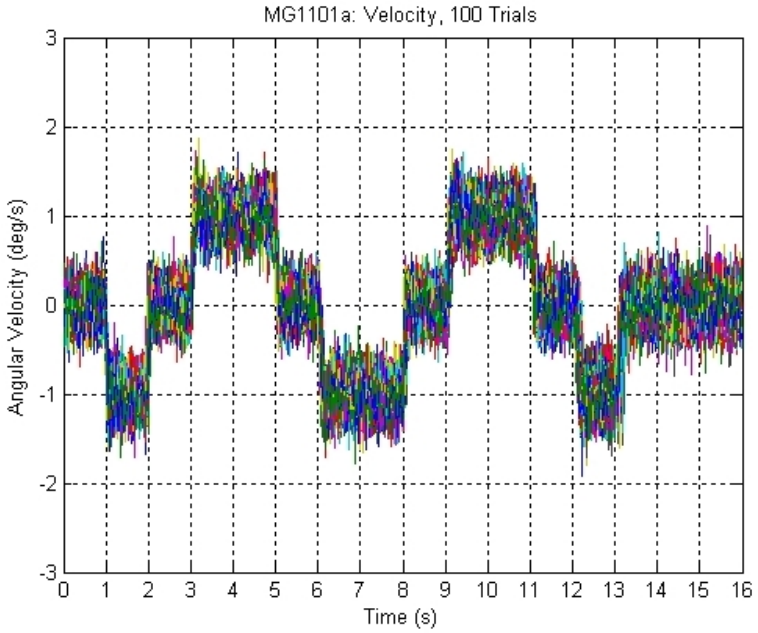


Figure 5.8: MG1101a Velocity, 1°/s

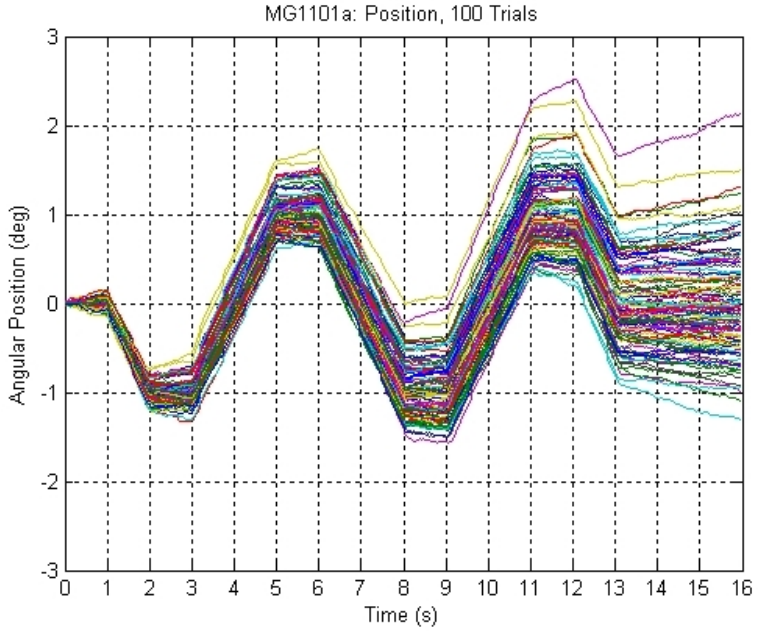


Figure 5.9: MG1101a Position, 1°/s

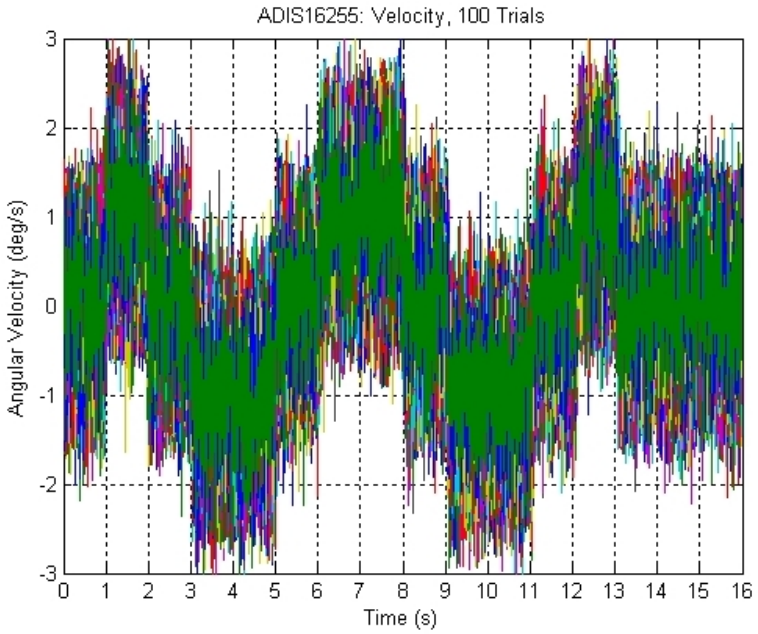


Figure 5.10: ADIS16255 Velocity, 1°/s

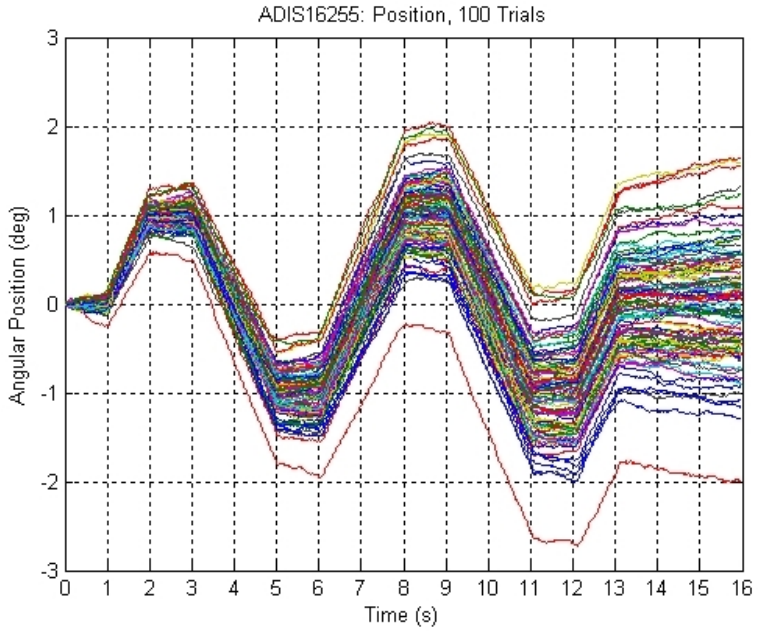


Figure 5.11: ADIS16255 Position, 1°/s

Figure 5.8 and Figure 5.10 illustrate the velocity profile used in all of the dynamic tests. These particular tests were performed with an angular rate of ±1°/s. Although Figure

5.10 appears much noisier than Figure 5.8, the positive, negative, and idle periods of the motion profile are clearly evident. Figure 5.9 and Figure 5.11 illustrate the angular position estimates corresponding to the motion profile. The position estimates are calculated from the rate output of the gyroscopes using (5.3).

Test Results for $\omega = 3°/s$

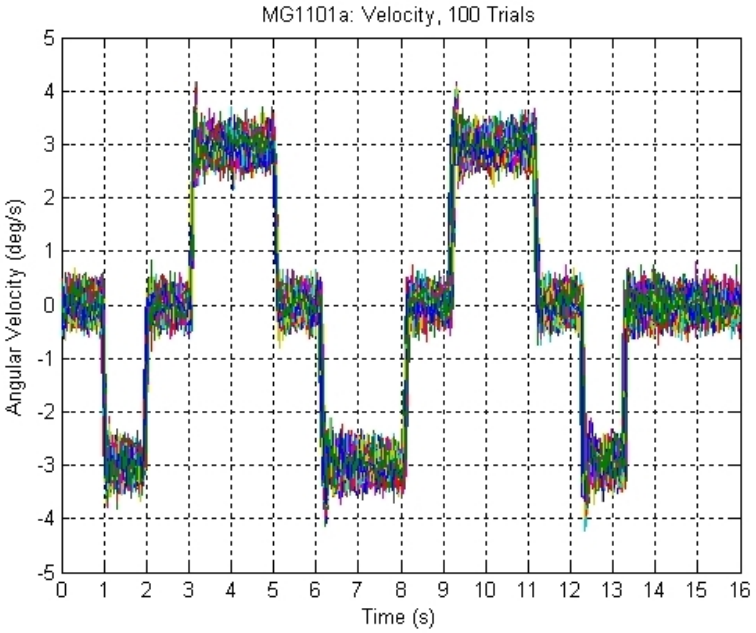


Figure 5.12: MG1101a Velocity, 3°/s

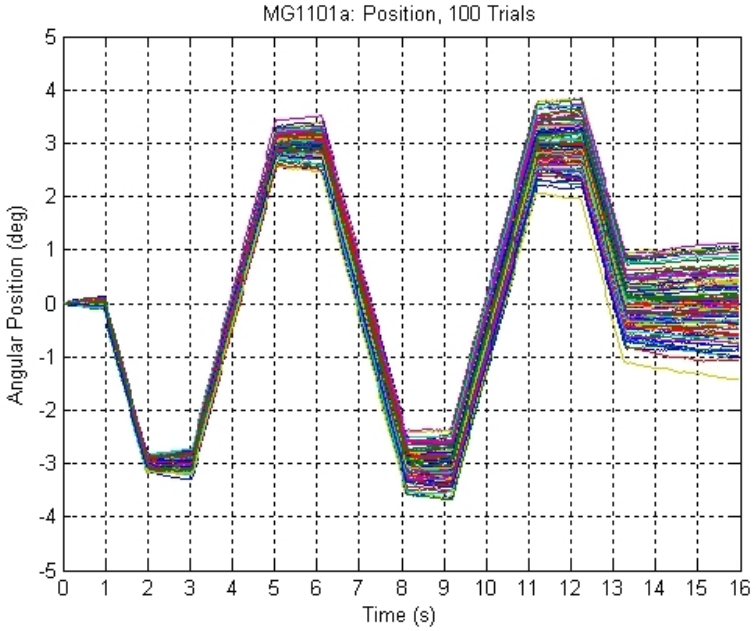


Figure 5.13: MG1101a Position, 3°/s

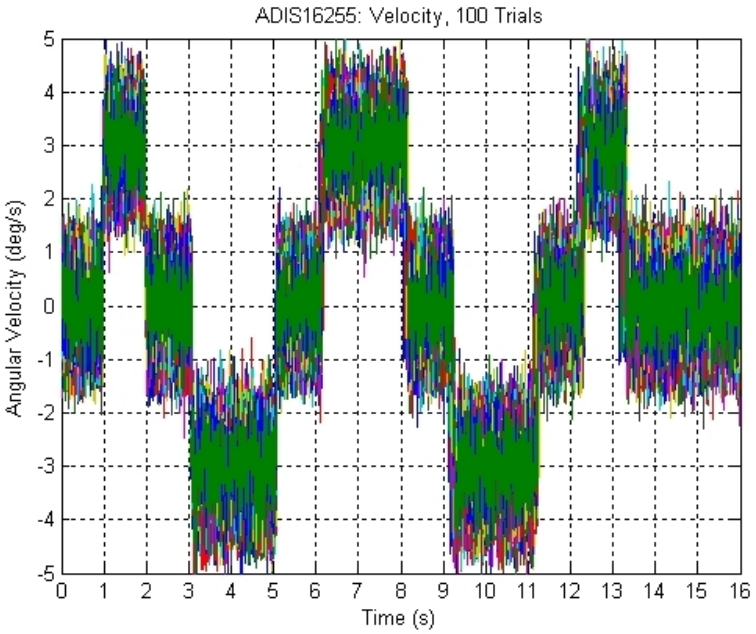


Figure 5.14: ADIS16255 Velocity, 3°/s

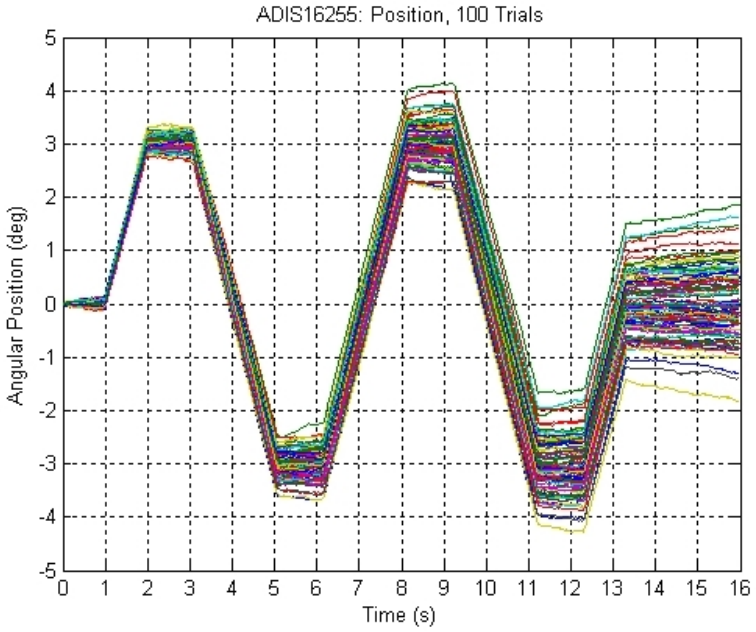


Figure 5.15: ADIS16255 Position, 3°/s

Test Results for $\omega = 5°/s$

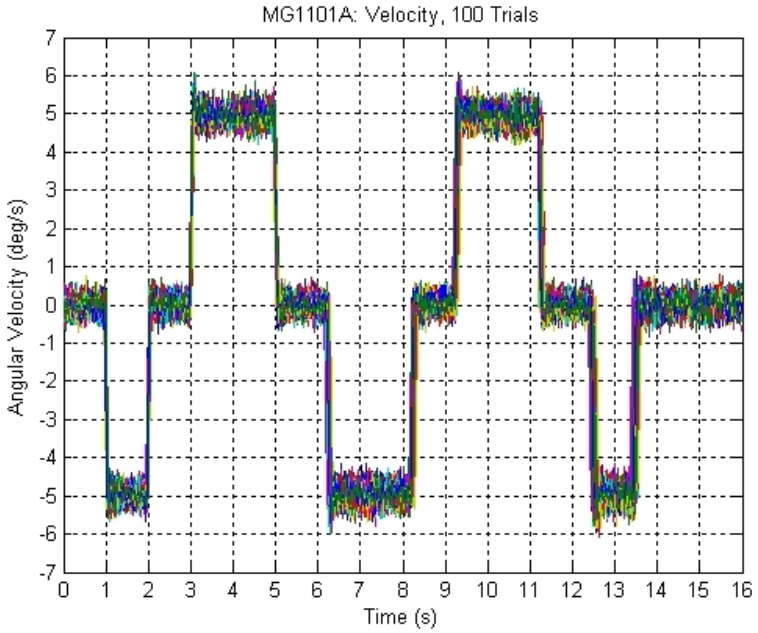


Figure 5.16: MG1101a Velocity, 5°/s

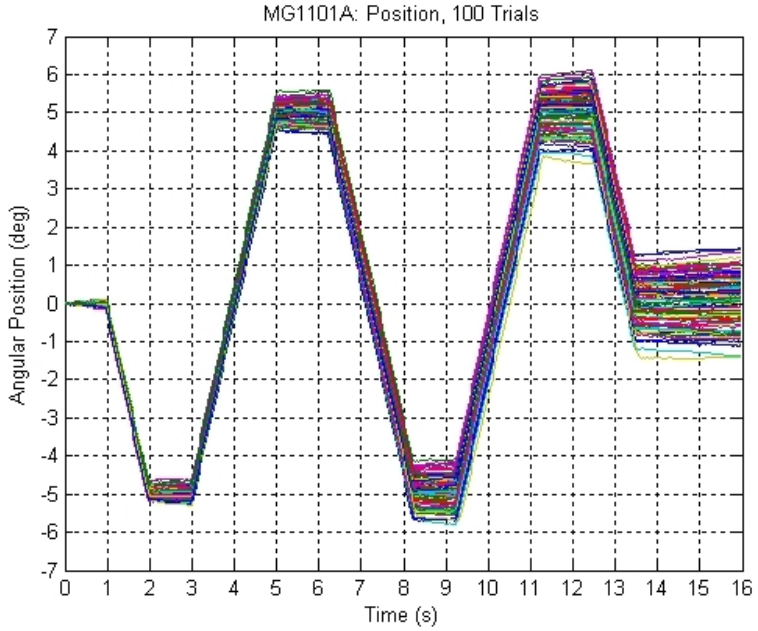


Figure 5.17: MG1101a Position, 5°/s

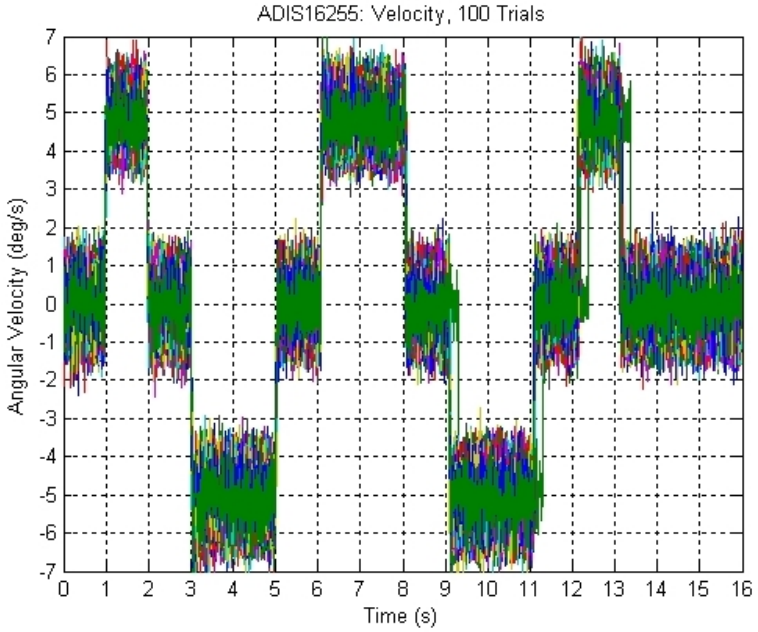


Figure 5.18: ADIS16255 Velocity, 5°/s

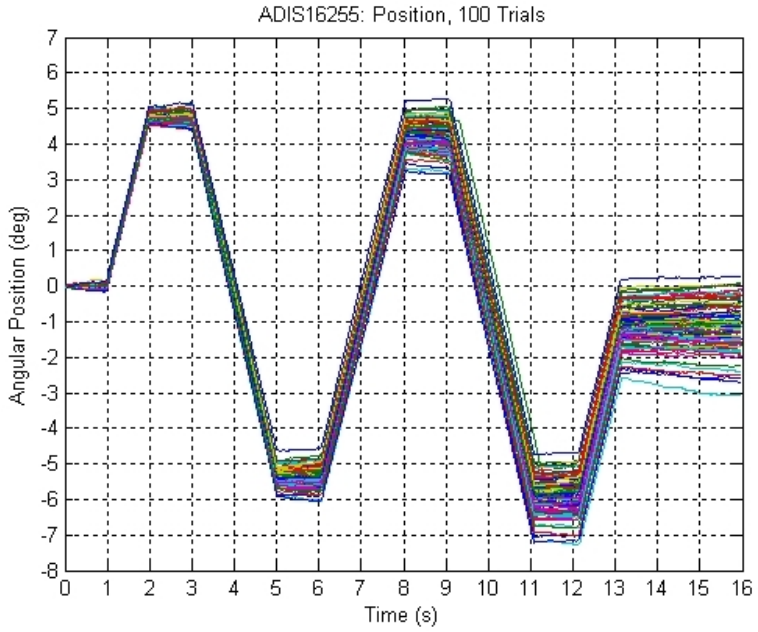


Figure 5.19: ADIS16255 Position, 5°/s

Test Results for $\omega = 10°/s$

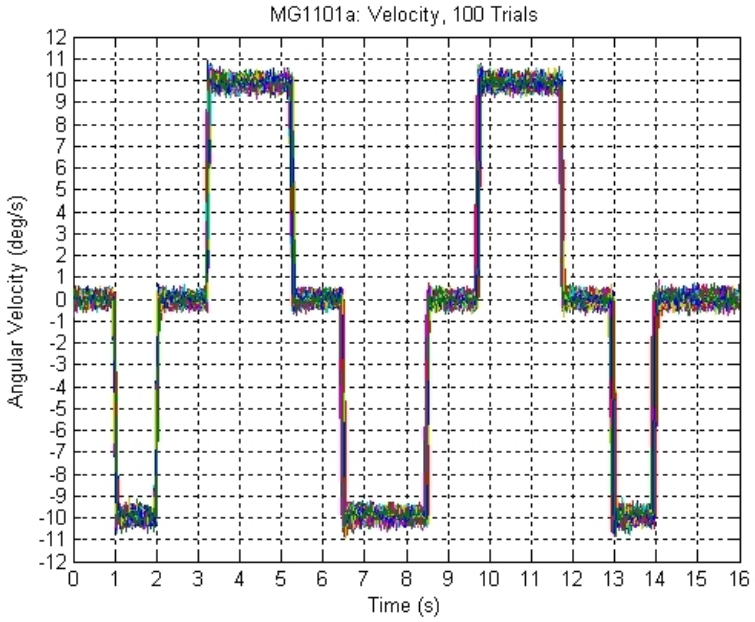


Figure 5.20: MG1101a Velocity, 10°/s

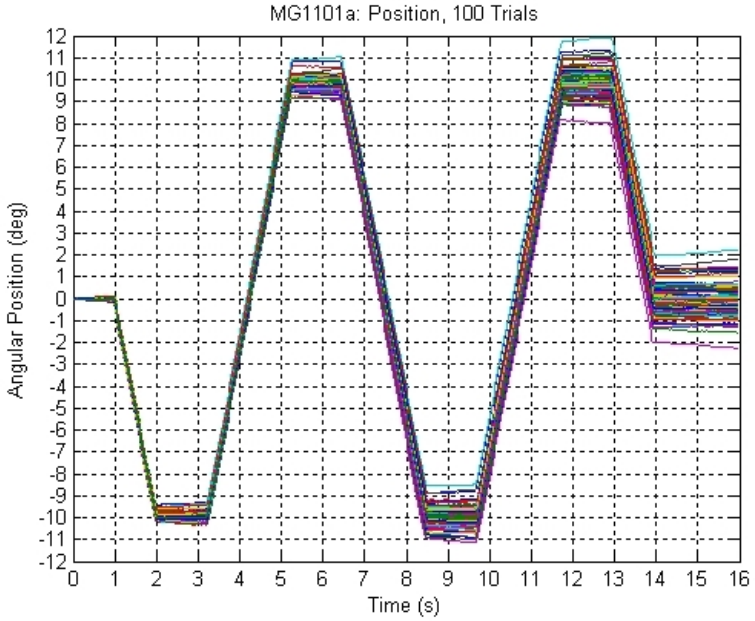


Figure 5.21: MG1101a Position, 10°/s

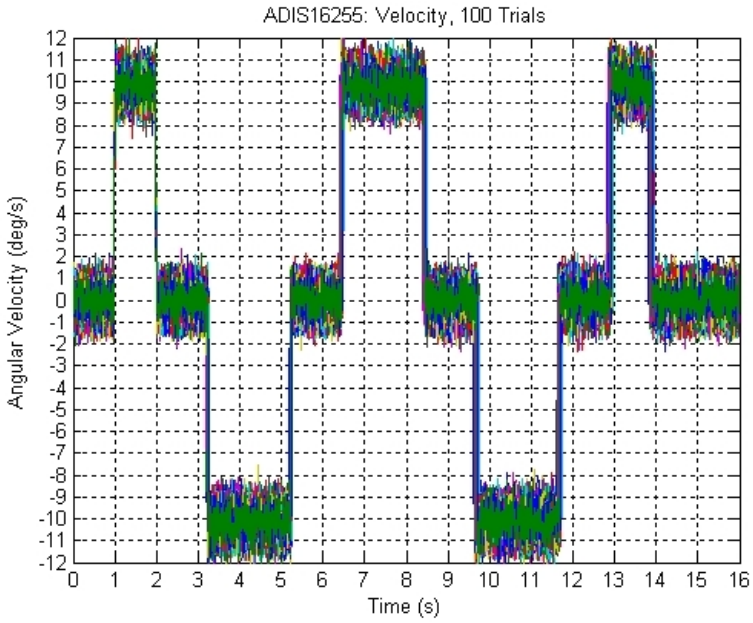


Figure 5.22: ADIS16255 Velocity, 10°/s

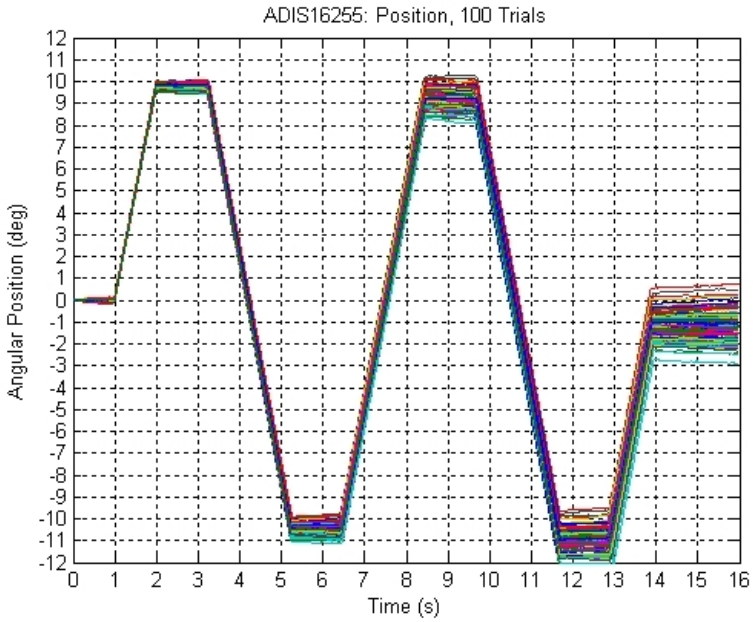


Figure 5.23: ADIS16255 Position, 10°/s

Test Results for the Rate Ramp Tests

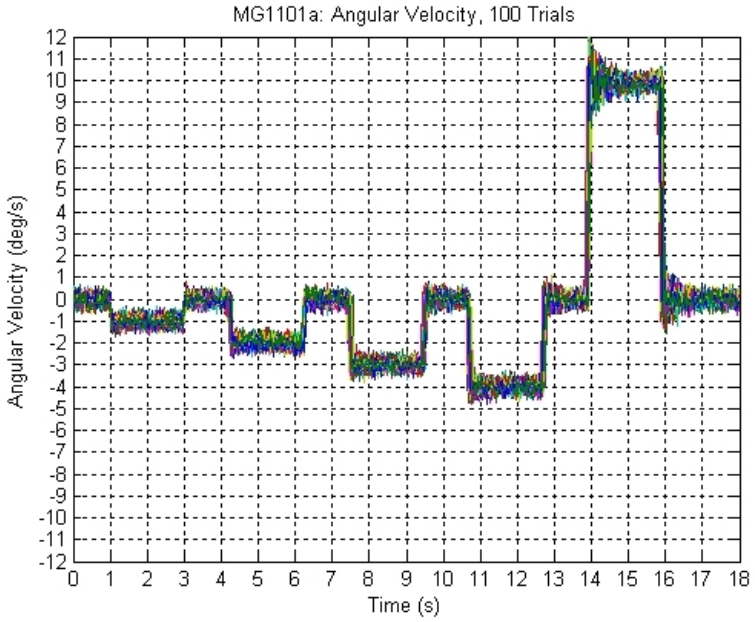


Figure 5.24: MG1101a Velocity, RateRamp

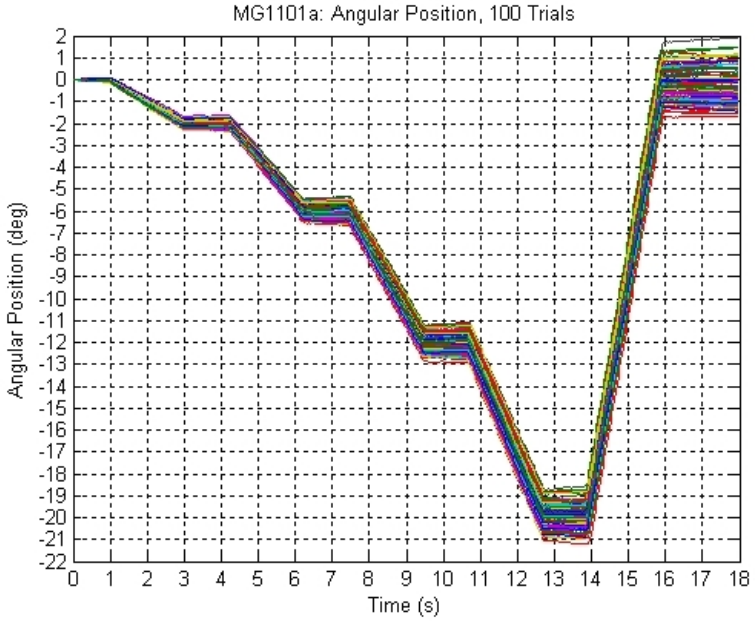


Figure 5.25: MG1101a Position, RateRamp

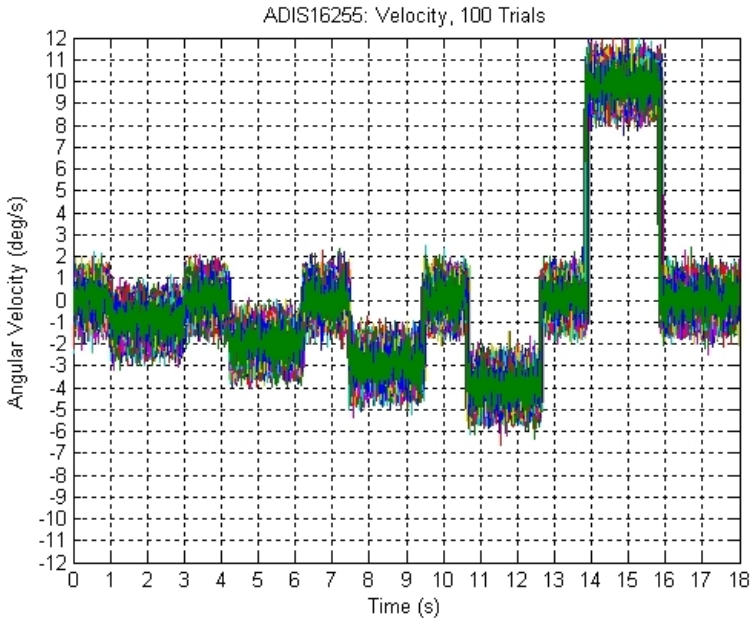


Figure 5.26: ADIS16255 Velocity, RateRamp

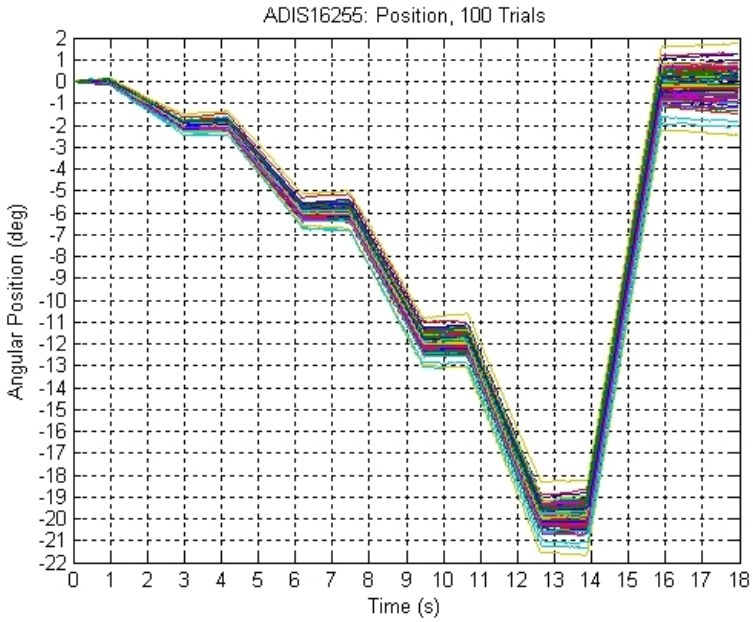


Figure 5.27: ADIS16255 Position, RateRamp

Table 5.4: MG1101a dynamic test velocity data summary

Rate (°/s)	Std Dev Positive	Std Dev Negative	Std Dev Static	Range Positive	Range Negative	Range Static	Mean Positive	Mean Negative	Mean Static
1	0.1892	0.1943	0.2007	0.3102	0.2536	0.3041	0.9944	-0.9913	0.0003
3	0.1959	0.2067	0.2027	0.2327	0.2168	0.2686	2.9755	-2.9766	0.0022
5	0.2141	0.2266	0.2072	0.2406	0.2466	0.2740	4.9684	-4.9461	0.0035
10	0.1954	0.2029	0.2022	0.3161	0.3741	0.3044	9.9008	-9.9043	-0.0016

Table 5.5: MG1101a dynamic test ending position data summary

Ending Positions			
Rate (°/s)	Std Dev	Mean	Range
1	0.6046	0.0012	3.4256
3	0.5490	-0.0183	2.5185
5	0.6797	0.0956	2.8476
10	0.7255	-0.0872	4.5026

Discussion

The dynamic tests were performed for two main purposes. First, the angular velocity data was analyzed to quantify the sensor noise in the presence of real motion. Second, the angular position data was analyzed to quantify the effects of bias and drift on the accuracy of the angular position estimate. These results were also compared to the results of the static tests as a means of evaluating the correlation between angular velocity and the noise and position error terms.

Table 5.4 and Table 5.6 summarize the results of the dynamic angular velocity tests. As illustrated in Figure 5.8, Figure 5.10, Figure 5.12, Figure 5.14, Figure 5.16, Figure 5.18, Figure 5.20, and Figure 5.22, these tests consisted of a series of forward and reverse motions at a fixed rate of $\pm 1°/s$, $\pm 3°/s$, $\pm 5°/s$, and $\pm 10°/s$ respectively. Each trial had a duration of 16 s and 100 trials were performed in each test series. For each test, the data was separated into positive, negative, and static sections. For each section the standard

Table 5.6: ADIS16255 dynamic test velocity data summary

Rate (°/s)	Std Dev Positive	Std Dev Negative	Std Dev Static	Range Positive	Range Negative	Range Static	Mean Positive	Mean Negative	Mean Static
1	0.5250	0.5294	0.5296	0.2649	0.2728	0.2242	1.0089	-1.0101	0.0053
3	0.5232	0.5334	0.5333	0.2176	0.2885	0.2813	3.0197	-3.0269	0.0049
5	0.4825	0.5289	0.5302	0.2574	0.2850	0.2808	4.8016	-5.0513	-0.0036
10	0.4398	0.4336	0.4249	0.1287	0.1598	0.1874	9.8403	-10.1297	0.0024

Table 5.7: ADIS16255 dynamic test ending position data summary

Ending Positions			
Rate (°/s)	Std Dev	Mean	Range
1	0.6494	0.0153	3.6538
3	0.6766	0.0254	3.7036
5	0.6476	-1.0904	3.3911
10	0.4346	-1.2419	1.2626

deviation, range, and mean was calculated across all 100 trials. Here, standard deviation is equivalent to RMS noise. The range is the amplitude of the noise and the mean is the actual angular rate that the sensor recorded. The results of these calculations are summarized for the MG1101a and ADIS16255 in Table 5.4 and Table 5.6 respectively.

The results of the static velocity testing indicated that the MG1101a has an average RMS noise of about $.18°/s$ (Table 5.1). The ADIS16255 has an average RMS noise of about $.53°/s$ (Table 5.1). The RMS noise values derived from the dynamic velocity data agree well with these figures. Direction and angular rate do not have an effect on the noise. It is important to note that, unlike the static tests, the rotation table could be contributing some of the measured noise. However, because we have no way to record that noise independent of the sensors, it must be assumed that all of the noise comes from the sensor itself. The fact that the measured noise still agrees with the static noise values indicates that the contribution of rotation table itself is negligible.

Table 5.5 and Table 5.7 summarize the results of the dynamic angular position tests. The end position data is derived from the velocity data by summation using (5.3).The calculated positions are illustrated in Figure 5.9, Figure 5.11, Figure 5.13, Figure 5.15, Figure 5.17, Figure 5.19, Figure 5.21, and Figure 5.23. For each angular velocity tested, the standard deviation, range, and mean of the derived ending position were calculated. These metrics define the distribution of outcomes and thus the accuracy of the sensor can be defined. Using the drift rate from Table 5.3, it is possible to calculate the expected drift for the dynamic tests. The value of 0.045°/s is used as an approximation for the drift rate derived for both sensors, since the actual results were very close. By multiplying the drift rate by the duration of a test run, the standard deviation of the position error can be predicted for any test. Comparing the expected value, derived from the static test data, to

Table 5.8: Rate ramp velocity data summary

	Std. Dev. -1°/s	Std. Dev. -2°/s	Std. Dev. -3°/s	Std. Dev. -4°/s	Std. Dev. +10°/s	Std. Dev. Static
MG1101a	0.1964	0.1943	0.1919	0.1923	0.1972	0.1920
ADIS16255	0.5277	0.5280	0.5317	0.5343	0.5334	0.5241
	Mean -1°/s	Mean -2°/s	Mean -3°/s	Mean -4°/s	Mean +10°/s	Mean Static
MG1101a	-1.0034	-1.9976	-2.9985	-4.0002	9.8478	-0.0107
ADIS16255	-0.9988	-1.9832	-2.9789	-3.9686	9.8943	0.0024
	Range -1°/s	Range -2°/s	Range -3°/s	Range -4°/s	Range +10°/s	Range Static
MG1101a	0.8900	0.8756	0.8550	0.8441	0.8778	0.8647
ADIS16255	3.1089	3.2161	3.2198	3.2637	3.2549	3.1663

Table 5.9: Rate ramp position data summary

	Std Dev	Mean	Range
MG1101a	0.7433	-0.0598	3.6608
ADIS16255	0.7196	-0.1477	4.1752

the experimental result, derived from the dynamic test data and summarized in Table 5.5 and Table 5.7, we can verify that the drift rate is independent of the motion of the sensor. The single speed dynamic tests each lasted 16s. The expected standard deviation of the calculated ending position is 0.73°. The duration of the rate ramp experiment was 18s, so the expected standard deviation of the calculated ending position is 0.82°. Table 5.5 lists the actual standard deviation of the ending positions for the dynamic tests performed on the MG1101a, and Table 5.7 lists the same for the ADIS16255. The MG1101a demonstrated some variation, but was at or below the predicted value in all cases. The ADIS16255 behaved more predictably and was also below the prediction in all cases. The standard deviation of the ending positions for the $10°/s$ test was only half of the predicted value and appears to be an outlier.

One phenomenon was observed that could not be explained. The ADIS16255 appeared to develop some type of bias during the 5°/s and 10°/s tests (Table 5.7). This resulted in a mean ending position of about $-1°$ in both cases when it should have been $0°$. This behavior was not observed with the MG1101a. It was also not observed in the dynamic tests performed on the ADIS16255 at lower angular rates. The mean of the ending position in the rate ramp test is approximately $-.15°$ which is significant. However, without understanding the underlying mechanism it is difficult to conclude whether there is common root cause. The

tests were repeated to verify that the observations were correct, and they were repeatable.

5.3 Conclusion

Overall, the MG1101a and ADIS16255 met or exceeded their specifications. The test results show very little difference between the two. For both sensors, the standard deviation of the positional uncertainty will grow at a rate of approximately 0.045°/s and the absolute positional uncertainty will grow at a rate of 0.23°/s. Therefore, assuming a test duration of 10 s, the positional uncertainty will be within 0.45° 68% of the time, within 0.9° 95% of the time, and always within 2.3°. In the absence of a clear difference, the choice of sensor must be made based on the other features and the ease with which it can be intergrated with the rest of the system. The descriptions from Chapter 3 show that the ADIS16255 has a much richer feature set than the MG1101a. In addition, it was assumed that the test results for the ADIS16255 would be representitve of the ADIS16350. The ADIS16350 is a complete 6 DoF intertial sensor that would greatly simplify the overall system. For these reasons, the ADIS16350 was selected.

For 3D ultrasound imaging the acceptable error is that amount of error which does not result in the assignment of data to the incorrect voxel. Previous work [24] has shown the acceptable error to be approximately 0.5°, at a scanning depth of 10 cm. Based on the results of this testing the probability of meeting that requirement is approximately 0.68 for any individual scan. The worst case error was found to be 2.3°, which could cause a reconstruction error of up to 3 voxels. It is important to note that this error would be spread out across the volume such that there would be three, single voxel errors distributed through the reconstructed volume as opposed to having a single, three voxel error in one location.

Chapter 6

Accelerometer Experiments

The purpose of the accelerometer tests was to determine if it was possible to use linear accelerometers as a means of tracking position. The test apparatus is described in Section 4.1.5 and Section 4.4. In order to estimate position using acceleration measurements it is necessary to integrate the acceleration signal twice; the integral of acceleration is velocity and the integral of velocity is position.

In addition to sensing acceleration caused by physical motion, linear accelerometers also sense the acceleration due to gravity. It is therefore necessary to subtract the component due to gravity from the observed acceleration prior to integration. To calculate the gravitational acceleration experienced by the sensor, one must have knowledge of the orientation of the sensor with respect to gravity. Using the definition of the inner product, given in (6.1), it is possible to resolve the gravitational acceleration acting on an arbitrary axis.

Figure 6.1 illustrates this in the case of a single axis accelerometer, where θ is the angle between the accelerometer's sensitive axis, x_b and horizontal, x_i. The gravitational acceleration observed by the sensor, g_b, is given by (6.2). If it is known that the sensor is stationary, then (6.2) can be reformulated to calculate the angle, θ, as shown in (6.3).

$$\vec{A} \cdot \vec{B} \equiv |\vec{A}||\vec{B}| \cos\theta \tag{6.1}$$

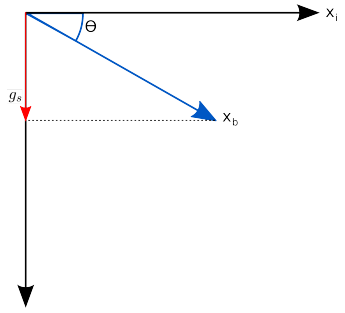


Figure 6.1: Diagram of the effect of gravity on a single axis accelerometer

$$\begin{aligned} g_s = \vec{A} \cdot \vec{B} = |\vec{X_b}||\vec{G_i}| \cos(-\frac{\pi}{2} - \theta) \\ g_s = |-g| \cos(-\frac{\pi}{2} - \theta) \\ g_s = -g \sin(\theta) \end{aligned} \tag{6.2}$$

$$\theta = \sin^{-1}(-\frac{g_s}{g}) \tag{6.3}$$

A 6 DoF tracking system has three linear degrees of freedom and the 5 DoF tracking system presented here has two linear degrees of freedom. The analysis in the preceding paragraph can be expanded to cover these cases. Consider three accelerometers oriented orthogonally with respect to one another, defining the body frame of reference, as in Figure 6.2. As in the previous case, it is necessary to express gravitational acceleration with respect to the body frame of reference, x_b, y_b, and z_b. However, there are now three degrees of freedom in orientation instead of one: the pitch angle, ρ, is defined as the angle of the x_b axis relative to ground, the roll angle, ϕ, is defined as the angle of the y_b axis relative to ground, and theta, θ, is the angle of the z_b axis relative to gravity. It is important to note that these are not equivalent to the Euler angles that are often used to represent orientation.

Equations (6.4) express the gravitation acceleration measured by each accelerometer axis given these angles, derived using the previous result. However, the presence of more than one sensitive axis allows a significant improvement to be made. The sin and cos functions are not

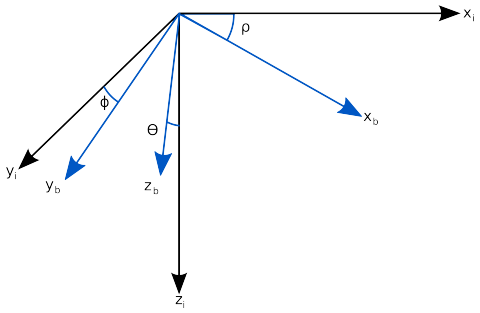


Figure 6.2: Diagram of the effect of gravity on a 3 axis accelerometer

linear, particularly as θ approaches 0°and 90°respectively. Using the Pythagorean theorem, as expressed in equation (6.5), the angles can be expressed as the ratio of the magnitudes of two sides of a right triangle using the tangent function. Equations (6.6) exploit this to yield functions that are much more linear. These equations can also be restated to derive the angles if it is known that the sensor is stationary, as in equation (6.7).

$$g_{xb} = -g\sin(\rho) \quad g_{yb} = -g\sin(\phi) \quad g_{zb} = -g\cos(\theta) \tag{6.4}$$

$$|\vec{G}| = \sqrt{g_{xb}^2 + g_{yb}^2 + g_{zb}^2} \tag{6.5}$$

$$\frac{g_{xb}}{\sqrt{g_{yb}^2 + g_{zb}^2}} = \tan(\rho) \quad \frac{g_{yb}}{\sqrt{g_{xb}^2 + g_{zb}^2}} = \tan(\phi) \quad \frac{\sqrt{g_{xb}^2 + g_{yb}^2}}{g_{zb}} = \tan(\theta) \tag{6.6}$$

$$\rho = \tan^{-1}\left(\frac{g_{xb}}{\sqrt{g_{yb}^2 + g_{zb}^2}}\right) \quad \phi = \tan^{-1}\left(\frac{g_{yb}}{\sqrt{g_{xb}^2 + g_{zb}^2}}\right) \quad \theta = \tan^{-1}\left(\frac{\sqrt{g_{yb}^2 + g_{zb}^2}}{g_{xb}}\right) \tag{6.7}$$

If the goal is to track position over time using accelerometers, as it was in these tests, knowledge of the initial pose is critical. When the sensor is stationary the acceleration measured along each sensor axis is attributable only to gravity. Therefore, (6.7) can be used to determine the initial pose.

6.1 Procedure

The testing performed on the LIS3LV02DL linear accelerometer was not intended to characterize the device's performance. Rather, the goal of these tests was to determine if MEMS accelerometers generally could be used to measure linear displacement in the tracking system. The EK3LV02DQ evaluation module was mounted to the print head of the HP7255A XY plotter, as illustrated in Figure 4.16. The plotter was commanded to move approximately 11cm in the positive x direction and then 11cm in the negative x direction. Only the data for the x-axis was used, the y and z data were ignored. The plotter allows the velocity to be controlled directly via a HP-GL command and was set to 1 cm/s to simulate a typical motion of a human hand during ultrasound scanning. The acceleration of the plotter head is not directly controllable and no effort was made to control it via microstepping or some other method.

6.2 Results and Discussion

Figure 6.3 illustrates the accelerometer output for the movement described in the previous section. The period from 3 to 14 s corresponds to a forward movement of 11 cm at a rate of 1 cm/s. The period from 16 s to 27 s is the reverse movement. It is immediately obvious that the stepper motors produce an enormous amount of noise. A sharp positive spike at the beginning and a sharp negative spike at the end of the forward motion, such as those visible in Figure 6.6, are expected. Although these spikes are partially visible, they are largely obscured by the noise.

The Fast Fourier Transform (FFT) of the accelerometer data is illustrated in Figure 6.4. The large spikes are the result of the step frequency of the stepper motors and their harmonics. Since the acceleration caused by the motors should have a mean of zero, the noise should not have a large effect on calculated velocity and position. However, the noise does make it harder to visualize the data.

Figure 6.6 is a plot of the acceleration data after the application of the FIR Low Pass

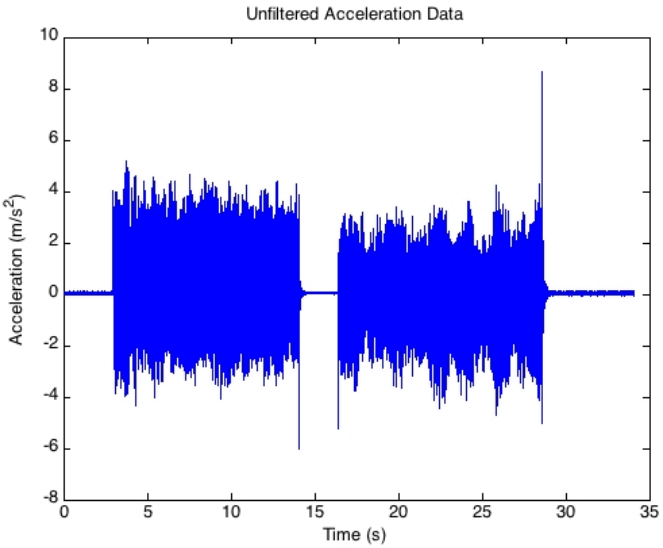


Figure 6.3: Unprocessed acceleration data from the x-axis accelerometer

Filter (LPF) shown in Figure 6.5. Now the expected spikes in acceleration are clearly visible. There are also a few spikes that were not expected. These are most likely the result of some imperfection in the experimental apparatus. It is also important to notice that there is a bias in the data. This bias appears to be composed of both a constant and a variable component. The constant component is most likely a slight misalignment of between the x-axis of the sensor and the force of gravity. The variable component appears to be the result of variation in the alignment of the plotter arm across its stroke. The bias is slightly higher between 15 and 25 s, which corresponds to the farthest distance between the plotter arm and the origin. The misalignment may be intrinsic to the device or it may be the result of the added weight of the cable pulling on the arm. Regardless of the source, the bias must be dealt with prior performing the integration. Failing to do so results in errors much larger than the true motion, as demonstrated in the next graphs.

Figure 6.7 illustrates both the expected velocity, in green, and velocity calculated as the first integral of the acceleration, in blue. As expected, a constant acceleration bias is reflected as a constant slope in the velocity. Notice that the positive and negative velocity are still visible.

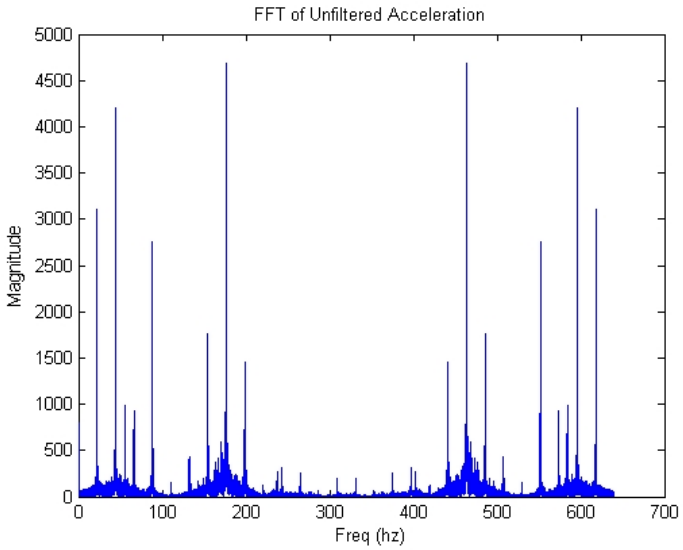


Figure 6.4: Fast Fourier transform of the unprocessed x-axis accelerometer data

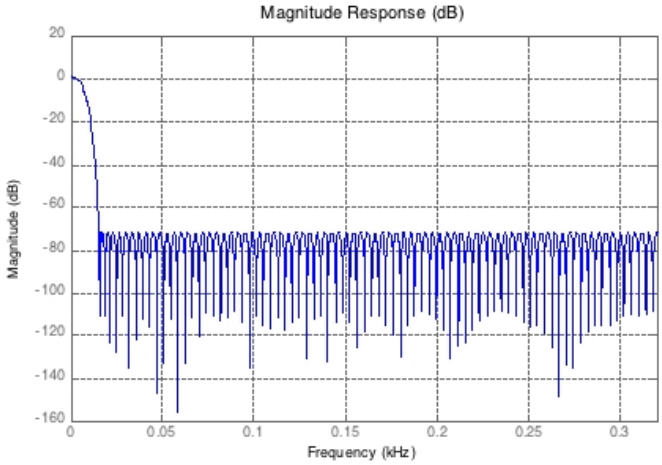


Figure 6.5: Transfer function of the linear phase FIR filter applied to the acceleration data

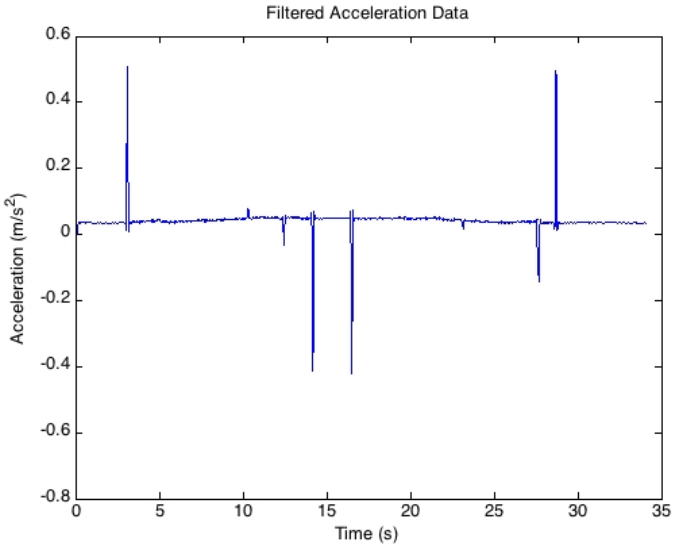


Figure 6.6: Acceleration data after filtering to remove high frequency content

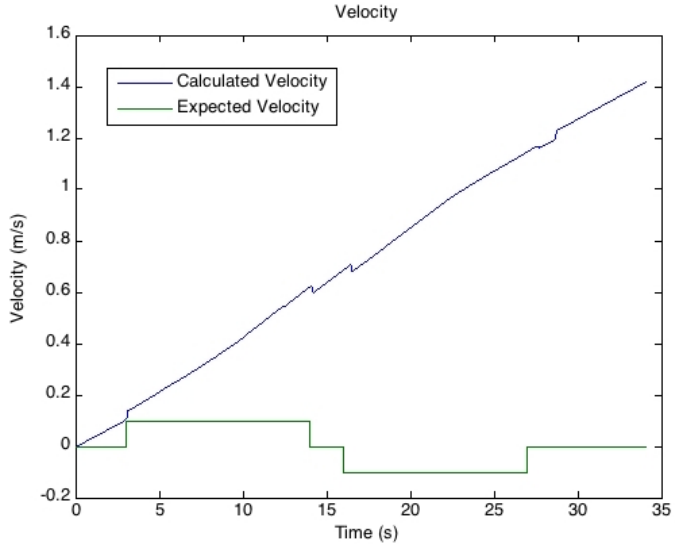


Figure 6.7: Velocity obtained by integrating the filtered accelerometer data

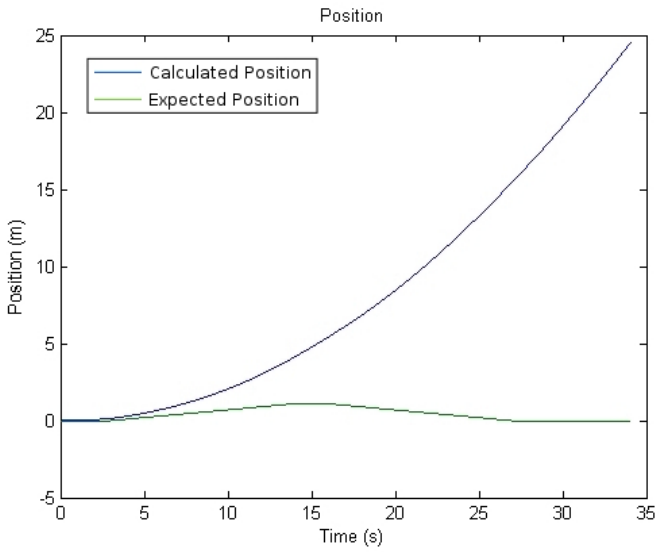
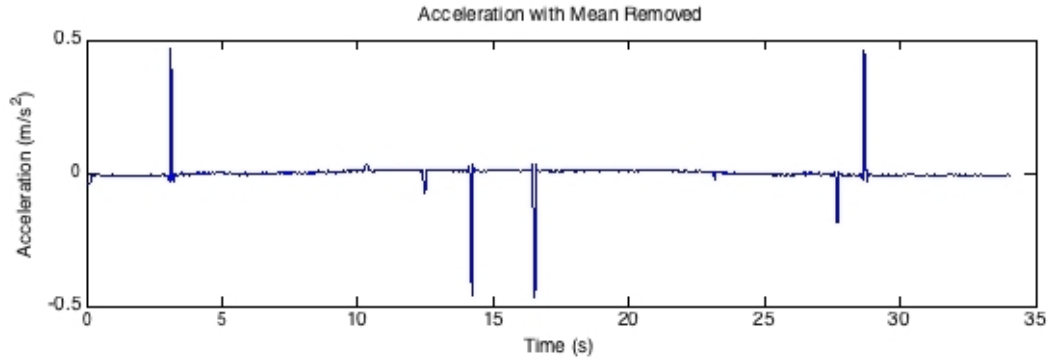


Figure 6.8: Position obtained by twice integrating the filtered accelerometer data

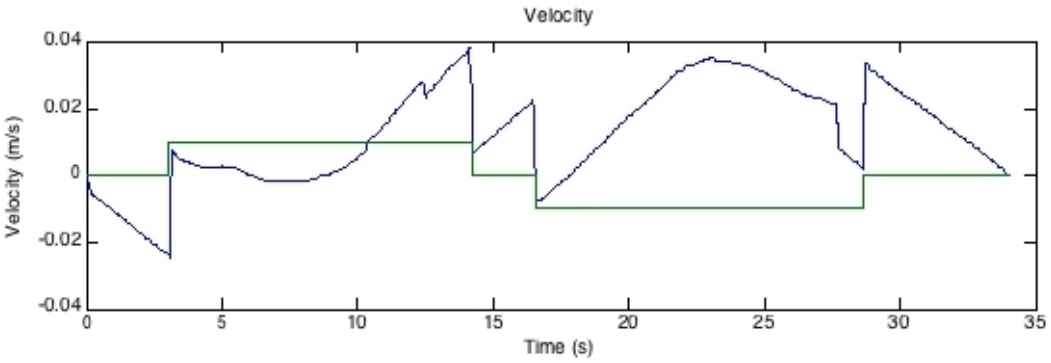
Figure 6.8 shows the expected position in green and calculated position in blue. As expected, the acceleration bias results in a position error that grows with the square of time. The maximum displacement was expected to be 0.11 meters and the absolute displacement at the end of the experiment should be 0. Although present, the effect of the motion is not even visible in the calculated position because the bias effects dominate.

One way of dealing with the bias in the acceleration data is to subtract the mean from each sample. As illustrated in Figure 6.9, the calculated velocity and position are now much closer to the expected result, in green. However, the mean is clearly not a good estimate of the bias because it does not take into account the fact that there are multiple components. Lumping the bias into a single term and subtracting gives the appearance of motion when the device is in fact at rest, as can be seen in between 0 and 3 s and 16-25 s in the middle graph.

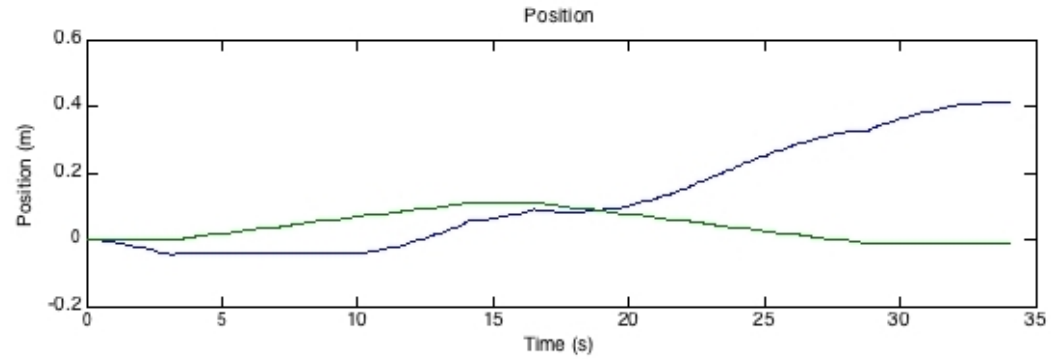
Another method for dealing with the bias is to remove only the static component. The static component of the bias term is estimated as the mean of the acceleration from 0-3s, when the sensor is at rest. The bias estimate is then subtracted from all the subsequent samples of acceleration data. The resulting calculated velocity is now much closer to the



(a)



(b)

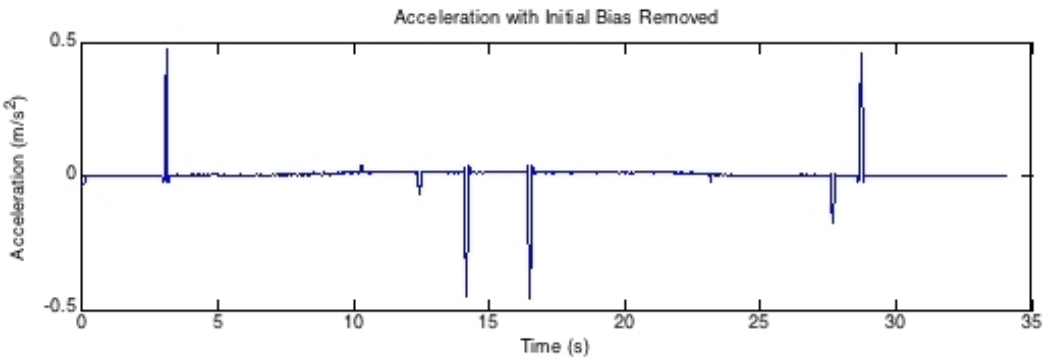


(c)

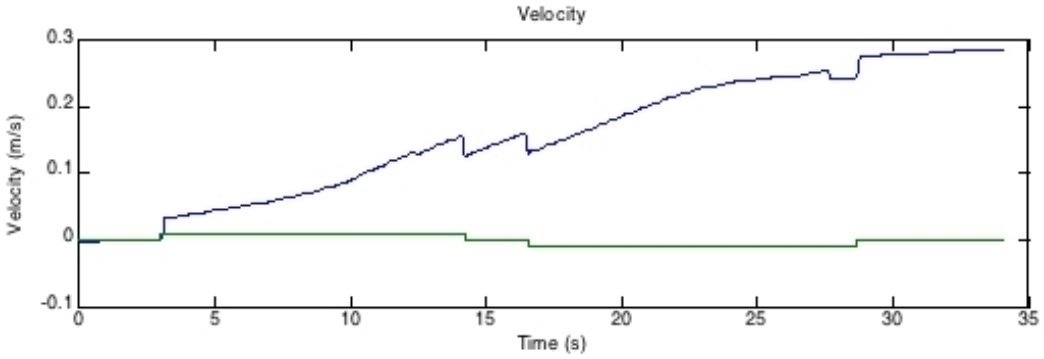
Figure 6.9: Effect of subtracting the data mean From each sample on the estimated velocity and position. The blue line is actual data and the green line is expected data.

expected velocity, at least in terms of its shape. The effects of the dynamic bias term are evident in the slope of the velocity curve between 10-25s in figure Figure 6.10(b). As illustrated in Figure 6.10(c), the dynamic bias still dominates the actual movement in the calculated position.

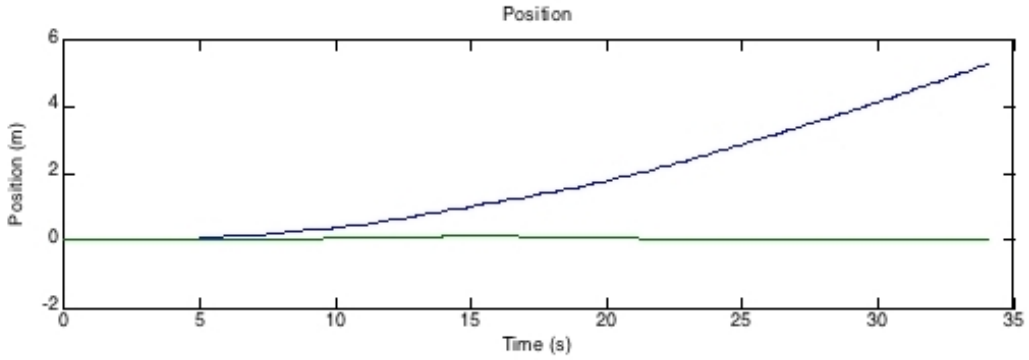
Figure 6.11 illustrates the results of attempting to compensate for both static and dynamic bias. The static bias compensation is carried over from the previous section. The dynamic bias is estimated as 0 at the beginning and end of the run (0-4 s and 27-34 s) because the bias here should have been accounted for by the static bias estimate. The maximum value of the dynamic bias is the mean of the acceleration between 14-17 s, which corresponds to a period of no motion where the plotter head is at its maximum displacement. Figure 6.12 illustrates the estimated dynamic bias. The mean value here is interpreted as a variation in the effect of gravity between the origin of the plotter arm and its maximum extension. This is most likely caused by an imperfection in the apparatus or the weight of the cable twisting the arm slightly as it extends. The complete dynamic bias estimate is shown in green in Figures 6.11(a), 6.11(b), and 6.11(c). This method is clearly better than the previous attempt but it is still inadequate.



(a)

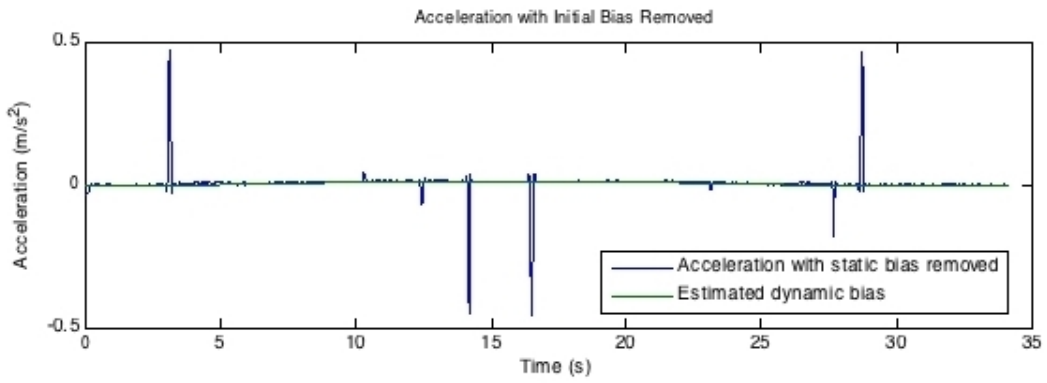


(b)

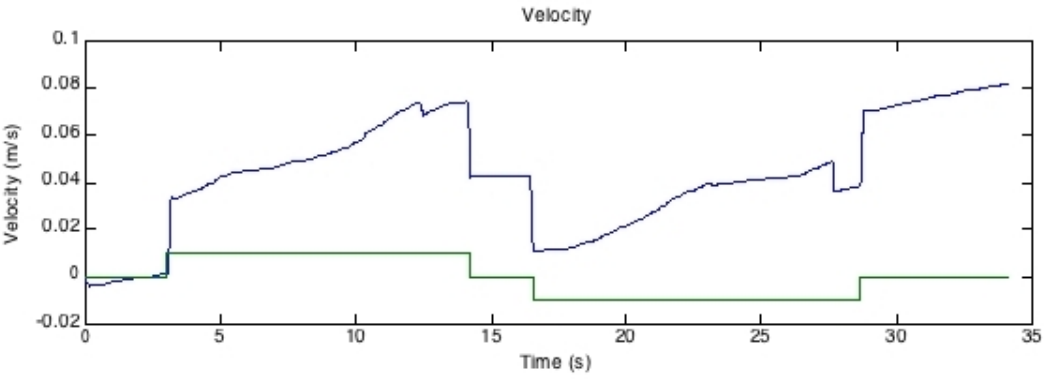


(c)

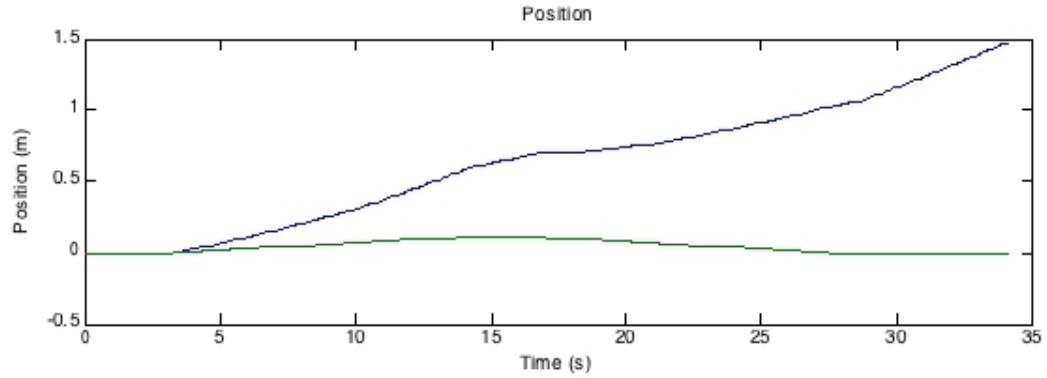
Figure 6.10: Effect of subtracting only the static bias from the accelerometer data prior to integration. The blue line is actual data and the green line is expected data.



(a)



(b)



(c)

Figure 6.11: Effect of subtracting the static and dynamic bias From the accelerometer data prior to integration. The blue line is actual data and the green line is expected data.

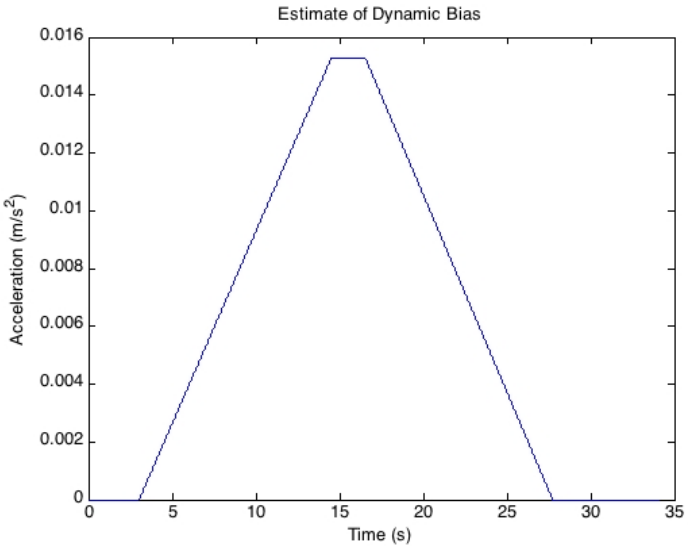


Figure 6.12: Estimate of dynamic bias present in the accelerometer data

6.3 Conclusion

The preceding experiments illustrate that position estimates based on measurement of linear acceleration are extremely sensitive to small errors. This is primarily because of the need to integrate the output twice to estimate position from the acceleration measurements. Even in situations where the sensitive axis of the accelerometer is nearly perpendicular to gravity, small misalignments can cause gravity to be the dominant term in the estimated position. The magnitude of gravity relative to the magnitudes of the accelerations we are trying to measure is so great that it is difficult to overcome.

Equation (6.8a) expresses the acceleration measured by the x_b accelerometer as the sum of the acceleration caused by motion and the component of gravitational acceleration along that axis, g_{tx}. The gravity term is expanded in equation (6.8b) to the sum of the true gravitational acceleration, g_x, and an error term, ϵ. As described earlier, the estimate of gravitational acceleration must be subtracted from the overall acceleration to yield the true

acceleration, a_t, resulting in equation (6.8c).

$$a_x = a_m + g_{tx} \tag{6.8a}$$

$$a_x = a_m + (g_x + \epsilon) \tag{6.8b}$$

$$a_t = a_m + \epsilon \tag{6.8c}$$

Equation (6.9) demonstrates the effect of the error in the acceleration measurement on the estimated velocity. The constant error term in the acceleration estimate becomes a linear error term in the velocity estimate.

$$v_x = \int (a_x + \epsilon_x)\, dt = a_x t + \epsilon_x t \tag{6.9}$$

When the second integral of acceleration is taken to arrive at the position estimate, the constant acceleration error becomes a squared error term in the position estimate, as illustrated in (6.10).

$$p_x = \iint (a_x + \epsilon_x)\, dt = \int (a_x + \epsilon_x)\, t dt = \frac{1}{2} (a_x + \epsilon_x)\, t^2 \tag{6.10}$$

Suppose that the only source of error in the tracking system is quantization error. If the accelerometer has 10 bits of precision and a full-scale range of ± 2 g, or 39.6 m/s^2, each bit represents 0.00391 g or 0.03831 m/s^2. The largest possible quantization error is $\frac{1}{2}$ LSB. On average, ϵ will be equal to $\frac{1}{4}$ LSB, or 0.00958 m/s^2. Thinking only in terms of acceleration, it is clear that an error of 9.58 mm/s^2 is unacceptably large in a system that requires sub-mm accuracy. Carrying this error through the integrations results in an accumulated velocity error of 0.0958 m/s and an accumulated position error of 0.4789 m after only 10 s.

This analysis represents an unrealistic scenario for using accelerometers to track position. In a real world system the orientation will be changing along with the position. The gravitational component of acceleration along each axis will have to be estimated using the integrated outputs of the gyroscopes. Not only are the gyroscopes less accurate than

the initial orientation estimates made using the accelerometers directly, Chapter 5 showed that they suffer from serious drift errors. The ADIS16255 demonstrated a 1σ drift rate of $0.045°/s$. An error in the angle estimate of $0.045°$, as would be likely after only 1 s, results in an error of 0.0078 m/s^2 in the gravity estimate. After 10 s, this gravity error will result in a position error of 0.385 m. This error is in addition to that attributable to quantization errors in the accelerometers.

The results of this testing indicate that neither MEMS accelerometers nor MEMS gyroscopes posses the accuracy and stability necessary to construct a sub-mm accurate inertial tracking system.

Chapter 7

Optical System

The optical system provides linear position information to the tracking system. To accomplish this, the ADNS-2610 optical mouse sensor is used to track features on the skin surface. The final goal is to integrate it directly into an ultrasound transducer. The prototypes presented here are small enough for this purpose. Because we do not have the means to manufacture our own transducer housings easily, the prototypes attach to the outside of the housing. In the future, the components of the prototypes can be integrated into the transducer, thereby achieving seamless integration.

Using the ADNS-2610 to track the skin surface presented several challenges that had to be addressed. First, when used as intended, the combination of the ADSN-2610 and the HDNS-2100 mouse lens has very limited depth of field. This means that the distance from the front of the lens to the tracking surface has to stay in a very small range for the surface to be in focus. Second, the HDNS-2100 mouse lens is not available in a form that can be easily integrated in to the system proposed here. However, there are no suitable replacements for it because of its unique characteristics. Third, there is very little room at the end of a typical ultrasound transducer in which to put the sensor and optics. For this reason, we decided to experiment with optical fiber bundles that would allow that ADNS-2610 to be mounted away from the end of the transducer. Lastly, because of the small scale of the components involved, it was very difficult to assemble a test apparatus, and later a

prototype, that could hold the mechanical tolerances necessary to make the system work. The following chapter documents the development of the optical tracking system from initial concept through working prototype.

Carsten Poulsen first demonstrated the effectiveness of the ADNS-2610 optical mouse sensor as a tracking device for 3D ultrasound in his master's thesis [24]. His initial prototype did not include an optical fiber bundle and was used to successfully to produce 3D ultrasound volumes. He then developed an optical tracking system based on an optical conduit and two plano-convex lenses, as a means of moving the image sensor away from the skin surface. Although that system was able to acquire identifiable images, it was unable to track reliably on the skin surface.

Irene Gouverneur picked up where Carsten left off. She analyzed the performance characteristics of the ADNS-2610 image sensor as well as several types of optical fiber and several lenses. The results allowed her to find an optimal combination of commercially available components for this application. At the end of her work she demonstrated that these components were able to track on simulated skin surfaces [17].

The work presented here builds upon the efforts of both Carsten and Irene. The results validate both the initial concept of tracking the skin surface with an optical mouse sensor and the idea of using several lenses in conjunction with an optical fiber bundle to allow the image sensor to be moved away from the skin surface.

7.1 Definition of Terms

There are a number common terms in the field of optics with which the reader may not be familiar. These terms are used extensively in the following section and they are defined in Table 7.1 for the benefit of the reader.

Table 7.1: Optical terminology and abbreviations

Term	Abbreviation	Definition
Image Distance	ID	Distance from the back of the Elmo lens to the front surface of the mouse lens
Object Distance	OD	Distance from the front of the Elmo lens to the tracking surface
Region of Interest	ROI	Physical area that is within view of the image sensor
Depth of Field	DOF	Range of object distances over which the optical system remains in focus
Contrast Ratio	CR	The difference between the maximum and minimum pixel values, divided by their sum
Line Profile Contrast Ratio	LPCR	Contrast ratio calculated using the values sampled by following a straight line through an image

7.2 Previous Work on Optical Tracking with the ADNS-2610 Optical Mouse Sensor

Irene Gouverneur performed a number of different experiments to identify an objective lens and an optical fiber bundle that would meet the requirements of the tracking system. The requirements were that the system include a flexible optical fiber bundle, the components be small enough to fit in an ultrasound transducer, and that it deliver enough contrast and resolution to the image sensor that it was able to track reliably. In order to characterize the components more completely, many of the experiments utilized a Pixelink PL-A781 6.6 megapixel camera instead of the 324 pixel ADNS-2610.

The fist experiment studied the effects of illumination type and intensity. This experiment did use the ADNS-2610 to capture images, to ensure that the illumination system was optimized for the final application. A 50 mW, 620 nm red LED of the type specified for use with the ADNS-2610, and a 3.5 mW, 635 nm laser diode were selected for evaluation. The results indicated that the LED provided superior image quality.

The effects of illumination angle were also investigated. The illumination angle is the angle between light rays emitted from the illumination source and the line perpendicular to the surface being imaged, as illustrated in Figure 7.1. When light rays reflect off of a rough surface at an angle, shadows are created. It is these shadows that provide the image contrast necessary for the ADNS-2610 identify and track surface features. This experiment quantified the image contrast over a range of illumination angles. The Pixelink camera was used for

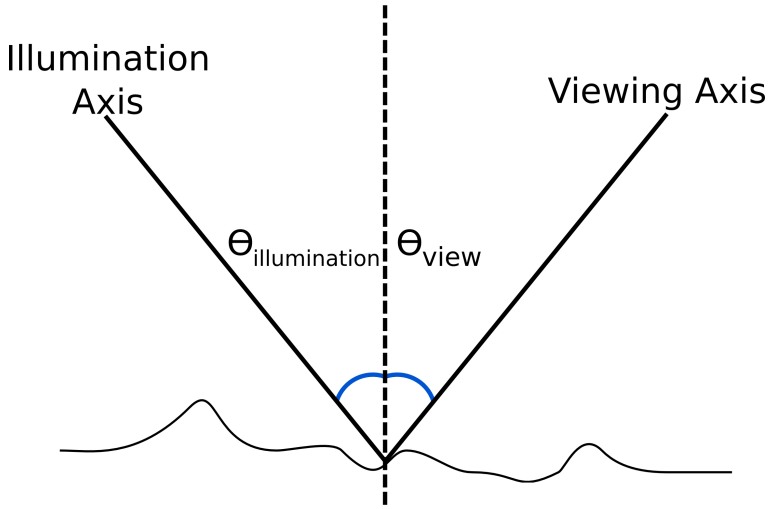


Figure 7.1: Graphical representation of illumination and sample angle

Table 7.2: Fiber bundles tested

Name	Material	Manufacturer	Diameter	Elements
Clad Rod 1	Glass	Schott	2 mm	1459
Clad Rod 2	Glass	Schott	2 mm	25200
Acid Leached Fiber Bundle	Glass	Schott	1. 5mm	18000
Plastic Fiber Bundle	Plastic	Mitsubishi	2.5 mm	6000
GRIN Rod	Glass	NSG America	2 mm	NA
Sumitomo	Glass	Sumitomo	2 mm	50000
5 mm OD	Glass	Unknown	5 mm	Unknown

this purpose because its higher resolution provided better contrast data. Irene concluded that an illumination angle of 50°and a sample angle of 30°provided the best image contrast.

The two previous experiments laid the foundation for the evaluation of the lenses and fiber bundles. Several different lenses (L2 in the following descriptions) and fiber bundles were selected to provide a good sample of what was commercially available, within the system's space limitations. The lenses that were tested are listed in Table 7.3 and the fiber bundles are listed in Table 7.2.

The experimental apparatus is illustrated in Figure 7.2. Light from the LED was focused onto the sample at an illumination angle of 30°. This angle was selected, rather than the experimentally determined optimum of 50°, because of physical constraints in the apparatus.

Table 7.3: Objective lenses tested

Name	Focal Length	Diameter
Elmo QT282AS	2.2 mm	5 mm
Elmo QT288	8 mm	5 mm
Elmo QT3515	15 mm	5 mm
Single Achromatic	7.5 mm	5 mm

The objective lens (L2), image guide, coupling lens (L3), and imager, were arranged as shown. Tests were conducted with both the Pixelink camera and the ADNS-2610 imager. L3 was an Olympus 4x lens for the Pixelink camera or the HDNS-2100 optical mouse lens for the ADNS-2610.

Modulation Transfer Function (MTF) and *Line Profile Contrast Ratio* (LPCR) were used as the performance metrics. The MTF is the magnitude response of the optical system to sinusoids of different spatial frequencies. In frequency domain image analysis, an image can be considered to be composed of plane waves. A Linear Shift Invariant (LSI) optical system images a sinusoid as another sinusoid [26]. For a number of reasons, the imaging process decreases the modulation depth, M, of the image relative to the original object. The modulation depth, M, is defined as the amplitude of the irradiance level divided by the bias level, as illustrated in (7.1).

$$M = \frac{A_{max} - A_{min}}{A_{max} + A_{min}} \tag{7.1}$$

Because M is dependent on the difference between A_{max} and A_{min}, it is a measure of contrast. The Modulation Transfer (MT) is the ratio of modulation in the image to that of the object. MT is frequency dependent, and represented as a function of spacial frequency, it becomes the MTF. This is stated mathematically in (7.2), where ξ represents the spatial frequency.

$$MTF(\xi) = \frac{M_{image}(\xi)}{M_{object}} \tag{7.2}$$

MTF is particularly convenient because a system's overall MTF is the product of the MTFs of the system's individual components. This allows the components to be characterized separately which simplifies testing.

As the name implies, LPCR is also a metric that is based on image contrast. It is the

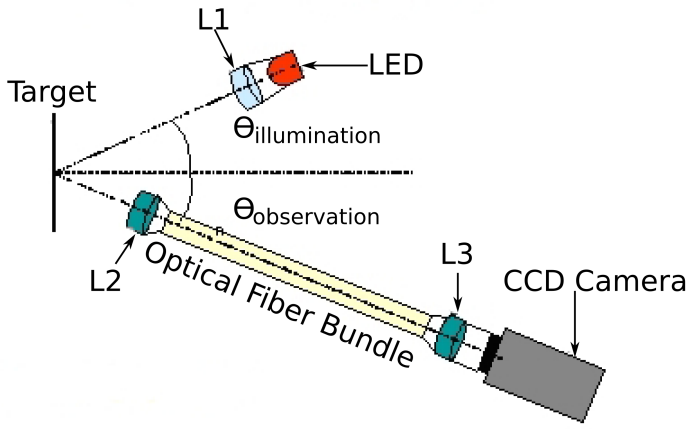


Figure 7.2: Experimental setup for MTF and LPCR testing

ratio of maximum to minimum image intensity taken over a line, as in (7.3). Figure 7.3 is a graphical representation of the LPCR. In these experiments, the LPCR was calculated using an element of the USAF-1951 test pattern. The pattern alternates between black and white. An ideal optical system imaging a pattern of total black and total white will achieve a LPCR of 1.

$$LPCR = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{7.3}$$

The first set of tests evaluated the MTF of the lenses by themselves. Images of the USAF test pattern were taken using the individual lenses and the Pixelink camera. Then PixelScope OpticStudio software was used to calculate the experimental MTF. First, L3 was tested. L3 was not one of the lenses that was being considered for the final system. L3 was necessary to couple light exiting the fiber bundle into the Pixelink camera. Without first deriving the MTF of L3 it would be impossible to derive the MTF of the fiber bundles alone. Next, the MTF of the objective lenses was derived. The data indicated that the Elmo F8 had the best MTF and the single achromatic lens had the worst MTF.

Because it had the best MTF, the Elmo F8 lens was used as the objective lens for

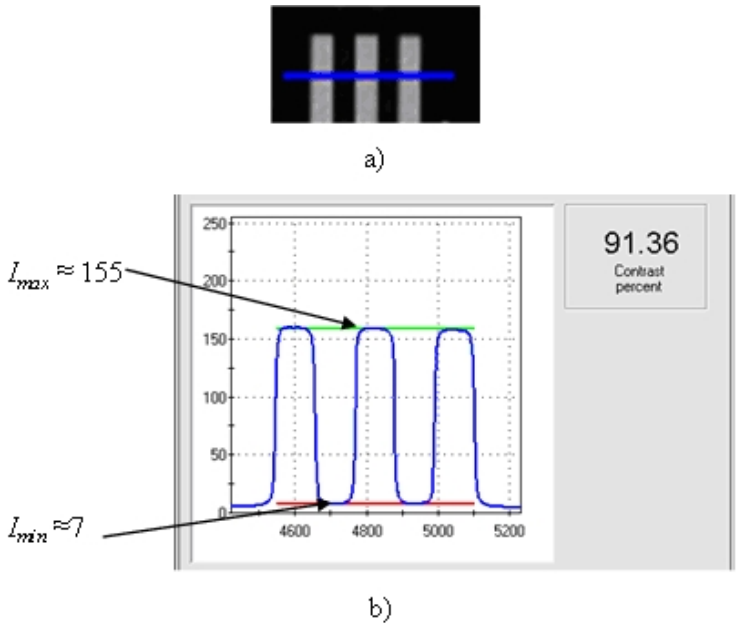


Figure 7.3: Graphical representation of LPCR [17]

evaluating the MTF of the fiber bundles. The GRIN fiber bundle did not use any objective lens because, in fibers of this type, the material's index of refraction is manipulated to create a lens within the fiber itself. As before, images were acquired using the Pixelink camera and the MTF was calculated using the PixelScope OpticStudio software. The results are illustrated in Figure 7.4.

From the group of fibers initially evaluated, only the Schott 12 μm and acid leached fiber bundles were selected for more detailed analysis. This was primarily because of the limited flexibility of the other fiber bundles. In the next test series, the contrast was evaluated using both the USAF test pattern and a piece of leather that mimicd the skin surface. The LPCR was calculated for each sample. The experiment was performed with both the Pixelink camera and the ADNS-2610 tracking sensor. The results indicated that the 12 μm fiber bundle performed slightly better than the acid leached bundle but the acid leached bundle was selected for the final test series because it was the most flexible.

The final set of tests were performed to evaluate the tracking performance of the overall system. The experiment was performed using the Elmo F8 objective lens, the Schott acid

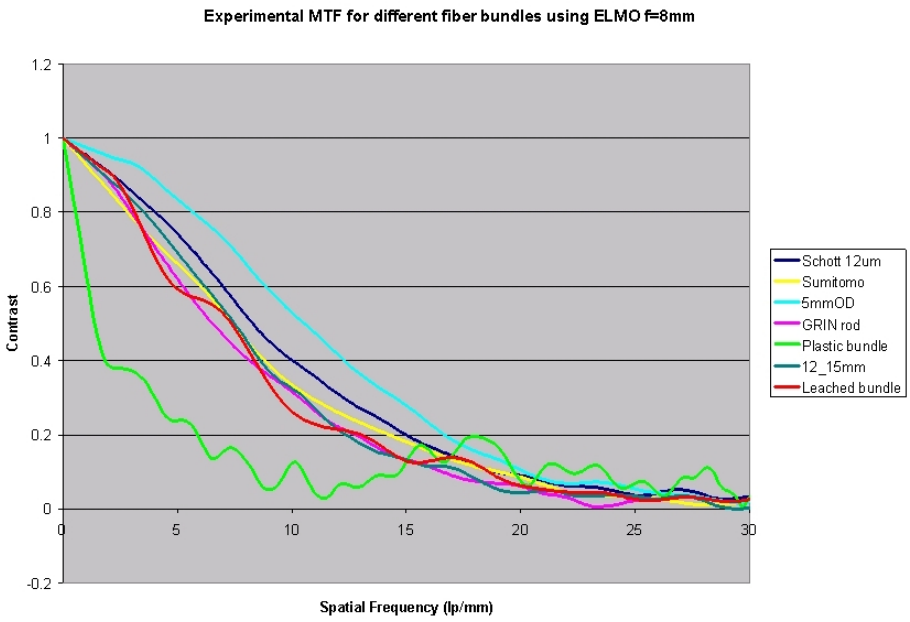


Figure 7.4: Experimental MTF of the fiber bundles

leached fiber bundle, the HDNS-2100 optical mouse lens, and the ADNS-2610 tracking sensor. The experimental setup was similar to that illustrated in Figure 7.2, with the exception that the illumination angle was 50°and the sample angle was 0°. Although the optimal contrast was achieved at a viewing angle of 30°, the angle must be 0°for tracking. If the tracking surface is at an angle with the imager, surface features in the near field will appear to move faster than objects in the far field. This condition prevents the imager from tracking. The leather sample was used as a tracking target to simulate tracking the skin surface. Tracking performance was evaluated at surface magnifications of 1.38x and 2.7x. The optical system was held fixed while the target was moved back and forth using a linear translation table. The tests were performed using ±5 mm and ±10 mm step sizes. The mean and the standard deviation of the detected movement were used as performance metrics. The results indicated that the tracking system had good repeatability at both magnifications.

7.3 Optical Tracking Configurations

The work performed by Irene Gouverneur provided clear evidence that an optical tracking system including an optical fiber bundle was feasible. The next step was to develop a miniaturized system that could attach to the outside of a transducer. The initial prototype had to attach to the outside of the transducer because we could not fabricate our own transducer with the features necessary to accommodate the tracking components internally. However, one of the goals of the prototype was to demonstrate the feasibility of placing those components inside the transducer. Therefore, it was important to make the system as small as possible.

Since earlier attempts to integrate an optical fiber bundle had failed, there was some concern that if this attempt also failed, it would hinder the development of the overall tracking system. To mitigate this risk, two different versions of the optical tracking system were developed and evaluated. In the first version, all of the components are mounted close together, at the end of the transducer. This arrangement is illustrated conceptually in Figure 7.5. LED light reflects off the skin surface and is coupled directly into the ADNS-2610 through the mouse lens and the objective lens. It will be referred to as the direct imaging configuration henceforth. The second version of the optical system includes an optical fiber bundle. In this version, LED light reflects off the skin surface and gets coupled into the optical fiber by an objective lens. The optical fiber transports the light away from the end of the transducer, to the remotely mounted mouse lens and image sensor, as illustrated in Figure 7.6. This version of the system will be referred to as the indirect imaging configuration.

Based on the results of Irene Gouverneur's work, the Schott fiber bundle was initially selected for the indirect imaging configuration because of its performance as well as its superior flexibility. However, that flexibility also made it very fragile and very expensive. Over the course of our experiments it became damaged, as described in Section 7.4.3, and we were forced to select a different fiber. Three standard fibers were selected and ordered from Edmunds Optical. The fibers were not flexible like the Schott fiber, but they were available

in much shorter lengths. The short length made it possible to integrate them despite their rigidity. The first was a 0.062 in diameter fiber with 3,012 elements. The other two were 0.125 in in diameter; one was the standard resolution model with 3,012 elements and the other was the high resolution model with 50,419 elements. Please refer back to Sections 3.3.4 and 3.3.5 for visual comparison.

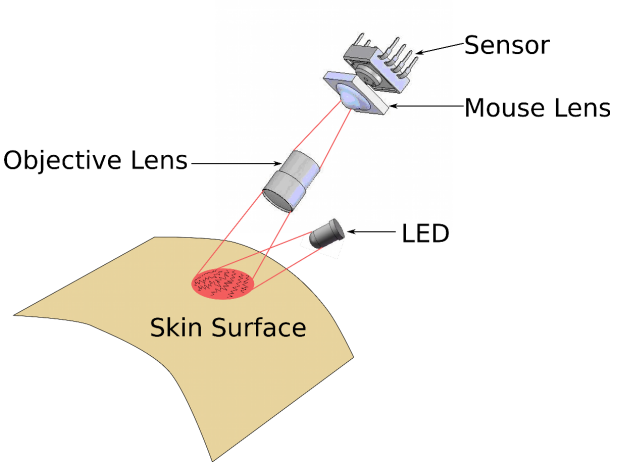


Figure 7.5: Conceptual view of the direct configuration

7.4 Experiments to Quantify the Optimal Dimensions of the Optical System

The goal of these experiments was to derive optimal values for the physical dimensions of the optical system. Two distinct sets of tests were performed; one for the direct imaging configuration and one for the indirect imaging configuration.

An experimental apparatus was developed to facilitate these experiments, illustrated in Figure 7.7. It was primarily composed of ThorLabs optical cage components mounted on an aluminum plate. There were also some components that had to be custom built to accommodate the Elmo lens and the fiber bundles. The LED is not pictured in the figures. Its position with respect to the target was constant for all of the experiments. The test target used in these experiments was the USAF 1951 test pattern, illustrated in Figure 7.8.

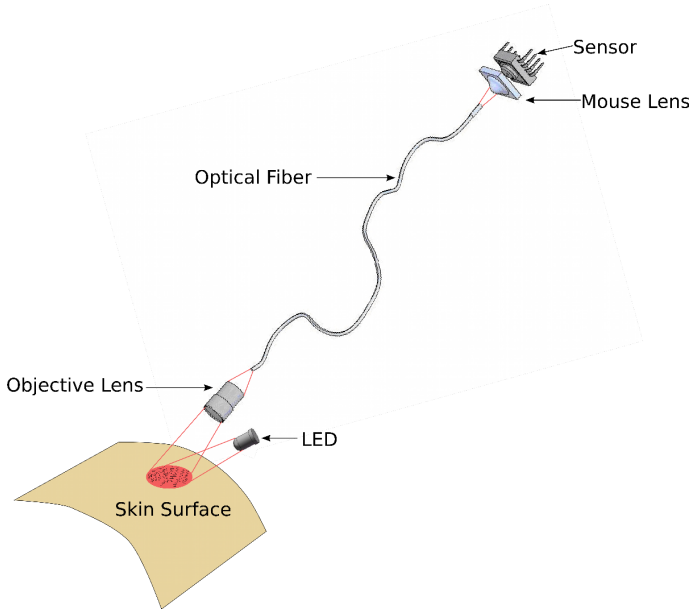


Figure 7.6: Conceptual view of the indirect configuration

Table 7.4 contains a description of each component along with its abbreviation. The abbreviations will be used throughout this section and are common with the figures. All dimensions are in mm unless otherwise specified.

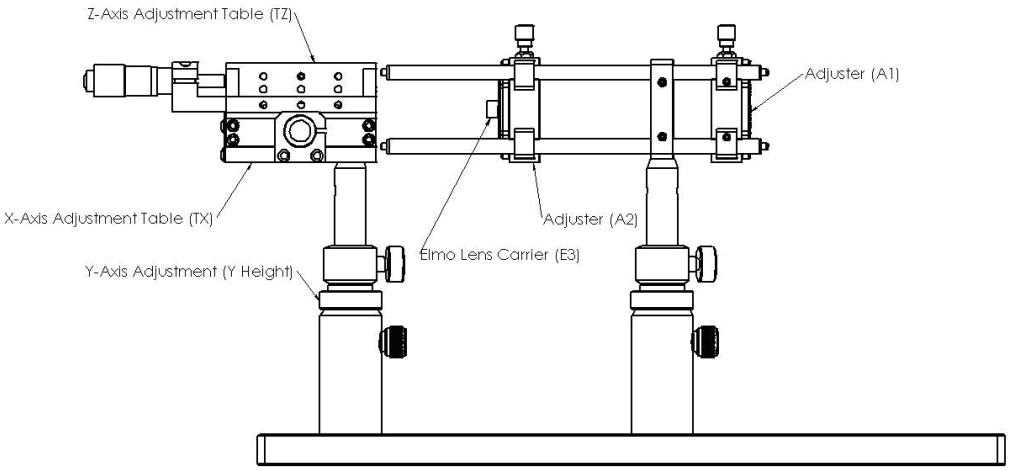


Figure 7.7: Common experimental apparatus

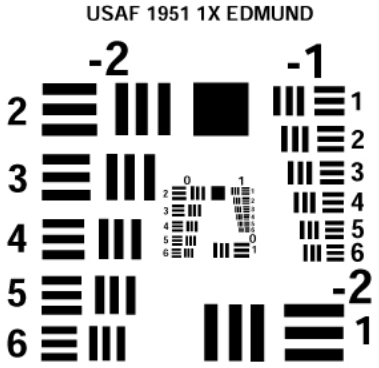


Figure 7.8: USAF 1951 test pattern

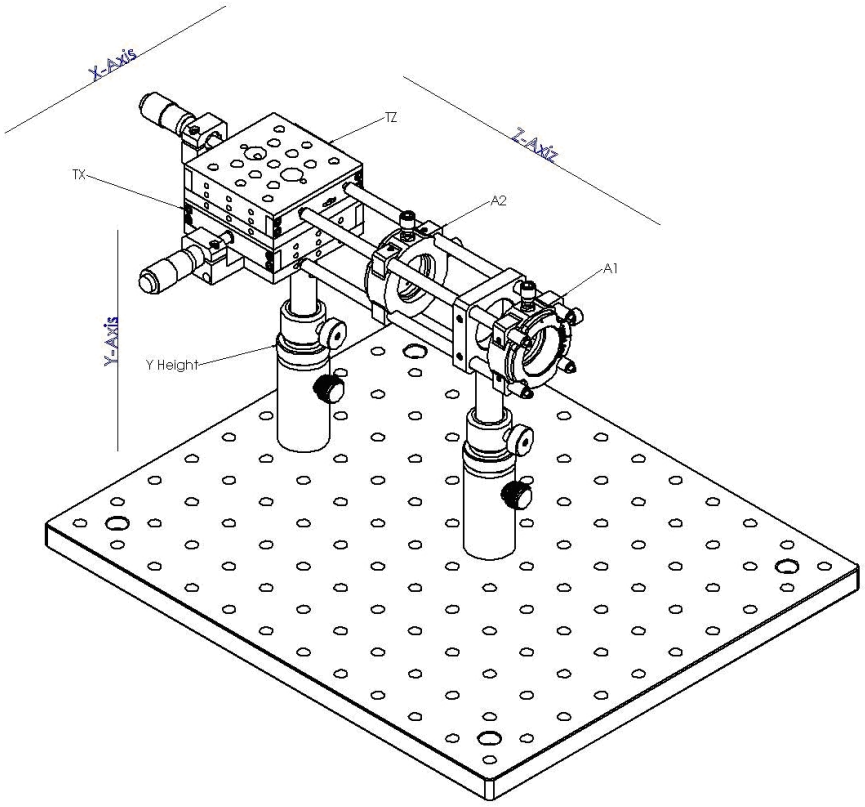


Figure 7.9: Common experimental apparatus with axes defined

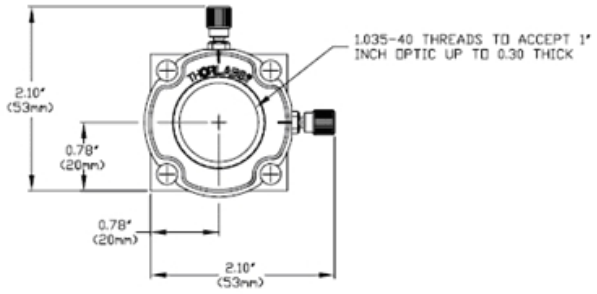


Figure 7.10: Axial Adjuster (A1 & A2)

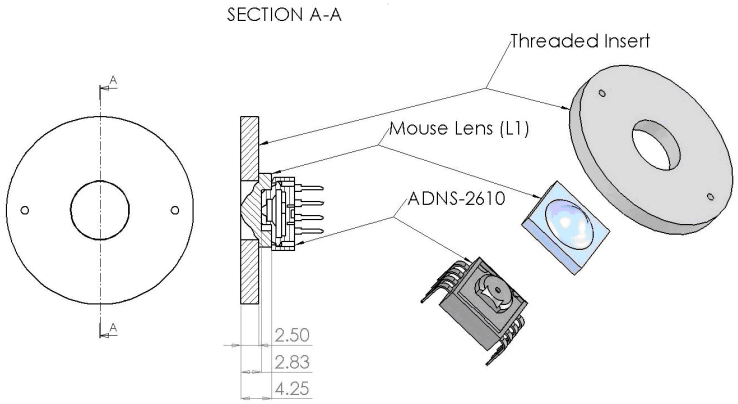


Figure 7.11: Imager Assembly (IA)

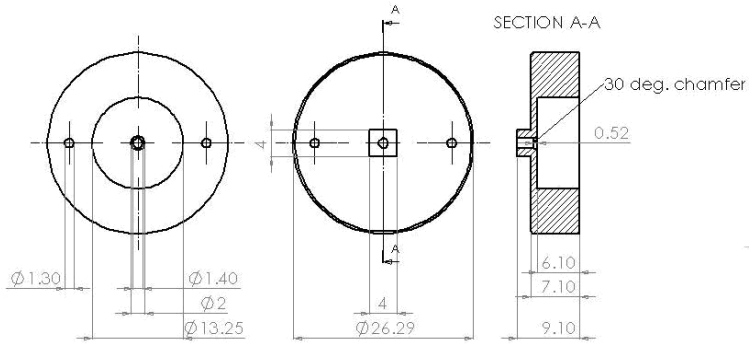


Figure 7.12: Short end fiber carrier (E1)

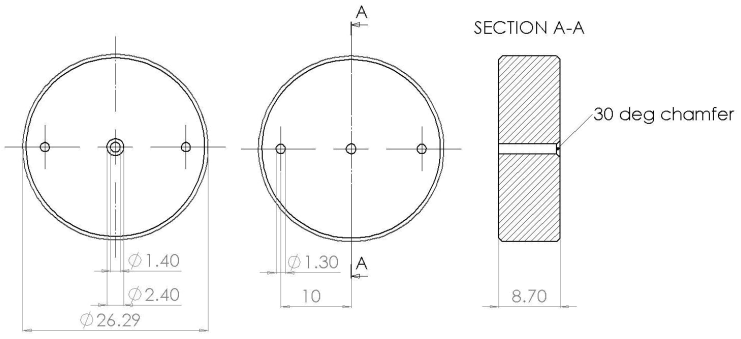


Figure 7.13: Long end fiber carrier (E2)

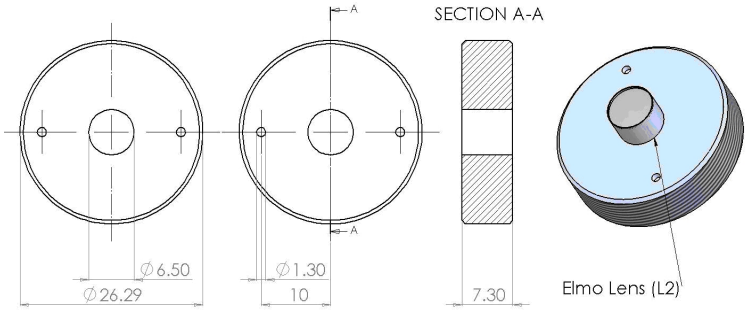


Figure 7.14: Elmo lens carrier (E3)

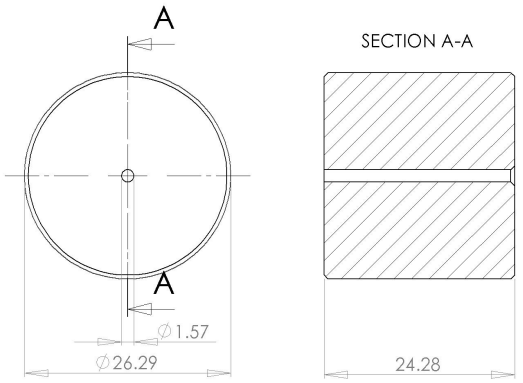


Figure 7.15: 0.062 in diameter fiber carrier (E4)

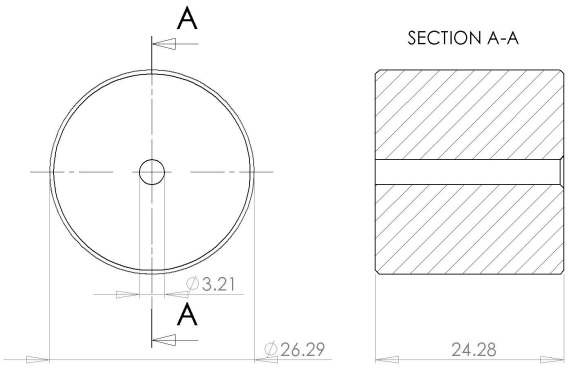


Figure 7.16: 0.125 in diameter fiber carrier (E5)

Table 7.4: Components of the optical experiment apparatus

Component Name	Abbrev.	Reference Figure	Definition
Z-Axis Translation Stage	TZ	7.7	Used to vary the position of the target along the z-axis
X-Axis Translation Stage	TX	7.7	Used to vary the position of the target along the x-axis
Imager Assembly	IA	7.11	This component is composed of 3 distinct parts: a 2.5 mm threaded insert, the ADNS-2610 sensor, and the mouse lens
Optical Mouse lens	L1	7.11	HDNS-2100 Plastic lens with very low numerical aperture that is designed to be used with ADNS-2610
Elmo Lens	L2	7.14	QT288 0.250 in micro video camera lens marketed by Elmo, Inc.
Schott Fiber Carrier	E1	7.12	Custom machined threaded insert used to mount the end of the optical fiber with the short steel sleeve
Schott Fiber Carrier	E2	7.13	Custom machined threaded insert used to mount the end of the optical fiber with the long steel sleeve
Elmo Lens Carrier	E3	7.14	Custom machined threaded insert used to mount the Elmo lens
0.062 in Diameter Fiber Carrier	E4	7.15	Custom machined threaded insert used to mount the 0.062 in diameter optical fiber
0.125 in Diameter Fiber Carrier	E5	7.16	Custom machined threaded insert used to mount the 0.125 in diameter optical fiber
Axial Alignment Adjuster	A1	7.7	A1 is used to house and align the IA with E1 in experiments that include the fiber, unused otherwise
Axial Alignment Adjuster	A2	7.7	Used to align E2 and E3 in experiments that include the fiber, used to align IA with E3 otherwise

7.4.1 Direct Imaging Configuration

There are two free parameters in the direct imaging configuration: the Image Distance (ID), and the Object Distance (OD). The experiments performed on this system configuration were devised to quantify the Region of Interest (ROI), DOF, and LPCR for all practical combinations of ID and OD. The term practical here means that the physical quantities of ID and OD are small enough to fit in a transducer and the ROI is sized properly for viewing features on the skin surface. The physical significance of ID and OD is illustrated in Figure 7.17, and with respect to the experimental apparatus in Figure 7.18 and Figure 7.19.

Derivation of Minimum Image Distance

The goal of the first experiment was to derive the minimum ID for the combination of L1 and L2. L2 is specified as having a focal length of 8 mm which should correspond to the ID in Figure 7.18. However, initial tests indicated that this was not the case. The ID was derived experimentally by shining the LED directly into L2 at a distance of approximately

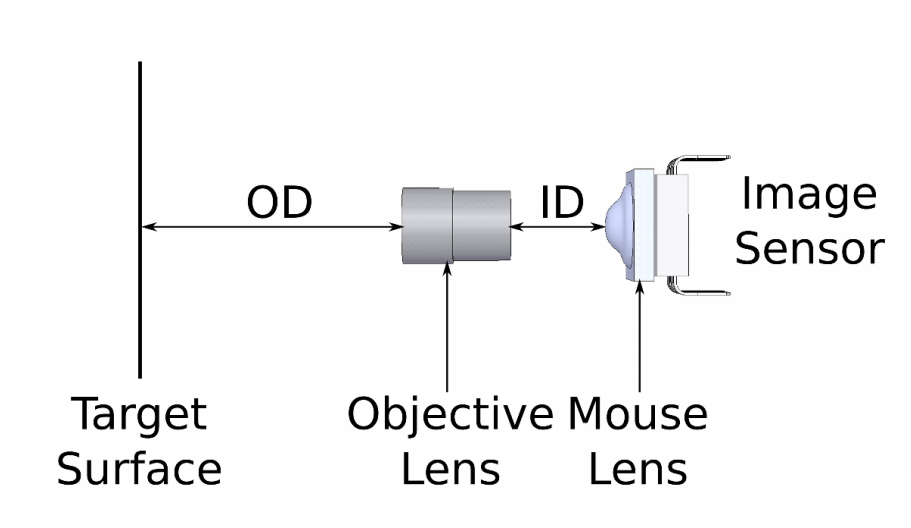


Figure 7.17: Graphical representation of ID and OD

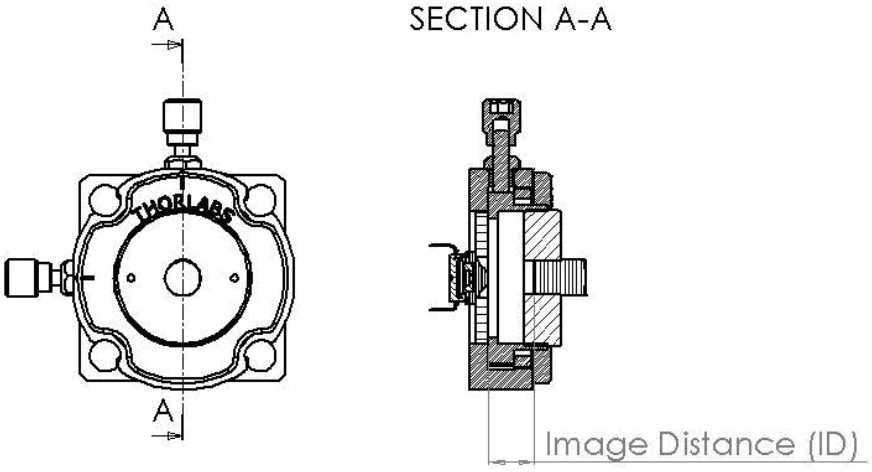


Figure 7.18: Graphical representation of ID with respect to the experimental apparatus

50 mm. A white piece of paper was taped to the back of A2 such that point of focus could be observed. The ID was then varied by threading E3 into and out of A2. The minimum ID was identified as the distance at which the point of light projected onto the paper was smallest. The minimum ID was found to be between 3.8 mm and 4 mm.

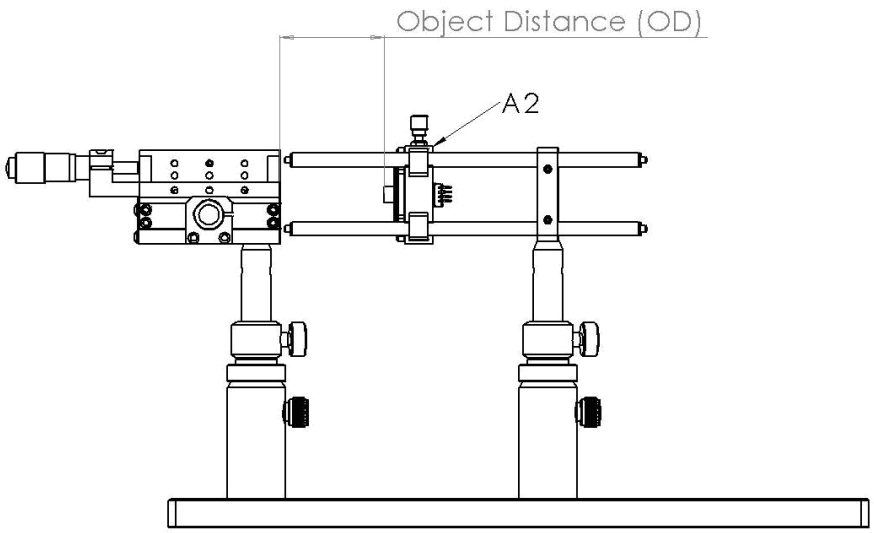


Figure 7.19: Graphical representation of OD with respect to the experimental apparatus

Derivation of Coarse Operational Ranges for ID and OD

The goal of this experiment was to define the broad operational ranges for ID and OD. There are many more possible combinations than there are functional combinations. This test was designed to rapidly identify ranges for each parameter, within which the system provided acceptable image quality. In these tests, "acceptable image quality" was a completely subjective measure. That is, focus was determined by visual inspection. Please refer back to Table 7.4 for definitions of the abbreviations used in the descriptions that follow.

The experimental apparatus is illustrated in Figure 7.19. E3 is threaded completely into A2 from the front. The IA is fully threaded into A2 from the back. The USAF target is mounted on TX. ID is controlled by threading E3 in or out, as illustrated in Figure 7.18. OD is controlled by sliding A2 on the rails, along the z-axis, as illustrated in Figure 7.19.

At the beginning of the test, TX and the y height are adjusted to optically center group 0, element 1 of the test pattern. The optical axis of the IA and E3 are then aligned by inspecting test images and adjusting the xy knobs on A2 such that all pixels contain useful data.

The experiment itself consisted of varying the ID in 1-turn increments, and for each

increment, varying the OD over a range of 60 mm in 5 mm increments. An image was recorded at each step. The ID was varied over the range from 5 turns (E2 threaded out 5 revolutions CCW from 0) to 12 turns and the OD was varied between 70 mm and 10 mm. Some variations were excluded when it was obvious that they would not yield useful data. The resulting images are illustrated in Figure 7.20, Figure 7.21, Figure 7.22, Figure 7.23, Figure 7.24, Figure 7.25, Figure 7.26, and Figure 7.27.

Figure 7.20: 5 turn direct imaging coarse image series (50 mm-70 mm)

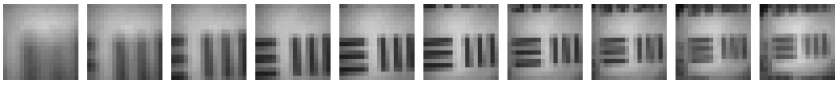


Figure 7.21: 6 turn direct imaging coarse image series (2 5mm-70 mm)

Figure 7.22: 7 turn direct imaging coarse image series (20 mm-70 mm)

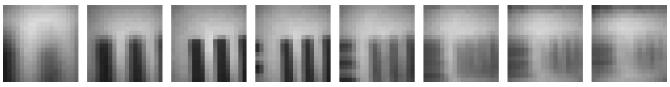


Figure 7.23: 8 turn direct imaging coarse image series (10 mm-40 mm)

Figure 7.24: 9 turn direct imaging coarse image series (10 mm-30 mm)

Figure 7.25: 10 turn direct imaging coarse image series (10 mm-30 mm)

Two clear trends are evident in the images. First, as the OD increases, so does the ROI. Second, as the ID increases the DOF decreases.

The lines in element 1, group 0 of the USAF 1951 test pattern are 0.5 mm wide. With this information it is possible to estimate the ROI at the ID where the image focus appeared

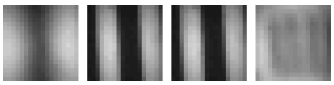


Figure 7.26: 11 turn direct imaging coarse image series (10 mm-25 mm)

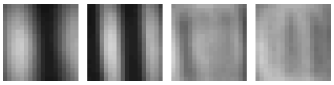


Figure 7.27: 12 turn direct imaging coarse image series (10 mm-25 mm)

optimal. Table 7.5 lists the number of turns, corresponding physical ID, and approximate ROI for each test series. Since one of the design objectives was to miniaturize the system, only the test runs that achieved focus at relatively short ODs were selected for further experiment. In addition, a relatively small ROI, on the order of 1 mm^2, was required to track on the skin surface. The tests that met these criteria were series 9, 10, 11, and 12.

Table 7.5: Summary of direct imaging experimental results

Series	ID (mm)	OD (mm)	ROI (mm^2)
5	7.33	60	6
6	7.96	50	5
7	8.60	35	4.5
8	9.23	30	3.6
9	9.87	20	2.5
10	10.50	15	1.8
11	11.14	15	1.8
12	11.77	15	1.8

Derivation of Fine Operational Ranges for ID and OD

This experiment built on the results of the previous experiment. ODs between 5 mm and 25 mm and IDs between 9 and 12 turns were studied in greater detail. The experimental apparatus was the same. However, in this experiment the OD was varied in 0.5 mm increments using TZ, instead of in 5 mm increments by moving A2.

One test series was run for each combination of ID (in number of turns) and nominal OD of interest. At the start of each series E3 was set to the desired ID. TZ was then set to 5 mm, which is half of its end-to-end travel. A2 was adjusted so that the distance from the target to the end of L2 was equal to the desired nominal OD. TZ was then set to 1 0mm (decreasing the OD by 5 mm) to begin the test.

The experiment itself consisted of moving the target away from L2 in 0.5 mm increments and acquiring an image at each step. For each image, the LPCR was calculated using (7.3). The ROI was also calculated for each image with sufficient focus. The DOF for each test series was calculated by estimating the range of OD over which the image remains in focus. This can only be done effectively by visual inspection.

The images acquired during the testing are presented below in Figures 7.28, 7.29, 7.30, and 7.31. The OD of each image increases from right to left and top to bottom.

Figure 7.32 illustrates the experimental relationship between OD and LPCR for each ID tested. The data indicate that contrast improves as the ID increases. Although all of the trials achieved good contrast, the highest contrast was achieved with the largest ID. However, the DOF was largest when the ID was smallest. Figure 7.33 illustrates the experimental relationship between OD and ROI for each ID tested. As is Figure 7.32, large values of ID perform better at small ODs. ROI and ID seem to have an inverse relationship, since the largest ROI was achieved in the trial with the smallest ID.

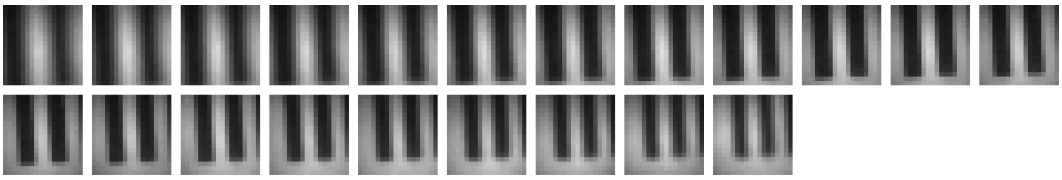


Figure 7.28: 9 turn direct imaging fine image series (15 mm-25 mm)

Figure 7.29: 10 turn direct imaging fine image series (15 mm-22 mm)

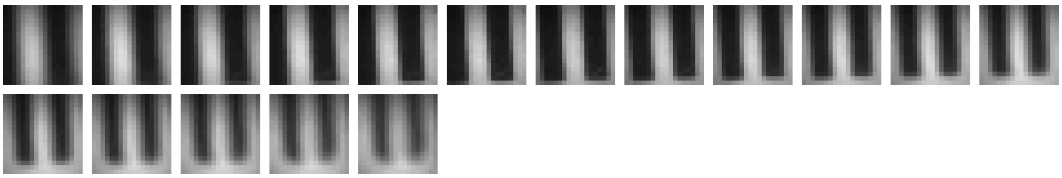


Figure 7.30: 11 turn direct imaging fine image series (12 mm-20 mm)

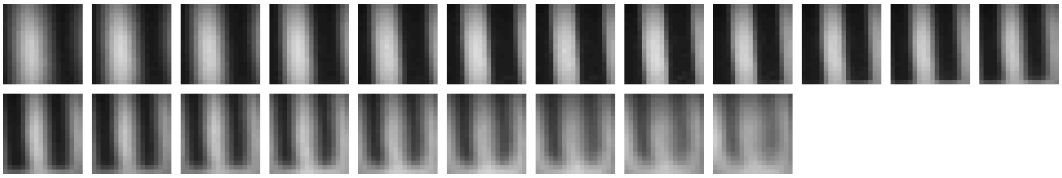


Figure 7.31: 12 turn direct imaging fine image series (10 mm-20 mm)

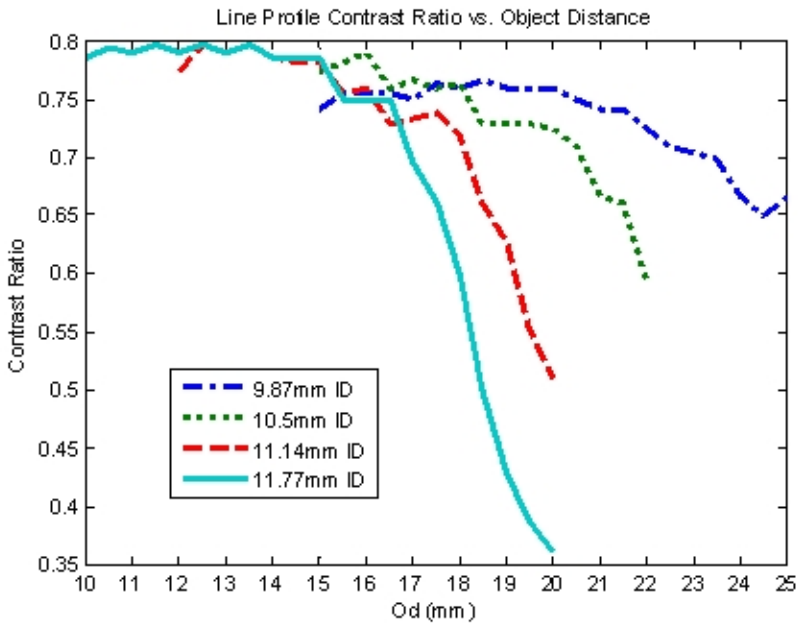


Figure 7.32: LPCR vs Object Distance (OD) for direct imaging system

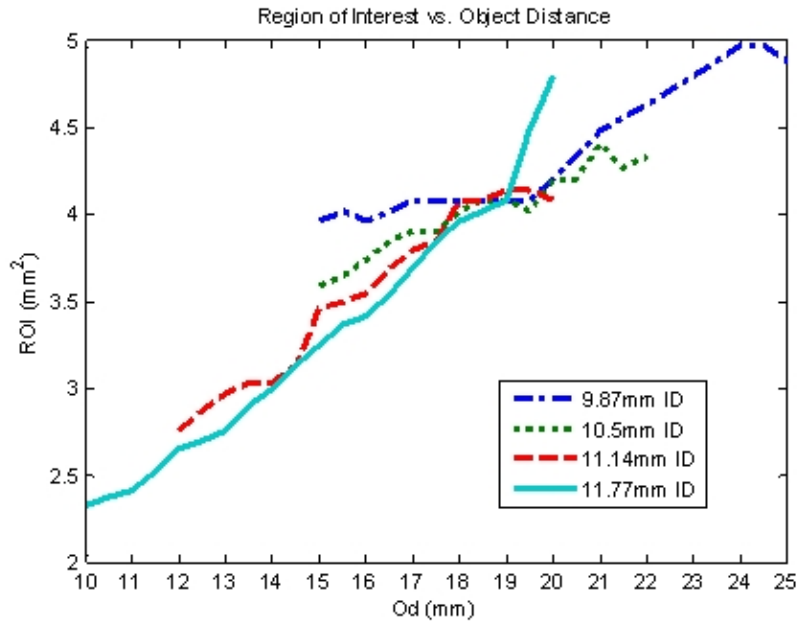


Figure 7.33: ROI vs. Object Distance (OD) for direct imaging system

7.4.2 Indirect Imaging with the Schott Fiber Bundle

The indirect imaging configuration is more complicated than the direct one. In addition to the ID and OD, the optimal spacing between the mouse lens and the end of the optical fiber needed to be determined. These tests were initially performed with the Schott fiber bundle and they were later repeated using the three rigid fibers from Edmunds Optical. The experimental apparatus was different in the latter tests, but the protocol remained consistent throughout.

Derivation of Imager-Fiber Spacing

The goal of this experiment was to investigate the spacing required between the fiber bundle's image surface and the mouse lens. Figure 7.34 illustrates this dimension. The imager-fiber spacing must be known and optimized prior to investigating the other unknowns in the system. If this dimension is incorrect, the images acquired will always be out of focus.

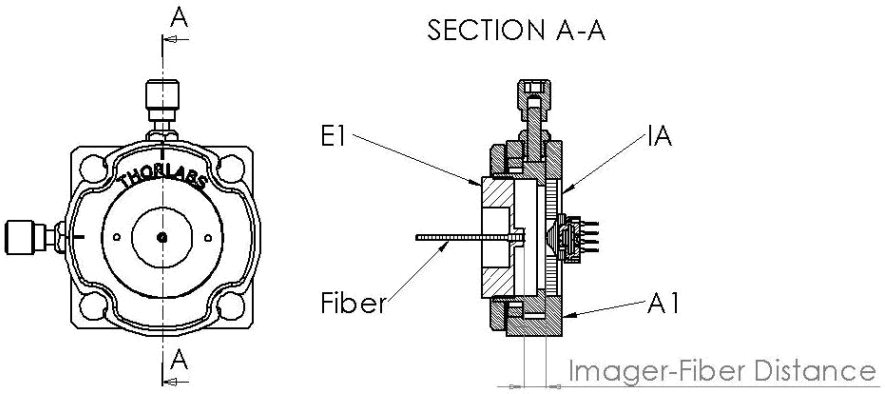


Figure 7.34: Graphical representation of imager-fiber distance

The experimental apparatus was similar to that used in the direct imaging experiments. The end of the fiber with the short sleeve was mounted in E1. The opposite end of the fiber was mounted in E2. A1 was used to control the alignment of the IA and E1. A2 was used to hold E2 stationary while the LED was shone directly into it.

E1 and the IA were both threaded into A1 completely. E1 was threaded out of A1 in

half rotation increments, each of which increased the imager-fiber distance by 0.3175 mm. An image was recorded at each distance and this process continued until the imager's entire DOF had been traversed. This determination was made by direct inspection of the images. The image sequence is presented in Figure 7.35.

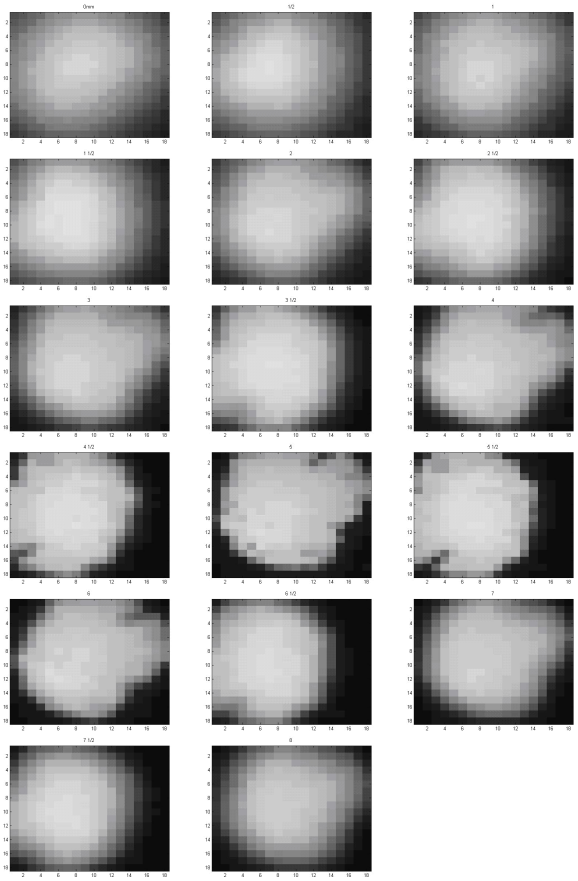


Figure 7.35: Image series acquired during the imager-fiber distance test

The focus is best around 5.5 turns, which corresponds to a distance of 3.4925 mm between L1 and the surface of the optical fiber. This is very close to 3.37 mm, which is the nominal distance from the lens surface to the object image, derived from information provided in the L1 datasheet. All images between 4.5 and 6 turns exhibit reasonable focus. This corresponds to 0.95 mm of travel and agrees well with the ±0.5 mm DOF, also specified in

the L1 datasheet. The agreement between the experimental result and the predicted result, based on the L1 specifications, helps confirm the validity of the overall test protocol.

A round, black region is apparent at the edge of all the images in Figure 7.35. It is caused by the diameter of the fiber. The ROI of the imager and L1 combination is specified as 1 mm^2 nominally. The diameter of the fiber bundle is also about 1 mm. The radial distance from the center of the image to the corner is $\frac{\sqrt{2}}{2}$ mm, while the radial distance to the outer diameter of the fiber bundle is 0.5 mm. This is illustrated in Figure 7.36. Simply put, the surface area of the fiber is smaller than the region of interest of the imager and L2 combination, at the specified image distance.

There is another important feature to note in these images and the ones to follow. The outline of the fiber is not round as one would expect and there are some dark spots protruding into the surface. After cleaning the fiber and inspecting the ends under a microscope it was determined that these features were the result of broken fibers in the bundle. These most likely occurred during handling. The remarkable flexibility of the Schott fiber bundle makes it very fragile and therefore very difficult to handle. Despite the damage, it was decided to continue with the testing in the hope that another fiber could be procured from Schott.

Derivation of Coarse Operational Ranges for ID and OD

This experiment was exactly the same as that described in Section 7.4.1. The goal of this experiment was to define the broad operational ranges for ID and OD. The experimental apparatus was also very similar to that illustrated in Figure 7.19, except that there were two adjusters, A1 and A2. A1 was used to align the imager assembly, IA, and the end of the fiber, housed in E1, as illustrated in Figure 7.34. This alignment was determined in the previous experiment and held fixed for the remaining experiments. A2 aligned the Elmo lens carrier, E3, with the other end of the fiber, housed in E2. E2 was threaded completely into A2 from the back. E3 was completely threaded into A2 from the front. The USAF target was mounted on TX. The ID was controlled by threading E2 in or out. The OD was controlled by sliding A2 on the rails, along the z-axis.

At the beginning of the test, TX and the y height were adjusted to optically center

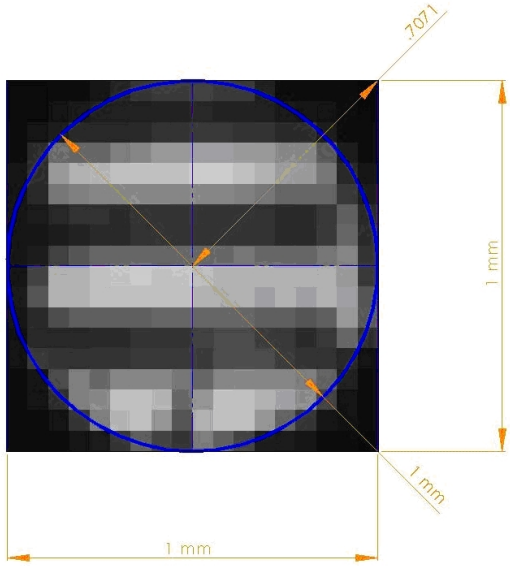


Figure 7.36: Detail drawing of the imager-fiber interface

group 0, element 1 of the test pattern. The optical axis of the E3 and E2 were then aligned by inspecting test images and adjusting the xy knobs on A2 such that all pixels contained useful data.

The experiment itself consisted of varying the ID in 1-turn increments and for each increment, varying the OD over a range of 60 mm in 5 mm increments. An image was recorded at each step. The ID was varied over the range from 5 turns (E3 threaded out 5 revolutions CCW from 0) to 12 turns and the OD was varied between 70 mm and 10 mm. Some variations were excluded when it was obvious that they would not yield useful data. The resulting images are illustrated in Figures 7.37, 7.38, 7.39, and Figure 7.40. The OD increases from left to right in the image sequences.

Figure 7.37: 1 turn, Schott fiber, coarse image series (30 mm-70 mm)

Figure 7.38: 2 Turn, Schott fiber, coarse image series (20 mm-70 mm)

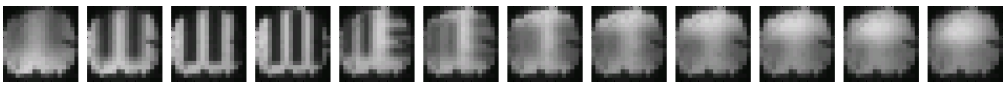


Figure 7.39: 3 Turn, Schott fiber, coarse image series (15 mm-70 mm)

Figure 7.40: 4 Turn, Schott fiber, coarse image series (10 mm-40 mm)

Derivation of Fine Operational Ranges for ID and OD

The procedure for this experiment was the same as that described in Section 7.4.1. The images from the previous experiment were inspected to identify operating ranges that should be investigated. The images recorded during the experiment are illustrated in Figures 7.41, 7.42, 7.43, 7.44, 7.45, and 7.46.

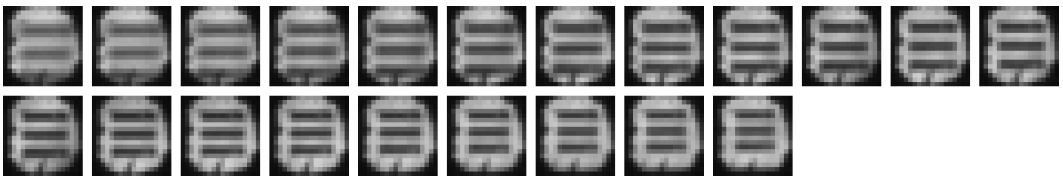


Figure 7.41: 3 Turn, Schott fiber, fine image series (20 mm-30 mm)

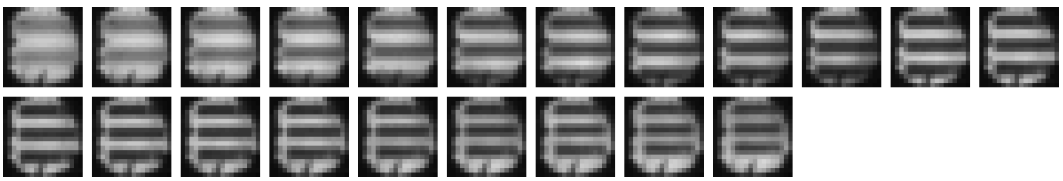


Figure 7.42: 4 Turn, Schott fiber, fine image series (15 mm-25 mm)

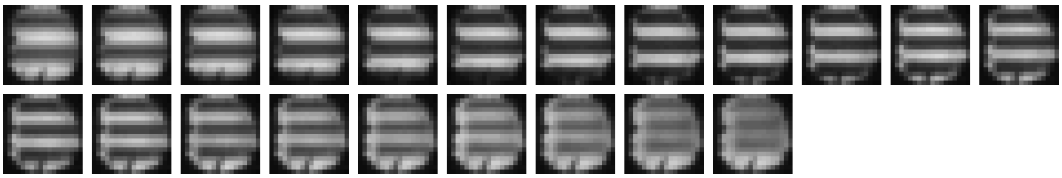


Figure 7.43: 5 Turn, Schott fiber, fine image series (15 mm-25 mm)

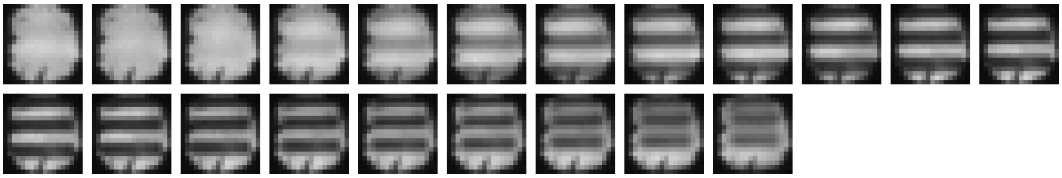


Figure 7.44: 5 Turn, Schott fiber, fine image series (10 mm-20 mm)

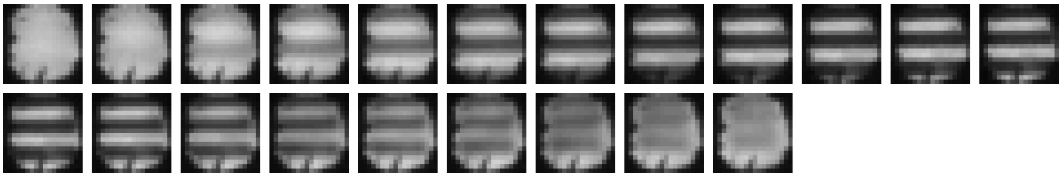


Figure 7.45: 6 Turn, Schott fiber, fine image series (10 mm-20 mm)

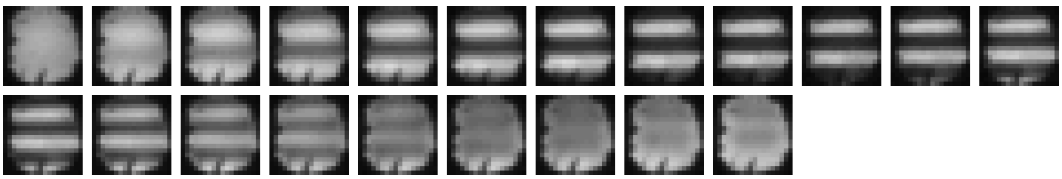


Figure 7.46: 7 Turn, Schott fiber, fine image series (10 mm-20 mm)

Figure 7.47 illustrates the experimental relationship between OD and LPCR for each ID tested. Like the results from the experiments on the direct imaging configuration, the data indicate that as the ID increases the maximum contrast ratio achieved in the experiment also increases. Conversely, the DOF decreases as the ID increases. Higher contrast ratios were achieved at smaller ODs.

Figure 7.48 illustrates the experimental relationship between OD and ROI for each ID tested. The data indicate that as the OD increases, so does the ROI. Likewise, for a fixed OD, increasing the ID also increases the ROI. This is exactly the expected behavior.

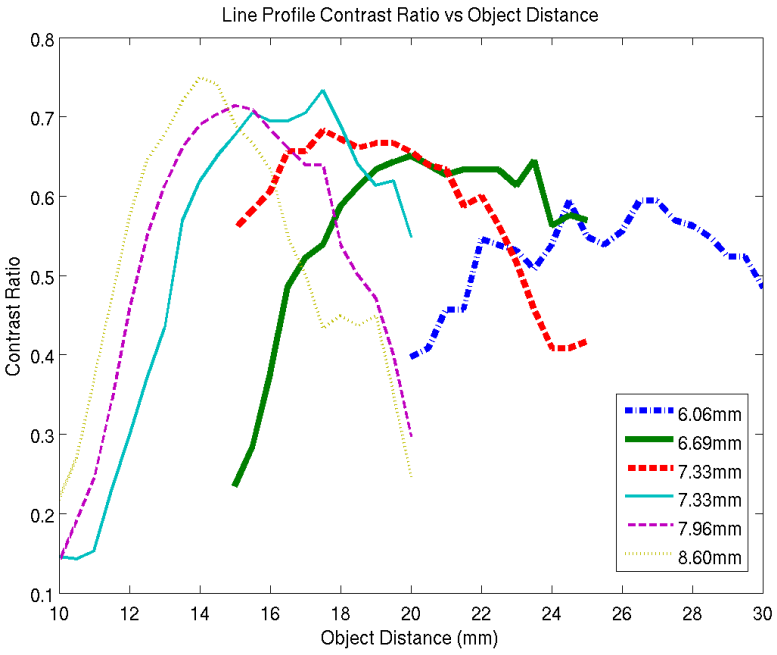


Figure 7.47: LPCR vs Object Distance for indirect imaging system

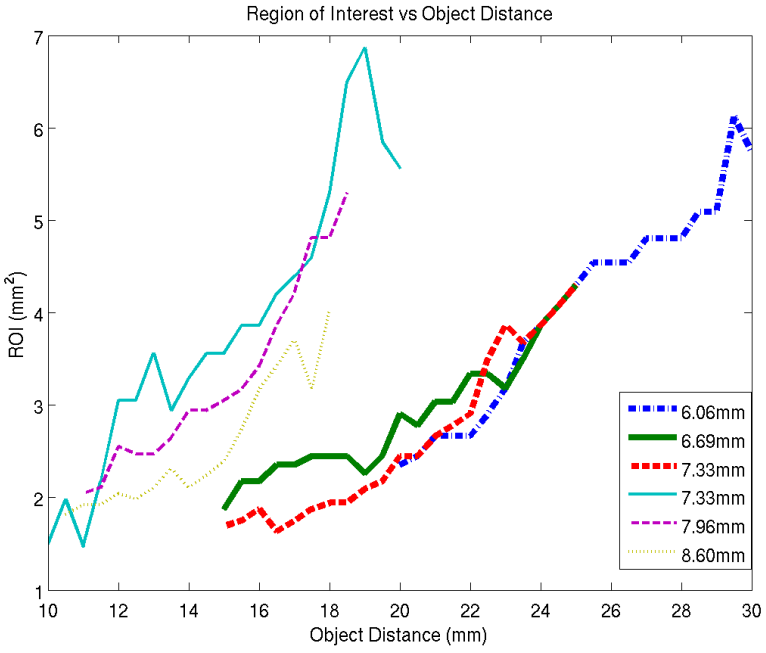


Figure 7.48: LPCR vs Object Distance for indirect imaging system

7.4.3 Indirect Imaging with the Edmunds Fiber Bundles

After the initial testing to quantify the parameters of the direct and indirect imaging systems was complete, the next step in the testing was to evaluate the tracking performance of the two systems. However, during the course of that testing it was discovered that the indirect imaging system with the Schott fiber bundle did not track well. The combination of the damage to the fiber bundle and the fact that the fiber diameter was not large enough to occupy all of the imager's field of view, as discussed in Section 7.4.2, were undoubtedly the causes.

The original Schott fiber bundle was provided free of charge because it was a quality reject. Schott did not have another to give us and because they do not have standard products, we were unable to procure another. After exploring possible alternatives it was decided to attempt to reevaluate the rigid fibers from Edmunds Optical. These types of fibers had been tested previously without success but based on the insights gained from the most recent round of tests, it was felt that they could be made to work. The downside to this course of action was that these types of fibers can only be bent by heating them to very high temperatures, which means they are more difficult to integrate into a prototype system. However, in a commercial system the rigid fiber would undoubtedly be the preferred choice, both because of their mechanical robustness and because they are approximately 100 times less expensive.

Three types of rigid fiber bundles were evaluated, all of which were one inch long and came from Edmunds Optical: a 0.062 in diameter 3,012 element bundle, a 0.125 in diameter 3,012 element bundle, and a 0.125 in diameter 50,419 element fiber bundle. Not all of the testing was repeated for these fibers. The data from the tests on the imager fiber spacing, presented in Section 7.4.2, remained valid for these fibers. In addition, the results on the previous coarse operational testing had provided enough insight into the general ranges of ID and OD. Based on those results the coarse range tests were not repeated.

All of the test procedures were exactly the same as those previously described in Section 7.4.2. There was a slight difference in the apparatus owing to the significant difference

between the Schott fiber and the Edmunds fibers. Because they were only an inch long, a single carrier was fabricated that completely contained the fiber. One was fabricated for the 0.062 in diameter fiber, as illustrated in Figure 7.15, and another was fabricated for the two 0.125 in diameter fibers, as illustrated in Figure 7.16. The two adjusters, A1 and A2, were threaded all the way onto the fiber carriers. The ID and imager fiber spacing could then be adjusted by threading E2 or the IA in and out, respectively. The axial alignment was performed exactly as previously described. The new apparatus had no effect on the process of manipulating the OD, as this was performed using TZ. The apparatus used to evaluate the indirect imaging configuration with the Edmunds fiber bundles is illustrated in Figure 7.49.

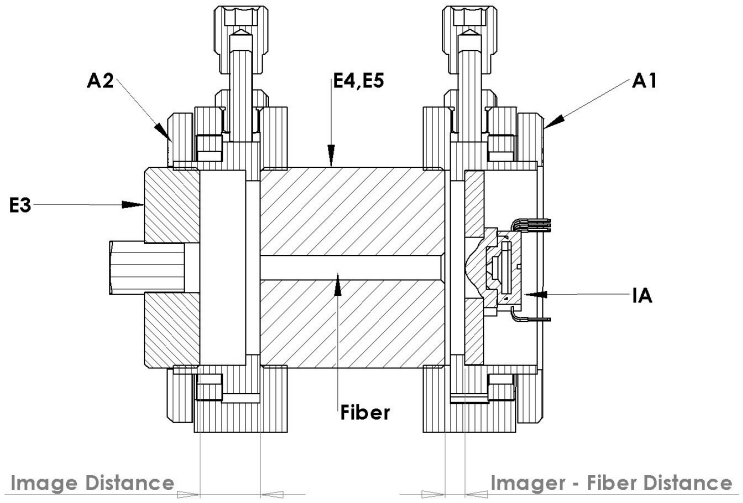


Figure 7.49: Graphical representation of Image Distance and imager-fiber distance for the Edmund's fibers

Since the procedures and apparatus have already been described in detail, the test data for all three of the fiber bundles will be presented in the next few pages. Figures 7.50, 7.51,7.52, 7.53, 7.54, 7.55, 7.56, and 7.57 illustrate the test images for the 0.062 in diameter fiber bundle. Figure 7.58 is the plot of LPCR versus OD for the 0.062 in diameter fiber and Figure 7.59 is the plot of ROI versus OD.

Figures 7.60, 7.61,7.62, 7.63, 7.64, 7.65, 7.66, and 7.67 illustrate the test images for the

0.125 in diameter standard resolution fiber bundle. Figure 7.68 is the plot of LPCR versus OD for the 0.125 in diameter standard resolution fiber and Figure 7.69 is the plot of ROI versus OD.

Figures 7.70, 7.71,7.72, 7.73, 7.74, 7.75, 7.76, and 7.77 illustrate the test images for the .125 in diameter high resolution fiber bundle. Figure 7.78 is the plot of LPCR versus OD for the .125 in diameter high resolution fiber and Figure 7.79 is the plot of ROI versus OD. The results discussed as a group at the end of this section.

Edmunds 0.062 in Diameter, 3,012 Element, Fiber Bundle

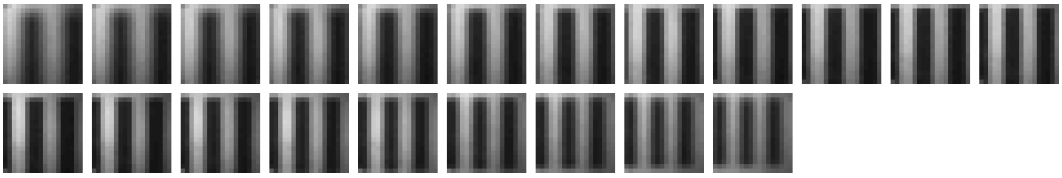


Figure 7.50: 0.062 in diameter, 3,012 element fiber 4 turn image series

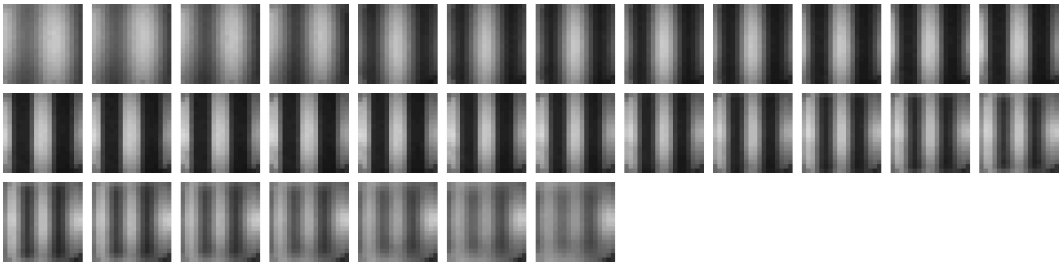


Figure 7.51: 0.062 in diameter, 3,012 element fiber 5 turn image series

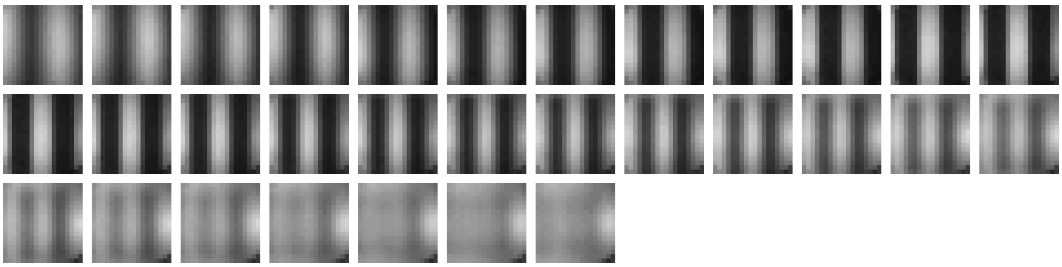


Figure 7.52: 0.062 in diameter, 3,012 element fiber 6 turn image series

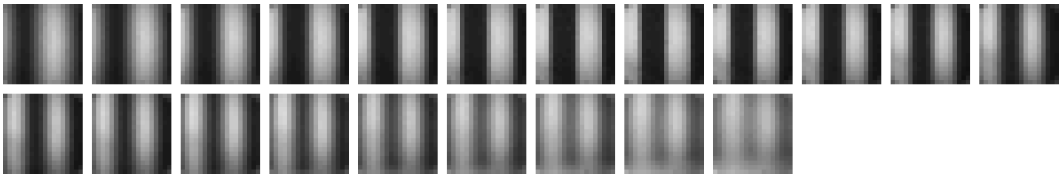


Figure 7.53: 0.062 in diameter, 3,012 element fiber 7 turn image series

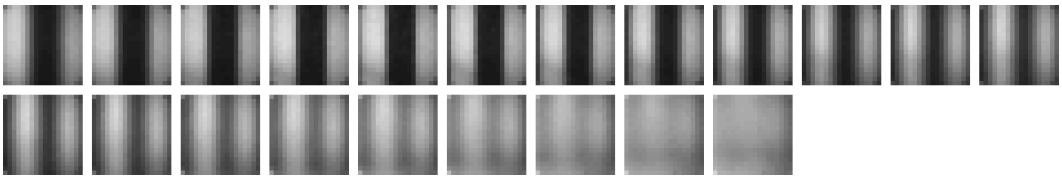


Figure 7.54: 0.062 in diameter, 3,012 element fiber 8 turn image series

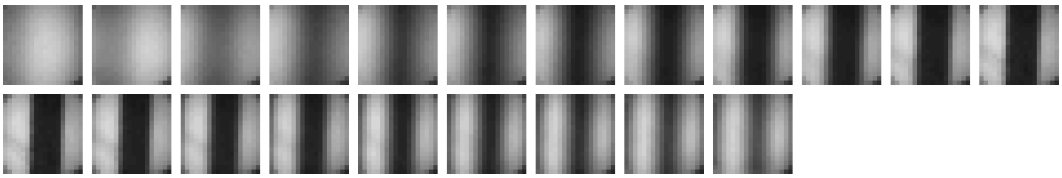


Figure 7.55: 0.062 in diameter, 3,012 element fiber 9 turn image series

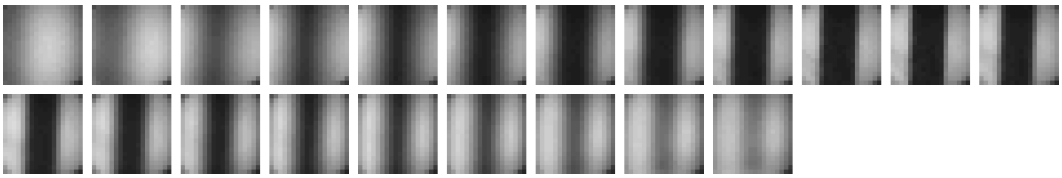


Figure 7.56: 0.062 in diameter, 3,012 element fiber 10 turn image series

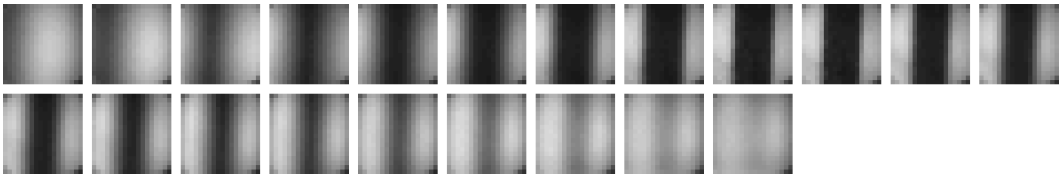


Figure 7.57: 0.062 in diameter, 3,012 element fiber 11 turn image series

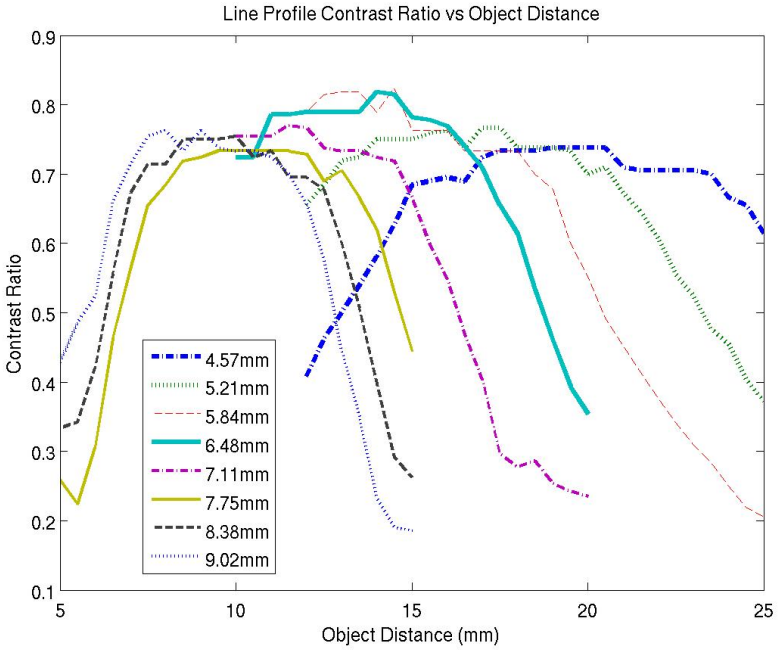


Figure 7.58: LPCR vs Object Distance for indirect imaging system with Edmunds 0.062 in diameter fiber

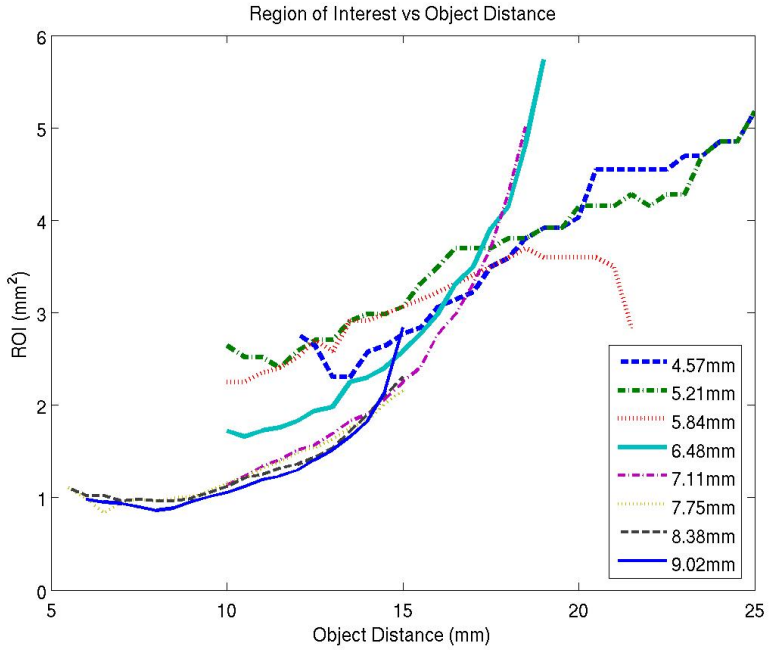


Figure 7.59: ROI vs. OD for indirect imaging system with Edmunds 0.062 in diameter fiber

Edmunds 0.125 in Diameter, 3,012 Element, Fiber Bundle

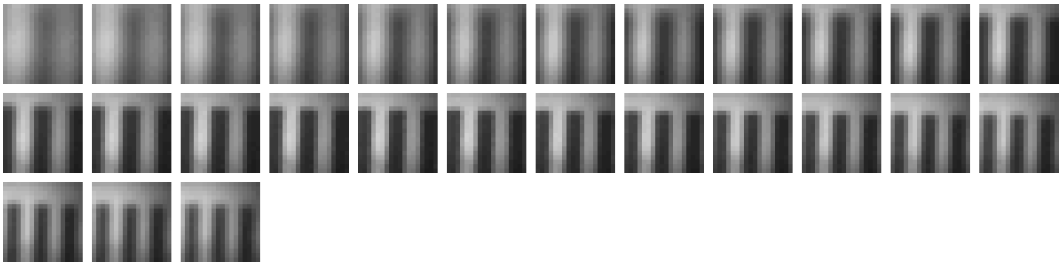


Figure 7.60: 0.125 in diameter, 3,012 element fiber 4 turn image series

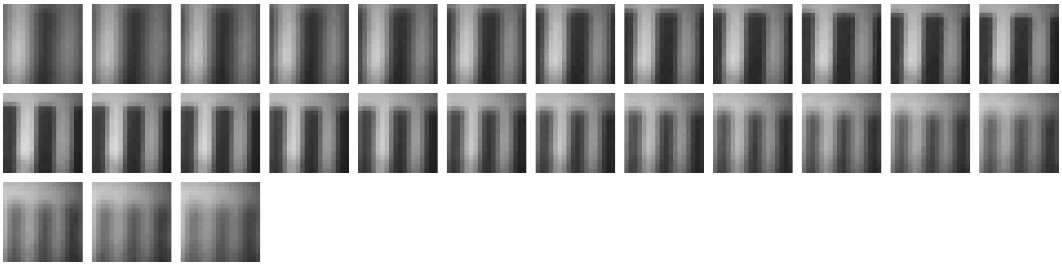


Figure 7.61: 0.125 in diameter, 3,012 element fiber 5 turn image series

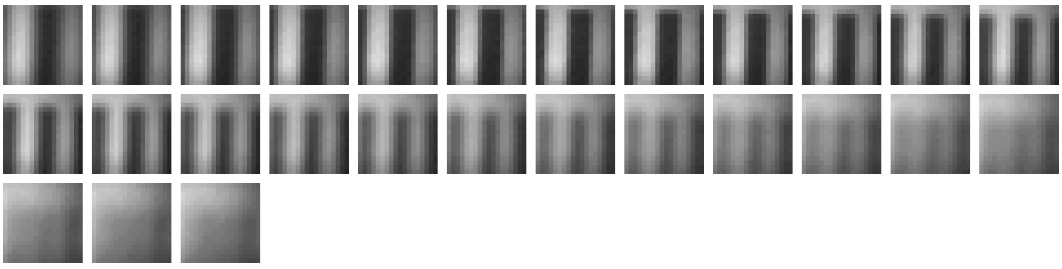


Figure 7.62: 0.125 in diameter, 3,012 element fiber 6 turn image series

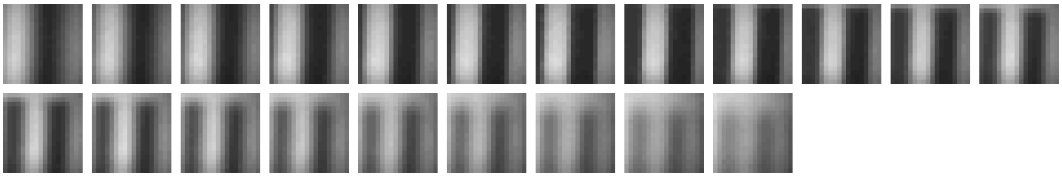


Figure 7.63: 0.125 in diameter, 3,012 element fiber 7 turn image series

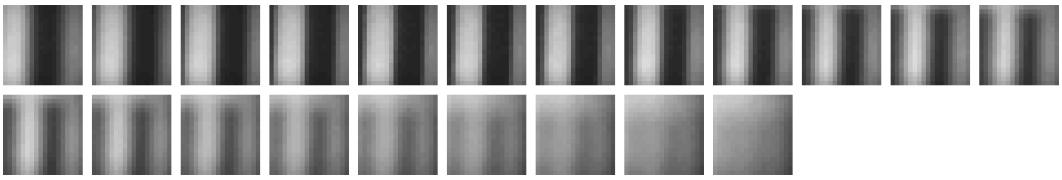


Figure 7.64: 0.125 in diameter, 3,012 element fiber 8 turn image series

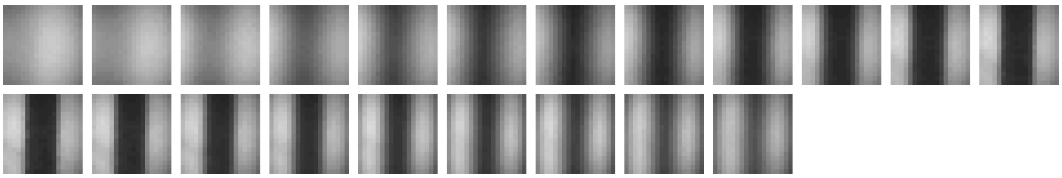


Figure 7.65: 0.125 in diameter, 3,012 element fiber 9 turn image series

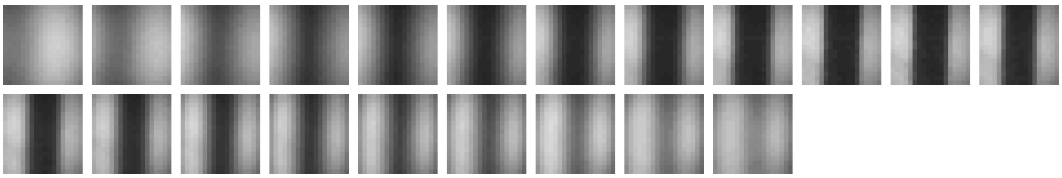


Figure 7.66: 0.125 in diameter, 3,012 element fiber 10 turn image series

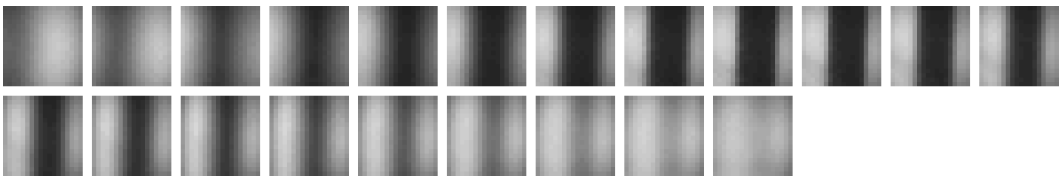


Figure 7.67: 0.125 in diameter, 3,012 element fiber 11 turn image series

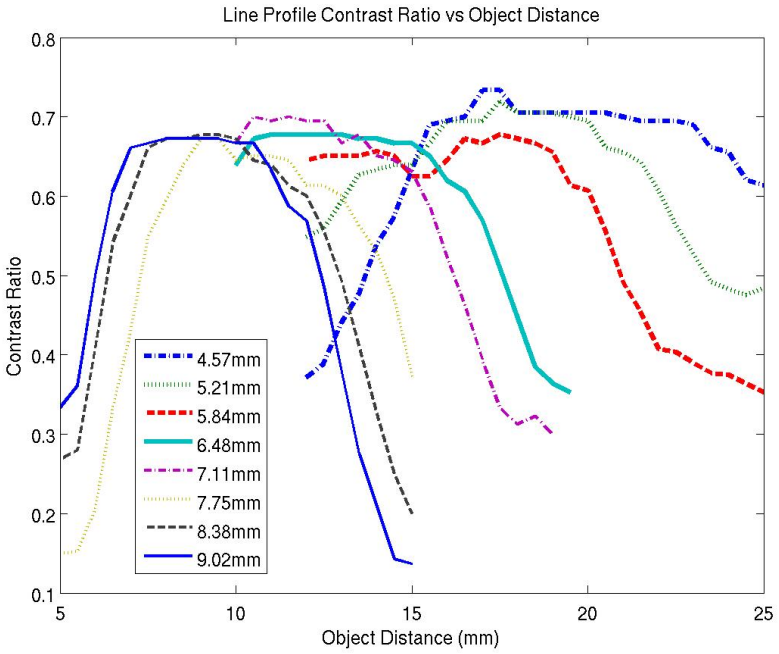


Figure 7.68: LPCR vs Object Distance for indirect imaging system with Edmunds 0.125 in diameter, 3,012 element fiber

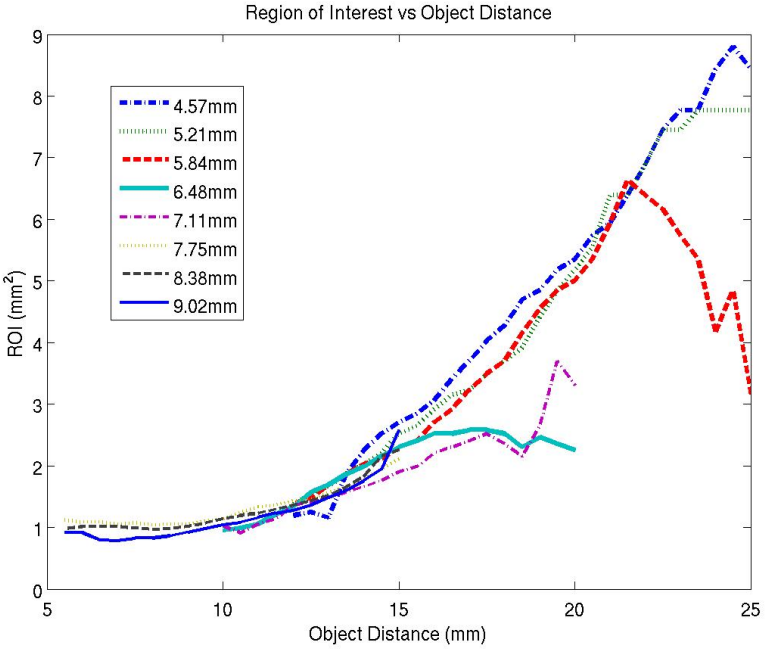


Figure 7.69: ROI vs. OD for indirect imaging system with Edmunds 0.125 in diameter, 3,012 element fiber

Edmunds 0.125 in Diameter, 50,419 Element, Fiber Bundle

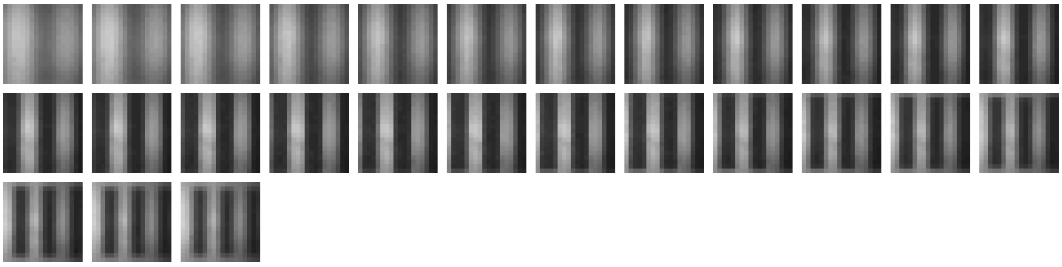


Figure 7.70: 0.125 in diameter, 50,419 element fiber 4 turn image series

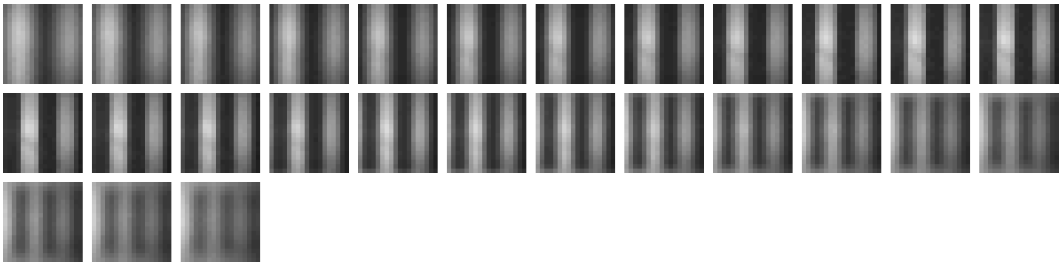


Figure 7.71: 0.125 in diameter, 50,419 element fiber 5 turn image series

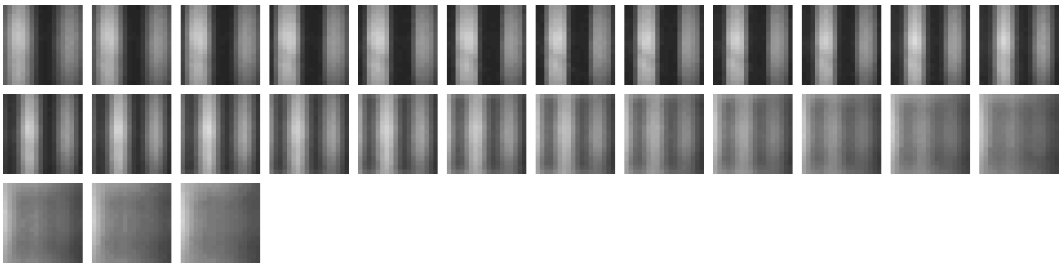


Figure 7.72: 0.125 in diameter, 50,419 element fiber 6 turn image series

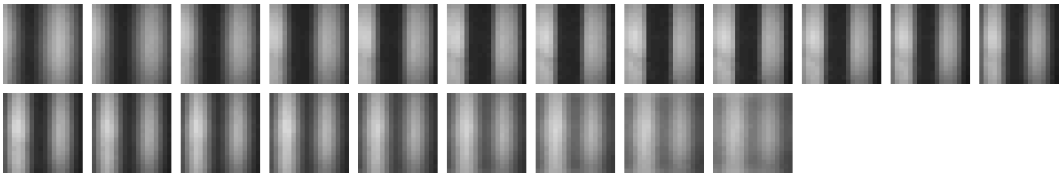


Figure 7.73: 0.125 in diameter, 50,419 element fiber 7 turn image series

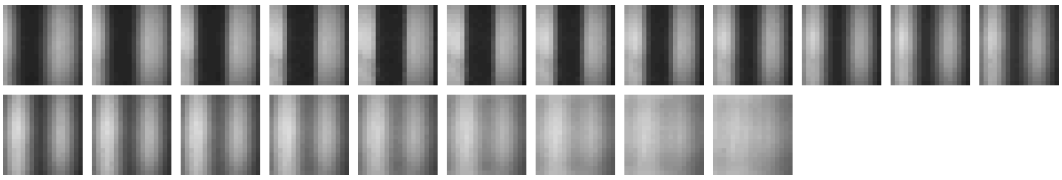


Figure 7.74: 0.125 in diameter, 50,419 element fiber 8 turn image series

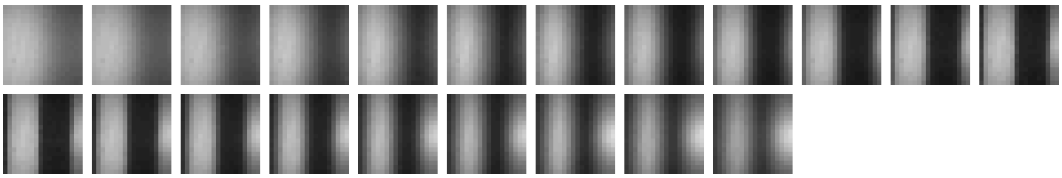


Figure 7.75: 0.125 in diameter, 50,419 element fiber 9 turn image series

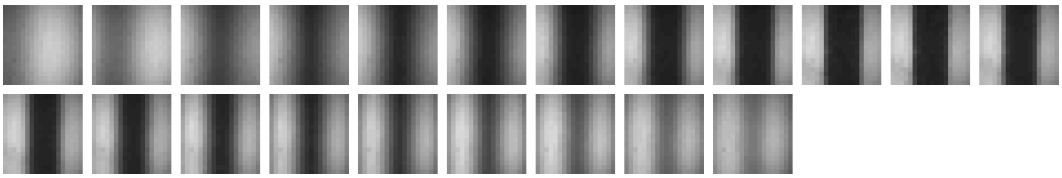


Figure 7.76: 0.125 in diameter, 50,419 element fiber 10 turn image series

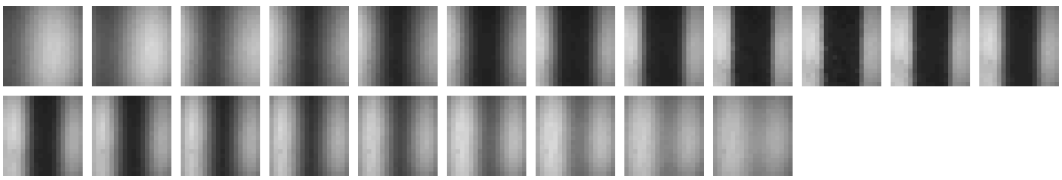


Figure 7.77: 0.125 in diameter, 50,419 element fiber 11 turn image series

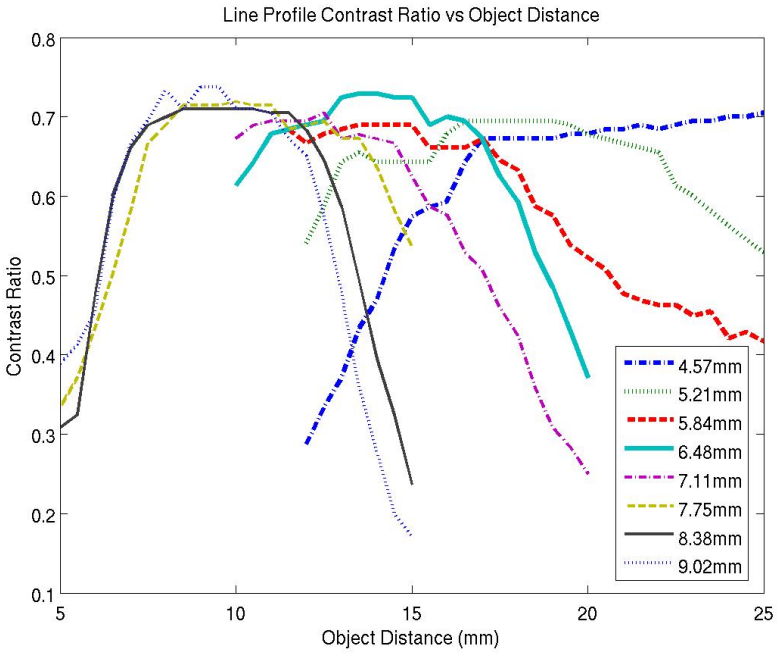


Figure 7.78: LPCR vs Object Distance for indirect imaging system with Edmunds 0.125 in diameter, 50,419 element fiber

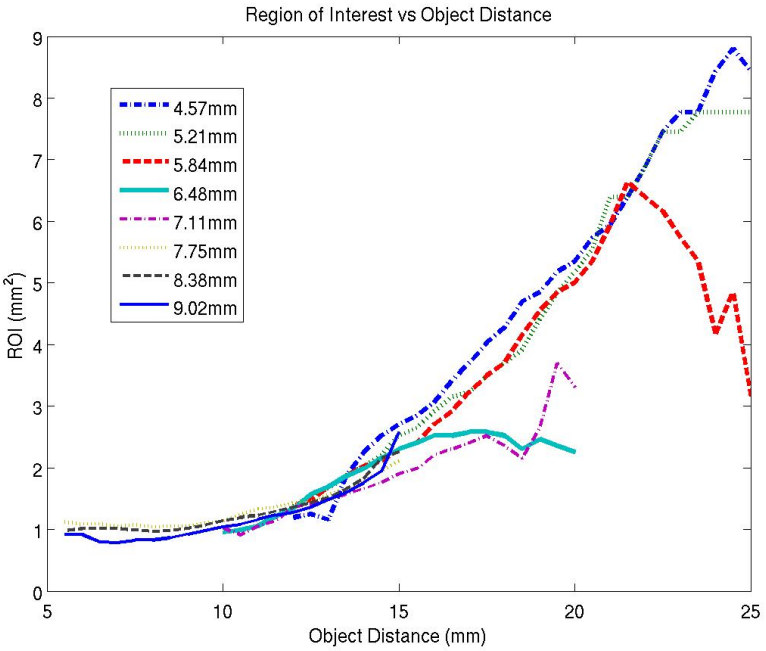


Figure 7.79: ROI vs. OD for indirect imaging system with Edmunds 0.125 in diameter, 50,419 element fiber

The data indicate that the three fibers have nearly identical performance. All of the fibers achieve a LPCR of about 0.7. This is slightly lower than was achieved with the direct imaging configuration, but still produces images with subjectively high contrast. The same trends that were observed in the direct imaging configuration test results are visible in this data. Figures 7.58, 7.68, and 7.78 indicate that DOF and ID have an inverse relationship. Figures 7.59, 7.69, and 7.79 indicate that ID is also inversely related to ROI. The inverse relationship between DOF and ID is desirable because a large DOF is an advantage when tracking on rough surfaces and a small ID will help keep the prototype optical system small. The relationship between ID and ROI is more problematic. A ROI between 1.5 mm and 2 mm is desirable but that limits the ID to a range of approximately 6.5 mm to 9mm. Ultimately, the choice of ROI should be governed by the characteristics of the tracking surface. There must be enough features within the ROI for the sensor to track reliably. The tests described in the following section deal with this issue.

7.4.4 Tracking Experiments

This section describes a set of experiments that were designed to test the tracking accuracy and repeatability of the direct tracking configuration. It was clear from the preceding test results that the direct imaging configuration was achieving acceptable contrast in the range of IDs and ODs that were required to fit it into a transducer housing. This test was meant to prove that good contrast translated into good tracking performance. Another purpose of this test was to derive the factor necessary to scale the raw counts output by the sensor into a physical quantity, in this case mm, for different operating points. Due to time constraints, this test was never performed on the indirect imaging configuration. In the previous tests, the direct imaging configuration achieved contrast ratios of about 0.8 in the range of ID and OD that meet the space requirements. In that same range, the indirect imaging configuration achieved contrast ratios of about 0.7. Although not identical, the values are close enough that the tracking performance of the direct imaging configuration should also apply to the indirect imaging configuration.

The experimental apparatus was similar to that described in section 7.4.1. Figures 7.32

and 7.33 were inspected and for each ID, 3 ODs were selected for evaluation. The three ODs were selected to correspond to the beginning, middle, and end of the range over which the system demonstrated good contrast for that particular operating point. This was determined by examining the LPCR graph as well as the raw images. It was important to reference the raw images because there are some cases where the contrast value is high but the actual focus is poor. This resulted in 12 distinct experiments.

The tracking target was a piece of leather which was meant to simulate the skin surface. It was attached to the front face of TZ. At the beginning of each trial the apparatus was adjusted to the desired ID and OD. Sample images were then taken to verify that the system was in focus. Then the Surface Quality (SQUAL) value was read out of the image sensor and recorded. SQUAL is a value that is calculated internally by the ADNS-2610 that represents the number of identifiable features that are visible to the sensor. The value of SQUAL ranges from 0 to 128, with higher values corresponding to better tracking. The test itself consisted of moving TX forward 10mm and then back 10 mm, in 1 mm increments. After each increment the measured distance, in raw counts, was read out of the image sensor and recorded. For each of the 12 experiments 5 trials were performed.

As a control experiment, a real optical mouse was modified so that it could be rigidly mounted to the test apparatus. The bottom of the mouse was brought into contact with the leather strip so that the mouse's optics would be in focus. The SQUAL value was read out of the image sensor and then 5 trials were performed as outlined in the previous paragraph. The experiment allowed a direct comparison to be made between the accuracy and repeatability of the image sensor when used as it was designed to be, with that of the image sensor when use in our direct imaging configuration.

The SQUAL value read out of the image sensor in the optical mouse configuration was 83. 83 is a good value and means that the image sensor was able to identify a large number of trackable features. Table 7.6 lists the tracking result for each trial. The mean is the average number of counts measured for each 1mm increment. The STD is the standard deviation of the number of counts recorded per increment over each trial. It is evident that the image sensor is quite capable of tracking on the leather sample when used in the standard optical

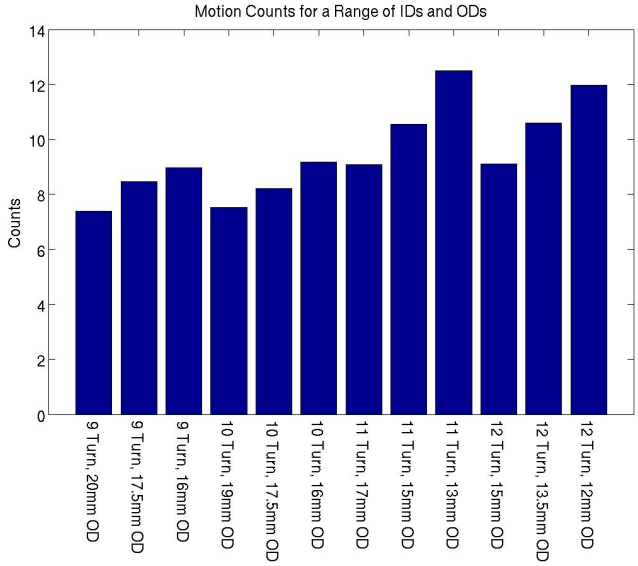
mouse configuration. The mean was nearly identical over the 5 trials which demonstrates that there is good repeatability. Each trial has a standard deviation of between 1.38 and 1.53 which shows that the tracking result was also very repeatable between individual increments. It is important to note that, because the motion was performed manually, there was some variation from increment to increment. Given that each count was approximately equal to 0.054 mm, the STD result seems very reasonable.

Table 7.6: Tracking performance and repeatability of a standard optical mouse on leather sample

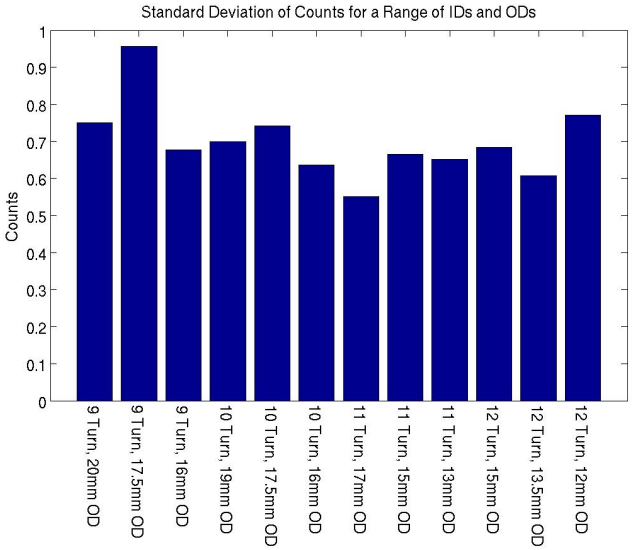
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Mean (cnt/mm)	19.25	19.24	19.18	19.20	19.20
STD (cnt/mm)	1.53	1.38	1.39	1.53	1.53

Figure 7.80(a) illustrates the experimental results for all of the tests performed on the direct imaging configuration. The values represented in the graph are the average number of counts per increment averaged over all 5 trials. It is clear that overall, decreasing OD increases the number of counts registered. This corresponds well with the previous experimental results, where it was shown that the area visible to the image sensor decreases with OD. The physical explanation is that the number of counts registered is relative to the size of the viewable area. When the viewable area is 1 mm^2, a feature that moves 1 mm appears to cross the entire field of view. When the viewable area is 4 mm^2 the same physical motion only crosses half of the field of view. Since the algorithm running in the sensor assumes that the ID and OD are fixed, as they would be in an optical mouse, varying these dimensions results in differences in the perceived motion, even when the absolute motion is constant.

Figure 7.80(b) illustrates the standard deviation of the number of counts for all test performed. The value represented by each bar is average over all trials of the standard deviation of each trial individually. There is no apparent correlation between the standard deviation and either the ID or the OD. This is the expected result and it indicates a high degree of repeatability.



(a) Average counts per increment for the direct imaging configuration



(b) Standard deviation of counts per increment for the direct imaging configuration

Figure 7.80: Direct configuration tracking performance

7.5 Final Optical System Implementation

In order to build a working prototype of the entire 3D ultrasound system it was necessary to design and fabricate both the direct and indirect versions of the optical tracking system

in a form that could be attached to a transducer. This required that they be as small as possible and rugged enough to protect the optical components. The testing described earlier in this section provided the information required to set the critical optical dimensions of the assemblies. The rest of the dimensions were then determined. Size was the primary driving factor but other considerations included ease of assembly and manufacturability.

7.5.1 Direct Imaging Configuration

Figure 7.81 is an exploded view of the entire direct imaging assembly. The housing was machined from aluminum to have an ID of 11.55 mm and OD of 15 mm. Aluminum was chosen because it is easy to machine and can be machined to very tight tolerances, unlike many plastics. The Elmo lens is inserted from the bottom and the LED is inserted from the side. Because we had only one Elmo lens and it needed to be tested in several assemblies it was held in place with Room Temperature Vulcanizing (RTV) silicone adhesive. There are features in the housing that match the flanges in both the lens and the LED to hold them in the correct position. The face into which the LED is inserted is designed to put the LED at a 50° angle with the tracking surface. This is the angle that was identified as optimal by Irene Gouverneur in [17]. Not pictured in the diagram is the small plastic window that fits in the relieved area at the base of the housing. The window is necessary to keep ultrasound transmission gel from fouling the lens. Both polycarbonate and acrylic were tested and found to work. The window was held in place with RTV silicone because it had to be easily removable and because of the excellent sealing characteristics of RTV. The entire lower face of the housing is rounded along the edge to keep it from catching on the surface being scanned.

As illustrated in Figure 7.82, the standard HDNS-2100 optical mouse lens had to be machined to make it fit the assembly. The finished dimension is approximately 5 mm square, to match the top of the housing. The top of the housing has a 1mm chamfer cut at 45°. The chamfer matches the contour of the lens and forces it to be centered with the optical axis of the housing. The 8 pins of the ADNS-2610 optical mouse sensor are carefully bent to point up instead of down. This allows a small PCB, not pictured here, to

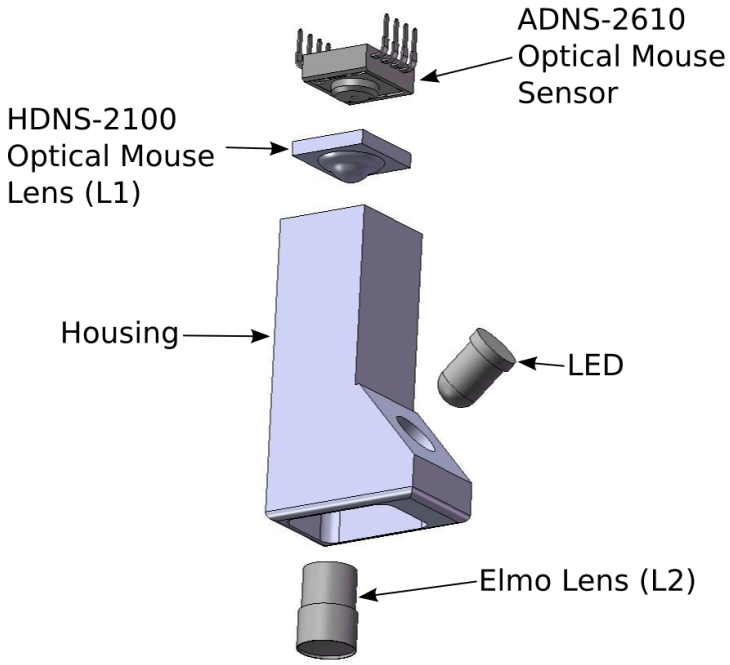


Figure 7.81: Exploded view of the prototype direct tracking assembly

be mounted on top of the sensor. this arrangement makes the assembly significantly smaller than if the sensor was mounted in its intended orientation. The sensor, mouse lens, and LED were held in place with epoxy.

7.5.2 Indirect Imaging Configuration

Figure 7.83 is an exploded view of the entire indirect imaging assembly. The main housing is very similar to that of the direct imaging assembly. It is machined in aluminum, the LED

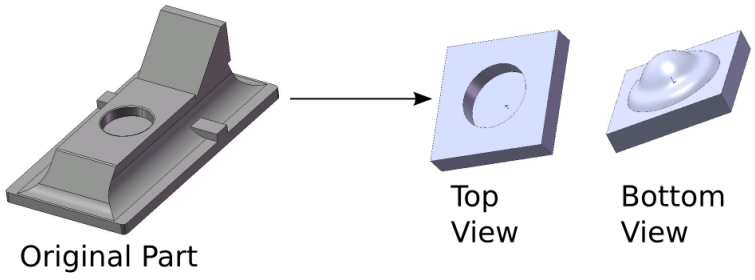


Figure 7.82: HDNS-2100 lens before and after modification

illumination angle is 50°, and the lower edge is rounded to keep it from catching on the scanning surface. Unlike the main housing of the direct imaging configuration, the main housing of the indirect imaging assembly was designed to be adjustable. The optical fiber is mounted inside the fiber carrier. The exterior of the fiber carrier and the interior of the main housing each have 40 Threads Per Inch (TPI) threads that allow the ID to be adjusted. The Elmo lens is glued into the lens carrier with RTV silicon. The lens carrier also has 40 TPI threads, which allow the OD to be adjusted. Mounting the optical fiber inside a carrier allows all three of the optical fibers that were evaluated in this chapter to be tested in the prototype tracking system without having to machine a complete housing for each one. Instead, a separate carrier was machined for each fiber and they could be swapped without even removing the housing from the probe.

The upper housing is where the ADNS-2610 optical tracking sensor and HDNS-2100 mouse lens mount. The upper housing does not attach rigidly to the fiber carrier because the fiber carrier rotates when it is adjusted. In order to allow the x and y axes of the sensor to align with the x and y axes of the probe, the upper housing must be allowed to rotate freely. The chamfer at the top of the fiber carrier matches the one in the bottom of the upper housing. The purpose of this feature is twofold. First, it forces the optical axes of the two parts to self align. Second, it sets the correct spacing between the end of the optical fiber and mouse lens. Once all of the adjustments were made, the upper housing was lightly glued to the fiber carrier to hold it in place. Like the direct configuration, the top surface of the upper housing also has a chamfer, which aligns the optical axis of the HDNS-2100 mouse lens.

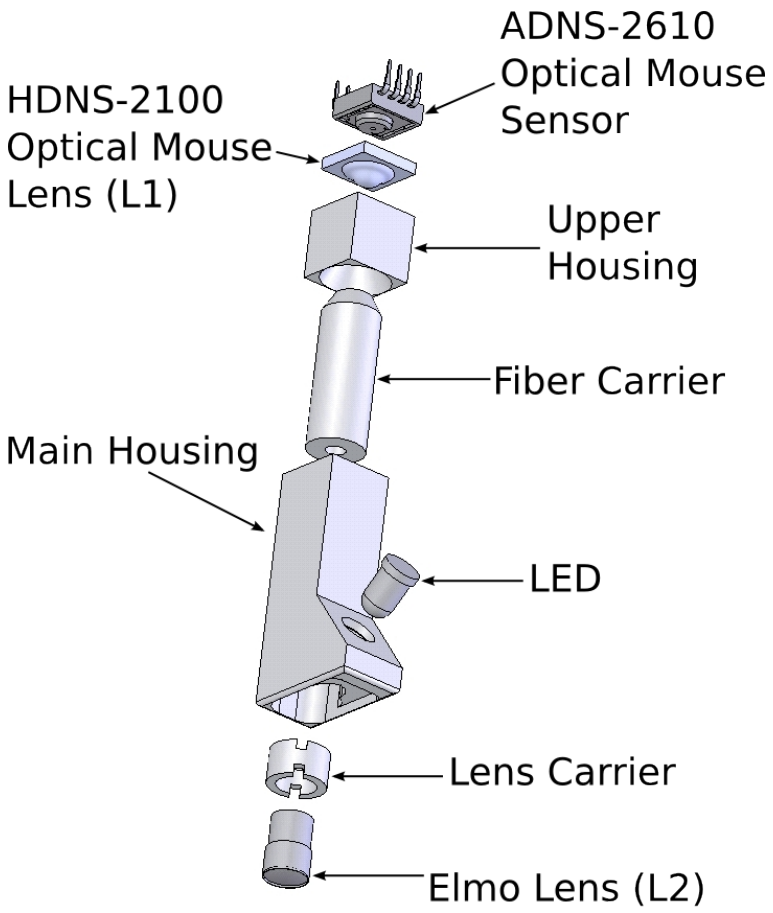


Figure 7.83: Exploded view of the prototype indirect tracking assembly

Chapter 8

Inertial Navigation

The inertial navigation algorithm tracks the real world position and orientation of the ultrasound transducer based on inputs from the sensors. Our tracking system is a modified version of what is traditionally called a strapdown Inertial Navigation System (INS). A traditional strapdown INS uses a set of 3 orthogonal accelerometers and 3 orthogonal gyroscopes to track position and orientation over time. The term "strapdown" refers to the fact that the sensor package is rigidly attached to the object being tracked, in contrast with other implementations such as stable platform systems.

Inertial navigation relies on the laws of classical mechanics. When a force acts on a body an acceleration results. This acceleration can be measured using an accelerometer and successively integrated to calculate change in velocity and position. In order to navigate using the acceleration measurements, it is necessary to keep track of the direction in which the accelerometers are pointing. The body's rotational motion can be measured using gyroscopes and this information is used to keep track of the orientation of the accelerometers. The orientation information can then be used to resolve the accelerations, which are measured in the body's inertial reference frame, into the real world coordinate system, before they are integrated. This process is illustrated in Figure 8.1. The real world coordinate system is represented by the set of axes labeled x y, the body being tracked is the gray box, and the body's inertial reference frame is represented by the set of axes labeled x' y'. Drawings (a)

and (b) represent the simple case, where the body's axes are perfectly aligned with the real world axes. A force acting along the body's x-axis causes the body to move from (x_0, y_0) to (x_1, y_1) in the real world. If a navigation system reported the position of the body in (b) as the second integral of the acceleration measured along the body's x-axis, it would be correct in this case. However, as illustrated in (c) and (d), this is not generally the case. Displacement resulting from a force acting only along the body's x-axis is expressed as displacement along both the x and y-axis in the real word coordinate system.

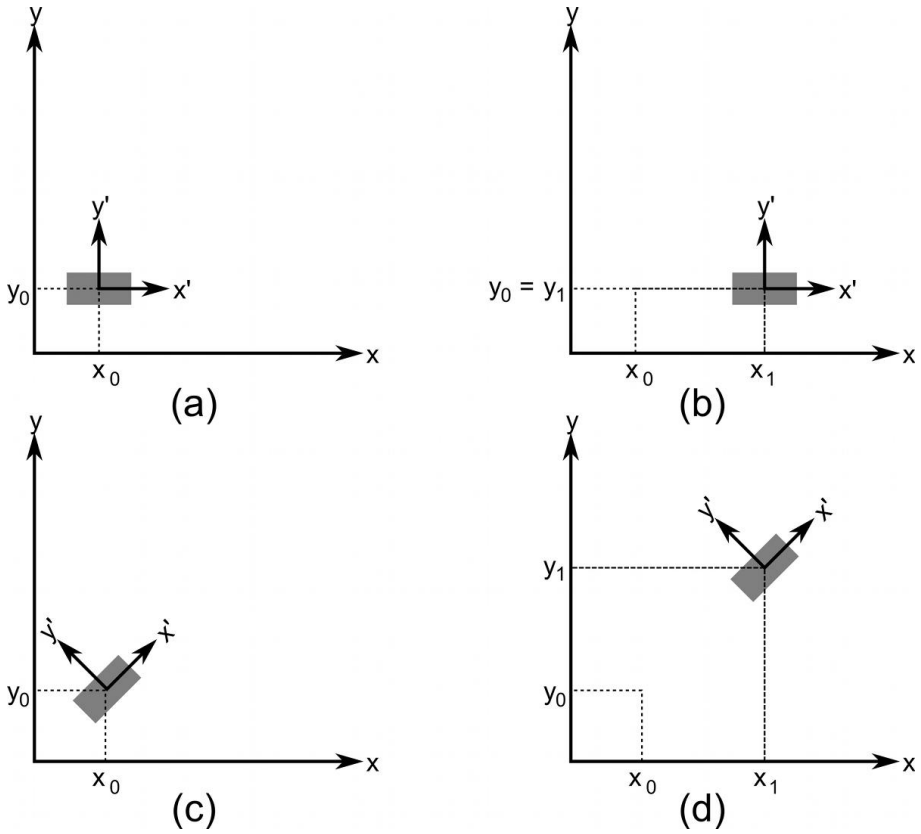


Figure 8.1: A simple inertial navigation example. In (a) and (b), a force acting along the x'-axis causes acceleration only along the x-axis. In (c) and (d), a force acting along the x'-axis causes acceleration along both the x and y axes.

8.1 Reference Frames

The above explanation demonstrates the need to define the various coordinate systems, or reference frames, that an INS must deal with. The reference frames described here are those necessary to understand how the prototype tracking system's navigation algorithm functions. In a typical INS, such as one would find in an aircraft, there are several others.

The inertial reference frame is the coordinate system that the sensor readings are expressed in. It is also known as the body reference frame because it remains fixed with respect to the body being tracked. The yaw, pitch, and roll angles are defined relative to the body reference frame around the z, y, and x axes, respectively. Figure 8.2 illustrates an example of a body reference frame.

The navigation reference frame is the coordinate system that the sensor measurements must be resolved in. In a traditional INS the axes of the navigation reference frame would have their origin at the location of the INS and would be aligned with north, east, and local vertical. The prototype tracking system uses a slightly different definition that is functionally equivalent. Because the prototype system has no way of knowing the which way north is, the positions it reports are always relative to the starting point of a scan. That means that the origin of the navigation reference frame is fixed at the starting point of the scan and its axes are aligned with those of the body reference frame at the start of the scan.

The Stradwin reference frame is the coordinate system that Stradwin uses internally. It has no physical significance but it does affect how data is displayed on the screen. Suppose that the prototype system's navigation frame has its z-axis aligned with local vertical and the probe is moved straight down the x-axis. This motion would appear horizontal to the operator, but depending on how Stradwin's reference frame is aligned, it might appear to be vertical motion on the computer screen. Stradwin reads operating parameters from a file at start up, and some of those parameters define the relationship between the navigation reference frame and Stradwin's internal reference frame.

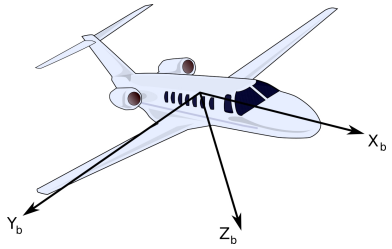


Figure 8.2: The body reference frame

8.2 Orientation Representations

The orientation of one coordinate system relative to another coordinate system can be represented as a *Direction Cosine Matrix* (DCM). In (8.1), $\mathbf{C}$ is a DCM that relates the body reference frame, denoted by the subscript b, back to the navigation reference frame, denoted by the superscript n. In the following explanation, DCMs will always be represented by $\mathbf{C}$. For DCMs only, the superscript position denotes the target reference frame of the transform and the subscript position denotes source reference frame of the transform. Therefore, $\mathbf{C}_{\mathsf{b}}^{\mathsf{n}}$ is the transform "from" the body reference frame "to" the navigation reference frame. The element in the ith row and jth column is the cosine of the angle between the i-axis of the reference frame and the j-axis of the body frame [27].

$$\mathbf{C}_{\mathsf{b}}^{\mathsf{n}} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{8.1}$$

Another method of representing the orientation of one coordinate system with respect to another is with Euler angles. In this representation, the transformation is expressed as 3 rotations: a rotation through angle ψ around the z-axis, a rotation through angle θ around the new y-axis, and a rotation through angle ϕ around the new x-axis. Each of these Euler rotations can be represented by a DCM, as in (8.2), (8.3), and (8.4).

$$\mathbf{C}_1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.2}$$

$$\mathbf{C}_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{8.3}$$

$$\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{8.4}$$

Equation (8.5) illustrates that the complete transformation from the navigation frame to the body frame is the product of these three matrices. The reverse transformation, from the body reference frame to the navigation reference frame, is the transpose of (8.5), equation (8.6). Equation (8.7) is the complete DCM in terms of Euler angles.

$$\mathbf{C}_{\mathsf{b}}^{\mathsf{n}} = \mathbf{C}_3\mathbf{C}_2\mathbf{C}_1 \tag{8.5}$$

$$\mathbf{C}_{\mathsf{n}}^{\mathsf{b}} = \mathbf{C}_{\mathsf{b}}^{\mathsf{n}T} = \mathbf{C}_1^T\mathbf{C}_2^T\mathbf{C}_3^T \tag{8.6}$$

$$\mathbf{C}_{\mathsf{n}}^{\mathsf{b}} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\psi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{8.7}$$

Conversion from DCM to Euler angles is accomplished using the equations in (8.8).

$$\begin{aligned} \phi &= \arctan\left[\frac{c_{32}}{c_{33}}\right] \\ \theta &= \arcsin\left[-c_{31}\right] \\ \psi &= \arctan\left[\frac{c_{21}}{c_{11}}\right] \end{aligned} \tag{8.8}$$

It is worth mentioning, for the sake of completeness, that there is a third common way of representing orientation, known as a quaternion. However, quaternions were not used in this work and therefore will not be covered.

8.3 Vector Transformation

The first goal of the tracking system is to track position over time. To do this, it is necessary to transform a displacement vector in the body reference frame to a displacement vector in the navigation reference frame. In physical terms, the displacements that are measured by the optical tracker on the probe have to transformed into displacements relative to the navigation reference frame, which is ultimately defined by the starting position of the probe. The transformation is accomplished by taking the product of the DCM and the displacement vector, as in (8.9).

$$\mathbf{r}^{\mathsf{n}} = \mathbf{C}_{\mathsf{b}}^{\mathsf{n}}\mathbf{r}^{\mathsf{b}} \tag{8.9}$$

8.4 Time Propagation of the DCM

The second goal of the tracking system is to keep track of the probe orientation over time. At the beginning of a scan, the body reference frame and the navigation reference frame are assumed to be perfectly aligned. The DCM representing this state is the identity matrix, I. A single angular rate sample can be represented as a vector of rotations that represent the turn rate of the body reference frame relative to the navigation reference frame, expressed in the body reference frame, as in (8.10). When attached to the quantities $\boldsymbol{\omega}$, $\boldsymbol{\Omega}$, and $\boldsymbol{\sigma}$,

the superscript denotes the reference frame the quantity is expressed in and the subscript denotes the object referred to by the quantity. For example, $\boldsymbol{\omega}$ is a general turn rate vector, $\boldsymbol{\omega}_{\mathsf{nb}}$ is the turn rate of the body reference frame relative to the navigation reference frame, and $\boldsymbol{\omega}_{\mathsf{nb}}^{\mathsf{b}}$ is the turn rate of the body reference frame relative to the navigation reference frame and expressed in the body reference frame. To update the DCM to reflect the latest angular rate sample, it is necessary to solve the matrix differential equation in (8.11). $\boldsymbol{\Omega}_{\mathsf{nb}}^{\mathsf{b}}$ is a skew symmetric matrix formed from the elements of the turn rate vector defined by (8.10) [27].

$$\boldsymbol{\omega}_{\mathsf{nb}}^{\mathsf{b}} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T \tag{8.10}$$

$$\dot{\mathbf{C}}_{\mathsf{b}}^{\mathsf{n}} = \mathbf{C}_{\mathsf{b}}^{\mathsf{n}} \boldsymbol{\Omega}_{\mathsf{nb}}^{\mathsf{b}} \tag{8.11}$$

$$\boldsymbol{\Omega}_{\mathsf{nb}}^{\mathsf{b}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{8.12}$$

A single iteration of the algorithm can be considered to take place over the interval t_k to t_{k+1}. The DCM at time t_{k+1} can then be expressed as (8.13). For notational convenience, $\mathbf{C}_{\mathsf{b}}^{\mathsf{n}}$ will be abbreviated as $\mathbf{C}$ for the remainder of this chapter. Assuming that the turn rate vector, $\boldsymbol{\omega}_{\mathsf{nb}}^{\mathsf{b}}$ remains fixed in space over the update interval, which is true as long as that interval is small, the integral of the skew symmetric turn rate matrix is $[\boldsymbol{\sigma}\times]$, as in (8.14). $\boldsymbol{\sigma}$ is an angle vector that relates the orientation of the body frame at time k to $k+1$. The magnitude and direction of $\boldsymbol{\sigma}$ are such that rotating the body frame an angle equal to the magnitude of $\boldsymbol{\sigma}$ about $\boldsymbol{\sigma}$ produces the rotation of the body reference frame from time k to $k+1$.

$$\mathbf{C}_{k+1} = \mathbf{C}_k \exp \int_{t_k}^{t_{k+1}} \boldsymbol{\Omega}_{\mathsf{nb}}^{\mathsf{b}} dt \tag{8.13}$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{\Omega}_{\mathsf{nb}}^{\mathsf{b}} dt = [\boldsymbol{\sigma}\times] \tag{8.14}$$

The expanded form of $\boldsymbol{\sigma}$ and its magnitude are defined in (8.15) and (8.16), respectively. The expanded form of $[\boldsymbol{\sigma}\times]$ is defined in (8.17).

$$\boldsymbol{\sigma}_{\mathsf{nb}}^{\mathsf{b}} = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z \end{bmatrix}^T \tag{8.15}$$

$$\boldsymbol{\sigma}_{\mathsf{nb}}^{\mathsf{b}} = \sqrt{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)} \tag{8.16}$$

$$[\boldsymbol{\sigma}\times] = \begin{bmatrix} 0 & -\sigma_z & \sigma_y \\ \sigma_z & 0 & -\sigma_x \\ -\sigma_y & \sigma_x & 0 \end{bmatrix} \tag{8.17}$$

Using the result in (8.14), (8.13) can be reduced to (8.18). The exponential function, e^x, is defined for square matrices and results in a matrix with the same dimensions as that in the exponent. Knowing this, (8.18) reduces to the product of two rotation matrices, as in (8.19).

$$\mathbf{C}_{k+1} = \mathbf{C}_k \exp\left[\boldsymbol{\sigma}\times\right] \tag{8.18}$$

$$\mathbf{C}_{k+1} = \mathbf{C}_k \mathbf{A}_k \tag{8.19}$$

Since $\mathbf{C}_k$ is the DCM at time k, $\mathbf{A}_k$ must be the DCM that transforms a vector in the body reference frame at time $k+1$ to a vector in the body reference frame at time k. In order to solve (8.19) we must find an expression for $\mathbf{A}_k$. The exponential of a square matrix is defined by the power series in (8.20).

$$e^{\mathbf{X}} = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{X}^k \tag{8.20}$$

Equation (8.21) shows the expansion of the first few terms of the series. Equation (8.21) is further simplified in (8.22).

$$\mathbf{A}_k = \mathbf{I} + [\boldsymbol{\sigma}\times] + \frac{[\boldsymbol{\sigma}\times]^2}{2!} + \frac{[\boldsymbol{\sigma}\times]^3}{3!} + \frac{[\boldsymbol{\sigma}\times]^4}{4!} + ... \tag{8.21}$$

$$\begin{aligned}
[\boldsymbol{\sigma}\times]^2 &= \begin{bmatrix} -(\sigma_y^2+\sigma_z^2) & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & -(\sigma_x^2+\sigma_z^2) & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & -(\sigma_x^2+\sigma_y^2) \end{bmatrix} \\
[\boldsymbol{\sigma}\times]^3 &= -(\sigma_x^2+\sigma_y^2+\sigma_z^2)[\boldsymbol{\sigma}\times] \\
[\boldsymbol{\sigma}\times]^4 &= -(\sigma_x^2+\sigma_y^2+\sigma_z^2)[\boldsymbol{\sigma}\times]^2 \\
&\vdots \\
\mathbf{A}_k &= \mathbf{I} + [\boldsymbol{\sigma}\times] + \frac{[\boldsymbol{\sigma}\times]^2}{2!} - \frac{\sigma^2[\boldsymbol{\sigma}\times]}{3!} - \frac{\sigma^2[\boldsymbol{\sigma}\times]^2}{4!} + ... \\
&= \mathbf{I} + \left[1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} - ...\right][\boldsymbol{\sigma}\times] + \left[\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - ...\right][\boldsymbol{\sigma}\times]^2
\end{aligned} \tag{8.22}$$

By representing the power series this way, (8.21) can be expressed with only the first and second powers of $[\boldsymbol{\sigma}\times]$, as in (8.23). This is desirable because it reduces the amount of computation required between each angular rate sample.

$$\begin{aligned}
\mathbf{A}_k &= \mathbf{I} + a_1[\boldsymbol{\sigma}\times] + a_2[\boldsymbol{\sigma}\times]^2 \\
a_1 &= 1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} - ... \\
a_2 &= \frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - ...
\end{aligned} \tag{8.23}$$

It is therefore possible to solve for the DCM after each new angular rate sample arrives by solving (8.23) for $\mathbf{A}_k$ and then applying (8.19). The accuracy of the algorithm, and also the amount of computation time required, is determined by the number of power series terms that are included in the computation of $\mathbf{A}_k$. As long as $\boldsymbol{\sigma}$ remains small, a small number

of terms are necessary to accurately calculate $\mathbf{A}_k$. This means that there is a trade off between maximum turn rate, sampling frequency, and computational complexity. A system with a high maximum turn rate will either have to sample at a higher rate or perform more computation than a system with a low maximum turn rate, to achieve the same accuracy.

Chapter 9

Prototype System Implementation

This chapter provides a detailed description of the prototype freehand 3D ultrasound system. The prototype system has several major components, which are illustrated in Figure 9.1. A PC provides the user interface and high level system control. The Stradwin software package implements real-time 3D volume reconstruction and visualization. Underneath Stradwin, Terason software controls the transducer electronics and the interface for grabbing image frames from the hardware. The ultrasound system is a Terason t3000 that connects to the PC using firewire. The t3000 supports many different probe modules but only a 7L3 linear array probe was used in this work. The position and orientation tracking system, or navigation computer, is implemented in a Xilinx Spartan 3E FPGA using a Microblaze soft-core embedded system. Two custom PCBs were developed to physically mount and electrically interface with the ADNS-2610 and ADIS16350 tracking sensors. Firmware running on the Microblaze embedded system implements an inertial tracking algorithm that converts raw sensor data into estimates of the ultrasound transducer's position and orientation, which are transmitted to the PC using the RS232 serial port. Both versions of the optical system were tested with the prototype.

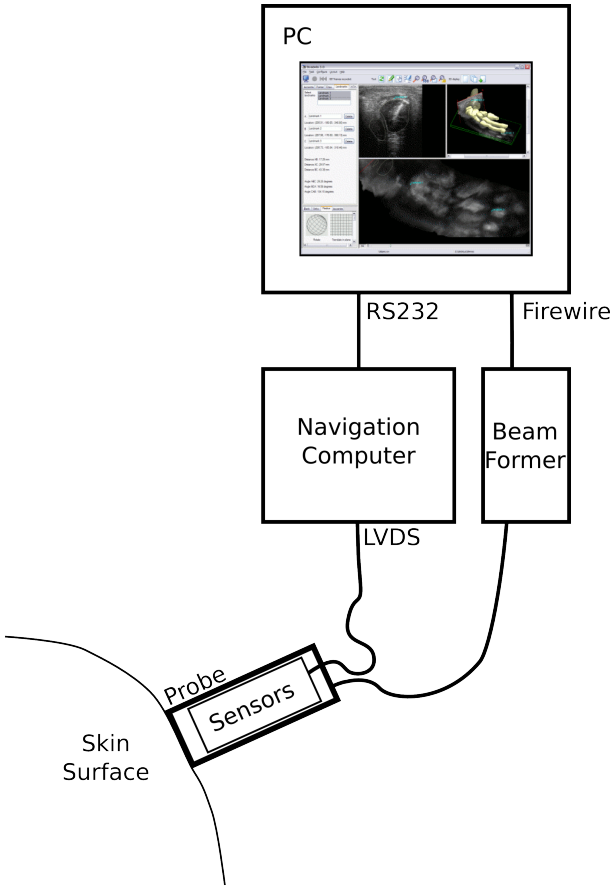


Figure 9.1: Diagram of the prototype freehand 3D ultrasound system

The individual sensors have already been described in Chapters 5 and 7. Stradwin is documented in Chapter 2. A users guide for the prototype system is included in Appendix A. With these components already documented, only the sensor interface electronics, the navigation computer hardware, and the navigation computer firmware remain. The first part of this chapter describes the design of the sensor interface electronics and the embedded system that forms the hardware part of the navigation computer. The second half of the chapter is dedicated to describing the firmware part of the navigation computer.

9.1 Sensor Interface Electronics

A custom PCB was designed for each of the two sensors used in the tracking system. Each PCB provides an electrical interface between the sensor and the navigation computer as well as mechanical features to mount the sensors to the PCBs and the PCBs to the ultrasound probe. In addition, a cable was fabricated to connect the sensor PCBs to the Spartan 3E demonstration board. These components are illustrated in Figure 9.2.

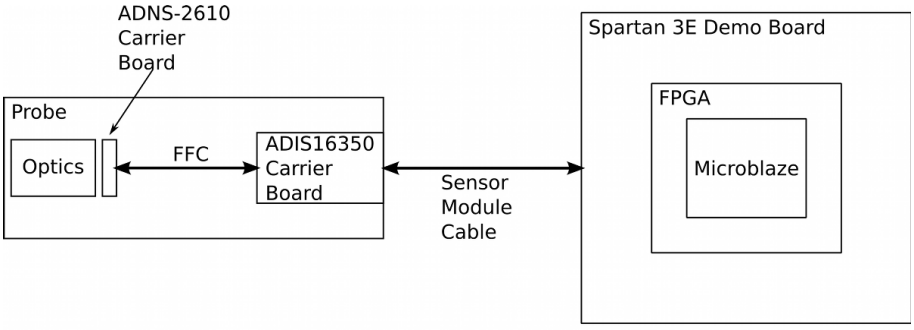


Figure 9.2: Diagram of the electronic components of the prototype system.

9.1.1 ADIS16350 Carrier Board

The ADIS16350 Carrier Board provides an electrical interface to the ADIS16350 inertial sensor, mechanical features for mounting the sensor to the PCB and the PCB to the ultrasound probe, and a mechanism for user input and output. It is pictured in Figure 9.3 without the ADIS16350 mounted. The PCB has several important mechanical features. The mounting hole pattern of the ADIS16350 was designed into the PCB so that it could be rigidly attached with #4-40 screws. Additional holes were placed in the middle and outer corners so that the PCB could be attached to the ultrasound probe. The overall size of the board was kept to minimum so that the entire assembly would fit on the probe.

Electrically, the ADIS16350 carrier board is quite simple. The schematic is illustrated in Figure 9.4. Signals from the demo board enter the carrier board through J1, which is a Hirose DX10G1M-26SE(50). The +5 v supply comes in on pins 11 and 12 and ground

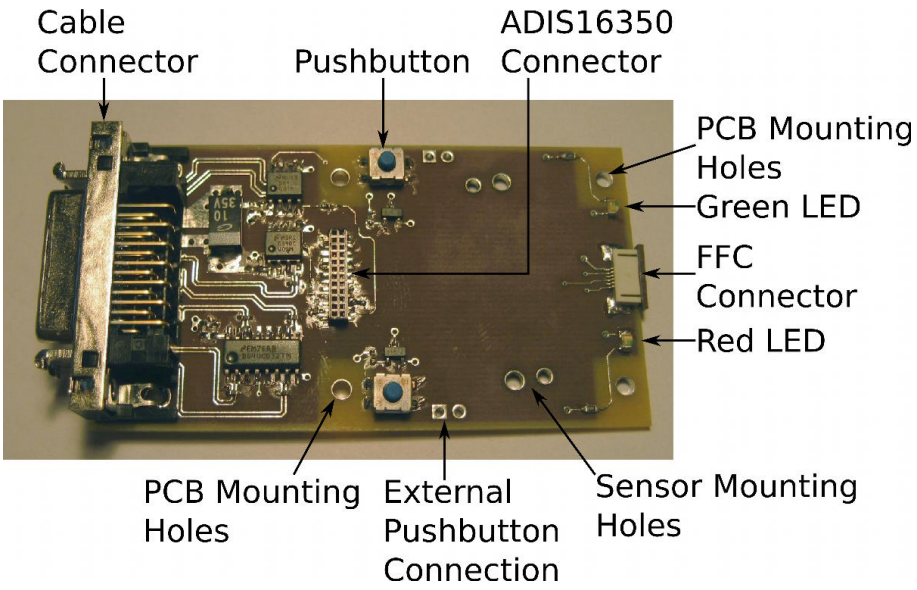


Figure 9.3: Picture of the ADIS16350 carrier PCB

returns on pins 5, 6, 13, 16, 20, and 26. Capacitor C7 provides bulk capacitance to supply current during electrical transients. A tantalum capacitor was selected because of its low internal resistance. D1 and D2 are red and green LEDs, respectively. Single ended signals from the FPGA drive the two NFETs, Q1 and Q2, which act as current switches to turn the LEDs on and off. The LEDs are used as system status indicators. SW1 and SW2 are momentary pushbutton switches that allow the operator to provide input to the system. J4 and J5, which are connected to the same nets as the switches, are provided so that switches could also be mounted remotely if desired.

Low Voltage Differential Signaling (LVDS) was employed to extend the range that the serial I/O signals can travel down the cable linking the demo board and ADIS16350 carrier board. The FPGA supports this signaling standard directly but the carrier board required LVDS receiver and driver ICs to complete the link. U1 and U2 are LVDS receivers; U1 has 4 channels and U2 has 2 channels. U3 is a two channel LVDS driver. The LVDS standard specifies that the cable maintain 100 ohm differential impedance and that the circuit be terminated with a 100 ohm resistor at the receiver. R1, R2, R3, R4, and R5 are the LVDS termination resistors. The receivers convert the differential input signals to 5 volt TTL compatible

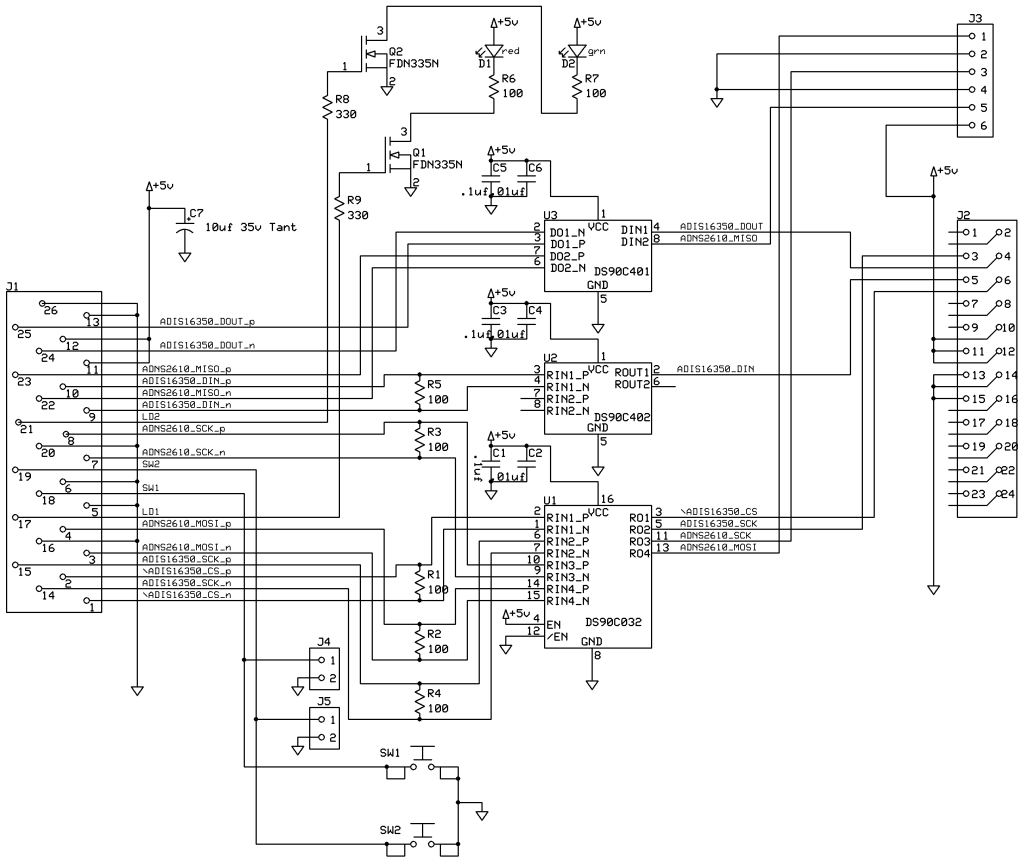


Figure 9.4: ADIS16350 Carrier PCB Schematic

outputs. The ADIS16350's serial inputs, labeled ADIS16350_CS, ADIS16350_SCK, and ADIS16350_DIN are routed to J2, the ADIS16350's electrical connector, which is a Samtec CLM-112-02-L-D. The ADIS16350's serial output, labeled ADIS16350_DOUT, is routed from J2 to U3, the LVDS driver, where its converted from 5 v TTL to LVDS, and transmitted back to the demo board. The ADNS-2610's serial inputs, labeled ADNS2610_SCK and ADNS2610_MOSI, are routed to J3. The ADNS-2610's serial output, labeled ADNS2610_MISO, is routed from J3 to U3, where it is converted to LVDS and transmitted back to the demo board.

A Flat Flex Cable (FFC) is used to connect the ADIS16350 carrier PCB to the ADNS-2610 carrier PCB, which is described in the next section. J3 is a 2 in long, 0.5 mm pitch

FFC connector, Molex part number 21020-0053, that matches the 6 circuit FFC, Molex part number 52746-0670. The FFC connection is pictured in Figure 9.5. The FFC was chosen because of the size of the connectors and because its thin flat profile allowed it to be taped to the transducer handle where it would be out of the way

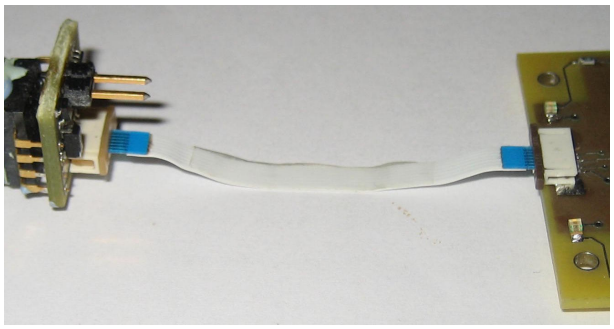


Figure 9.5: FFC connection between the ADIS16350 carrier PCB and the ADNS-2610 carrier PCB

9.1.2 ADNS-2610 Carrier Board

The ADNS-2610 carrier board was designed to provide an electrical interface to the ADNS-2610 optical mouse sensor and to mount the external components it requires to function. It is pictured in Figure 9.6. In order to make the optical tracking assembly as small as possible it was important to keep the size of this board to a minimum. It is just slightly larger than the footprint of the sensor itself. The board was designed to have the sensor mounted backwards. To mount the sensor the pins must be bent 180°from their nominal position. This arrangement minimizes the size of the PCB by allowing components to be mounted underneath the sensor, where a hole would normally be located for the optical aperture. The boards were ordered from a vendor that had a minimum PCB size that was larger than the size required for components. The ground and power planes were kept out of the excess area so that it could be removed with a saw. The excess area is the lighter region on the lower left hand side of Figure 9.6.

The ADNS-2610 carrier board's schematic is illustrated in Figure 9.7. J2 is the FFC connector, Molex part number 52559-0672, that connects the ADNS-2610 carrier to the

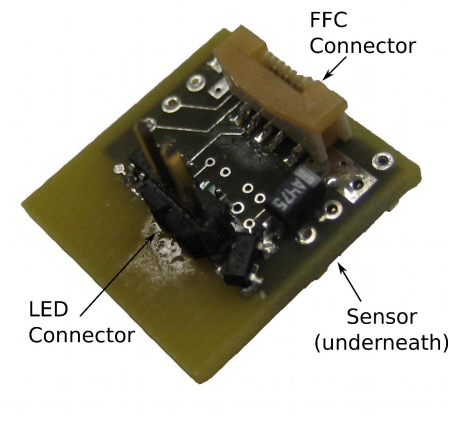


Figure 9.6: Picture of the ADNS-2610 carrier PCB

ADIS1630 carrier. Pin 6 is the +5 v supply and pins 2 and 3 are ground. U1 is the ADNS-2610, X1 is a 24 MHz surface mount resonator, and Q1 is a PNP transistor that supplies the current to drive the LED. J1 is a single row 2 x 0.100 in header that is used to connect the LED, which is mounted remotely. The ADNS-2610 is designed to use a two wire serial interface in which a single wire is used to send and receive serial data. However, the Microblaze SPI port has separate data in and data out connections, Master In Slave Out (MISO) and Master Out Slave In (MOSI). For data to be exchanged with the ADNS-2610, MISO and MOSI must both be connected to the ADNS-2610's SDIO pin. This is a problem because when MOSI is idle it is driven high. The result is contention between the driver in the Microblaze SPI port and the driver in the ADNS-2610. U2 is a dual NFET that is used to implement an open drain non-inverting buffer. The buffer breaks the contention by isolating MOSI from SDIO. The output of this buffer does not actively drive high, it is only pulled up by the 4.7 kohm resistor. The buffered version of MOSI is connected to SDIO and MISO. The current sinking capability of the driver in the ADNS-2610 is large enough to overcome the pullup resistor, which allows it to take control of the bus.

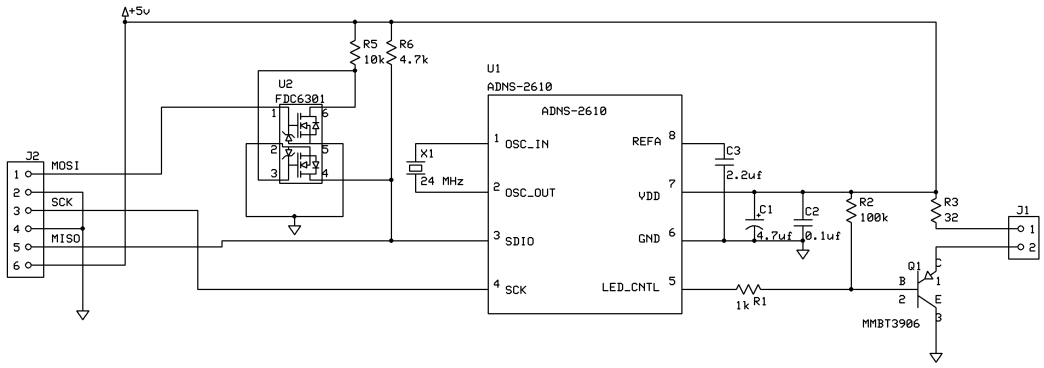


Figure 9.7: ADNS-2610 Carrier PCB Schematic

9.1.3 Sensor Module to Demo Board Cable

A cable connects the Spartan 3 demo board to the ADIS16350 carrier board. The cable itself is 2 m long and has 16 100 ohm differential impedance twisted pairs of 24 AWG solid copper wire. One end connects to J3 on the demo board using a Hirose FX2BA-100SA-1.27R connector. The side that attaches to the ADIS16350 carrier board uses a Hirose DX40M-26P connector. Table 9.1 is a pin to pin connection table for the cable.

Table 9.1: Pinout of the sensor module cable

ADIS16350 carrier board pin #	Signal Name	Spartan 3E board J3 pin #	FPGA pin #
1	ADIS16350_CS_n	A20	D11
2	ADIS16350_CS_p	A21	C11
3	ADNS2610_MOSI_n	A16	F8
4	ADNS2610_MOSI_p	A17	E8
5	GND	B2	-
6	GND	B5	-
7	ADNS2610_SCK_n	A8	D5
8	ADNS2610_SCK_p	A9	C5
9	ADIS16350_DIN_n	A24	E12
10	ADIS16350_DIN_p	A25	F12
11	+5v	A49	-
12	+5v	A50	-
13	GND	B24	-
14	ADIS16350_SCK_n	A10	A6
15	ADIS16350_SCK_p	A11	B6
16	GND	A46	-
17	LED1	A13	F7
18	SW1	A15	C7
19	SW2	A12	E7
20	GND	A48	-
21	LED2	B47	D7
22	ADNS2610_MISO_n	A22	F11
23	ADNS2610_MISO_p	A23	E11
24	ADIS16350_DOUT_n	A6	B4
25	ADIS16350_DOUT_p	A7	A4
26	GND	A48	-

9.2 Microblaze Embedded System

The heart of the tracking system is the Spartan 3E 1600 development board manufactured by Digilent Incorporated. This board was selected because it was one of the few demonstration boards available that supported the largest, in terms of logic gates and memory, Spartan 3E FPGA. The Spartan 3E family of FPGA was selected because it is a low cost FPGA with features specifically targeted for embedded processing applications. Other important features of the demonstration board are the RS232 connector, configuration Programmable Read Only Memory (PROM), and an expansion connector for cabling to the sensor module. The board is pictured in Figure 9.8.

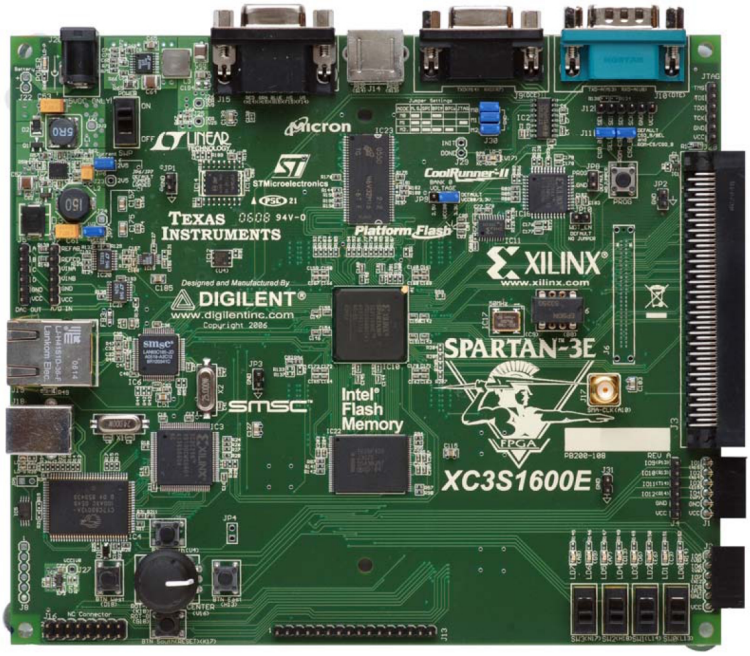


Figure 9.8: Digilent Spartan-3 1600E Demonstration Board

The Xilinx Embedded Platform Studio (XPS) toolsuite was used to develop an embedded system in which to implement the tracking system. Figure 9.9 is a block diagram of the system. At the center is a Microblaze 32 bit Reduced Instruction Set Computing (RISC) microprocessor. The Microblaze utilizes a *Harvard Architecture*, which means that the in-

struction and data memory address spaces are separate. The processor itself has many configuration options that affect its capabilities, performance, size in the FPGA, and maximum operating frequency. The version used in this design was optimized for speed at the expense of size. The size is larger because the speed optimized version features a five stage pipeline as opposed to a three stage pipeline in the size optimized version. A deeper pipeline reduces the delay between stages and therefore increases the maximum operating frequency.

Several optional functional units were included in the processor. These were the Floating Point Unit (FPU), barrel shifter, 32 bit integer multiplier, and pattern comparator. The floating point unit was critical because Stradwin required the position and orientation values to be single precision floating point values. The overhead incurred by doing the floating point processing in software would have prevented the system from operating in real time. The other functional units were also included to improve processing performance. Two major options that were not included were caches and a Memory Management Unit (MMU). If the system utilized the external memories available on the demo board, a cache would have improved performance because the external memories cannot be accessed in a single clock cycle. However, all of the memory used in the system is implemented in block RAM inside the FPGA. The block RAM operates at the same frequency as the processor so there is no advantage to including a cache. A MMU is unnecessary because the entire application fits in memory and memory protection is not required.

The processor implements two different types of bus to connect to other system components. The Local Memory Bus (LMB) is a high speed bus that is optimized to have single cycle latency to local memories. The LMBs are represented by the orange and red lines in the diagram, for the data and instruction memory spaces, respectively. The Processor Local Bus (PLB) is a more general purpose interconnect. It supports multiple masters, burst and block transfers, timeouts, multiple wait states, and many other features that make it flexible enough to connect with a wide variety of devices. The PLB is the bus that connects the processor to the peripherals. It is represented by the green line in the diagram. Unlike the LMBs, there is only one PLB that spans both the instruction and data memory spaces. This is only possible because the PLB supports multiple masters. In this implementation,

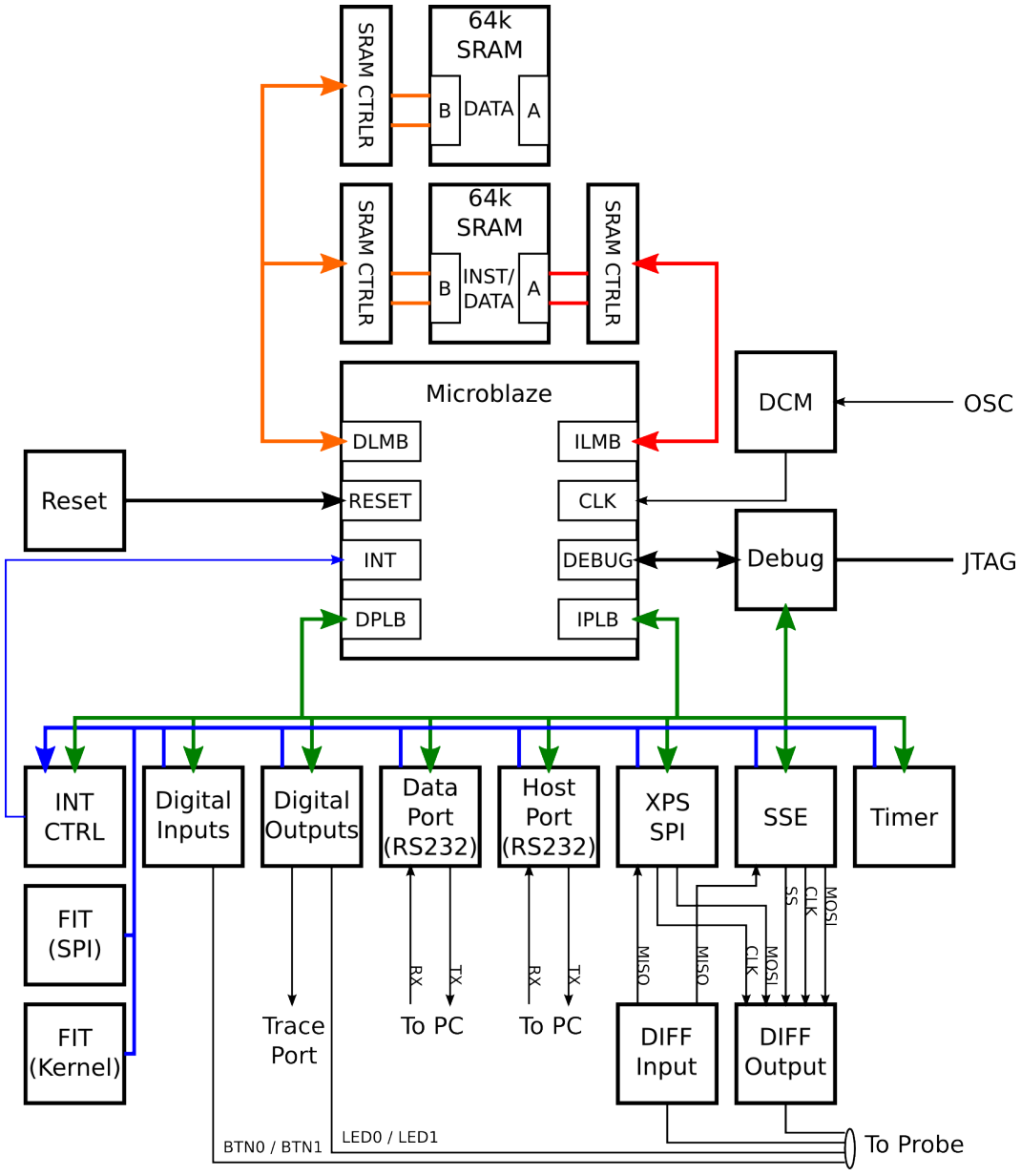


Figure 9.9: Block diagram of the microblaze embedded system

the PLB is never accessed from the instruction memory space because there is no memory attached to the PLB that could hold instructions. All of the accesses to the PLB are from the data memory space and they are all for memory mapped I/O to the peripherals.

There are two 64k blocks of memory in the system. Both blocks are implemented in on-chip block RAM which allows the processor to access the memory in a single clock cycle. The RAM blocks and their associated interface controllers appear above the processor in Figure 9.9. The interface controllers are responsible for address decoding, byte steering, and physically connecting the bus signals to the RAM interface signals. The first RAM block is shared between the instruction and data memory spaces and the second block is mapped only to the data memory space. The memory system is configured this way because the blocks can only be configured in power of two sizes. As the FW was being developed a point was reached when the executable code would no longer fit in a 32k block. In order to not waste the unused part of a 64k block connected only to the instruction address space, the block was connected to both. This works because the block RAMs in Xilinx FPGAs are dual ported. If a block RAM were shared between two data memory spaces, as in a block RAM shared between two separate processors, there would be the potential for data consistency issues. However, because the instruction memory space is read-only, that is not a problem in this system.

Many different peripherals were included in the system. With the exception of the Synchronous Serial Engine (SSE) block, all of them were included in the XPS development environment. Most are connected to the PLB and are accessed through the data memory space but some have no bus connections. All of the blocks in Figure 9.9 that are not the processor and memory can be considered peripherals. There are two digital I/O blocks in the system, one for input and one for output. The digital input block is used to read the state of the two push buttons located on the ultrasound probe. The digital output block is used to drive the two status LEDs located on the ultrasound transducer and to drive the trace port. The trace port signals were connected to J16 on the demo board. These signals were used to verify system timing during firmware development. Individual pins were assigned to each thread so that the order of thread execution could be observed on an

oscilloscope. Two RS232 UART blocks are included in the system. The block labeled Host Port was used to communicate with the Stradwin program running on the host PC. The block labeled Data port was used as a secondary interface to configure and stream data out of the system without interrupting the connection to Stradwin. The Data Port was used primarily as debug tool. The block labeled XPS SPI is an 8 bit SPI serial port. It was used to communicate with the ADNS-2610 optical mouse sensor. The block labeled SSE is a custom designed peripheral that was used to interface with the ADIS1350 inertial sensor. It will be covered in greater detail later in this chapter. The timer block implements two 32 bit timers. The timers are used by the firmware where timing more precise than the kernel's software timers was required.

The debug module connects the Microblaze's debug port to the FPGA's Joint Test Action Group (JTAG) interface. A JTAG bus connects to the FPGA and the PROM on the demo board and is bridged to the PC over USB. This provides a means of programming them and also allows the PC to control the processor's execution remotely. The debug module allows breakpoints and watchpoints to be set, memory initialization, code download, and single stepping the processor, all of which are critical for firmware development.

The interrupt controller block allows multiple interrupt sources to share the single interrupt input to the processor. Each interrupt source can be individually enabled and configured for polarity. Priority is set by the order in which the individual signals are connected to the interrupt controller's input port. The blue line in Figure 9.9 illustrates the interrupt connections present in the system.

The remaining blocks are not connected to any of the system busses. Their configuration is set in the XPS tool and fixed during synthesis. There are two Fixed Interval Timer (FIT) blocks. The first is used to provide a periodic interrupt for the XMK kernel tick. This timer sends the interrupt controller and interrupt request every 100 ms. The second is used to trigger the thread that polls the ADNS-2610. It interrupts every 20 ms. The reset block brings the various system components out of reset in the correct order. A Digital Clock Manager (DCM) is a digitally controlled Phase Locked Loop (PLL) that is permanently embedded in the FPGA. DCMs are used for clock management and implement advanced

features like phase and duty cycle adjustment, clock multiplication and division, and skew compensation. The DCM block is used to configure the hardware. In this application it was used to increase the 50 MHz clock provided by the crystal on the demo board to 66 MHz for increased performance. The DIFF Input and DIFF Output blocks provide access to the FPGA's LVDS input and output buffers, respectively. They basically just provide a mechanism for informing the synthesis tool that a signal represented by single net in the design needs to be converted to a differential signal at the I/O pad. LVDS I/O was used on all of the serial signals used to communicate with the sensors. This will be explained in greater detail later in this chapter.

9.2.1 Synchronous Serial Engine

The ADIS16350 inertial sensor is designed to interface with a 16 bit SPI serial port. This proved to be a problem because the SPI serial port that Xilinx provides with XPS is an 8 bit device. It is possible to interface the 8 bit port to the 16 bit port by breaking each 16 bit transfer into two consecutive 8 bit transfers. The ADIS16350 is polled at a rate of just over 819.2 Hz and every time it is polled twelve 16 bit words are transferred. Using the 8 bit serial port, this requires 24 transactions per sample period, which results in 19660 8 bit transfers per second. The processing overhead incurred by the XMK kernel's interrupt system and the time spent polling between 8 bit transfers proved to be too great to both handle communication and service all of the other system tasks in real time.

A custom serial port was developed in VHSIC Hardware Description Language (VHDL) to solve the problem. The SSE was designed to interface directly to the Microblaze's PLB just like the standard peripherals that come with XPS. XPS has a wizard that allows the user to specify the parameters of the bus interface and then generates VHDL template files that the user then customizes to fit his or her needs. Figure 9.10 is a block diagram of the SSE.

The PLB interface block on the left is the module that was generated by XPS. The block then provides data in and data out busses, select signals, and read and write enables to the user logic. The PLB interface also implements the interrupt and reset control logic. Interrupt

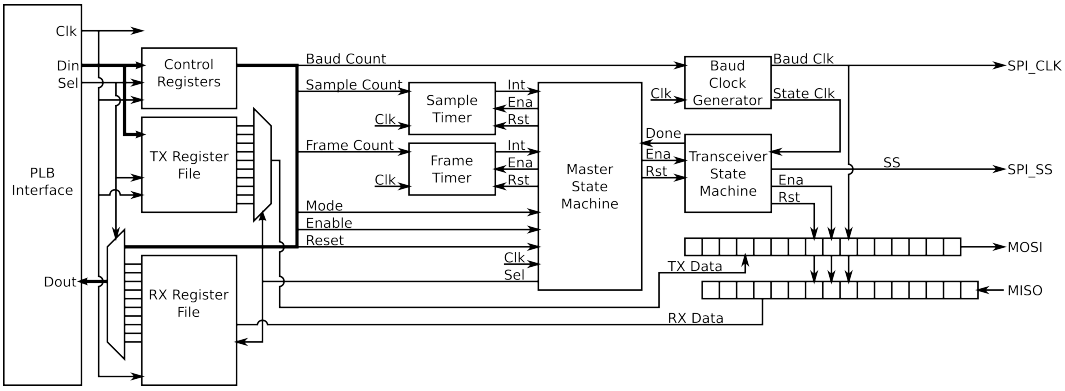


Figure 9.10: Synchronous Serial Engine block diagram

status and enable resisters inside the PLB interface control how interrupts generated by the user logic are handled. The SSE has only one interrupt, which is generated whenever a transfer completes. The interrupt pulse is latched into the interrupt status register and if the corresponding bit is set in the interrupt enable register, the signal is forwarded to the system interrupt controller. The PLB interface's reset register can be used to send a reset signal to the SSE hardware.

This design uses set of registers as the connection between the PLB and the user logic. The control register block consists of 4 32 bit registers named Control, Baud Count, Sample Count, and Frame Count. Only bits 31 to 29 of the control register are used; 31 is the mode select bit, 30 is the enable bit, and 29 is the reset bit. The three count registers are used to set the overflow value of the three timers in the system: the Sample Timer, the Frame Timer, and the Baud Clock Generator. The Sample Timer sets the overall sampling rate. The value written to the Sample Count register should be the result of dividing the system clock frequency by the desired sampling rate. To sample the ADIS16350 at its maximum rate of 819.2 Hz, the count value should be 66.667 MHz / 819.2 Hz = 81,381, or 0x00013DE5 in hex as is would be written to the register. The Frame Timer sets the time between the start of individual 16 bit transfers within a sample period and is equivalent to $t_{DATARATE}$ in Figure 9.11, which is a timing diagram from the ADIS16350 datasheet. The minimum allowable $t_{DATARATE}$ is 40 μs. To set Frame Timer to 40 μs we use the same

procedure as before, and arrive at 2666.66, or 0x00000A6B in hex as it would be written to the register. The Baud Clock Generator generates State Clock, which is the clock that drives the Transceiver State Machine, and the Baud Clock, which clocks data into and out of the shift registers. The ADIS16350 data sheet specifies that the maximum frequency of the SPI clock is 2 MHz. The value written to the Baud Count register should be the result of dividing the system clock frequency by twice the desired SPI clock frequency. To generate a 2 MHz SPI clock, 66.667 MHz is divided by 4 MHz to get the a baud count value of 17, or 0x00000011 in hex as it would be written to the Baud Count register.

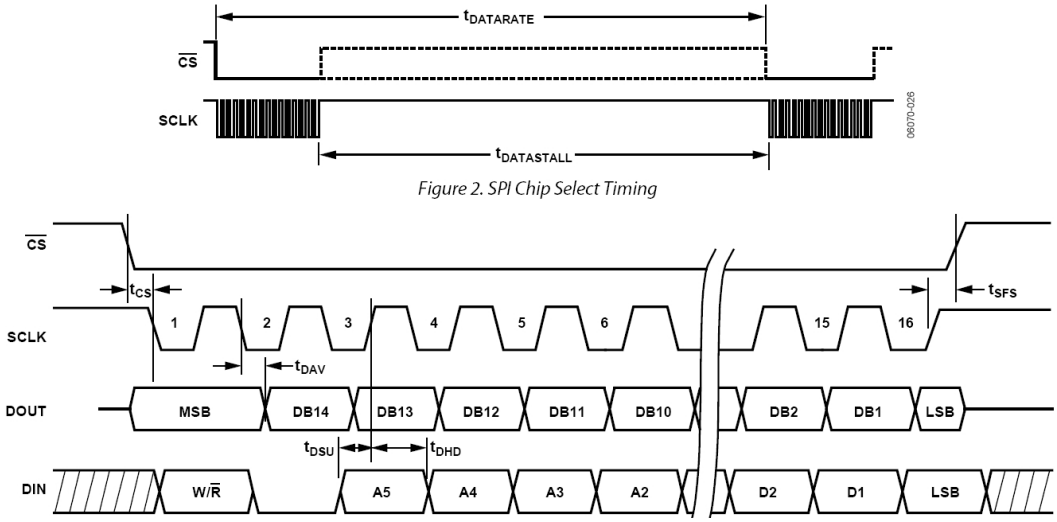


Figure 9.11: ADIS16350 SPI Timing Diagram [8]

The TX Register file and the RX Register file are used to store the outgoing and incoming data, respectively. They are mapped to the same physical addresses which means that data cannot be read from the TX Register File. Reading from the TX Register File is equivalent to reading from the RX Register File. The Control Registers, however, can be read back. The Master State Machine module controls the overall behavior of the SSE. The SSE is active when the Enable bit, bit 30, of the Control Register is set. The SSE has two modes of operation. In single transfer mode the SSE will transfer a single 16 bit data word. This mode is useful for configuration of the ADIS16350, when the user only wants to read or write a single register in the ADIS16350. In this mode, the transmit data should be written

to TX Register(0) and the receive data will be stored in RX Register(0). When the cycle is complete the Master State Machine asserts the interrupt signal which is then processed in the PLB interface. Single transfer mode is active when the mode bit of the control register is set to 1.

The second mode of operation is continuous mode. In continuous mode, the SSE will automatically transfer a series of sixteen 16 bit words and do so every time the Sample Timer overflows. This mode allows all of the sensor's output registers to be polled continuously. In each sample period, the contents of each TX Register are transmitted sequentially from 0 to 15. The received data is stored in each RX Register, from 0 to 15. After all 16 transfers are complete the Master State Machine asserts the Interrupt signal which is then processed in the PLB interface. Continuous transfer mode is active when the mode bit of the control register is set to 0.

9.3 Navigation Computer Firmware

The navigation computer firmware runs on the Microblaze embedded system described in the previous section. The primary function of the firmware is to implement the inertial navigation algorithm described in Chapter 8. In addition, the firmware transfers data from the sensors, responds to requests for position and orientation data from the PC, and implements several testing and debugging features.

The Xilinx Micro Kernel (XMK) operating system is the foundation of the firmware system. XMK is a Portable Operating System Interface (POSIX) compliant operating system that Xilinx provides free of charge with XPS. It provides basic services such as memory management, interrupt management, multithreading, inter-thread communication mechanisms, and thread synchronization mechanisms such as semaphores and mutexes. Xilinx also provides drivers for all of peripherals that are included in XPS, such as the UART, interrupt controller, SPI serial port, and GPIO module. The source code for the XMK kernel and the application code are written in C. There are GCC C/C++ cross compilers and associated tools that are used to compile firmware for the Microblaze processor. This

includes ports of the C/C++ standard libraries.

The system tasks are divided between four threads. One of these is used for system initialization and the other three are used during normal operation. Each of the three main threads has an ISR associated with it that causes the thread to be scheduled in response to an event. Priority scheduling is employed to make sure that critical tasks take precedence over less critical ones. When the system first starts, the initialization thread is the only thread known to the scheduler. Its priority does not matter because it there are no other threads to compete with. It performs basic system initialization tasks then creates the other system threads. After it completes it returns to the kernel and never executes again.

The `fastrack_protocol_stack` thread manages communication with the Stradwin software running on the host PC and it has the lowest execution priority. The UART module dedicated to host communication is configured to interrupt when a character is received. The UART ISR does nothing more than acknowledge the interrupt and *post()* to a semaphore called `sem_fastrack`. After performing some basic initialization tasks, the `fastrack_protocol_stack` thread enters an infinite loop and *wait()s* on the `sem_fastrack` semaphore. When the XMK kernel sees the *post()* to `sem_fastrack`, the `fastrack_protocol_stack` thread is scheduled to execute after any higher priority threads in the ready queue.

The navigation computer communicates with Stradwin by emulating the serial protocol used by the Ascension Fastrack, a commercially available 6 DoF tracking device. The `fastrack_protocol_stack` thread only implements the subset of the Fastrack protocol that is actually used by Stradwin. Of the rest of the commands in the protocol, some were redefined to implement debug and test features, and the rest are unused. All of the commands are ASCII characters sent from the PC to the navigation computer. The response, if there is one, is sent from the navigation computer to the PC. The format of the response varies depending on the command. Table 9.2 documents the protocol commands and response formats. Quantities in quotes are ASCII characters and brackets are used to indicate that the value at a particular position is variable. For values that are not ASCII, the type is indicated in the definition of the field.

Table 9.2: Serial communication protocol used between the navigation computer and the PC

Command Name	Request	Response
Status	'S'	'29S00000000000000000000000000000000000000000000000000000000000000'
Position	'P'	'000'[X0-3][Y0-3][Z0-3][A0-3][P0-3][R0-3]'00' [X0-3] = X Position, float32 [Y0-3] = Y Position, float32 [Z0-3] = Z Position, float32 [A0-3] = Yaw Angle, float32 [P0-3] = Pitch Angle, float32 [R0-3] = Roll Angle, float32
Reset Data Port	'R'	NONE
LED on/off	'L'['R'|'G']['0'|'1']	NONE
	['R'|'G'] - 'R' = red, 'G' = green ['0'|'1'] - '0' = off, '1' = on	
Get Image	'I'	[SQUAL][D0...D323] [SQUAL] = value of SQUAL register, uchar [D0...D323] = raw pixel data, uchar
Enable / Disable Tracking	'T'['E'|'D'] ['E'|'D'] - 'E' = disable, 'D' = disable	NONE
Raw Gyro Command	'GA'[addr][data] [addr] = ADIS16350 register address, uchar [data] = ADIS16350 register data, uchar	[D0][D1] [D0] - LSB of result, uchar [D1] - MSB of result, uchar
Gyro normal / program mode	''GB'['N'|'P'] ['N'|'P'] - 'N' = normal mode, 'P' = program mode	NONE
Gyro Precision Calibrate	'GC'	[Xg0-3][Yg0-3][Zg0-3][Xa0-3][Ya0-3][Za0-3] [Xg0-3] - X gyro offset, float32 [Yg0-3] - Y gyro offset, float32 [Zg0-3] - Z gyro offset, float32 [Xa0-3] - X accelerometer offset, float32 [Ya0-3] - Y accelerometer offset, float32 [Za0-3] - Z accelerometer offset, float32
Data Port	'D'['0'|'1']['A'|'B'] ['0'|'1'] - '0' = off, '1' = on ['A'|'B'] - 'A' = none, 'B' = raw	NONE

The `mouse_comm_thread` manages communication with the ADNS-2610 optical mouse sensor. It has the middle priority, which means that it can preempt the `fastrack_protocol_stack` thread but not the other thread. When the `mouse_comm_thread` first starts it configures the SPI serial port that is used to communicate with the ADNS-2610 and initializes the sensor. It then enters an infinite loop and *wait()*s on the `sem_mouse` semaphore. This thread uses two timers. One of the system's FIT timers is configured to interrupt every 20 ms. When the interrupt occurs the ISR acknowledges the interrupt and *post()*s to the `sem_mouse` semaphore. This causes the scheduler to place the `mouse_comm_thread` in the ready queue to be executed. One of the system's general purpose timers is configured to have a period of 105 μs, which is the minimum time the ADNS-2610 requires between a read and the subsequent transfer. The serial communication routines use the timer to meet this requirement by starting the timer and then polling its interrupt flag.

When the `mouse_comm_thread` executes, it reads the ADNS-2610's Delta_X and Delta_Y registers, converts the raw binary values to floating point, scales them to mm, and calls the *update_position()* function to convert the movement from the probe reference

Table 9.3: SSE register addresses and ADIS16350 read commands

Address	Name	Value	Comment
0xCA000018	TXRX[0]	0x00000200	read gyro supply voltage
0xCA000020	TXRX[1]	0x00000400	read x gyro rate
0xCA000024	TXRX[2]	0x00000600	read y gyro rate
0xCA000028	TXRX[3]	0x00000800	read z gyro rate
0xCA00002C	TXRX[4]	0x00000A00	read x acceleration
0xCA000030	TXRX[5]	0x00000C00	read y acceleration
0xCA000034	TXRX[6]	0x00000E00	read z acceleration
0xCA000038	TXRX[7]	0x00001000	read x temp
0xCA00003C	TXRX[8]	0x00001200	read y temp
0xCA000040	TXRX[9]	0x00001300	read z temp
0xCA000044	TXRX[10]	0x00003C00	read status
0xCA000048	TXRX[11]	0x00003C00	read status
0xCA00004C	TXRX[12]	0x00003C00	read status
0xCA000050	TXRX[13]	0x00003C00	read status
0xCA000054	TXRX[14]	0x00003C00	read status
0xCA000058	TXRX[15]	0x00003C00	read status

frame to the navigation reference frame. The *update_position()* function implements the coordinate transformation part of the navigation algorithm, which was described in Chapter 8.3. The result of the transformation is a motion vector that describes the linear motion of the probe over the last sample interval, expressed in the navigation frame of reference. The transformed motion vector is then added to a running sum that keeps track of the probe's position relative to the start of the scan. After the position update is complete, the `mouse_comm_thread` goes back to *wait()*ing until the next interrupt occurs.

The `gyro_comm_thread` handles communication with the ADIS16350 gyroscope and implements the DCM update part of the inertial navigation algorithm. It has the highest priority in the system which means it can only be preempted by interrupts. When the thread first starts it configures the SSE and initializes the ADIS16350. The SSE has 16 TX Command registers that store the SPI commands used to read data from the ADIS16350's sensor data registers. Table 9.3 lists the register names, absolute addresses, and the actual binary command values. Only the first ten command slots are used for real data. The last six are filled with the read status command to prevent invalid commands from being issued. After initializing the SSE, the `gyro_comm_thread` enters an infinite loop and *wait()*s on the `sem_sspi` semaphore.

Once every sample period, which is approximately every 1.2 ms, the SSE interrupts to signal that new data is available. The ISR copies the result data from the SSE's RX regis-

ters to a buffer in main memory, acknowledges the interrupt, and *post()*s to the `sem_sspi` semaphore. The `gyro_comm_thread` is immediately scheduled by the kernel because it has the highest priority. The thread first checks if it is in calibration mode or not. If it is in calibration mode it calls the *calibrate()* routine, otherwise it continues down the normal execution path. When not in calibrate mode, the raw angular rate values from the ADIS16350 are converted to floating point and scaled to real units. Then *update_rotation_matrix()* is called to perform the DCM update. The DCM update algorithm was described previously in Chapter 8.4. After the DCM has been updated the values of the equivalent Euler angles are computed using (8.8). This is necessary because Stradwin requires that the orientation be represented as Euler angles. When the conversion is complete, the `gyro_comm_thread` goes back to *wait()*ing on `sem_sspi`.

9.4 Calibration

Before every scan the operator must calibrate the system to compensate for the zero offset and bias in the gyroscopes and to reset the position and orientation to zero. A detailed description of this process is available in Appendix A, what is presented here is a description of the algorithm. The routine averages the sensor output over 2500 samples and stores the result individually for each of the gyroscopes. During normal operation the offset value is subtracted from each new sample before it is used by the INS algorithm. After the last calibration sample has been received, the position and orientation values are reset. The position values reset to zero and the DCM resets to the identity matrix.

Chapter 10

Performance Testing

The purpose of these tests was to evaluate the tracking and 3D volume reconstruction performance of the overall system. A set of tests were devised to provide performance information on individual motion axes. The tests were performed separately on the direct and indirect version of the tracking system so that the performance of the two versions could be compared. Each test was carried out by scanning a CIRS model 044 ultrasound phantom. The phantom contains inclusions of known sizes and positions that were compared with the test results. Data was acquired using Stradwin 3D ultrasound software. Stradwin was also used to manually segment the inclusions. The volume of the segmented inclusions and the accuracy of the segmented 3D surface were compared against the ground truth to yield the performance estimates.

10.1 Test Apparatus

The test apparatus was composed of the tracking system, a Terason t3000 Firewire ultrasound system with a 7L3 linear array transducer, Stradwin 3D ultrasound software, and a CIRS model 044 ultrasound resolution phantom. During testing, Stradwin communicated with the tracking system to get position and orientation information, while simultaneously acquiring ultrasound images from the Terason system. Stradwin displayed the data in real

time, stored it for later analysis, and was also used to perform the manual segmentations necessary to analyze the data.

The CIRS model 044 ultrasound phantom is composed primarily of a water based polymer called zyrdine. This polymer was designed to have ultrasound propagation characteristics very similar to that of soft tissue. The phantom has 4 groups of cylindrical targets with dimensions of 1.5 mm x 2.4 mm, 3 mm x 3 mm, 3 mm x 6 mm, and 12 mm x 18 mm. The targets range in depth from 10 mm to 150 mm. The 12 mm x 18 mm target group at a depth of 15 mm was used for this testing. A single cylindrical inclusion, with an attenuation of -6 dB compared to the surrounding medium, was used for all tests.

The surface of the CIRS phantom is quite smooth and prevented the tracking system from functioning. A random pattern was printed on a sheet of plastic like that used for overhead transparencies. A thin layer of ultrasound gel was applied to the phantom and then the plastic sheet was placed over it, in such a way as to ensure that there were no air bubbles trapped between the plastic and the surface of the phantom. Side by side comparison indicated that the presence of the plastic did not have a significant impact on the quality of the ultrasound images. The tracking pattern is visible on the surface of the phantom in Figure 10.1.

10.2 Experimental Procedure

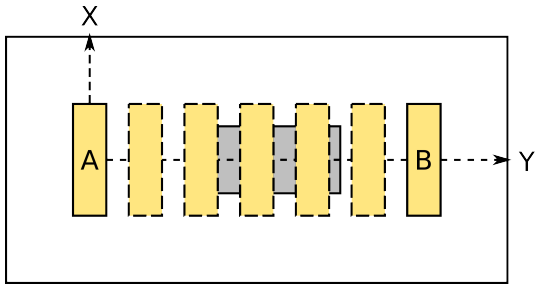
The experiments were designed to evaluate the tracking system's performance in individual degrees of freedom. In addition, the overall motion of the probe during any particular test was limited so as to mimic the motion the probe might undergo during a typical scan in a clinical environment. It is important to note that, because the scanning was performed by hand, there was always some variation present in each degree of freedom. However, great care was taken so that the variation was primarily in the degree of freedom being tested. Five different tests were devised, each isolating a particular type of motion. Each test was repeated ten times to provide a basis for statistical analysis. The test suite was performed individually on the direct and indirect versions of the tracking system. Figure

10.1 illustrates the degrees of freedom that were tested with respect to the probe and the phantom.

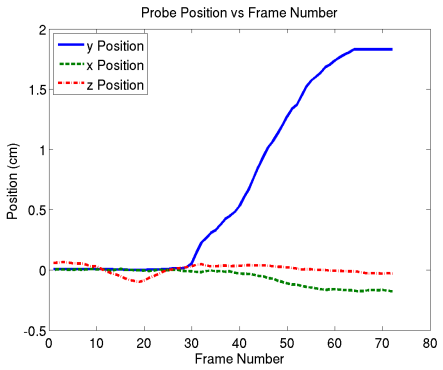
Figure 10.1: Reference coordinate system relative to the ultrasound phantom. The x and y axes illustrated here correspond with those used throughout the testing.

The following tests were performed:

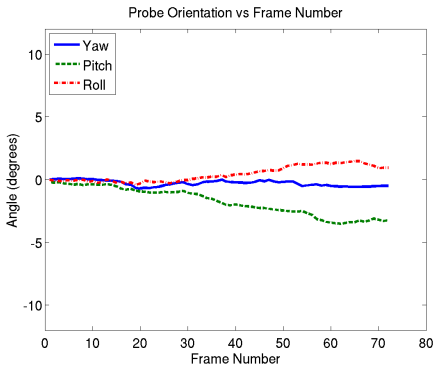
- Test 1 consisted of uniform movement along the y-axis. The yaw and pitch angles as well as the position along the x-axis were held as close to constant as possible. Figure 10.2(a) illustrates the scan path. The yellow box represents the transducer and the gray box represents the inclusion in the phantom. The scan starts at point A and ends at point B. The yellow boxes with the dashed borders represent the intermediate probe positions. They are evenly spaced in this figure to illustrate that the rate of change along the y-axis was relatively uniform. Figure 10.2(b) is a graph of the probe's x, y, and z position for each image frame in the dataset. The blue line in Figure 10.2(b) demonstrates that the linear motion in test 1 was primarily in the Y direction. Figure 10.2(c) is a graph of the probe's yaw, pitch, and roll angle versus image frame number. The rotational motion indicated by this plot was not intentional and represents the operator's best effort not to rotate the probe.



(a) Graphical representation of the test 1 scanpath



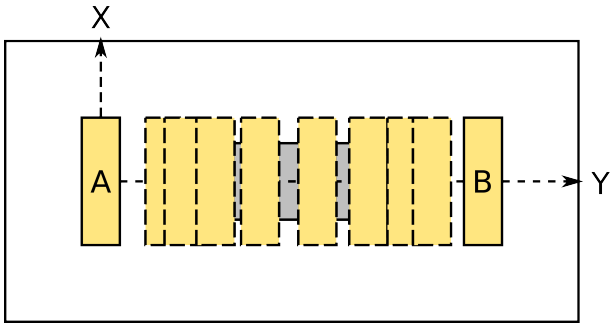
(b) Test 1 linear motion profile



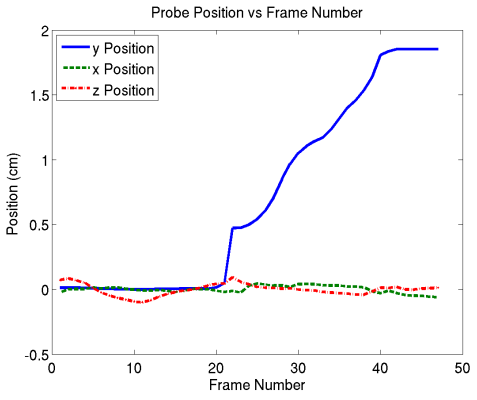
(c) Test 1 rotational motion profile

Figure 10.2: Sample test 1 motion profiles

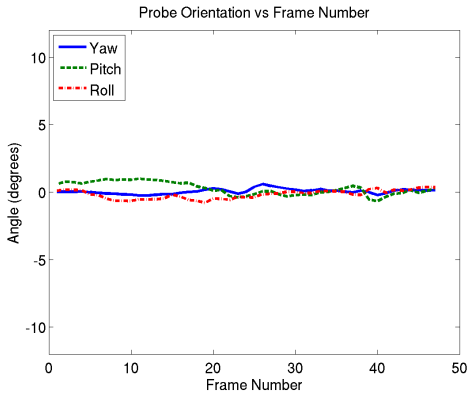
- Test 2 consisted of movement along the y-axis at a variable rate. The yaw and pitch angles as well as the position along the x-axis were held as close to constant as possible. Point A in Figure 10.3(a) represents the probe position at the start of the scan and point B represents the probe position at the end of the scan. Here, the dashed boxes representing the intermediate probe positions are unevenly spaced to illustrate that the rate of change along the y-axis was variable. The variation is also visible in Figure 10.3(b), where 3 distinct slopes are present in the plot of y position. As in the previous example, the rotational motion illustrated in Figure 10.3(c) was not intentional.



(a) Graphical representation of the test 2 scanpath



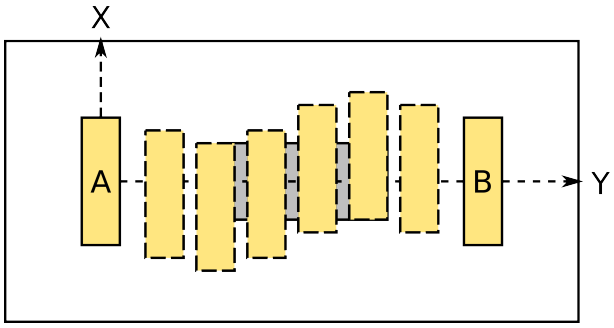
(b) Test 2 linear motion profile



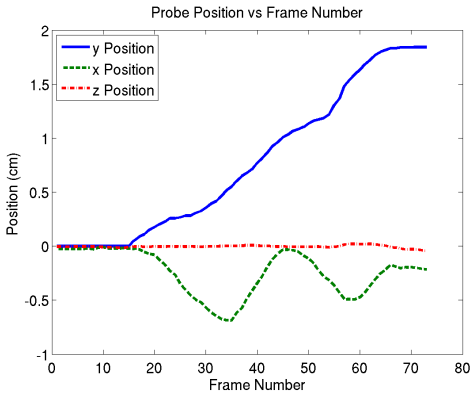
(c) Test 2 rotational motion profile

Figure 10.3: Sample test 2 motion profiles

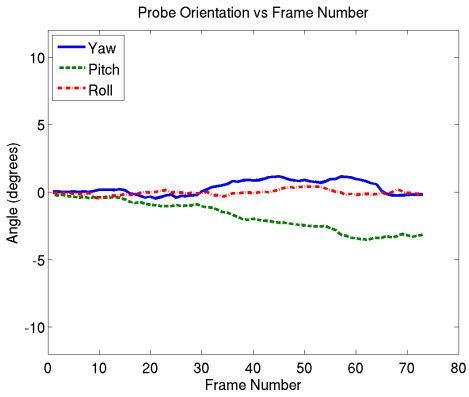
- Test 3 consisted of uniform movement along the y-axis combined with variation in position along the x-axis. The yaw and pitch angles were held as close to constant as possible. Point A in Figure 10.4(a) represents the starting position of the probe and point B represents the ending position of the probe. The dashed boxes representing the intermediate probe positions are shifted up and down along the x-axis as the scan progresses, to illustrate the motion of the probe. The x-axis variation also appears as the green line in Figure 10.4(b), in addition to the y-axis variation present in the previous examples. Once again, the rotational motion indicated by Figure 10.4(c) was incidental.



(a) Graphical representation of the test 3 scanpath



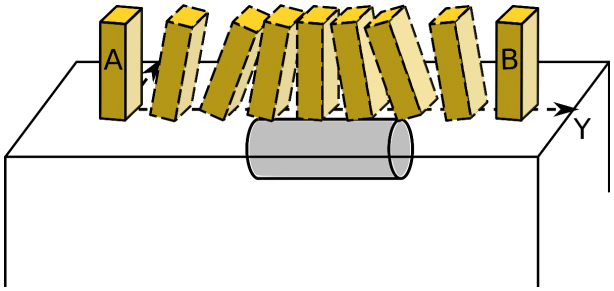
(b) Test 3 linear motion profile



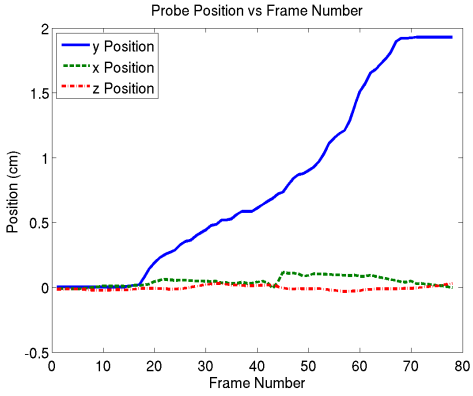
(c) Test 3 rotational motion profile

Figure 10.4: Sample test 3 motion profiles

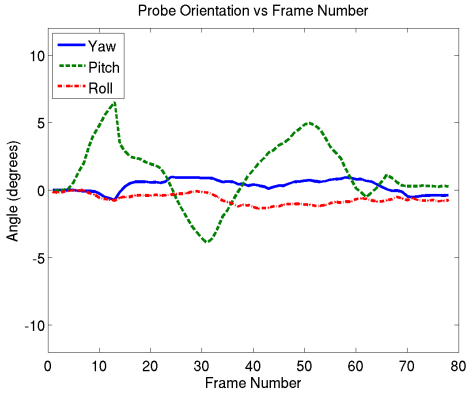
- Test 4 consisted of uniform movement along the y-axis combined with variation in the pitch angle. The yaw angle and the position along the x-axis were held as close to constant as possible. Figure 10.5(a) is a 3D version of the previous scan path illustrations. The yellow box represents the probe, which moves from point A to point B over the course of the scan. The boxes with the dashed outlines represent the intermediate position of the probe, and illustrate the angular motion about the probe's pitch axis. Figure 10.5(b) illustrates that the linear motion in this test was predominantly in the y direction and was relatively constant. The variation in pitch angle is also indicated by the green line in Figure 10.5(c).



(a) Graphical representation of the test 4 scanpath



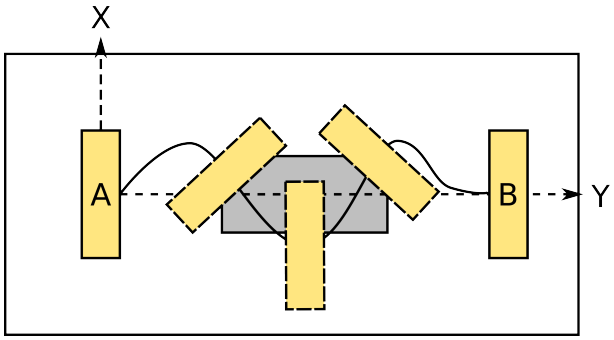
(b) Test 4 linear motion profile



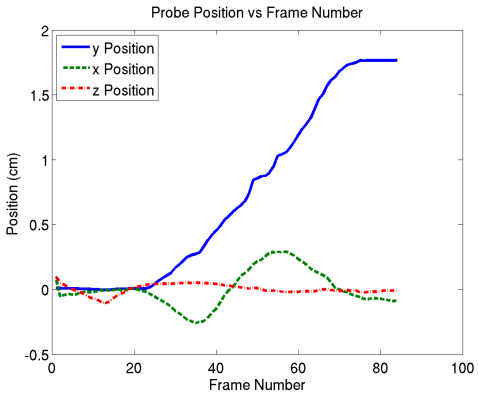
(c) Test 4 rotational motion profile

Figure 10.5: Sample test 4 motion profiles

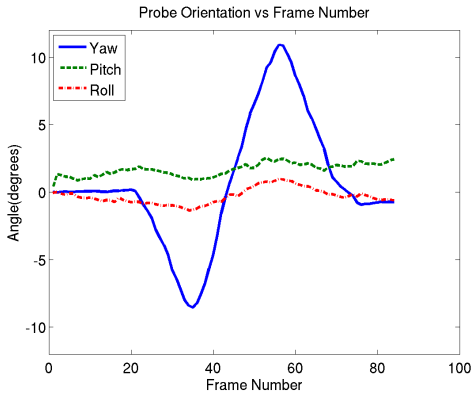
- Test 5 consisted of movement along both the x and y axes combined with variation in the yaw angle. The pitch angle was held as close to constant as possible. As illustrated in 10.6(a), the path of the probe was curved as it moved from point A to point B. The boxes with the dashed outlines illustrate the rotation of the probe at different points during the scan. The y and x variation are also clearly visible in Figure 10.6(b), as the green and blue lines, respectively. The variation in yaw angle is illustrated by the blue line in Figure 10.6(c).



(a) Graphical representation of the test 5 scanpath



(b) Test 5 linear motion profile



(c) Test 5 rotational motion profile

Figure 10.6: Sample test 5 motion profiles

At the beginning of each test run the system was calibrated to compensate for the gyroscope bias and to reset the position and orientation of the tracking system to zero. When

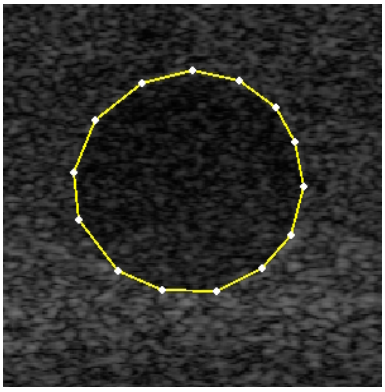
the calibration was complete Stradwin was set to record data and the test run commenced. Once the data displayed on the screen indicated that the entire inclusion had been scanned the test was terminated and the data saved.

10.3 Segmentation Technique and Performance Metrics

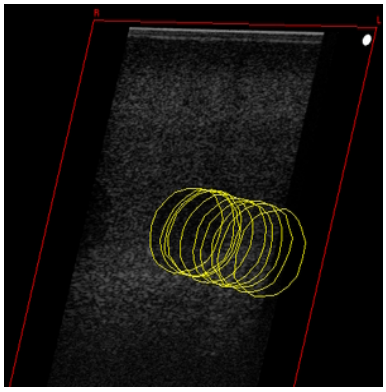
Two metrics were used to rate the reconstruction performance of the system: volume accuracy and surface accuracy. The volume accuracy metric compares the volume of the segmented region to the known volume of the inclusion in the CIRS phantom. The surface accuracy metric describes how closely the surface of the segmented region matches a reference surface. In this case, the reference surface is an artificially generated surface with dimensions matching those of the inclusion in the CIRS phantom.

The first step in calculating both metrics was to manually segment the data. This was performed within Stradwin. Contours were drawn around the region of interest on the original B-scans. This process is illustrated in Figure 10.7(a), where the white points were placed by hand and the yellow lines connecting them are generated automatically. This process was repeated every few image frames. Figure 10.7(b) illustrates a series of contours prior to surface generation. Once the desired region was contoured, the operator clicked the *Update All* command to have Stradwin fit a surface around the contours. Stradwin offers several levels of smoothing strength that can be applied to the surface. The "Very Little" setting was used for this work to preserve as many of the segmented features as possible while still maintaining the smoothness of the circular contours. Figure 10.7(c) illustrates the surface that was generated to match the contours drawn in Figure 10.7(b). Since a contour was not drawn on every B-scan, it was possible that the automatically generated contour did not match the underlying structure well. As illustrated in Figure 10.7(d), the outline of the surface appears on the b-scans that were not contoured by hand. This allowed the operator to review the surface and make corrections as necessary.

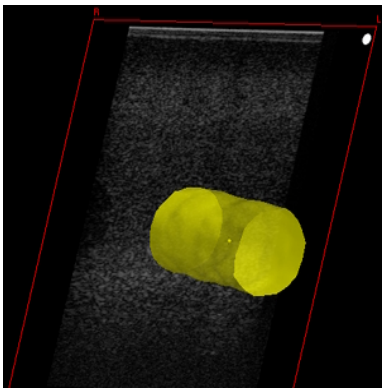
The volume enclosed within the surface is displayed in the Stradwin user interface. The volume of each segmentation was recorded and compared to the known volume of the



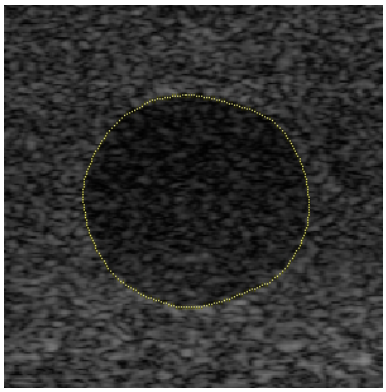
(a) Contour drawn on an individual B-scan



(b) Complete sequence of contours prior to surface generation



(c) Surface generated to fit contours



(d) Intersection of the 3D Surface with a 2D B-scan

Figure 10.7: Stradwin manual segmentation process

inclusion, which was calculated using the well known formula for the volume of a cylinder in (10.1).

$$Volume\ of\ Cylinder\ =\ \pi \cdot radius^2 \cdot length = (6mm)^2 \cdot 18mm = 2035.75mm^3 = 2.035ml \tag{10.1}$$

The difference between the segmented volume and that of the ground truth was then represented as an error percentage relative to ground truth using (10.2).

$$\%\ Error = \left| \frac{Segmented\ Volume - Ground\ Truth\ Volume}{Ground\ Truth\ Volume} \right| \cdot 100 \tag{10.2}$$

By itself, volume accuracy is not enough to quantify reconstruction performance. This is because two different objects of equal volume would achieve a perfect volume accuracy. J.D. Quatararo developed the surface accuracy metric to deal with this issue [28]. The surface accuracy metric characterizes the degree of similarity between two objects by comparing the RMS distance between points on their surfaces. The two surfaces are first aligned using the Iterative Closest Point (ICP) algorithm. The IPC algorithm finds the affine transformation between the two point sets that minimizes the squared distance between points in the segmentation result and the ground truth. The transformation is then applied to the data [29].

To compute the surface accuracy, the Euclidean distance from every point in the segmentation result, S_i, to every point in the ground truth point set, G_i, is calculated and the smallest one is recorded. This is expressed as an equation in (10.3), where $dist(x, y)$ is the Euclidean distance given by (10.4).

$$d_i = min_{j \in J} \left(dist(S_i, G_j) \right) \tag{10.3}$$

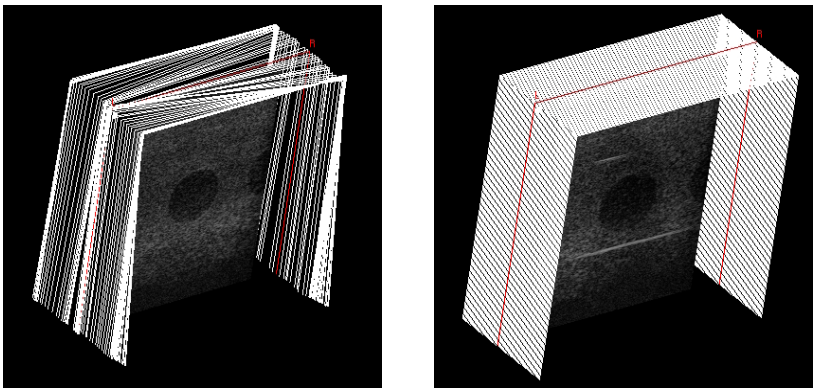
$$dist(a, b) = \sqrt{(x_A - x_b)^2 + (y_A - y_b)^2 + (z_A - z_b)^2} \tag{10.4}$$

The average RMS distance between the surfaces is then calculated using (10.5).

$$RMS = \sqrt{\frac{1}{I} \sum_{i=1}^{I} d_i^2} \tag{10.5}$$

To perform the calculation, the two objects must each be reduced to a set of points on the surface of the object. A ground truth cylinder was generated with dimensions corresponding to the known size of the inclusion in the CIRS phantom. The segmentations that were performed in Stradwin were exported as VRML files. These files contain the list of x, y, and z coordinates that are required for the calculation as well as a lot of other information. A Matlab script was used to parse the VRML files, pull out the data, and write out new files in the correct format.

In order to provide a basis for comparison, the volumes were also reconstructed without using the tracking information. To make the comparison fair, the assumption was made that each image was equally spaced between the true starting and ending positions. Otherwise, Stradwin would have placed each image on top of one another. All other degrees of freedom were held constant. Figure 10.8 illustrates the relationship between the scanplanes of the same piece of data, reconstructed with (10.8(a)) and without (10.8(b)) the position and orientation information. After reconstruction, these volumes were manually segmented and processed for volume and surface accuracy in the same manner as the volumes that were reconstructed with position information



(a) (b)

Figure 10.8: 3D volume reconstruction with and without position information

10.4 Results

The results of the testing are presented here in tabular form. The data is split between that recorded using the direct imaging configuration and that recorded using the indirect imaging configuration. The difference is that the indirect version uses a fiber bundle in the optical system and the direct version does not. Please refer back to Section 7.3 for more information. For each version of the system, the absolute volume, volume error as a percentage of the ground truth volume, and surface accuracy are presented.

10.4.1 Volume Accuracy

Tables 10.1 and 10.3 contain the raw volume data in ml for the direct and indirect imaging configurations respectively. Tables 10.2 and 10.4 represent the difference between that data and the ground truth as a percentage. The last 2 rows of each table lists the mean and standard deviation independently for each of the five tests performed.

Table 10.1: Segmentation volume data for direct imaging configuration with position correction

Trial	Test 1 (ml)	Test 2 (ml)	Test 3 (ml)	Test 4 (ml)	Test 5 (ml)
1	2.044	1.982	1.996	2.034	1.790
2	2.047	2.085	1.986	1.954	2.031
3	1.977	1.931	1.852	2.102	1.956
4	2.051	1.971	1.835	2.165	2.203
5	2.038	2.172	2.102	1.941	2.002
6	1.973	1.919	1.941	2.089	1.864
7	2.063	1.934	2.288	2.065	1.863
8	2.046	2.116	2.193	2.130	2.036
9	2.007	2.035	2.019	1.902	1.995
10	1.990	1.964	2.001	1.976	1.938
Mean	2.024	2.011	2.021	2.036	1.968
STD	0.034	0.087	0.141	0.089	0.116

Table 10.2: Segmentation volume error data for direct imaging configuration with position correction

Trial	Test 1 (%)	Test 2 (%)	Test 3 (%)	Test 4 (%)	Test 5 (%)
1	0.405	2.640	1.953	0.086	12.072
2	0.553	2.419	2.444	4.016	0.233
3	2.886	5.146	9.026	3.254	3.918
4	0.749	3.181	9.861	6.349	8.216
5	0.110	6.693	3.254	4.654	1.658
6	3.082	5.735	4.654	2.616	8.437
7	1.338	4.998	12.391	1.437	8.486
8	0.503	3.942	7.724	4.630	0.012
9	1.412	0.037	0.823	6.570	2.002
10	2.247	3.525	1.707	2.935	4.802
Mean	1.329	3.832	5.384	3.655	4.983
STD	1.070	1.922	4.051	2.034	4.127

Table 10.3: Segmentation volume data for indirect imaging configuration with position correction

Trial	Test 1 (ml)	Test 2 (ml)	Test 3 (ml)	Test 4 (ml)	Test 5 (ml)
1	2.115	2.215	2.137	2.106	2.299
2	2.169	2.268	2.020	1.818	2.376
3	2.089	2.351	2.155	2.149	2.217
4	2.005	2.063	1.901	2.064	2.015
5	2.075	2.148	2.095	2.153	2.289
6	2.009	2.146	2.026	2.074	2.035
7	2.119	2.343	2.136	1.859	2.252
8	2.183	2.097	1.985	2.258	2.215
9	2.128	2.087	2.010	2.066	2.081
10	2.121	2.180	1.947	2.162	2.076
Mean	2.101	2.190	2.041	2.071	2.186
STD	0.059	0.103	0.087	0.136	0.125

Table 10.4: Segmentation volume error data for indirect imaging configuration with position correction

Trial	Test 1 (%)	Test 2 (%)	Test 3 (%)	Test 4 (%)	Test 5 (%)
1	3.893	8.805	4.973	3.451	12.931
2	6.545	11.408	0.774	10.696	16.714
3	2.616	15.486	5.858	5.563	8.903
4	1.511	1.338	6.619	1.388	1.019
5	1.928	5.514	2.910	5.759	12.440
6	1.314	5.416	0.479	1.879	0.037
7	4.089	15.093	4.924	8.682	10.623
8	7.233	3.009	2.493	10.917	8.805
9	4.531	2.517	1.265	1.486	2.223
10	4.188	7.086	4.360	6.202	1.977
Mean	3.785	7.567	3.466	5.602	7.567
STD	2.012	5.053	2.189	3.610	5.852

Tables 10.5 and 10.7 contain the raw volume data of the reconstructions performed without position data, for the direct and indirect imaging configurations respectively. Tables 10.6 and 10.8 represent the difference between that data and the ground truth as a percentage. The last 2 rows of each table lists the mean and standard deviation independently for each of the five tests performed.

Table 10.5: Segmentation volume data for direct imaging configuration without position correction

Trial	Test 1 (ml)	Test 2 (ml)	Test 3 (ml)	Test 4 (ml)	Test 5 (ml)
1	1.676	1.687	1.679	2.013	1.540
2	1.567	1.728	1.712	1.722	1.940
3	1.792	1.616	1.545	1.666	1.969
4	1.660	1.360	1.585	1.927	1.871
5	1.589	1.806	1.675	1.701	1.744
6	1.508	1.692	1.597	1.576	1.776
7	1.455	1.533	1.622	1.625	1.630
8	1.506	1.642	1.580	1.960	1.542
9	1.603	1.639	1.645	1.184	1.577
10	1.529	1.766	1.615	1.722	1.589
Mean	1.589	1.647	1.625	1.710	1.718
STD	0.100	0.127	0.052	0.237	0.166

Table 10.6: Segmentation volume error data for direct imaging configuration without position correction

Trial	Test 1 (%)	Test 2 (%)	Test 3 (%)	Test 4 (%)	Test 5 (%)
1	17.672	17.131	17.524	1.118	24.352
2	23.026	15.117	15.903	15.412	4.704
3	11.974	20.619	24.107	18.163	3.279
4	18.458	33.194	22.142	5.342	8.093
5	21.945	11.286	17.721	16.444	14.331
6	25.924	16.886	21.552	22.584	12.760
7	28.528	24.696	20.324	20.177	19.931
8	26.022	19.342	22.387	3.721	24.254
9	21.258	19.489	19.194	41.840	22.535
10	24.893	13.251	20.668	15.412	21.945
Mean	21.970	19.101	20.152	16.021	15.618
STD	4.899	6.257	2.552	11.642	8.132

Table 10.7: Segmentation volume data for indirect imaging configuration without position correction

Trial	Test 1 (ml)	Test 2 (ml)	Test 3 (ml)	Test 4 (ml)	Test 5 (ml)
1	1.990	1.793	1.810	1.623	2.103
2	2.011	1.717	1.615	2.033	2.247
3	1.813	1.988	1.705	2.133	2.460
4	1.734	1.713	1.660	1.687	1.680
5	1.696	1.824	1.813	1.911	1.964
6	1.468	1.790	1.685	1.707	1.836
7	1.699	1.910	1.773	1.925	2.373
8	1.914	1.754	1.847	1.883	2.035
9	1.790	1.858	1.829	1.629	2.053
10	1.770	1.985	1.830	1.729	2.132
Mean	1.788	1.833	1.757	1.826	2.088
STD	0.160	0.101	0.083	0.176	0.235

Table 10.8: Segmentation volume error data for indirect imaging configuration without position correction

Trial	Test 1 (%)	Test 2 (%)	Test 3 (%)	Test 4 (%)	Test 5 (%)
1	2.247	11.924	11.089	20.275	3.303
2	1.216	15.658	20.668	0.135	10.377
3	10.942	2.346	16.247	4.777	20.840
4	14.823	15.854	18.458	17.131	17.475
5	16.689	10.402	10.942	6.128	3.525
6	27.889	12.072	17.230	16.149	9.812
7	16.542	6.177	12.907	5.440	16.566
8	5.981	13.840	9.272	7.503	0.037
9	12.072	8.732	10.156	19.980	0.847
10	13.054	2.493	10.107	15.068	4.728
Mean	12.145	9.950	13.708	11.259	8.751
STD	7.837	4.948	4.086	7.229	7.444

Figures 10.9 and 10.10 compare the volume accuracy of reconstructions performed with and without position information, for the direct and indirect imaging configurations respectively. For the direct imaging configuration the percent volume error ranged between 1.3% and 5.3% with the position information, versus 15.6% to 21.9% without. This represents a 3.1x improvement in the worst case (test 5) and 16.5x improvement in the best case (test 1). For the indirect imaging configuration the percent volume error ranged between 3.7% and 7.6% with the position information, versus 8.7% to 13.7% without. This represents a 1.2x improvement in the worst case (test 5) and 4.0x improvement in the best case (test 3).

Figure 10.11 compares the volume accuracy of segmentations performed on data from direct and indirect imaging configurations. This comparison provides information on how the two configurations performed relative to each other. With the exception of test 3, the direct imaging configuration outperformed the indirect configuration by 2-4%.

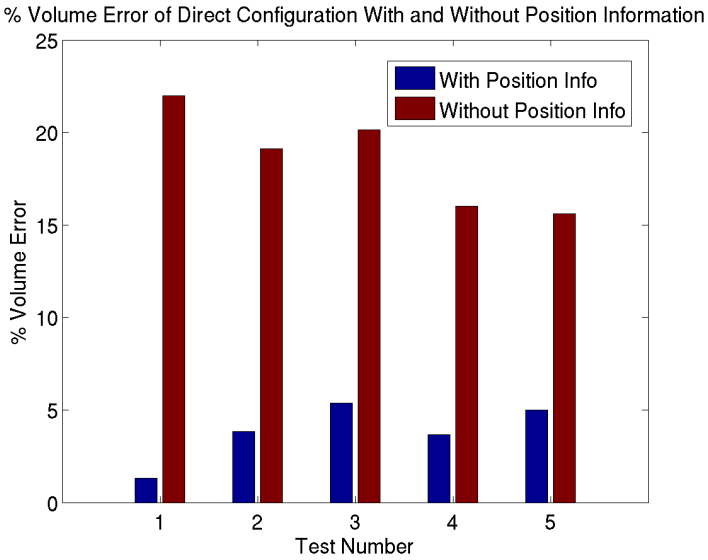


Figure 10.9: Comparison of volume accuracy with and without position information for the direct imaging Configuration

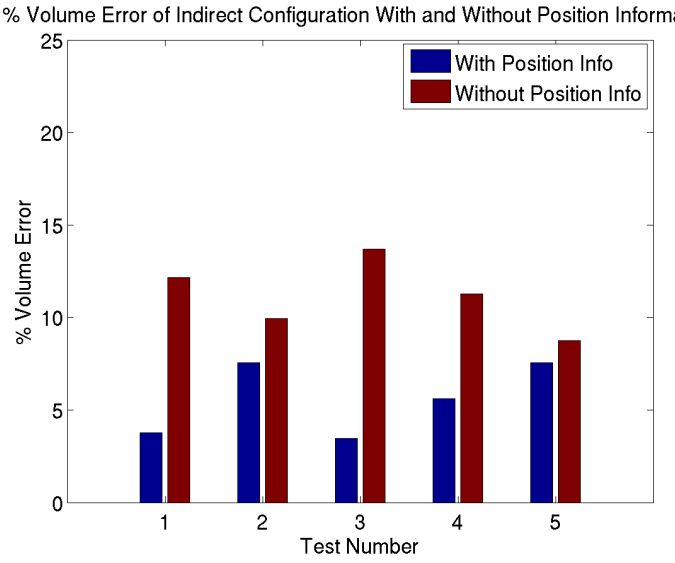


Figure 10.10: Comparison of volume accuracy with and without position information for the indirect imaging Configuration

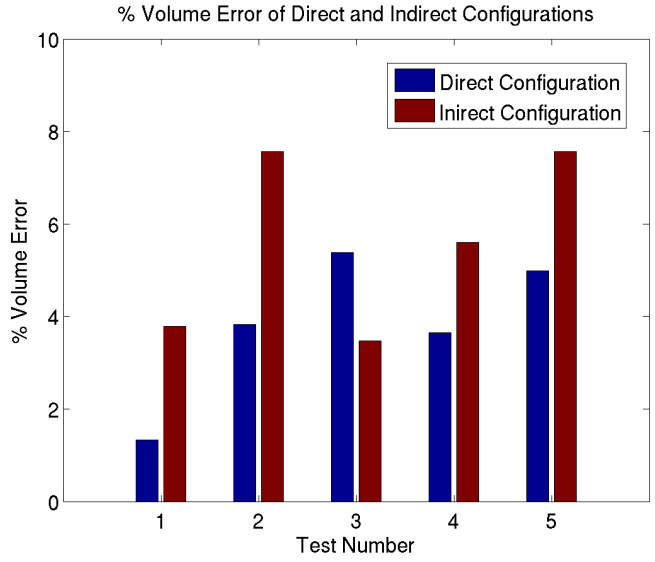


Figure 10.11: Comparison of volume accuracy for the direct and indirect imaging configurations, with position information

10.4.2 Surface Accuracy

Tables 10.9 and 10.10 list the surface accuracy data for the direct imaging configuration, with and without position information. Tables 10.11 and 10.12 list the surface accuracy data for the indirect imaging configuration, with and without position information. Figures 10.12 and 10.13 present the same data in graphical form. As expected, the reconstructions performed with the position information had significantly lower RMS surface error than those done without position information. This true for both the direct and indirect imaging configurations.

Table 10.9: Surface accuracy data for direct imaging configuration with position correction

Trial	Test 1 (mm)	Test 2 (mm)	Test 3 (mm)	Test 4 (mm)	Test 5 (mm)
1	0.283	0.318	0.546	0.349	0.411
2	0.332	0.365	0.396	0.369	0.289
3	0.323	0.340	0.389	0.398	0.553
4	0.291	0.290	0.365	0.462	0.462
5	0.338	0.374	0.266	0.424	0.476
6	0.349	0.378	0.477	0.337	0.437
7	0.314	0.394	0.447	0.334	0.612
8	0.318	0.363	0.312	0.464	0.350
9	0.332	0.264	0.423	0.384	0.302
10	0.312	0.295	0.354	0.374	0.730
Mean	0.319	0.338	0.397	0.390	0.462
STD	0.021	0.044	0.081	0.047	0.139

Table 10.10: Surface accuracy data for direct imaging configuration without position correction

Trial	Test 1 (mm)	Test 2 (mm)	Test 3 (mm)	Test 4 (mm)	Test 5 (mm)
1	0.622	0.746	1.768	0.423	0.881
2	0.754	0.622	1.105	0.638	0.510
3	0.526	0.786	1.617	0.674	0.471
4	0.716	0.994	1.316	0.535	0.523
5	0.780	0.609	1.293	0.681	0.754
6	0.853	0.787	1.195	0.919	0.656
7	0.938	0.904	1.212	0.771	0.774
8	0.767	0.732	0.867	0.476	1.077
9	0.872	0.712	0.792	1.039	1.099
10	0.830	0.666	1.447	0.626	0.814
Mean	0.766	0.756	1.261	0.678	0.756
STD	0.122	0.120	0.304	0.191	0.223

Table 10.11: Surface accuracy data for indirect imaging configuration with position correction

Trial	Test 1 (mm)	Test 2 (mm)	Test 3 (mm)	Test 4 (mm)	Test 5 (mm)
1	0.307	0.397	1.002	0.395	0.693
2	0.423	0.409	0.459	0.539	0.773
3	0.253	0.748	0.605	0.408	0.594
4	0.339	0.289	0.427	0.344	0.338
5	0.305	0.330	0.334	0.481	0.571
6	0.306	0.372	0.349	0.355	0.396
7	0.325	0.615	1.007	0.405	0.451
8	0.345	0.379	1.139	0.516	0.370
9	0.309	0.303	1.388	0.393	0.339
10	0.352	0.475	1.026	0.377	0.273
Mean	0.326	0.432	0.774	0.421	0.480
STD	0.044	0.145	0.380	0.067	0.169

Table 10.12: Surface accuracy data for indirect imaging configuration without position correction

Trial	Test 1 (mm)	Test 2 (mm)	Test 3 (mm)	Test 4 (mm)	Test 5 (mm)
1	0.400	0.625	1.767	0.638	0.442
2	0.400	0.698	1.213	0.440	1.327
3	0.495	0.402	1.333	0.476	0.884
4	0.623	0.638	1.591	0.656	0.713
5	0.680	0.618	1.193	0.509	0.460
6	0.950	0.523	1.149	0.690	1.095
7	0.685	0.411	2.377	0.445	0.650
8	0.481	0.596	2.236	0.498	0.507
9	0.572	0.450	1.916	0.762	0.426
10	0.598	0.424	1.797	0.735	0.447
Mean	0.588	0.538	1.657	0.585	0.695
STD	0.164	0.110	0.439	0.124	0.314

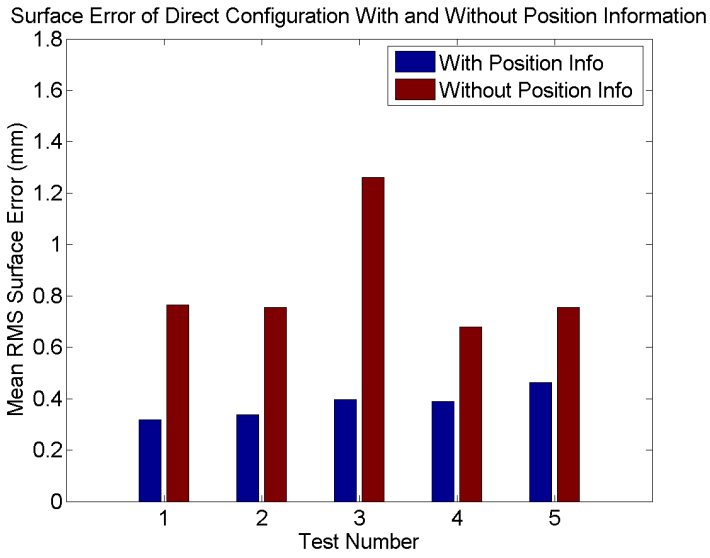


Figure 10.12: Comparison of surface accuracy with and without position information for the direct imaging configuration

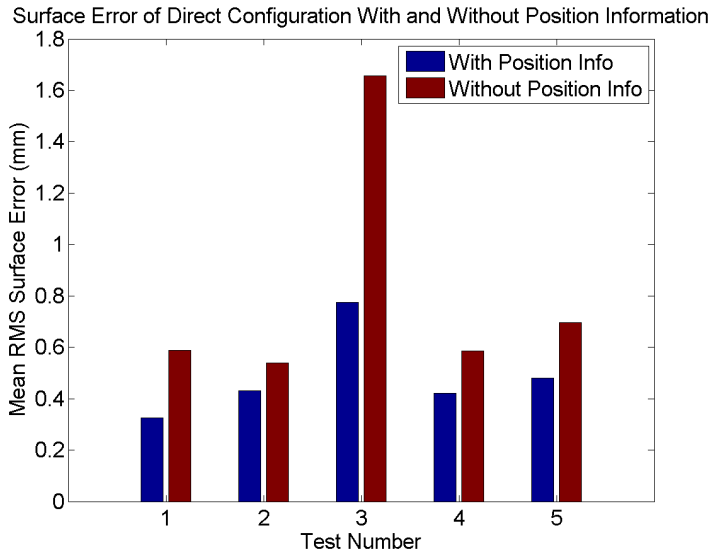


Figure 10.13: Comparison of surface accuracy with and without position information for the indirect imaging configuration

Figure 10.14 compares the surface accuracy of the direct and indirect imaging configurations with position information. The graph indicates that, with the exception of test 3, the direct configuration outperformed the indirect configuration by a small margin. This result agrees well with the volume error data presented previously, where the direct configuration outperformed the indirect configuration by a 2-4% margin.

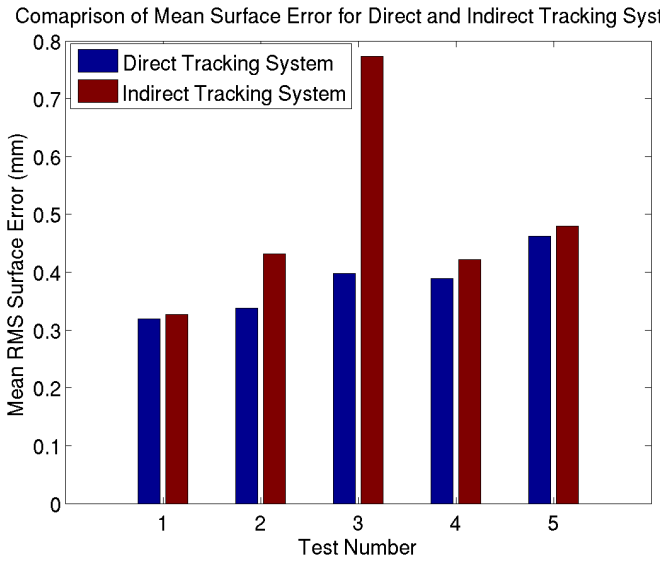


Figure 10.14: Comparison of surface accuracy for the direct and indirect imaging configurations with position information

10.5 Discussion

The data indicate that reconstructions using data from the tracking system were significantly more accurate than those that did not use the tracking information. Overall, the two versions of the tracking system were quite comparable. Although there was a 2-4% difference between them using volume accuracy as the metric, their relative performance using the surface accuracy metric was very close. The discrepancy can most likely be explained by the dimensionality of the metrics. The volume is measured in ml or mm^3, with 1 ml being equal to 1000 mm^3. The surface accuracy is measured in mm. Inspection of Figure 10.14 indicates that the difference in surface accuracy was about 0.5 mm on average, excluding test 3. Table 10.13 lists the possible effects of a 0.5 mm error in either the radius or length of the cylinder. The first line of the table is the ground truth volume.

As one would expect, variations in the radius have a more significant impact than variations in length. Table 10.13 indicates that the small differences in surface accuracy between the two versions of the tracking system could easily account for the larger relative difference

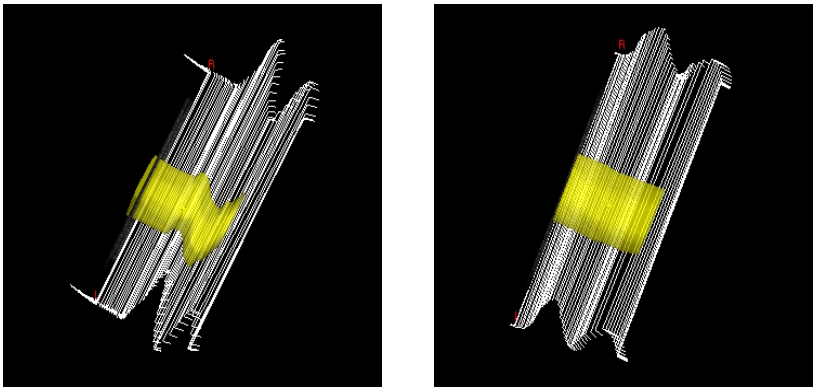
Table 10.13: Possible effects of 0.5 mm error in the radius or length of a cylinder

Radius (mm)	Length (mm)	Volume (mm^3)	Volume (ml)	% Volume Error
6.0	18.0	2035.75	2.035	0.0
5.5	18.0	1710.59	1.711	16.0
6.5	18.0	2389.18	2.389	17.4
6.0	17.5	1979.20	1.979	2.7
6.0	18.5	2092.30	2.092	2.7

in volume error. Because surface accuracy is an RMS value, a 0.5 mm error is unlikely to be entirely positive or negative, as was assumed in this example. It is more likely that the error would be distributed more evenly between positive and negative values. In that case, the volume errors in column five of Table 10.13 would be smaller and closer to the 2-4% range observed in the volume error measurements.

Figure 10.14 indicates that there is a serious discrepancy in performance between the direct and indirect versions of the system for test 3. However, this difference is only observed using the surface accuracy metric. The volume accuracy data for test 3 does not indicate such a large disparity. This is a situation where, because of the characteristics of the test, volume accuracy fails as a performance metric. Test 3 consisted of linear motion in both the y and x directions. Although the motion was some of the most severe, in a relative sense, it was not the kind that would significantly alter the volume. Because the transducer remained perpendicular to the cylindrical inclusion for the entire scan, the cross sectional area of the inclusion in the individual images remained constant. Figure 10.15 illustrates test 3 reconstructions performed using the direct and indirect versions of the system. The volumes are very similar but the surfaces are not.

Although figure 10.15 illustrates an extreme example, the indirect configuration was clearly less accurate than the direct version. Because the orientation sensors were the same in both cases the difference must be in the optics. This conclusion is supported by the fact that the larger performance disparity was in test 3, which also had the most linear motion. It is likely that the indirect imaging system was not as well calibrated as the direct imaging system. The indirect imaging system is also much more complex than the direct version. As a result, it is not surprising that it did not perform as well. The contrast of the overall



(a) (b)

Figure 10.15: Test 3 reconstructions using the indirect and direct imaging systems

system was not as good and there were more places where small errors could be introduced.

There is another source of error that affected both versions of the optical tracking system. Both versions were mounted on the outside of the ultrasound transducer, which means that they were approximately 1 cm away from the center of rotation of the transducer's pitch axis. As a result, when the pitch angle varied, the distance between the optical tracker and the tracking surface also varied. This distance affects the magnification of the optical system. The scale factor that relates raw sensor output to a physical distance is a function of magnification. Therefore, pitch variation can cause errors in the distance measured by the sensor. This problem can be solved in future implementations by moving the optical tracker inside the transducer housing so that the axis of the optical system is aligned with the pitch axis' center of rotation.

Chapter 11

Conclusion

In this thesis, a freehand 3D ultrasound system was developed. The system functions in real time and the ultrasound probe is tracked in 5 DoF using only sensors attached directly to it. Early in the work, a great deal of time was spent testing linear accelerometers and MEMS gyroscopes. The linear accelerometer testing proved that a fully inertial 6 DoF tracking system was not possible. The gyroscope experiments provided the drift rate data that was needed to understand what kind of tracking performance was attainable. The miniaturized optical tracking system is one of the major accomplishments of this work. The optical system was rigorously characterized which allowed the optimal operating point to be identified. Fine mechanical components were then fabricated to house and align the optical components. Another major accomplishment is the implementation of the navigation computer. An embedded computer system composed of both standard and custom logic was developed to meet the performance requirements. The firmware running on the embedded computer implemented the inertial navigation algorithm.

This work has demonstrated that it is feasible to implement a tracking system for freehand 3D ultrasound in which all of the sensors are contained within the ultrasound probe. Although the components were mounted on the outside of the probe in the prototype, the sensors are clearly small enough to fit inside the probe if the transducer housing were slightly modified. The prototype also achieved the goal of being inexpensive. It cost less than $1000

to construct the prototype and that cost would be significantly reduced if mass produced. Commercial tracking systems typically cost several thousand dollars.

The ADIS16350 proved to be a good choice for the angular sensing device. It would have been nearly impossible to fabricate a 3-axis gyroscope in a package that size ourselves. In addition, the digital interface greatly simplified the overall hardware design. The sensor's accuracy is more than adequate for our application, but its drift rate limits the duration of a scan to between 10 and 20 seconds with reasonable accuracy. However, there are no competitive products that have significantly better performance in this respect. We can therefore say that the angular drift performance of our system is near the maximum achievable using only gyroscopes, with the current technology. Another important thing to remember is that an average ultrasound scan only lasts between 10 and 20 seconds, so the drift problem does not seriously detract from the system's usefulness.

Both versions of the optical tracking system were able to track the skin surface effectively. However, the version with the optical fiber did not perform as well as the version without it. Based on the data in Chapter 7, I believe that the performance with the fiber can be made to match the performance of the version without the optical fiber, with additional refinement of the calibration constants. The contrast data between the two versions is quite similar and visual inspection of the experimental images confirms that they have comparable image quality. One major source of error in both systems is the fact that the optical axis is not aligned with the pitch axis of the probe. This is because the optical tracker had to be mounted externally. As the pitch angle varies, so does the object distance, which causes the magnification to vary. Since the factor that relates distance as measured by the optical sensor to a physical distance is a function of magnification, pitch variation induces errors in the measured linear movement. This problem can be easily addressed by moving the tracking sensor inside the transducer housing, aligned with the pitch axis.

Using an FPGA to implement the navigation computer provided a great deal of flexibility during the implementation process. It was difficult to anticipate all of the resources that would be required. The Xilinx EDK allowed peripheral modules and Microblaze processor features to be added to the system as necessary. The ability to implement custom hardware

in the FPGA was critical to the success of this project. The overhead of communicating with the ADIS16350, using the 8 bit SPI peripheral supplied by EDK, would have prevented the system from operating in real time. Because an FPGA was used, I was able to implement a custom serial port that handled most of the communication without processor intervention. The XMK real time operating system was another critical tool. The navigation computer had to perform many tasks in parallel and in real time. The XMK provided solid multi-threaded firmware framework that allowed me to concentrate on intricacies of the tracking algorithm itself.

This system would not have worked without the Stradwin 3D ultrasound software that was used for system control and 3D reconstruction. Without it we would have essentially had to write our own reconstruction software from scratch. Our experience with CustusX, described in Chapter 2, illustrates just how challenging that would have been. Without Stradwin, this work would have resulted in a bare tracking system. With Stradwin, I was able to construct a freehand 3D ultrasound system that functions in real time.

11.1 Future Work

The most obvious shortcoming of the prototype tracking system is that it tracks in only 5 degrees of freedom. Therefore, sensor technologies capable of adding the 6th degree of freedom are natural choices for further research. I wish I could suggest some candidates, but I am not optimistic that anything short of a fully inertial tracking system will be able to achieve this goal.

The development of a probe that fully integrates the transducer and all of the tracking sensors would greatly improve the system performance. This is true because of the optical phenomenon explained in the previous section and also because having all of the sensors permanently fixed in position would allow for much better system level calibration. Such a housing would also allow from much greater freedom of movement and allow the system to be effectively tested on human subjects.

Gyroscope drift compensation is another area that can be improved by further work.

Even when a scan is taking place, gravity is the dominant force sensed by the linear accelerometers. The gravity vector can be used to estimate the orientation of the probe and is not subject to drift like the gyroscopes are. A more advanced drift compensation algorithm should be able to use the data from the accelerometers to improve the overall angular accuracy.

Bibliography

[1] G. Frey and R. Chiao, "4z1c real-time volume imaging transducer," 2008.

[2] "Acuson s2000 ultrasound system transducers," 2008.

[3] J. Beutel, J. M. Fitzpatrick, S. Horii, Y. Kim, H. Kundel, M. Sonka, and R. V. Metter, *Handbook of Medical Imaging*. SPIE Press, 2000.

[4] J. Green and D. Krakauer, "New iMEMS angular-rate-sensing gyroscope," *Analog Dialogue*, vol. 37, 2003. Analog Dialogue is a journal published by Analog Devices.

[5] Gyration, Inc., Minatec BHT, 7 parvis Louis Neel, 38040 Grenoble cedex 09, France, *MicroGyro MG1101*, March 2005. Document# DE01300-001

[6] Analog Devices, Inc., One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, *iSensor Evaluation Tools*, June 2008. Marketing Presentation.

[7] "Analog devices' breakthrough mems sensor brings sophisticated motion and navigation control to industrial applications," June 2007. Press Release.

[8] Analog Devices, Inc., One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, *ADIS16350 Tri Axis Interial Sensor*, June 2007. Document# D06070-0-3/07(B)

[9] H. Weinberg, "Dual axis, low g, fully integrated accelerometers," *Analog Dialogue*, vol. 33-1, 1999. Analog Dialogue is a journal published by Analog Devices.

[10] Avago Technologies, Pte., 350 W. Trimble Rd. Bldg. 90, San Jose, CA, 95131, *Understanding Optical Mice*, January 2006. An Avago White Paper.

[11] Avago Technologies, Pte., 350 W. Trimble Rd. Bldg. 90, San Jose, CA, 95131, *Agilent HDNS-2100 Solid-State Optical Mouse Lens*, April 2001.

[12] Microchip Technology, Inc., 2355 West Chandler Blvd., Chandler, AZ 85224-6199, *PICDEM FS-USB Demonstration Board User's Guide*, November 2004. Document# DS51526A

[13] Analog Devices, Inc., One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, *ADIS16255 Programmable Low Power Gyroscope*, March 2007. Document# D06070-0-3/07(B)

[14] USB Implementers Forum, Inc., 5440 SW Westgate Dr., Portland, OR 94221, *Universal Serial Specification Revision 2.0*, April 2000.

[15] C. Peacock, "Usb in a nutshell." November 2002.

[16] IEEE, 3 Park Avenue, New York, NY 10016-5997, *IEEE Standard for Inertial Sensor Terminology*, November 2001.

[17] I. Gouverneur, "Miniaturization of a flexible imaging system for a 3d ultrasound positioning sensor," Master's thesis, Worcester Polytechnic Institute, 2007.

[18] R. Prager, A. H. Gee, and L. Berman, "Stradx: Real-time acquisition and visualization of freehand 3d ultrasound," *Medical Image Analysis*, 1998.

[19] R. Prager, A. H. Gee, and L. Berman, "Surface interpolation from sparse cross-sections using region correspondence," Tech. Rep. CUED/F-INFENG/TR342, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, England, March 1999.

[20] J. C. Carr, J. L. Stallkamp, M. M. Fynes, A. H. Gee, R. Prager, G. M. Treece, C. Overton, and L. Berman., "Design of a clinical free-hand 3d ultrasound system," in *Medical Imaging 2000 (Ultrasonic Imaging and Signal Processing)*, vol. 3982, SPIE, 2000.

[21] R. W. Prager, R. N. Rohling, A. H. Gee, and L. Berman, "Rapid calibration for 3-d freehand ultrasound," *Ultrasound in Medicine and Biology*, 1998.

[22] R. A. Serway, *Physics for Scientists and Engineers.* Brooks Cole, 7 ed., 2007.

[23] F. S. Rovati, P. Gardella, P. Zambotti, and D. Pau, "Spatial-temporal motion estimation for image reconstruction and mouse functionality with optical or capacitive sensors," in *Consumer Electronics, International Conference On*, ICCE, 2003.

[24] C. Poulsen, "Development of an optical positioning system for 3d ultrasound," Master's thesis, Worcester Polytechnic Institute, 2005.

[25] Microchip Technology, Inc., 2355 West Chandler Blvd., Chandler, AZ 85224-6199, *MPLAB C18 Compliler Libraries*, November 2004. Document# DS51297F

[26] G. D. Boreman, *Modulation Transfer Function in Optical and Electro-Optical Systems.* SPIE - The International Society for Optical Engineering, 2001.

[27] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology.* American Institute of Aeronautics and Astronautics, Inc., 2004.

[28] J. Quatararo, "Semi-automated segmentation of 3d medical ultrasound images," Master's thesis, Worcester Polytechnic Institute, 2008.

[29] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

Appendices

Appendix A: 5DOF 3D Ultrasound User's Guide

5DOF 3D Ultrasound User's Guide

Abraham Goldsmith

Worcester Polytechnic Institute

Deptartment of Electrical Engineering

100 Institute Rd

Worcester, MA 01609

1 Introduction

This document decribes the configuration and use of the 5DOF 3D ultrasound acquisition system. The system is composed of a WindowsXP PC, a Terason t3000 firewire ultrasound probe system, Strawin 3D ultrasound acquisition software, Inertial Measurement Unit (IMU), and Spartan3E-1600 Demonstration board.

2 Scope

This document assumes that the reader already understands 3D ultrasound conceptually and the process of acquiring ultrasound data more generally. Only tasks specific to the configuration and operation of Stradwin and the 5DOF tracking system are dealt with. The use of the Stradwin and Terason software is covered insofar as it directly pertains to acquiring data. For more detailed information the user is directed to their respective websites, which are listed in Section 8. In addition, this guide does not deal with programming the developemnt board with the the netlist and system firmaware for the tracking system, attachment of the IMU and optical tracker to the probe, or low-level calibration. These are all very complicated topics that are well outside the scope of this document.

3 Requirements

The following sytem components must be available to proceed:

- A PC running WindowsXP SP2 with a P4 processor and 4Gb of RAM. It may or may not be possible to use a computer with lower specs. This is the only setup that was tested so I will treat its specifications as the minimum.
- A Terason t3000 ultrasound system and at least 1 probe module. The probe should have the IMU and optical tracking assembly already attached, as in Figure 5.
- A Spartan-3E 1600E Development board, pictured in Figure 1. This board was manufactured by Digilent Inc (www.digilentinc.com). They appear to have discontinued

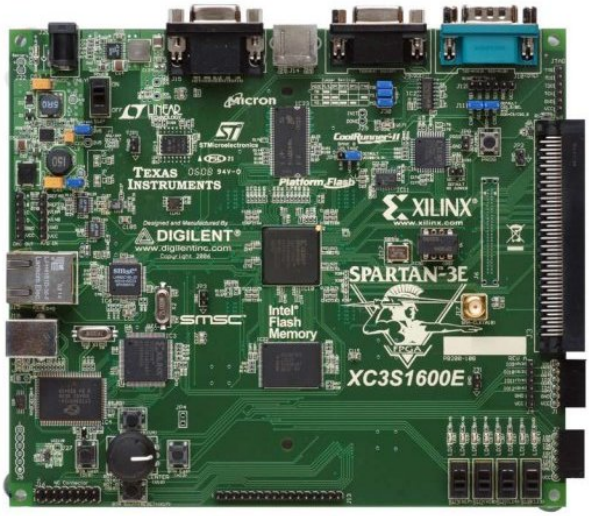


Fig. 1: Spartan3E-1600 Demonstration Board

the 1600E model and now only sell a version with the a lower gate-count FPGA, so don't break that one!

4 Stradwin Installation

This section descibes the initial installation of Strawin. It is intended to help users who need to setup a new system from scratch. Users who have access to a preconfigured system can skip this section.

1. The first step is to aquire a copy of the Stradwin installation package. The creators of stradwin made a special version for use with this system. It includes a few display modifications that make the system easier to work with but are not absolutely required. The custom version is on the data DVD, otherwise, download the latest Stradwin package from the Stradwin website.

2. Double click the .msi file and follow the instructions. Choose the default setting for everything.

3. On the desktop, find the shortcut named **Stradwin 3.4** or **Stradwin 3.5** and delete it.

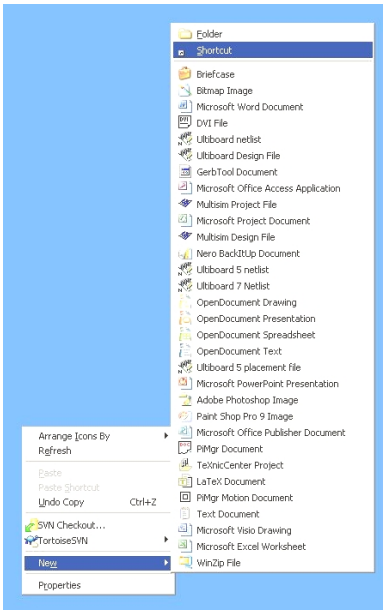


Fig. 2: Create New Shortcut

4. On the desktop, right click and select **new** → **shortcut**

5. Browse to **c:\Program Files\Stradwin** and select stradwin.exe or just enter **C:\Program Files\Stradwin\stradwin.exe** directly into the box. Click next and enter a name for the new shortcut.

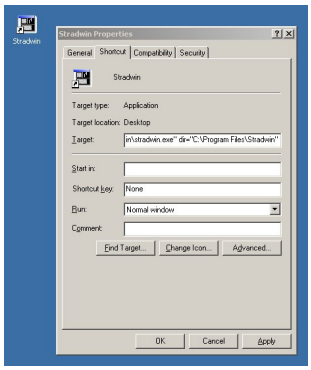


Fig. 3: Edit Shortcut Properties

6. Right click on the shortcut and select properties. In the box labeled Target, add **"dir=C:\Program Files\Stradwin"** so that the entire line reads **"C:\Program Files\Stradwin\stradwin.exe" dir="C:\Program Files\Stradwin"**.

7. Copy the vertical.swt file into the **C:\Program Files\Stradwin** directory. Vertical.swt is not part of the Stradwin package. It is the file that configures Stradwin to work with the particular probe, image size, and IMU orientation used in this system. The text contents of the file are in Appendix A if you do not have access to a digital copy of the file.

5 Physical Connections

This section describes all of the electrical connections between the PC, t3000 ultrasound probe, Spartan development board, and IMU.

5.1 Spartan Development Board

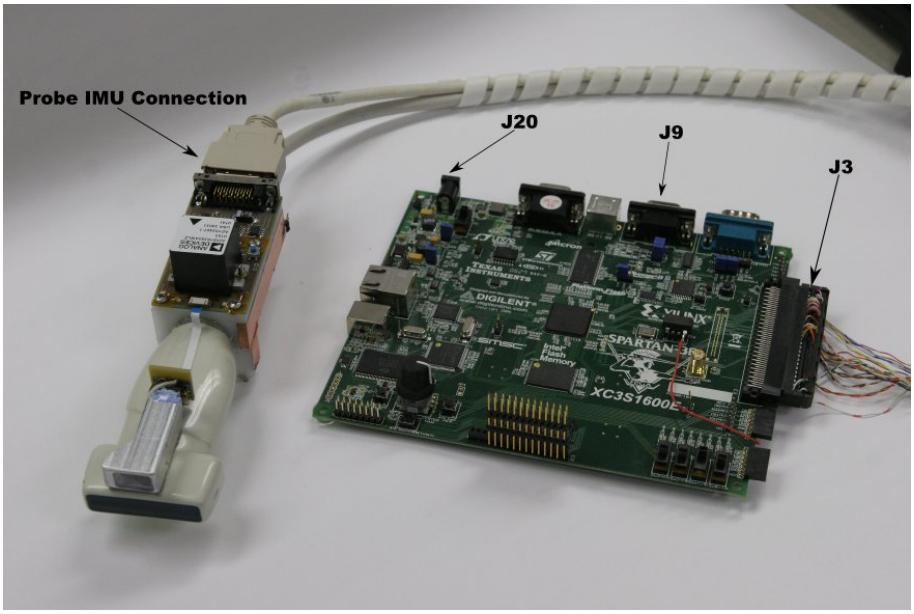


Fig. 4: Spartan Development Board Connections

Plug the external power supply into J20. Connect the RS232 serial cable to J9 on the board and into the PC's COM port. A USB to RS232 adapter can also be used to connect the RS232 cable to the PC on systems that do not have a RS232 COM port. Connect the probe cable to the IMU on one end and to J3 on the Spartan Dev. board.

5.2 Terason t3000 Ultrasound Probe

Plug the t3000 base module into the PC's firewire port. If you are using a laptop you may need to either plug in an external power source or use the inline power splice cable. Then plug the probe into the base module and lock it into place.

5.3 Ground Strap

Connect the earth ground harness to the probe, as in Figure 5 and instert the plug into a 3 prong AC outlet. It is important to be sure that the outlet is grounded. If you don't connect the earth ground you risk serious damage to the electronics should an ESD event occur.

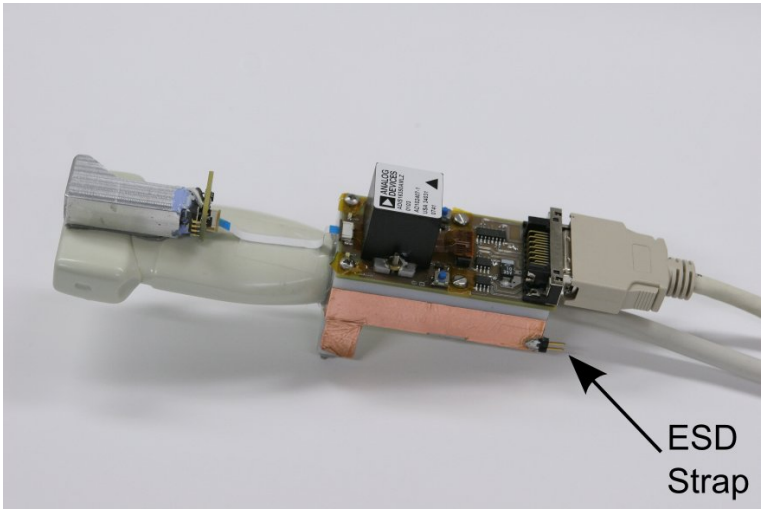


Fig. 5: ESD Strap Connection

6 Configuration

1. Start the terason control program (ultrasound.exe) and configure the probe as desired. When finished, minimize the window. Typically, the Terason application can be launched from the start menu by selecting **Start** → **Programs** → **Terason** → **Terason**.

2. Double click on the Stradwin desktop shortcut, created during the installation step, to start Stradwin. The Terason application window will disappear but that is ok.

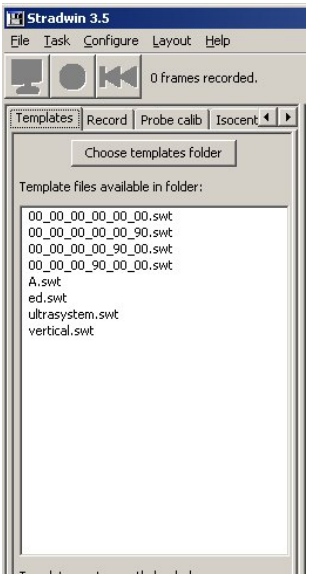


Fig. 6: Template Selection Tab

3. On the left side of the tradwin window, select the **Templates** tab, shown in Figure 6, and then double click on vertical.swt. If no files are visible you can browse to another location by clicking on the **Choose templates folder**. You should see a screen appear that looks like Figure 7. Click **OK**.

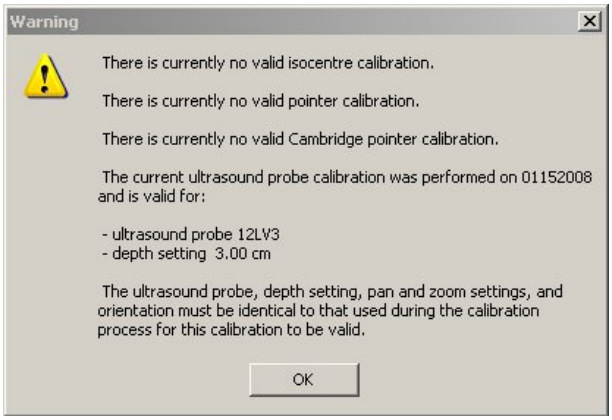


Fig. 7: Template Selection Warning

4. If Stradwin was able to communicate with the tracking system, it should say **Fastrak Initialized** along the bottom of the screen, as in Figure 8. You can now click on the **Record** tab and configure the acquisition parameters. You should see the image source and the tracking system listed as in Figure 9. If you PC is powerful enough you can vary the frame rate. Otherwise, set it maximum and take what you can get. Set the live display limit to 5Hz to limit associated workload. Leave everything else in its default state.

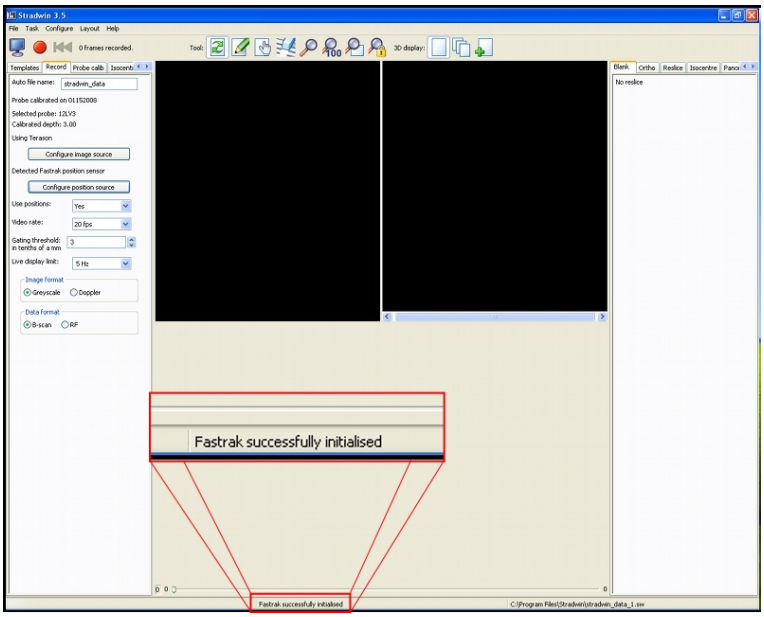


Fig. 8: Stradwin Position Initialization Message

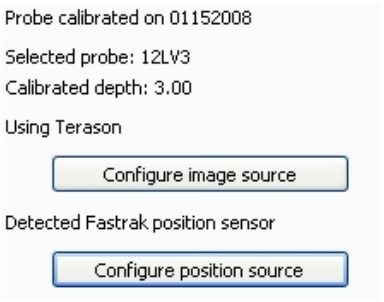


Fig. 9: Stradwin Image and Position Sources Identified after Configuration

7 Acquisition

Once you have configured the software you are ready to begin acquiring ultrasound data. The first step in this process is to calibrate the tracking system. When the Spartan-3E demonstation board is first powered on the red LED will be illuminated, indicating that the system has not been calibrated. The system can be calibrated at any time by pressing button 2 and holding the probe perfectly still until the green LED comes on. The orientation that the probe is held in during the calibration process will be the zero reference for all the angular measurements and the position of transducer during calibration will be the zero position reference. The red and green LEDs and button 2 are indicated in Figure 10.

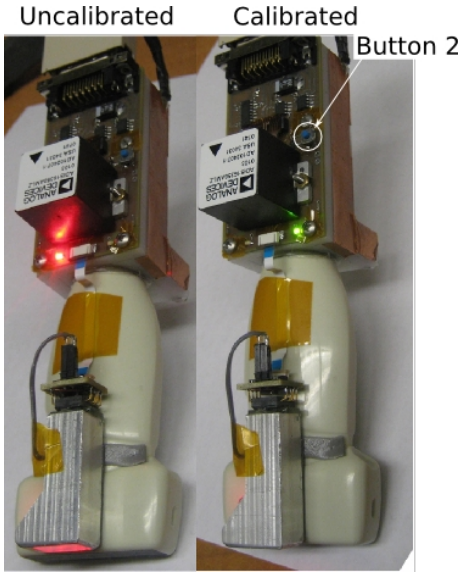


Fig. 10: Transducer Calibration State Indicators and Calibrate button

1. Click the **Live Display** button in the upper left hand corner of the screen to begin displaying live data. You should now see the current B-scan in the left-upper portion of the display and the sensor poition and coordinate axes displayed in the upper-right hand portion of the display, as illustrated in Figure 11.

2. The system must be calibrated prior to each acquisition session. Hold the probe perpendicular to the surface being scanned and hit button 2 on the probe, as shown

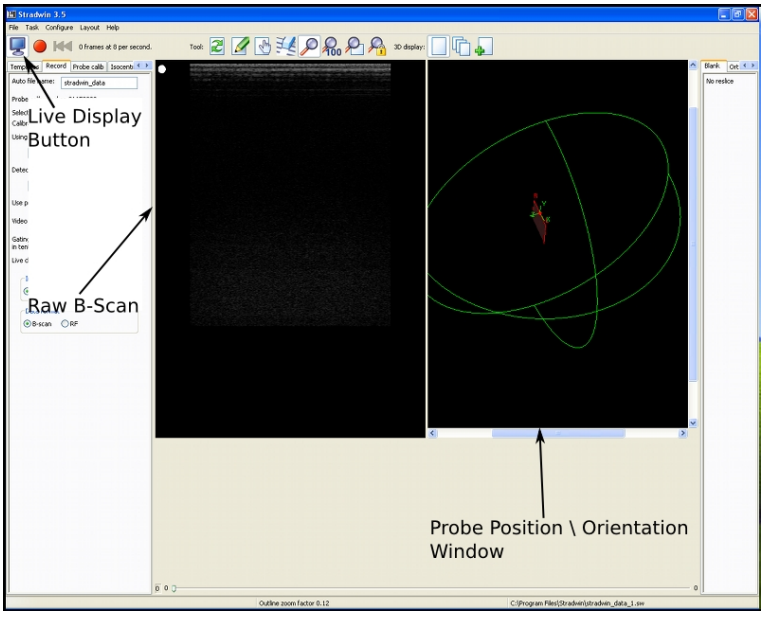


Fig. 11: Stradwin Live Image Display

in Figure 10. The calibration process begins as soon as the button is pressed. It is very important that the probe remain still until the green LED comes on; any motion will be included in the gyroscope bias estimates, which will result in inaccurate data. Calibration can be confirmed by inpecting the probe orientation in the Probe Position Window. Depending on the state of the system prior to calibration, the image frame may not be at the origin in the Probe Position Window. If the live display is active the image frame may jump back to the origin when calibration is complete.

3. Once the calibration is complete you are ready to aquire data. You should begin aquiring as soon as possible after the calibration. Gyroscope bias accumumulates with time and leads to errors in the reported position and orientation of the probe. As a rule of thumb, the length of acuision should be limited to 20s after calibration.

   Press the **Record** button to start acquisition. It is the red button to the right of the **Live Display** button, pictured in Figure 11. The Raw B-Scan window will now display each new image frame as it is recorded. The transducer position and pose, along with a white frames indicating the relative position of each image frame, will be

displayed in the Probe Position Window.

4. To stop acquisition, click on the **Record** button again. Live image and position infomation will continue to be displayed while the live display mode is active. In order to work with your new data you will need to exit live display mode. To do so, click on the **Live Display** button again. Once you have done this the Probe Position Window will display only a series of white frames inicating the relative positions of the recorded image frames. The data in the Raw B-Scan window represents the currently selected image.

5. Before you do anything else, you will probably want to save your data. To do this, go to **File** → **Save**. Now you can interact with your data using one of the several visualization modalities built into stradwin. For more information please see the Stradwin documentation, which is availble in the Stradwin help file as well as on the Stradwin website.

8 Additional Resources

- Offical Stradwin website: http://svr-www.eng.cam.ac.uk/~rwp/stradwin/
- Offical Terason website: http://www.terason.com/
- Complete documentation and schematics for the Spartan 3E-1600 Development Board: http://www.digilentinc.com/Products/Detail.cfm?Prod=S3E1600&Nav1=Products&Nav2=P

A Vertical.swt

```
RES_BUF_WIDTH 512
RES_BUF_HEIGHT 512
RES_POS_REC true
RES_BUF_RF false
RES_RF_VECTORS 127
RES_RF_SAMPLES 3336
RES_END_HEADER
RES_VID_CARD TERASON Terason
RES_VID_PORT 2
RES_VIDEO_STANDARD 0
RES_POS_DETECT FASTRAK
RES_SERIAL_PORT COM6
RES_SERIAL_SPEED 115200
RES_POLARIS_PORT 0
RES_POLARIS_POINTER 0
RES_TEMP_CALIB 0
RES_BUF_DOPPLER false
RES_AUTO_CORRECT_POS true
RES_VID_XPOS 0
RES_VID_YPOS 0
RES_VID_RATE 20
RES_VID_MOVE_THRESH 0.030000
RES_VID_GREY_THRESH 24
RES_VID_CHROMA_THRESH 20
RES_CAL_DATE 01152008
RES_CAL_PROBE 12LV3
RES_CAL_DEPTH 3.000000
RES_XTRANS 0.0
RES_YTRANS 2.5
RES_ZTRANS 0.00
RES_AZIMUTH -90.000000
RES_ELEVATION 0.000000
RES_ROLL -90.0000
RES_XSCALE 0.009766
RES_YSCALE 0.009766
RES_RF_DISPLAY BSCAN
RES_RF_DUAL_DISPLAY false
RES_RF_PROBE_NAME
RES_RF_PROBE 0
RES_RF_VECTOR_OFFSET 0
RES_RF_SAMPLE_OFFSET 0
RES_RF_SCALE 0.010000
RES_RF_FREQ 0
RES_RF_FOCUS 0
RES_RF_FOCII 1
RES_RF_TX_FIRE 1.500000
RES_TX_POLARITY true
RES_RF_FRAME_SYNC 2.500000
RES_FRAME_POLARITY false
RES_STRAIN_SUBSAMPLE 5
RES_STRAIN_ALGORITHM 1
RES_STRAIN_EWEIGHTING 1
RES_STRAIN_PWEIGHTING 1
RES_STRAIN_LOGCOMPRESS false
RES_STRAIN_LATERAL true
RES_STRAIN_ELEVATIONAL false
RES_STRAIN_HALF_CYCLES 6
```

```
RES_STRAIN_WINDOW_2D true
RES_STRAIN_WINDOW_3D false
RES_STRAIN_SKIP_PIXELS 6
RES_STRAIN_DIFF_2D true
RES_STRAIN_DIFF_3D false
RES_STRAIN_DROPOUTS 5
RES_STRAIN_TRACK_FROM_ZERO false
RES_STRAIN_TRACK_FROM_TOP true
RES_STRAIN_DIFF_OFFSET 10
RES_STRAIN_GRADIENT_2D true
RES_STRAIN_GRADIENT_3D false
RES_STRAIN_WEIGHTING false
RES_STRAIN_POSTFILTER_ALGORITHM 1
RES_STRAIN_FILTER_RANGE 5
RES_STRAIN_PERSISTENCE 20
RES_STRAIN_NORMALISE 3
RES_STRAIN_NORM_RANGE 2
RES_STRAIN_NORM_SCALE 45
RES_STRAIN_NOISE 50
```

Appendix B: CustusX-WPI Users Guide

CustusX-WPI User's Guide

Abraham Goldsmith
1/14/07

Table Of Contents

1 Introduction

CustusX is a 3D visualization tool developed by SINTEF; an independent research group headquartered in Trondheim, Norway. It is based on the ITK and VTK toolkits from Insight. CustusX renders 2D views of 3 dimensional data such as that generated by ultrasound scanners. Dr. Peder Pedersen contracted SINTEF to build a custom version of CustusX to run on windows (it was originally developed for platforms running XFree86 and Xorg). This version was to serve as the visualization frontend for portable 3D ultrasound system.
This document describes the features and functionality of CustusX as of January, 2007. At the time of writing the program supports several data import modes, visualization, and a single data export mechanism. It is still very rough, with many unhandled exceptions and frequent crashes.

2 System Requirements

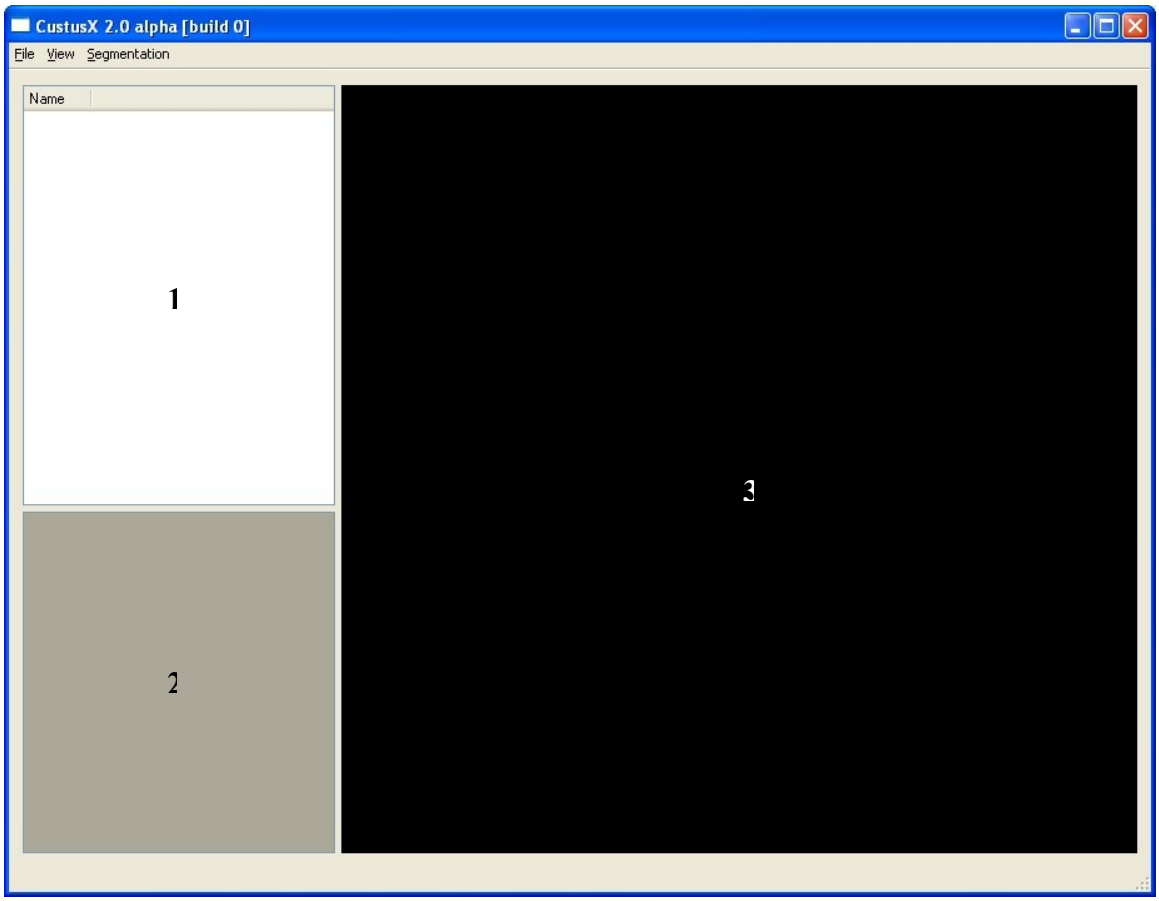
- P4 2.0 GHz or better
- 1Gb RAM or greater
- Graphics hardware supporting OpenGL 2.0
- Windows XP
- 10Mb Free hard disk space

3 Installation

CustusX is provided as a self-installing windows installer package (.msi). Double clicking on the .msi file will invoke the installer which will guide the user through a series of dialogues. The only real option is the installation location. The program can be uninstalled through the **Add and Remove Programs** option in the **Windows Control Panel.**

4 User Interface

All operations are performed from within the graphical user interface shown below.



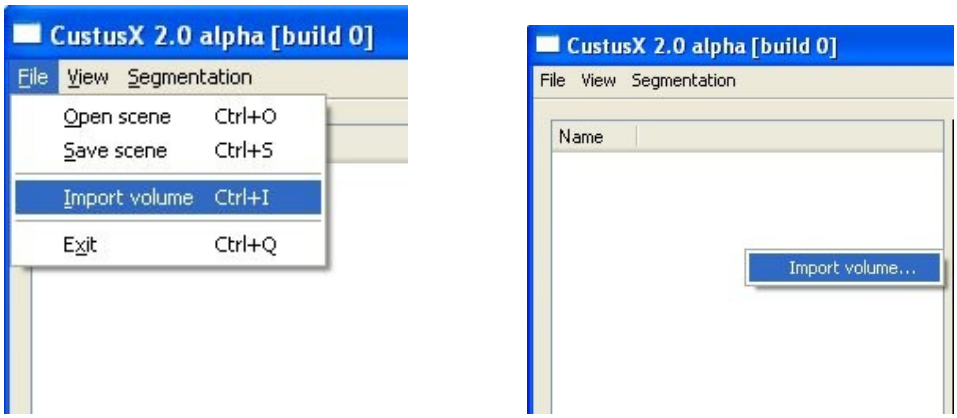
The screen is divided into 3 sections:

1. **Volume Browser**
2. **Transfer Function**
3. **Volume Display**

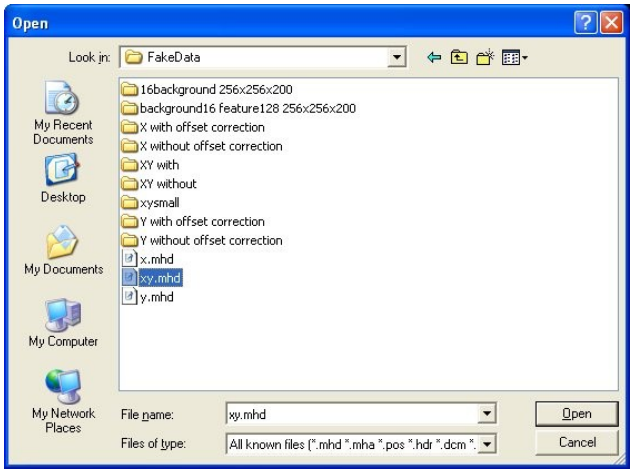
The **Volume Browser** displays a list of volumes that are currently loaded into memory. The **Transfer Function** controls allow the user to adjust the visualization parameters. The actual controls are only displayed when at least one volume is loaded. In the image above no volumes are loaded so all sections of the screen appear blank. The **Volume Display** visualizes the data according to the settings in the **Transfer Function** section.

From the main screen the user can import a volume in one of two ways:

- **File -> Import Volume**
- By right clicking in the **Volume Browser** and selecting **Import Volume**



In either case, the user will be presented with a standard browser window. Specifics about what file types can be opened are provided in section 5. Browse to the location of the file(s), select it, and hit the open button.



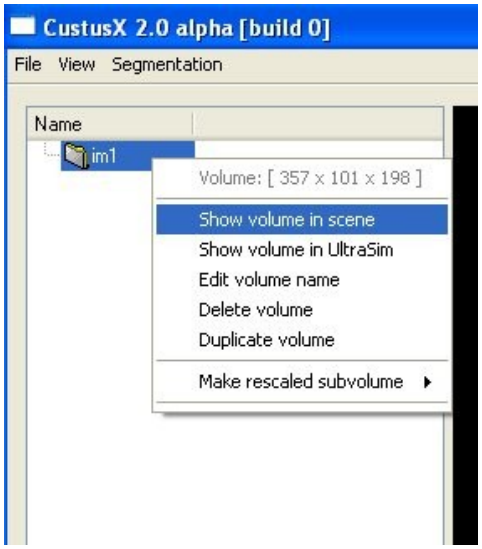
If the file selected was the first in a series of separate .txt and .bmp files, a warning will be displayed. Hit **OK** to continue.

4.1 Volume Browser

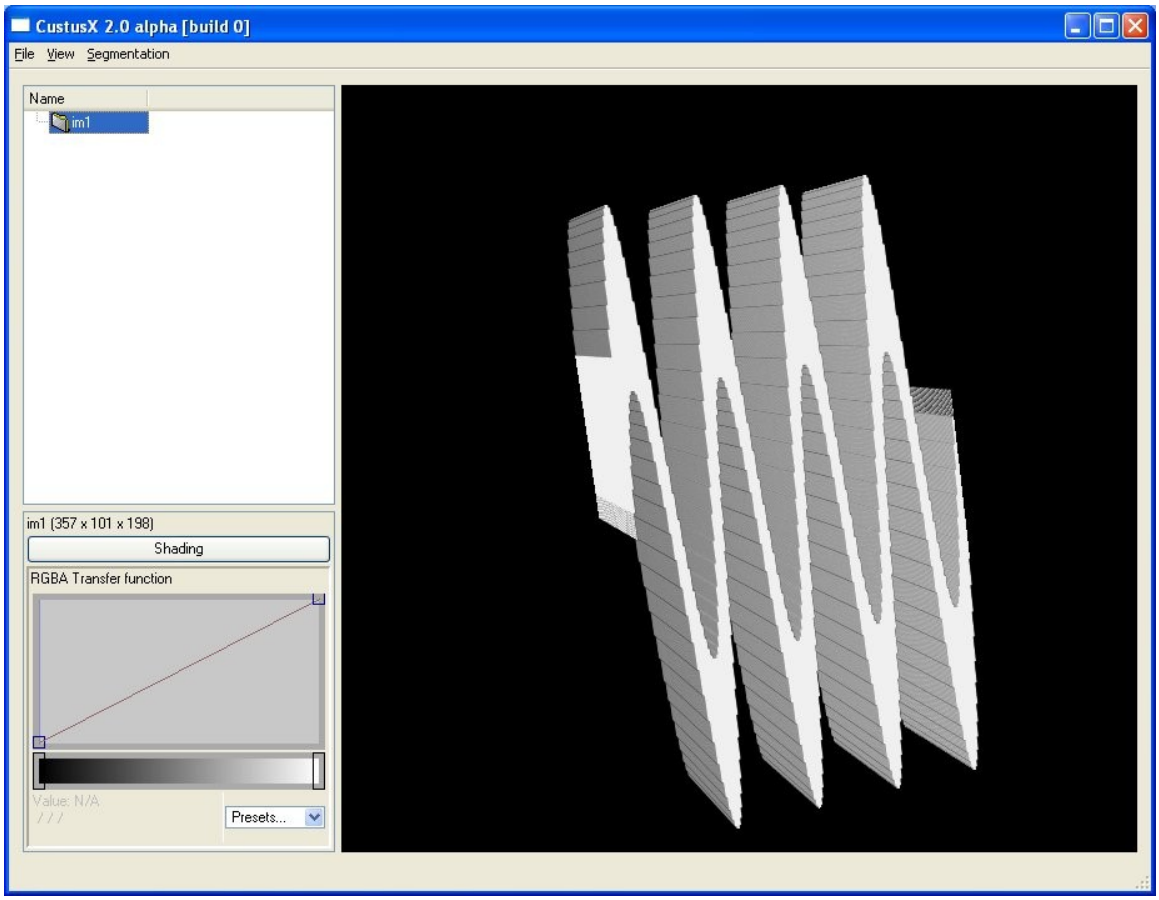
After the volume has been read in, its name will appear in the **Volume Browser**. The name assigned depends on the import mode used. Data imported with a .mhd header file will use the name of the .mhd file. Data imported using the WPI dll will be assigned names sequentially, starting with im1.

There are a number of actions that can be performed on a volume once it has been loaded. They can be accessed by right clicking on the name ff the volume in the **Volume Browser**. The available actions are:

- **Show Volume in Scene** – This option will render the volume and display it in the **Volume Display**
- **Show Volume in UltraSim** – This option does nothing
- **Edit Volume Name** – This option allows the user to change the volume name displayed in the **Volume Browser**
- **Delete Volume** – This option removes the volume from memory and removes its name from the list of loaded volumes in the **Volume Browser**
- **Duplicate Volume** – This option makes a copy of the selected volume. Warning: This action causes the program to crash.
- **Make Rescaled Subvolume** – This option allows the user to make a rescaled copy of the original volume. The user can select from a list of predefined scaling ratios or select **Custom** to define their own.

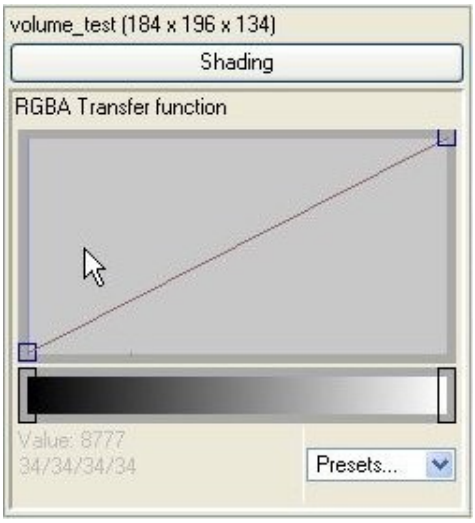


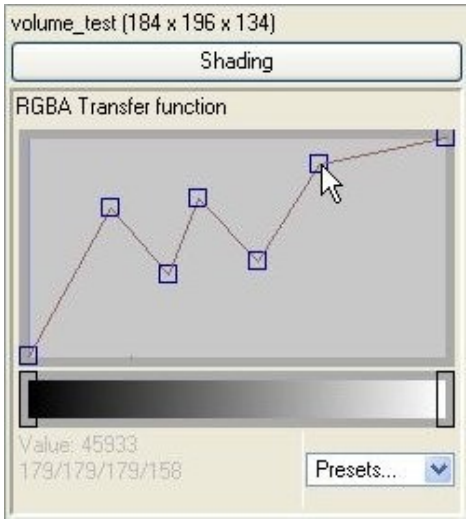
Selecting **Show Volume in Scene** causes the **Transfer Function** and **Volume Display** sections to become active.



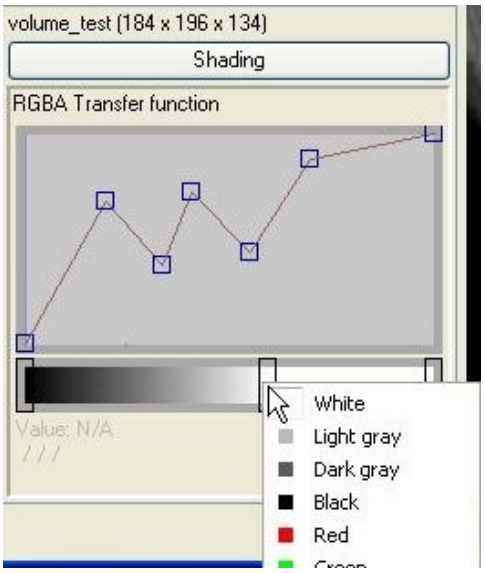
4.2 Transfer Function

Once a volume has been displayed the **Transfer Function** section can be used to alter the display. The graph titled **RGBA Transfer Function** is basically a histogram. For some volumes vertical bars can be seen representing the relative number of voxels containing a particular value. For others, no bars are visible. I do not know why this happens. The X or horizontal axis represents intensity values with the lowest values on the left and the highest values on the right. The Y or vertical axis represents transparency. High values correspond to low transparency. The red line defines the transparency of the surface mapped to each intensity value.



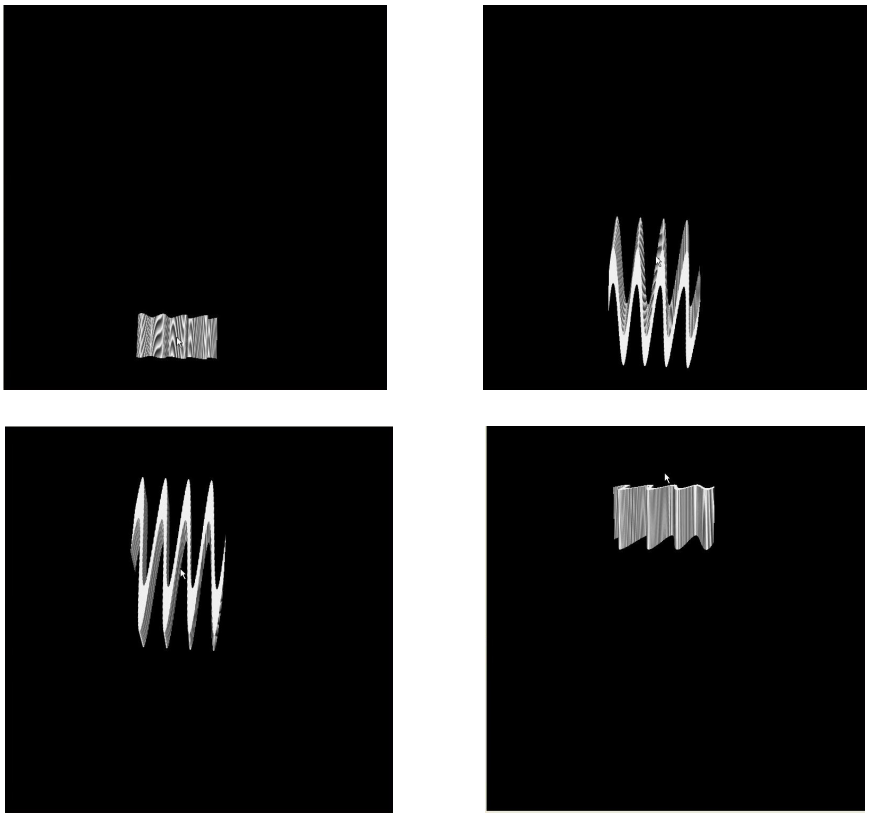


The greyscale, red (R), green (G), and blue (B) intensities and the alpha (A, means transparency) are displayed in the lower left hand corner of the **Transfer Function** view when the cursor is positioned over the histogram. Left mouse clicking on the histogram adds adjustment points to the transfer function. Right mouse clicking on the adjustment points allows them to be dragged to a new location.



Below the transfer function histogram is the color bar. The color bar is used to assign colors to particular intensity values. Like the histogram, new points can be added to the color bar by right mouse clicking over the bar and points can be moved by right clicking and dragging. The **Presets** dropdown box provides some predefined color configurations and an editor that allows them presets to be altered.

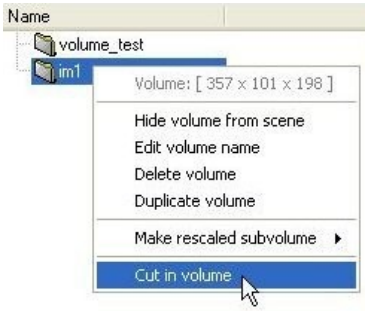
4.3 Volume Display

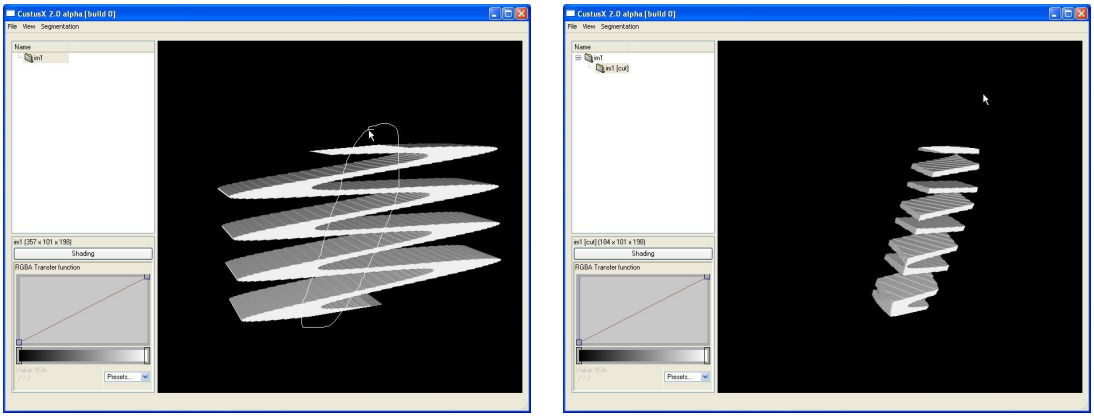


The rendering is displayed in the **Volume Display** area of the screen. There are a number of ways to manipulate the view:

- Left clicking and dragging in this area the image can be rotated in any direction
- Holding the shift key while left clicking and dragging moves the center of rotation
- Holding the CTRL key while left clicking and dragging rotates the image about the center of rotation along the axis pointing straight into the screen only.
- Clicking and dragging with both the left and right buttons simultaneously zooms in and out. This can also be accomplished with the mouse scroll wheel, if present.
- **View -> Nearest Neighbor Texture Filtering** toggles between the two texture filtering modes.

4.4 Cuts





Cuts are used crop volumes to isolate areas if interest. When a volume is being actively displayed, cut mode is activated by right mouse clicking on the name of the volume in the **Volume Browser** and selecting **Cut in Volume**. The mouse cursor will now turn into a set of crosshairs when moved over the **Volume Display**. Left click and drag the crosshairs around the area of interest. When the mouse button is released the area of the volume outside of the line will be cropped off.

4.5 Saving and Opening Scenes

A CustusX session can be saved at any time by selecting **File -> Save Scene**. A browser window will open allowing the operator to select the name and location of the folder containing the data. A saved scene contains all currently loaded volumes, subvolumes, and display settings. A saved session can be loaded at any time by selecting **File -> Open Scene**.

5 Data Import

5.1 MetaIO

CustusX uses the MetaIO medical image file IO library, which is integrated into ITK. This library supports a wide variety of standard medical imaging formats. It also supports importing custom data types via a header file describing the data to be imported. For a complete guide to data import using MetaIO please refer to the documentation in appendix A. Import header file examples can be found in appendix XXX.

5.2 6DOF Extension to MetaIO

CustusX also implements an extension to the MetaIO library that allows data to be imported along with information about the orientation of the probe, the position of each 2D slice in the volume with respect to the starting point, and mask that can be used to crop off unwanted portions of each 2D scan as the volume is reconstructed. The additional information is described by individual ASCII files as follows:

- .raw – A file containing a series of 2D images stored as raw binary data
- .mha / .mhd – Describes the layout of the data in the .raw file
- .msk – A raw 2D image used to mask out a section of each input image
- .cal – An ASCII file containing a 4x4 transformation matrix describing the relationship between the ultrasound probe and the image
- .pos – An ASCII file containing a series of 3x4 transform matrixes (they are actually 4x4 but the last line row is always 0 0 0 1 so it is omitted) that describe the position and orientation of the probe. Please see appendix B for a complete description of the matrixes. There do not need to be the same number of position matrixes as there are images in the source volume.

To use this import mode, select the .pos file instead of the .mhd file when importing the volume. All the files must have the same name up to the extension.

5.3 WPI DLL

CustusX also supports an import mode that was developed specifically for WPI. The original goal was to develop a method for dynamically acquiring a variable number of image frames directly from an ultrasound scanner. SINTEF and WPI agreed on an API that would allow CustusX to read in an arbitrary number of frames as if they were files. The functionality was implemented in a dynamically linked library by WPI.

At the time of writing this feature only works as a proof of concept. Instead of actually acquiring ultrasound image frames, the dll opens a sequence of bitmap files and passes the image data to CustusX. Each bitmap has a corresponding text file that describes the

position of the frame with respect to the first frame in the sequence. CustusX uses this data to construct a 3D volume. The number of frames read in is fixed to 200 to avoid the difficulty of implementing a mechanism for starting and stopping image acquisition. It turns out that the MetaIO import mechanism also provides this functionality and much more. For this reason, it is recommended that the user use only the MetaIO import method.

6 Data Export

Data can be exported from CustusX using the **File -> Save Scene** feature described in section 4.4. When a scene is saved a directory is created containing all of the data. An XML file describes all of the data and display settings and each volume is stored in an individual raw data file. The raw data files are organized in a brick-of-bytes arrangement which can be imported into most other software tools as long as the user provides information about the dimension, voxel spacing, and data type. Most of this information is in the XML file but the data type is not. CustusX uses 16 bit intensity values. That means that even if the volume had 8 bit samples when it was initially imported into CustusX, it will have 16 bit samples when it is exported. It is also important to note that color and transparency information cannot be exported for use with other programs. The raw data files contain only the raw intensity values. If position information was used during the import process the exported volume will contain the reconstructed volume, not the original image slices.

7 Appendix A: MetaIO Reference

This material is taken directly from the MetaIO Users Manual

MetaIO
Medical Image I/O Made Simple

Abstract

Our goal was to create a simple, flexible, cross-platform image file format that supported medical image application development.
This meant that the image file format and its associated image I/O library had to support:

1. Image acquisition information essential to the correct processing and alignment of medical images (e.g., voxel/element size and spacing; and image position and orientation)
2. MSB (Most-significant-bit = Mac/Sun) and LSB (Least-significant-bit = PC) byte ordering
3. Arbitrary atomic pixel types (char, unsigned char, short, float, etc.)
4. N dimensional data
5. Image data stored in one file or in one file per sub-dimensional (e.g., N-1) slice
6. Text-based headers that are easily read and edited
7. Optionally storing data in a file(s) separate from the header to simplify conversion to/from other formats

MetaImage is the text-based tagged file format for medical images that resulted. We have now extended that file format to support a variety of objects that occur in medicine such a tubes (for vessels, needles, etc.), blobs (for arbitrary shaped objects), cubes, spheres, etc. The complete library is known at MetaIO. The central code of MetaImage/MetaIO is quite stable. MetaImage has been in use for several years by a wide range of research at UNC, Chapel Hill. New features are occasionally added, but backward compatibility will always be maintained.
Current info on MetaObjects is available from
http://caddlab.rad.unc.edu/technologies#metaObjects

Obtaining MetaIO

MetaIO is being distributed over the WWW by its developers in the CADDLab at UNC:
http://caddlab.rad.unc.edu/technologies#metaObjects
MetaIO is also being distributed with the National Library of Medicine's Insight Tookit (ITK) for medical image segmentation and registration:
http://www.itk.org

Installing The MetaIO Package

MetaIO is a hierarchy of C++ classes and functions. We have yet to find a modern C++ compiler that does not compile MetaIO. Know compatible compilers include G++ v2.95 and beyond (and probably previous), Microsoft Visual C++ 6.0, Sun's CC on Solaris 2.6 and beyond, Intel compiler and compilers on other workstations including HPs, SGIs, and Alpha systems. Please contact us (Stephen R. Aylward, aylward@unc.edu, Julien Jomier mailto:jomier@unc.eduor via http://caddlab.rad.unc.edu) if you encounter any incompatibilities between our code and your compiler.

The hierarchy of the software in the stand-alone MetaIO package is as follows:

MetaIO/
    doc/
    tests/

The top level contains the source files, the header files, and the CMakeLists.txt file that is used by the CMake program to compile MetaIO. This document and the MetaObjects www pages are in the doc directory. A sequence of simple tests is available in the tests directory.

The hierarchy of the software in the ITK distribution is as follows:

Insight/Code/Utilities/MetaIO/
    doc/
    tests/
Insight/Code/IO/
    Insight/Examples/
    MetaImageReadWrite/
    MetaImageViewer/
    MetaImageColorViewer/
    MetaImageImporter/

The files in Insight/Code/Utilities exactly match those in the stand-alone MetaIO package. Routines to wrap metaImage so that it can be accessed via the itkImageIO Object Factory are in /Insight/Codc/IO. The details of the Examples are provided later in this documents.

MetaImage as a stand-alone package

If you downloaded MetaIO separate from Insight, you must also download and install CMake from www.CMake.org. That www site provide details regarding installing CMake and using it to compile another program. Follow those directions to create the MetaIO library and tests.
A minimal description of the process is presented next.

1. Install CMake
2. On Windows:
    a. Run CMake.exe
    b. In the source directory field, browse to the top-level of the MetaIO directory
    c. In the binary directory field, enter the path and name of the source directory, but add "-VC++" to the directory name (this assumes that you are compiling using VC++, otherwise add an equally descriptive name for the directory to store the binaries in such as "MetaIO-bcc" or Borland).
    d. Press the "configure" button. It will ask if you want to create the binary directory, press "yes."
    e. Specify your compiler in the pull-down menu field.
    f. Press the "configure" button a second time, and then press "okay" to generate the Makefiles for your compiler (e.g., MetaIO.dsw and related files for VC++) in the binary directory.
3. On *nix/Cygwin machines:
    a. At an appropriate spot, create a directory "MetaIO-g++" or give it an equally descriptive name based on your compiler
    b. "cd MetaIO-g++"
    c. Run "ccmake <path to MetaIO source directory>"
    d. Press "g" to generate the makefiles for your compiler in the binary directory.

To use MetaIO in your stand-alone programs:

1. add the MetaIO directory to your include paths,
2. add "#include <metaImage.h>" to the top of the your file that performs your image IO,
3. add the MetaIO binary directory to your link paths, and
4. link with the MetaIO library.

See the files in MetaIO/tests for examples of how to read/write MetaImages and other MetaObjects.

MetaIO with the NLM's Insight toolkit (ITK)

If you have downloaded and are installing MetaIO as part of the Insight toolkit, follow the standard installation procedure of Insight. This will create the MetaIO library and the ITK specific wrapping, examples and tests. Certain examples, such as the MetaImageViewer, also require FLTK (a cross-platform user interface library available from http://fltk.org). Install FLTK and then ITK to have every MetaIO example built. Numerous other ITK examples also rely on FLTK.

See the file /Insight/Examples/MetaImageReadWrite for a working example on how to develop a program using MetaImage for IO. Work is underway to add access to other MetaObjects (e.g., tubes, spheres, etc.) via ITK's SpatialObjects in /Insight/Code/SpatialObject.

Quick Start: Data conversion via MetaHeaders

This section assumes that you have some data that you wish to process using an application that reads MetaImages. This section gives examples on how "convert" your data to the MetaImage format.

For uncompressed data (i.e., data stored in a raw format as is often done for DICOM, BMP, and PNG formats), "conversion" to MetaImage is actually just a matter of specifying a MetaImage Headerfile (a "MetaHeader") that describes and points to the file(s) containing your data.

For compressed data (i.e., data stored in JPEG or GIF formats), you must first convert your data to a noncompressed format. Currently, no data compression methods are supported by MetaImage, but adding support for .gz, RLE, and other compression formats is high on our To-Do list. One of the most robust image conversion software packages is ImageMagick (http://www.imagemagick.org/; Unix and PC versions available). It has an application called "convert" that handles most of the popular 2D image formats.

IF YOU HAVE INSTALLED METAIO AS PART OF ITK, USE THE PROGRAM Insight/Examples/MetaImageImporter. IT WILL ASK A SERIES OF QUESTIONS AND PRODUCE A METAIMAGE HEADER FOR YOU.

Reading a Brick-of-Bytes

A “brick of bytes” is a volume of image data stored in a single file possibly with preceding and trailing non-image data.

To correctly load these images, the minimal information that you need to know is:

1. Number of dimensions
2. Size of each dimension
3. Data type
4. Name of the data file

For example, let’s say the data was 3 dimensional, had 256 x 256 x 64 voxels, used an unsigned short to represent the value at each voxel, and was stored in the file “image.raw”. The resulting MetaHeader (our naming convention would call this file “image.mhd”) file would read

```
ObjectType = Image
NDims = 3
DimSize = 256 256 64
ElementType = MET_USHORT
ElementDataFile = image.raw (this tag must be last in a MetaImageHeader)
```

That’s it, but this assumes quite a bit about the image data. Specifically, it assumes

1. There are not any non-image data bytes (header data) at the beginning of the image data file “image.raw”.
2. The voxels are cubes – the distance spanned by and between a voxel in each coordinate direction is 1 “unit”, e.g., 1x1x1mm voxel size and voxel spacing
3. The byte-order of the data in image.raw matches the byte ordering native to the machine the application is running on (e.g., PC’s use LSB ordering and Suns/Macs use MSB ordering).

If these assumptions are false, the data will not be loaded correctly by the application. To fix these problems….

1. To skip the header bytes in the image data file, use
   `HeaderSize = X`
   where X is the number of bytes to skip at the beginning of the file before reading image data. If you know there are no trailing bytes (extra bytes at the end of the file) you can specify
   `HeaderSize = -1`
   and MetaImage will automatically calculate the number of extra bytes in the data file, assume they those bytes are at the head of the data file, and automatically skip them before beginning to read the image data.

2. To specify the spacing of the voxels, use
   `ElementSpacing = X Y Z`
   where X is the distance between of the centers of the voxels along the x-dimension, Y is the spacing in the y-dimension, and Z is the spacing in the z-dimension. Therefore, to specify a 1x1x3mm voxel spacing, use
   `ElementSpacing = 1 1 3`

   NOTE: If ElementSpacing is not specified, it is assumed to be equal to ElementSize. If neither is specified, both are assumed to be 1.

3. To specify a voxel size, use
   ElementSize = X Y Z
   where X Y Z represent the size in the x, y, and z-dimensions respectively.

   NOTE: If ElementSize is not specified, it is assumed to be equal to ElementSpacing. If neither is specified, both are assumed to be 1.

4. To specify a particular byte ordering, use
   ElementByteOrderMSB = True
   or
   ElementByteOrderMSB = False
   MSB (aka big-endian) ordering is common to SPARC and Motorola processors (e.g., Macintoshes). LSB (aka little-endian) ordering is common to Intel processors (e.g., PC compatibles).

Putting it all together, to "convert" a file containing the image data in a continuous block at the end of the file, specify the header

```
ObjectType = Image
NDims = 3
DimSize = 256 256 64
ElementType = MET_USHORT
HeaderSize = -1
ElementSize = 1 1 3
ElementSpacing = 1 1 1
ElementByteOrderMSB = False
ElementDataFile = image.raw
```

Reading DICOM and Other One-Slice-Per-File Data Formats

If the data is split to be one slice per file, as is done with DICOM data, only the ElementDataFile tag's option needs to change.

Since the MetaLibrary cannot automatically parse DICOM headers, those headers must be skipped and the user must specify the image dimensions and other essential image information. For DICOM files, the MetaLibrary must automatically calculate the header size of each file (luckily for almost every type of DICOM object in common use, the image data is stored at the end).

To specify which files comprise the volume, they can be specified as an ordered list in the MetaHeader using the ElementDataFile=LIST option. The filenames should be listed at the end of the MetaHeader, after the ElementDataFile option, and the filenames should be separated by whitespace:

```
ObjectType = Image
NDims = 3
DimSize = 512 512 100
ElementType = MET_USHORT
HeaderSize = -1
ElementSize = 1 1 3
ElementSpacing = 1 1 1
ElementByteOrderMSB = False
ElementDataFile = LIST
filenameOfSlice1
filenameOfSlice2
filenameOfSlice3
filenameOfSlice4
.
. (one hundred filenames must be specified to specify the 100 slices in the volume)
.
```

The second way of specifying a series of files can be used if the filenames are numerically distinguished. That is, the files names should be able to be specified using a numeric substitution into a c-style printf-string, for a range of values. In pseudo-code:

```
for i=numBegin to numEnd step numStep
        sprintf(sliceName, "baseName.%03d", i);
end
```

The parameters of this system are numBegin, numEnd, numStep, and the c-style printf string (e.g., "baseName.%03d"). The begin, end, and step parameters appear in order after the c-style printf string:

```
ObjectType = Image
NDims = 3
DimSize = 512 512 100
ElementType = MET_USHORT
HeaderSize = -1
ElementSize = 1 1 3
ElementSpacing = 1 1 1
ElementByteOrderMSB = False
ElementDataFile = baseName.%03d 1 100 1
```

This MetaImage will cause the files "baseName.001" to "baseName.100" to be read to create a 100-slice volume.

In this case, because of the overlap of the slices, it may be helpful to only consider every-other slice in the volume. Changing the slice spacing and the ElementDataFileNumStep enacts this…

```
ObjectType = Image
NDims = 3
DimSize = 512 512 50
ElementType = MET_USHORT
HeaderSize = -1
ElementSize = 1 1 3
ElementSpacing = 1 1 2
ElementByteOrderMSB = False
ElementDataFile = baseName.%03d 1 100 2
```

8 Appendix B: Matrix Representation of Frame Position and Orientation

This information is taken from Wikipedia, the internet encyclopedia, at:
http://en.wikipedia.org/wiki/3D_projection

Data necessary for projection

Data about the objects to render is usually stored as a collection of points, linked together in triangles. Each point is a set of three numbers, representing its X,Y,Z coordinates from an origin relative to the object they belong to. Each triangle is a set of three such points. In addition, the object has three coordinates X,Y,Z and some kind of rotation, for example, three angles *alpha*, *beta* and *gamma*, describing its position and orientation relative to a "world" reference frame.

Last comes the observer (or *camera*). The observer has a set of three X,Y,Z coordinates and three *alpha*, *beta* and *gamma* angles, describing the observer's position and the direction in which it is pointing.

All this data is usually stored using floating point values, although many programs convert them to integers at various points in the algorithm to speed up the calculations.

First step: world transform

The first step is to transform the point's coordinates, taking into account the position and orientation of the object they belong to. This is done using a set of four matrices: (The matrix we use is column major, i.e. v' = Matrix*v, the same in OpenGL but different in Directx).

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — object translation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — rotation about the *x*-axis

$$\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — rotation about the *y*-axis

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — rotation about the *z*-axis.

The four matrices are multiplied together, and the result is the *world transform matrix*: a matrix that, if a point's coordinates were multiplied by it, would result in the point's coordinates being expressed in the "world" reference frame.

Note that unlike multiplication between numbers, the order used to multiply the matrices is significant; changing the order will change the results too. When dealing with the three rotation matrices, a fixed order is good for the necessity of the moment that must be chosen. The object should be rotated before it is translated, since otherwise the position of the object in the world would get rotated around the centre of the world, wherever that happens to be.

World transform = Translation × Rotation

To complete the transform in the most general way possible, another matrix called the scaling matrix is used to scale the model along the axes. This matrix is multiplied to the four given above to yield the complete world transform. The form of this matrix is:

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — where s_x, s_y, and s_z are the scaling factors along the three co-

Since it is usually convenient to scale the model in its own model space or co-ordinate system, scaling should be the first transformation applied. The final transform thus becomes:

World transform = Translation × Rotation × Scaling

(as in some computer graphics books or programming API's such as DirectX, it uses matrices with translation vectors in the bottom row, in this scheme, the order of matrices would be reversed.)

$$\begin{bmatrix} s_x \cos\gamma \cos\beta & -s_y \sin\gamma \cos\beta & s_z \sin\beta & x \\ s_x \cos\gamma \sin\beta \sin\alpha + s_x \sin\gamma \cos\alpha & s_y \cos\gamma \cos\alpha - s_y \sin\gamma \sin\beta \sin\alpha & -s_z \cos\beta \sin\alpha & y \\ s_x \sin\gamma \sin\alpha - s_x \cos\gamma \sin\beta \cos\alpha & s_y \sin\gamma \sin\beta \cos\alpha + s_y \sin\alpha \cos\gamma & s_z \cos\beta \cos\alpha & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

final result of Translation × x × y × z × Scaling

Note with respect to CustusX: In most cases 1:1 scaling will be used. The scaling factors can be set to 1 removed from the equation above.

Appendix C: Gyration MG1101a Specifications

Parameter	Conditions	Symbol	MIN	NOM	MAX	UNITS
Dynamic Range		MAV	-500		+500	deg/sec
Offset	No rotation applied	OFF	-180	0	+180	deg/sec
Offset Temperature Coefficient	Measured at 10ºC Intervals	OFF_TC		± 0.25	± 1.0	deg/sec / ºC
Offset Supply Voltage Coefficient	Measured from 1.8V to 3.6V at T_{AM}=25°C	OFF_VC	-0.75	0.22	+1.25	deg/sec / V
Rate Sensitivity	Measured at Full Scale Rate at T_{AM}=25°C	SF	31.0	32.0	33.0	LSB / deg/sec
Rate Sensitivity Temperature Coefficient	% change from SF at T_{AM}=25°C	SF_TC	-0.12	-0.04	+0.01	% / ºC
Cross Axis Coupling Rotation on Axis B, change in A output Rotation on Axis A, change in B output		 CAXA CAXB	 -3 -3	 0 0	 3 3	% of applied rotation rate
Drift	Measured from 1 to 180 seconds after power up at T_{AM}=25°C	TOD	-2.0		+2.0	deg/sec
Power on Reset threshold (Brown out detection)		V_{POR}	1.5		1.75	V
Power Up Time		t_{GRNR}		55	150	msec
Power On Settling Time		t_{POST}		800	1000	msec
Rotational Rate Nonlinearity		NLIN		± 0.05	± 0.5	% FS
Gyro Digital Output Report Rate (refresh of rotation rate output after new ADC conversion)			26.7	29	31.3	Hz
Rotational Rate Input Bandwidth	3dB attenuation, 45 deg phase shift input to output	BW	13.4	14.5	15.7	Hz
Noise floor: Axis A Axis B	No vibration applied	 NZ_A NZ_B		 0.11 0.14	 0.30 0.35	 deg/sec (rms) deg/sec (rms)
Sleep Mode Current	With serial bus idle, after <Init> Bit is set.	I_{SLEEP}		3.0	10	uA
Steady State Supply Current	Full Mode, Normal Operation	I_{FULL}		7.5	10.0	mA
Temperature Sensor Sensitivity		T_{SLOPE}	4.757	5.115	5.500	LSB / Kelvin
Voltage Sensor Sensitivity		V_{SLOPE}	200.0	204.7	210.0	LSB / V
Mass		M		6.8		g

Appendix D: Analog Devices ADIS16250 Specifications

Parameter	Conditions	Min	Typ	Max	Unit
SENSITIVITY[1]	Clockwise rotation is positive output				
	25°C, dynamic range = ±320°/sec[2]		0.07326		°/sec/LSB
	25°C, dynamic range = ±160°/sec		0.03663		°/sec/LSB
	25°C, dynamic range = ±80°/sec		0.01832		°/sec/LSB
Initial Tolerance	25°C, dynamic range = ±320°/sec		±0.2	±1	%
Temperature Coefficient	ADIS16250 (see Figure 8)		225		ppm/°C
	ADIS16255 (see Figure 11)		25		ppm/°C
Nonlinearity	Best fit straight line		0.1		% of FS
BIAS					
In Run Bias Stability	25°C, 1σ		0.016		°/sec
Turn-On-to-Turn-On Bias Stability	25°C, 1σ		0.05		°/sec
Angular Random Walk	25°C, 1σ		3.6		°/√hour
Temperature Coefficient	ADIS16250 (see Figure 7)		0.03		°/sec/°C
	ADIS16255 (see Figure 10)		0.005		°/sec/°C
Linear Acceleration Effect	Any axis		0.2		°/sec/*g*
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		1.0		°/sec/V
NOISE PERFORMANCE					
Output Noise	At 25°C, ±320°/sec range, no filtering		0.48		°/sec rms
	At 25°C, ±160°/sec range, 4-tap filter setting		0.28		°/sec rms
	At 25°C, ±80°/sec range, 16-tap filter setting		0.14		°/sec rms
Rate Noise Density	At 25°C, f = 25 Hz, ±320°/sec range, no filtering		0.056		°/sec/√Hz rms
FREQUENCY RESPONSE					
3 dB Bandwidth	See the Analog Bandwidth section for adjustment		50		Hz
Sensor Resonant Frequency			14		kHz
SELF-TEST STATE					
Change for Positive Stimulus	320°/sec dynamic range setting	439	721	1092	LSB
Change for Negative Stimulus	320°/sec dynamic range setting	−439	−721	−1092	LSB
Internal Self-Test Cycle Time			20		ms
TEMPERATURE SENSOR					
Output at 25°C			0		LSB
Scale Factor			6.88		LSB/°C
ADC INPUT					
Resolution			12		Bits
Integral Nonlinearity			±2		LSB
Differential Nonlinearity			±1		LSB
Offset Error			±4		LSB
Gain Error			±2		LSB
Input Range		0		2.5	V
Input Capacitance	During acquisition		20		pF
ON-CHIP VOLTAGE REFERENCE			2.5		V
Accuracy	At 25°C	−10		+10	mV
Temperature Coefficient			±40		ppm/°C
Output Impedance			70		Ω

Appendix E: Analog Devices ADIS16350 Specifications

Parameter	Conditions	Min	Typ	Max	Unit
GYROSCOPE SENSITIVITY	Each axis				
Initial Sensitivity	25°C, dynamic range = ±300°/s	0.0725	0.07326	0.0740	°/s/LSB
	25°C, dynamic range = ±150°/s		0.03663		°/s/LSB
	25°C, dynamic range = ±75°/s		0.01832		°/s/LSB
Temperature Coefficient	ADIS16350, see Figure 6		600		ppm/°C
	ADIS16355, see Figure 9		40		ppm/°C
Gyroscope Axis Nonorthogonality	25°C, difference from 90° ideal		±0.05		Degree
Gyroscope Axis Misalignment	25°C, relative to base-plate and guide pins		±0.5		Degree
Nonlinearity	Best fit straight line		0.1		% of FS
GYROSCOPE BIAS					
In Run Bias Stability	25°C, 1 σ		0.015		°/s
Angular Random Walk	25°C		4.2		°/√hr
Temperature Coefficient	ADIS16350, see Figure 7		0.1		°/s/°C
	ADIS16355, see Figure 10		0.01		°/s/°C
Linear Acceleration Effect	Any axis, 1 σ (linear acceleration bias compensation enabled)		0.05		°/s/*g*
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		0.25		°/s/V
GYROSCOPE NOISE PERFORMANCE					
Output Noise	25°C, ±300°/s range, 2-tap filter setting		0.60		°/s rms
	25°C, ±150°/s range, 8-tap filter setting		0.35		°/s rms
	25°C, ±75°/s range, 32-tap filter setting		0.17		°/s rms
Rate Noise Density	25°C, f = 25 Hz, ±300°/s, no filtering		0.05		°/s/√Hz rms
GYROSCOPE FREQUENCY RESPONSE					
3 dB Bandwidth			350		Hz
Sensor Resonant Frequency			14		kHz
GYROSCOPE SELF-TEST STATE					
Change for Positive Stimulus	±300°/s range setting	432	723	1105	LSB
Change for Negative Stimulus	±300°/s range setting	−432	−723	−1105	LSB
Internal Self-Test Cycle Time			25		ms
ACCELEROMETER SENSITIVITY	Each axis				
Dynamic Range		±8	±10		*g*
Initial Sensitivity	25°C	2.471	2.522	2.572	mg/LSB
Temperature Coefficient	ADIS16350, see Figure 8		100		ppm/°C
	ADIS16355, see Figure 11		40		ppm/°C
Axis Nonorthogonality	25°C, difference from 90° ideal		±0.25		Degree
Axis Misalignment	25°C, relative to base-plate and guide pins		±0.5		Degree
Nonlinearity	Best fit straight line		±0.2		% of FS
ACCELEROMETER BIAS					
In-Run Bias Stability	25°C, 1 σ		0.7		m*g*
Velocity Random Walk	25°C		2.0		m/s/√hr
Temperature Coefficient	ADIS16350, see Figure 12		4		m*g*/°C
	ADIS16355, see Figure 15		0.5		m*g*/°C

Appendix F: Avago ADNS-2610 Specifications

Parameter	Symbol	Minimum	Typical	Maximum	Units	Notes
Operating Temperature	T_A	0		40	°C	
Power Supply Voltage	V_{DD}	4.1	5.0	5.5	Volts	Register values retained for voltage transients below 4.10V but greater than 3.9V
Power Supply Rise Time	V_{RT}			100	ms	
Supply Noise	V_N			100	mV	Peak to peak within 0-100 MHz bandwidth
Clock Frequency	f_{CLK}	23.0	24.0	25.0	MHz	Set by ceramic resonator
Serial Port Clock Frequency	SCLK			$f_{CLK}/12$	MHz	
Resonator Impedance	X_{RES}			55	Ω	
Distance from Lens Reference Plane to Surface	Z	2.3	2.4	2.5	mm	Results in ±0.2 mm DOF (See Figure 9)
Speed	S	0		12	in/sec	@ frame rate = 1500 fps
Acceleration	A			0.25	g	@ frame rate = 1500 fps
Light Level onto IC	IRR_{INC}	80 100		25,000 30,000	mW/m^2	λ = 639 nm λ = 875 nm
SDIO Read Hold Time	t_{HOLD}	100			µs	Hold time for valid data (Refer to Figure 22)
SDIO Serial Write-write Time	t_{SWW}	100			µs	Time between two write commands (Refer to Figure 25)
SDIO Serial Write-read Time	t_{SWR}	100			µs	Time between write and read operation (Refer to Figure 26)
SDIO Serial Read-write Time	t_{SRW}	250			ns	Time between read and write operation (Refer to Figure 27)
SDIO Serial Read-read Time	t_{SRR}	250			ns	Time between two read commands (Refer to Figure 27)
Data Delay after PD deactivated	$t_{COMPUTE}$	3.1			ms	After $t_{COMPUTE}$, all registers contain data from first image after wakeup from Power-Down mode. Note that an additional 75 frames for AGC stabilization may be required if mouse movement occurred while Power Down. (Refer to Figure 10)
SDIO Write Setup Time	t_{SETUP}	60			ns	Data valid time before the rising of SCLK (Refer to Figure 20)
Frame Rate	FR		1500	2300	frames/s	See Frame_Period register section

Appendix G: ST Microelectronics LIS3LV02DL Specifications

Symbol	Parameter	Test conditions	Min.	Typ.(2)	Max.	Unit
FS	Measurement range(3)	FS bit set to 0	±1.7	±2.0		g
		FS bit set to 1	±5.3	±6.0		
Dres	Device resolution	Full-scale = ±2 g ODR1=40 Hz		1.0		mg
		Full-scale = ±2 g ODR2=160 Hz		2.0		
		Full-scale = ±2 g ODR3=640 Hz		3.9		
		Full-scale = ±2 g ODR4=2560 Hz		15.6		
So	Sensitivity	Full-scale = ±2 g 12 bit representation	920	1024	1126	LSb/g
		Full-scale = ±6 g 12 bit representation	306	340	374	
TCSo	Sensitivity change vs temperature	Full-scale = ±2 g 12 bit representation		0.025		%/°C
Off	Zero-g level offset accuracy(4),(5)	Full-scale = ±2 g X, Y axis	-70		70	mg
		Full-scale = ±2 g Z axis	-90		90	
		Full-scale = ±6 g X, Y axis	-90		90	
		Full-scale = ±6 g Z axis	-100		100	
LTOff	Zero-g Level offset long term accuracy(6)	Full-scale = ±2 g X, Y axis	-4.5		4.5	%FS
		Full-scale = ±2 g Z axis	-6		6	
		Full-scale = ±6 g X, Y axis	-1.8		1.8	
		Full-scale = ±6 g Z axis	-2.2		2.2	
TCOff	Zero-g Level Change Vs Temperature	Max Delta from 25°C		0.2		mg/°C

Appendix H: Edmunds Fiber Optic Bundle Specifications

Minimum Bend Radius	4 x Rod Diameter
Numerical Aperture	0.53 std. res.; 0.55 high res.
Max. Operating Temperature	454°C (850°F)
Softening Point	704°C (1300°F)
Conduit Resolution (500 / Fiber Diameter)	12µm: 42 lp/mm 24µm: 20 lp/mm 25µm: 20 lp/mm 50µm: 10 lp/mm 100µm: 5 lp/mm
Dimensional Tolerance	Length: ±0.030" Diameter: ±0.010"
Core Refractive Index	1.58
Cladding Ref. Index	1.49 std. / 1.48 high
Rod Fiber Count	3,012 std. res.; 50,419 high res.
Packing Ratio	100%
Approx. Transmission	35 - 45% in the range of 400 to 750nm; (larger diameter = higher transmission)