The 3D Interaction Tool: A Pointing Device for Virtual Reality Applications

A senior design project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering Sciences at Harvard University

Lukas Gemar

Faculty Advisors:
Avi Uttamchandani and Krzysztof Gajos

Harvard University School of Engineering and Applied Science

Cambridge, MA

April 1, 2016

Contents

List of Figures

I. Abstract

The 3D Interaction Tool is a hand-held device that enables a user to interact with virtual reality applications. Since both the position and orientation of the tool are tracked, user input with the tool accomplishes tasks such as selecting, translating, and rotating three-dimensional objects, or free-hand drawing in space. The tools position is tracked by a software application that filters observations from a camera attached to the virtual reality display, and the orientation is found by means of observations from an inertial measurement unit attached to the tool itself. Simple computer-aided design and drawing applications are possible with the capabilities of the 3D Interaction Tool.

II. Nomenclature

Position vectors definitions in the inertial coordinate frame:
$\vec{p}_r \equiv$ position of the tool rear
$\vec{p}_f \equiv$ position of the tool front
$\vec{p}_l \equiv$ span vector that connects in the rear of the tool to the front of the tool
$\vec{p}_c \equiv$ position of the tracking camera
$\vec{p}_d \equiv$ position of the virtual reality display

Rotated position vectors:
$\vec{p}'_r \equiv$ position of the tool rear in display coordinates
$\vec{p}'_f \equiv$ position of the tool front in display coordinates
$\vec{p}'_l \equiv$ span vector in the body coordinate system of the interaction tool

Attitude matrices:
$\mathbf{A}_c \equiv$ attitude of the tracking camera
$\mathbf{A}_d \equiv$ attitude of the virtual reality display
$\mathbf{A}_t \equiv$ attitude of the interaction tool
$\mathbf{A}'_t \equiv$ attitude of the interaction tool in display coordinates

Body-referenced quaternion representations of attitude:
$\overline{\mathbf{q}}_c \equiv$ attitude of the tracking camera
$\overline{\mathbf{q}}_t \equiv$ attitude of the interaction tool

Position error covariances in inertial coordinate frame:
$\mathbf{\Sigma}_{pr} \equiv$ covariance of the position of the rear of the interaction tool
$\mathbf{\Sigma}_{pf} \equiv$ covariance of the position of the front of the interaction tool

Position error covariances in rotated body coordinates:
$\mathbf{\Sigma}'_{pr} \equiv$ covariance of the position of the rear of the interaction tool in rotated coordinates
$\mathbf{\Sigma}'_{pf} \equiv$ covariance of the position of the front of the interaction tool in rotated coordinates

General:
$\overline{\mathbf{q}} \equiv$ quaternion
$A(\overline{\mathbf{q}}) \equiv$ rotation matrix from the world coordinate frame to the orientation specified by $\overline{\mathbf{q}}$
$\delta\overline{\mathbf{q}}_t \equiv$ small orientation errors in the quaternion
$\delta\theta \equiv$ small orientation errors

Kalman Filter $Q \equiv$ process noise matrix of the Kalman filter $R \equiv$ measurement noise matrix of the Kalman filter $K \equiv$ Kalman gain $H \equiv$ measurement sensitivity matrix $\mu \equiv$ innovation

III. Introduction

A. Motivation

As much as $25 billion could be spent on virtual reality (VR) and augmented reality (AR) software applications by 2025 [**?**]. There are two primary modes of user-input that are possible for these applications: indirect and direct. Indirect input devices translate the user's physical movements into the motion of a virtual cursor by means of a fixed transformation. Common examples of indirect input devices are a mouse or joystick. Direct input devices also translate physical to virtual movements, but this transformation is one-to-one. For example, a touch screen is a direct input device because the physical location of the user's press is collocated with the virtual click.

With the advent of virtual reality applications the following question must be answered: which of these input paradigms – indirect or direct – will prove most useful for interacting with virtual reality applications? While indirect input devices have achieved widespread popularity for two-dimensional displays, the direct input paradigm for VR applications embraces the immersion of a user in a VR experience. Direct input devices allow users to reach out and interact with the objects of their virtual space, just as they would with objects in their physical space.

B. Design Approach

The 3D Interaction Tool is a novel, low-cost 3D input device. In particular, the tool is well-suited for VR applications that use a mobile-phone for the virtual reality display, such as for those applications that run on the Google Cardboard or other low cost solutions. This versatility of the 3D Interaction Tool comes from the simple requirement that the VR display have a front-facing camera attached. This makes the tracking system compatible with any smartphone. The purpose of the 3D Interaction Tool is to demonstrate the functionality of this versatile tracking system. While the tool itself is designed for use with a tracking system on a VR display, the scope of this report is to evaluate the tool's performance in a more controlled environment. The simplifying assumptions necessary to do this are specified below.

IV. Background

A. Commercial Solutions

Recent market trends suggest that direct input devices are on the rise for consumers. Two of these direct input devices, the Oculus Touch controller and the HTC Valve controller are achieving widespread publicity surrounding the commercial release of virtual reality headsets [**?**] [**?**] [**?**]. These two controllers are shown in Figure 1, with the Oculus Touch depicted on the left and the HTC Controllers depicted on the right. While originally scheduled for release alongside the Oculus VR headset at the end of March 2016, the Oculus Touch controller will be available in the second half of 2016 [**?**]. Meanwhile the Valve controllers have been reviewed in preparation for the release of the HTC Vive headset at the beginning of April, 2016, and their utility has been demonstrated with a three-dimensional sketching program by Google called TiltBrush [**?**]. These input devices track both their position and orientation, making them degree-of-freedom (6DOF) controllers. While the 6DOF controllers are reported to work well with their respective headsets, the headsets are expensive – both retail for more than $600 – and the controllers are not useful with other VR displays. Thus, two of the drawbacks of these commercial VR controllers are their cost and their compatibility requirements.

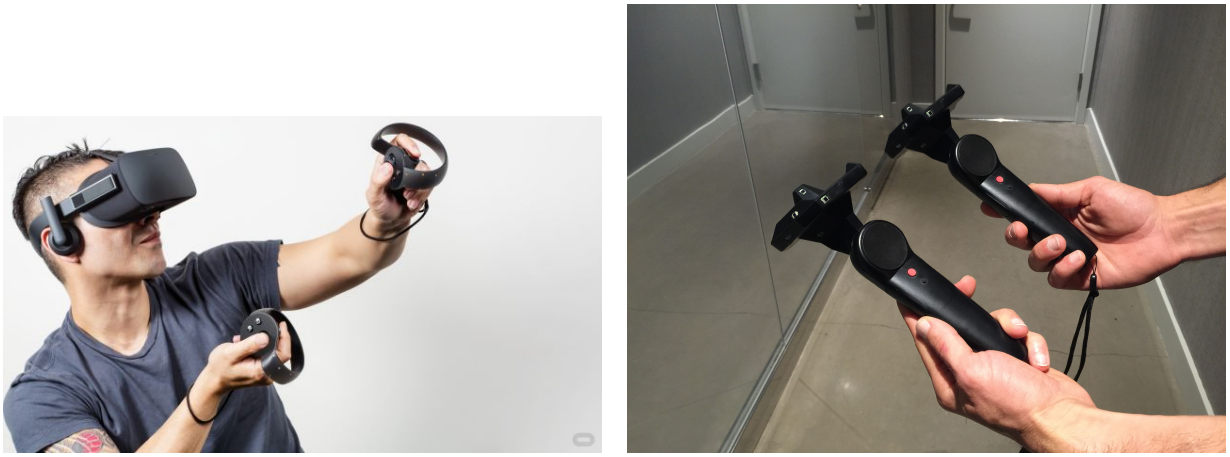

Figure 1: Commercial VR input devices

B. 3D Interaction Techniques and their Applications

Three-dimensional interaction techniques can be separated into several categories. First, there are interactions that manipulate objects. These interactions accomplish tasks such as rotating, translating, or scaling three dimensional objects. Second, there are interactions that manipulate 3D viewpoint, allowing the user to adjust their field of view. Lastly, there are interactions that control an application. Application control interactions include navigating menus or widgets associated with the application logic. The first set of tasks, those that manipulate objects, have largely been the focus of three-dimensional input device design [**?**], and the 3D Interaction Tool is designed the object manipulation tasks in mind.

Object manipulation tasks are required in artistic and design applications. Several projects have demonstrated the usefulness of 3D input techniques to perform these tasks. For example, the HoloSketch project demonstrated a 3D wand for object creation and manipulation. In HoloSketch the wand could be used for drawing and animating three dimensional objects in a virtual reality environement (VE) [**?**]. The HoloSketch wand also enabled a user to create free-hand drawings in three-space.

The CavePainting research also allowed a user to manipulate objects using an input device – and not just one but many such input devices, such as paint brushes and paint buckets. These physical props were used to create artistic works in VE [**?**]. The props gave users a very tactile experience of virtual object manipulation, and the tracking system, known as the Cave, was able to track these objects while the user walked around in the virtual environment. Both of these solutions investigated the 3D input device interface for the process of artistic creation.

In addition to artistic applications, 3D interface design has been explored for its potential use for CAD applications. For example, the 6D hands project enabled a user to interact with objects in a 3D modelling environment to more quickly and efficiently perform object manipulation tasks [**?**].

C. Tracking Systems for 3D Input Devices

The methods of tracking objects in virtual environments are numerous [**?**]. These tracking methods are a subset of six degree of freedom tracking approaches, and none of these tracking systems are without drawbacks, leading researchers to develop tracking systems that are well-suited to the needs of their particular application [**?**]. The main categories of 6DOF trackers for virtual environments are mechanical, inertial, acoustic, magnetic, and optical [**?**].

V. Design Goals

There are two broad approaches to 3D input device design. First, there are approaches focused on designing the user-interface for the device. This user-interface would be suitable for a particular

application. In short, this approach attempts to answer the question: will this device be usable? Second, there are approaches that explore the functionality of the device for a number of different types of applications, asking the question: will this device be high performance? This report seeks to answer this second question for a three-dimensional input device tracked with a particular configuration of a 6DOF tracking system.

A. Usability

The usability requirements presented in Figure 2 must be met for the tool to succeed as an *input* device for VR software applications.

Usability	
Objective	Target
Portability	The interaction tool tracking system should be easily translatable between uses
Natural form factor	Easy to learn and provides high degree of control
Large tracking volume	$\approx 1m^3$
Low cost	<$20

Figure 2: Usability design objectives

The first of these criteria is portability. The user should be able to use the input device wherever they are using a virtual reality display, without having to precalibrate their environment. Note that this portability objective is different from a *mobility* objective. Mobility would allow unrestricted movement of the device. The reason portability is chosen as a design requirement rather than mobility is that low-cost virtual reality displays such as Google Cardboard are extremely portable and not very mobile. When a smartphone is used as a virtual reality display, it is not aware of its position in its environment. This is true of other low-cost virtual reality displays as well, such as the Samsung GearVR [**?**]. Thus, the device is chosen to be as portable as the displays with which it would be used.

The second usability design objective is a natural form factor for the device. The natural form factor serves the ends of making the device easy to learn, preventing fatigue, and providing user's with a large degree of control over the manipulation of the device.

Third, the tracking volume – the spatial range over which the position of the device is tracked – should not constrain the interactions that are possible with the device. Since the interaction tool should be usable with a number of different applications, no assumptions should be made about what size tracking volume these applications will require. A reasonable assumption is that the user will not use the device to directly interact with objects outside of the reach of their arms, so the tracking volume should cover the reach of an average user. This tracking volume is on the order of magnitude of a meter.

Finally, as mentioned before the 3D Interaction Tool should be compatible with low-cost virtual reality displays. Therefore, the tool itself must be low cost.

B. Performance

The performance requirements presented in Figure 3 must be met for the tool to succeed as a *direct* input device for VR software applications.

Performance	
Objective	Target
RMSE (Position)	<5cm
RMSE (Orientation)	<5°
Latency	<60ms.
Robustness	Robust to tool velocity, tool occlusions

Figure 3: Performance design objectives

The tracking system must be accurate in its position and orientation estimation to produce a high fidelity reconstruction of device movement in the user's virtual reality application. For the device to truly be a direct input device, the accuracy of the mapping between physical and virtual worlds should be as low as 1mm, as is required for the physical and virtual registration to be imperceptible [**?**]. In virtual reality, as opposed to augmented reality, the accuracy constraint can be relaxed somewhat. The input device will be usable as long as movements are precise and repeatable, as is the case with indirect input devices. Thus position error of about 5% along each axis of the trackign volume is acceptable, leading to an accuracy target of 5cm. For orientation, device usability studies have shown that physical and virtual orientation differences of 5° are imperceptible to a user [**?**]. Therefore, the target for orientation estimation is an accuracy better than 5°.

To quantify the accuracy, the root-mean-squared error model is used. The root-mean-squared error (RMSE) is the square root of the variance between the estimated values and the position or orientation reference. This error model is useful because it can be decomposed into bias and variance components.

Requirements for low latency and high robustness fill out the design specifications. The latency should be as low as possible since the discrepency between virtual motion and physical motion has been known to induce simulator sickness in virtual environments [**?**]. Best practices place the maximum latency that is comfortable for a user at 60ms [**?**]. As far as robustness, the tracking of the device should be robust to tool occlusions and tool velocities. This will ensure that the user has control of the device in the virtual environment at all times, no matter how the tool is configured or moved in the physical environment.

VI. Design Approach

A. Requirements

In order to enable direct input with the interaction tool, the tool's tracking system must estimate both the position and orientation of the interaction tool. Additionally, rendering a virtual representation of the tool requires measuring its position and orientation relative to the virtual reality display orientation, given by the attitude matrix $\mathbf{A}_d$. To track the tool's absolute trajectory, its position is determined relative to the world – that is, relative to a non-accelerating, inertial coordinate system in the user's physical environment. The position and orientation of the tool in the world coordinates of the inertial frame are $\vec{p}_f$ and $\mathbf{A}_t$, respectively. The position of the tool in the coordinate frame of the virtual reality display is denoted by a 'prime' sign: the tip or front of the tool in display coordinates is $\vec{p}'_f$. Thus, there are four tracking targets: $\vec{p}_f$, $\mathbf{A}_t$, $\vec{p}'_f$, and $\mathbf{A}_d$; these are shown in Figure 4.

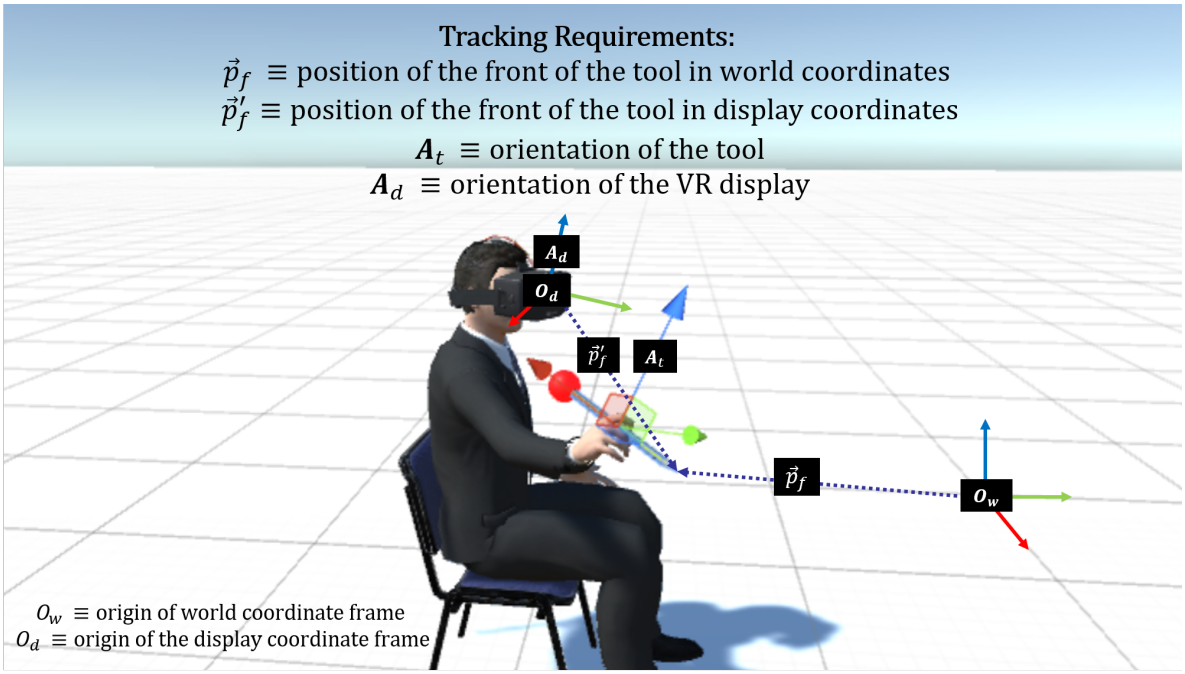

Figure 4: Position and orientation tracking requirements

B. Geometry

The geometry of the tool tracking system affects both the usability and the performance of the interaction tool. The display-referenced geometry is chosen because it eliminates the burden on the user to pre-calibrate their environment before providing direct input to VR applications. At the same time, this geometry is compatible with VR applications with which a user interacts while seated.

B.1 Option 1: World-referenced geometry

The world-referenced tracking geometry is shown in Figure 5.

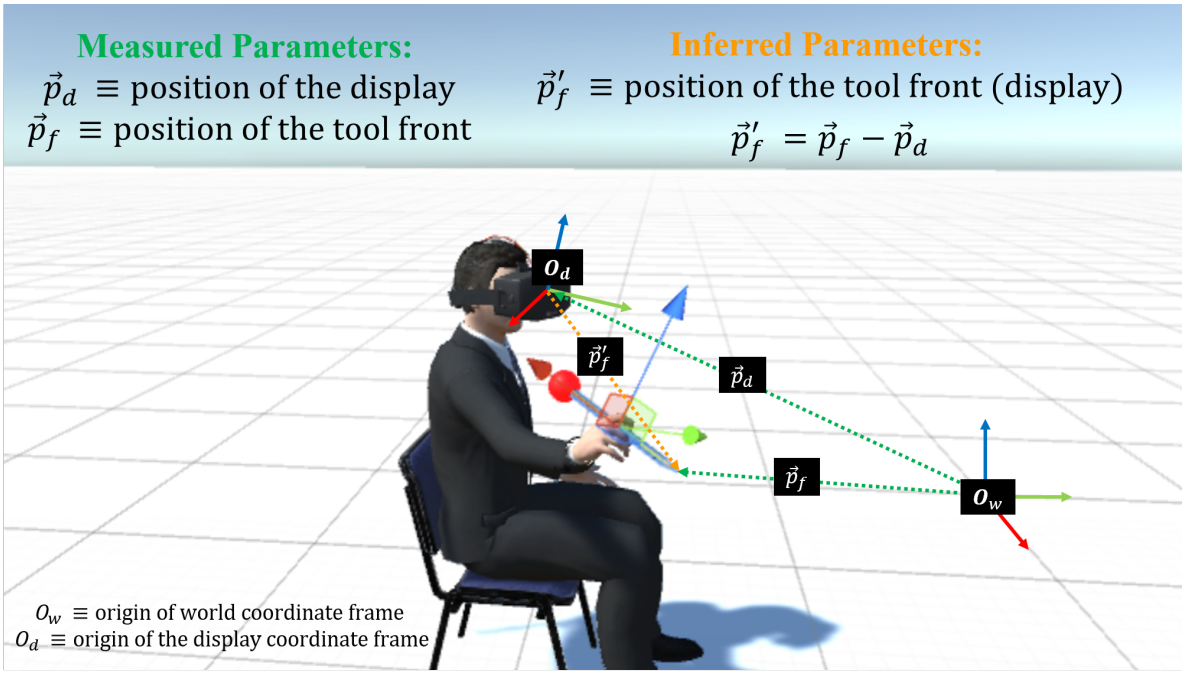

Figure 5: Position tracking with a world-referenced geometry

B.2 Option 2: Display-referenced geometry

The display-referenced tracking geometry is shown in Figure 6.

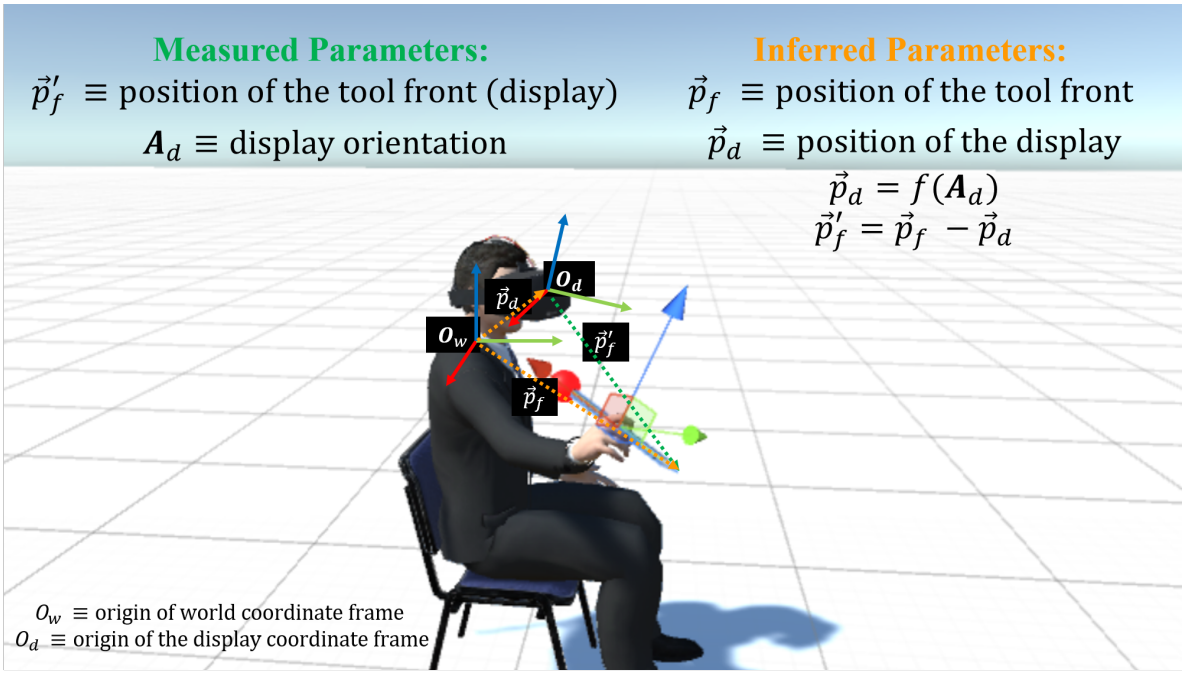

Figure 6: Position tracking with a display-referenced geometry

B.3 Evaluating Tracking Geometries: Comparing portability and tracking volume

Tracking geometries 1 and 2 have advantages and disadvantages when evaluated according to the usability design objectives. Relative to the usability criteria, the two geometries differ greatly in their portability and tracking volume.

The world-referenced tracking geometry requires a fixed coordinate system in the user's physical environment, decreasing portability by increasing the configuration overhead for using the direct input device. However, once this coordinate system is established, the user can provide direct input to a VR application over a very large tracking volume, up to the size of the room in which the tracking system is located. The display-referenced geometry, on the other hand, has the advantage of enabling a user to provide direct input to a VR application without pre-calibrating the physical environment in which they are using the VR display. The trade-off is that the tracking volume is small. Since the origin of the world coordinate system in the display-referenced geometry is located beneath the user's head, the user must remain seated while using a direct input device with a display-referenced tracking geometry.

As discussed previously, *portability* is preferred to *mobility* in order to make the input device compatible with low-cost VR displays that are also highly portable.

C. Technology

A combination between optical and inertial sensors was chosen to track the 3D Interaction Tool's position and orientation. The chart in Figure 7 summarizes the comparison between various alternatives for the choice of the tracking technology.

Tracking Technology	Advantages	Disadvantages
Mechanical	Very low **bias** and **variance.** Good **portability.**	Small **tracking volume.**
Magnetic	Very low **bias** and **variance**. Low **latency**. Good r**obustness**.	Very high **cost.**
Ultrasound	Low **bias** and acceptable **variance.**	Poor **portability.** Incompatible with slender **form factor**. Poor **robustness** to acoustic noise.
Optical	Low **bias** and **variance**. Very low **configuration overhead.** Extremely **cost** effective**.**	Poor **robustness** to occlusions and tool velocity. High **latency.** Narrow **field of view.**
Inertial	Very low **latency**. Low **cost.** Acceptable levels of **bias** and **variance** measurements at short time scales.	Large **bias** of inertial measurements. Low **accuracy** and **precision** of position over large time scales.

Figure 7: Comparison of five tracking technologies

Magnetic and mechanical sensors were quickly eliminated as tracking options due to the high cost of the former and the limited tracking volume of the latter. Acoustic sensors were considered but rejected due to suboptimal sensor geometries [**?**]. These geometries were incompatible with the slender design of the interaction tool.

D. Final Design

D.1 System Overview

The final system layout is shown in figure 8. The genius of this design is that it is completely robust to tool occlusions, provided that the user is holding the interaction tool as they would a pen or stylus. In this geometry the back of the interaction tool, which is the tracking target of the camera, is always in view. Moreover, this design achieves a large tracking volume because it only requires that the viewpoint of the user be directed in the general direction of the interaction tool for the tracking camera to find the tool in its field of view. This is possible because the display and the tracking camera are collocated.

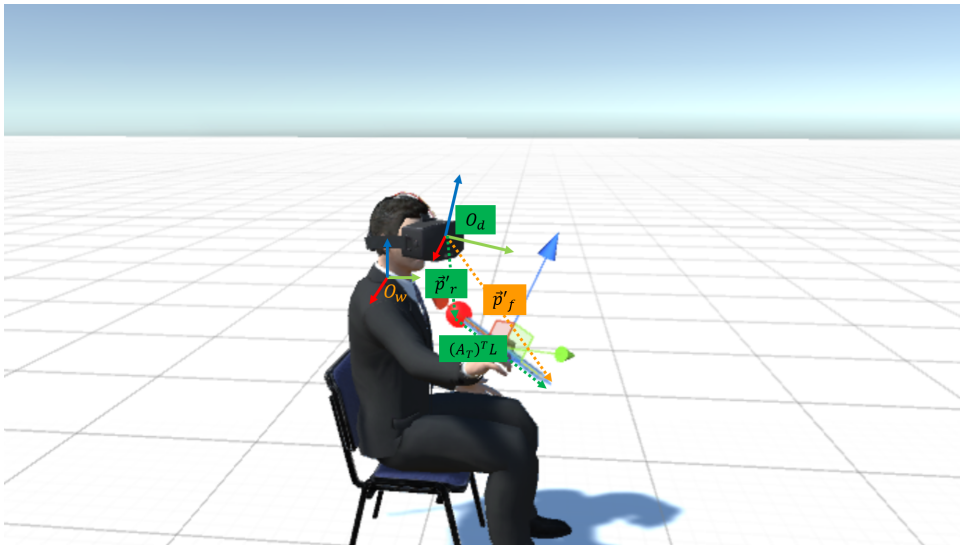

Figure 8: Final tracking system design: a display-referenced tracking topology

However, one of the downsides of this tracking design is that the position of the front of the tool $\vec{p}_f$, the position of interest, is largely occluded from the field of the camera and must be inferred from the measurements of the position of the rear of the tool and the orientation of the tool. Since both error in the tracking of the tool rear $\vec{p}_r$ and the tracking of the orientation of the tool $\mathbf{A}_t$ (and the span vector $\vec{p}_l$) will lead to error in the estimation of the position of the end of the tool, both of these measurements must be made with as little bias and variance as possible.

D.2 Tracking measurements

Not all of these three tracking targets are measured directly. The tracking targets are inferred from the measurements of the position of the rear of the interaction tool relative to the camera, the attitude of the camera, and the attitude of the tool. The position of the tool relative to the camera is denoted by $\vec{p}'_r$, the attitude of the camera relative to the inertial coordinate system by $\mathbf{A}_c$ and the attitude of the tool relative to the inertial coordinate system by $\mathbf{A}_t$. The vector $\vec{p}'_r$ is measured by finding the position of the tracking target in the camera field of view and measuring its size to determine its depth. Using its depth and its position in the field of view, its three-dimensional position relative to the camera can be estimated. This model will be presented in more detail in a later section of the report.

D.3 Inferring the tool position

The position of the front of the tool in world coordinates is given by $\vec{p}_f$. This position is inferred through measurements of the position of the rear of the tool and the tool's orientation. The equation for the position of the front of the tool is

$$\vec{p}_f = \vec{p}_r + \vec{p}_l = \vec{p}_r + \mathbf{A}_t^{-1}\vec{p}'_l$$

where the position of the rear of the tool, $\vec{p}_r$ is given by

$$\vec{p}_r = \mathbf{A}_c^{-1}\vec{p}'_r - \vec{p}_c$$

and $\vec{p}_l$ is the span vector that relates the rear of the tool to the front of the tool – it is essentially the length of the tool rotated from the body coordinate system of the tool to world coordinates. The attitude matrix given by, $\mathbf{A}_t^{-1}$, is the rotation from the body coordinates of the tool to the world coordinate system.
In order to render the tool on the display for the user, the position of the tool must be known to the VR application in the coordinate system of the VR display. The position of the front of the tool in display coordinates is $\vec{p}'_f$ and it is found by multiplying the tool's position in the world coordinate system by the attitude of the display. Additionally, the position of the display must be added to the tool position before multiplying by the display attitude matrix:

$$\vec{p}'_f = \mathbf{A}_d(\vec{p}_f + \vec{p}_d)$$

The tool attitude in the coordinate system of the display is denoted by $\mathbf{A}'_t$ and it is found by composing the rotation from the tool's body coordinates to world coordinates with the display attitude matrix:

$$\mathbf{A}'_t = \mathbf{A}_d \mathbf{A}_t^{-1} \vec{p}_l$$

This is the tool orientation visible to the user.

VII. Implementation

The system's physical layout is shown in 9. For simplicity, the tracking camera is assumed to be fixed. The model for this system is geometrically equivalent to the one presented in Figure 8. The webcam of the computer is used as the tracking camera. The inertial measurement unit, the Adafruit BNO055, is attached to the breadboard. The Adafruit sensor was chosen because it has built-in calibration routines for its three sensors: an accelerometer, a gyroscope, and a magnetometer. Since all of these sensors have bias that must be subtracted off, the calibration routines of the BNO055 are of great help to the prototyping process.

A colored foam sphere is attached to the rear of the interaction tool to serve as the tracking target for the camera.

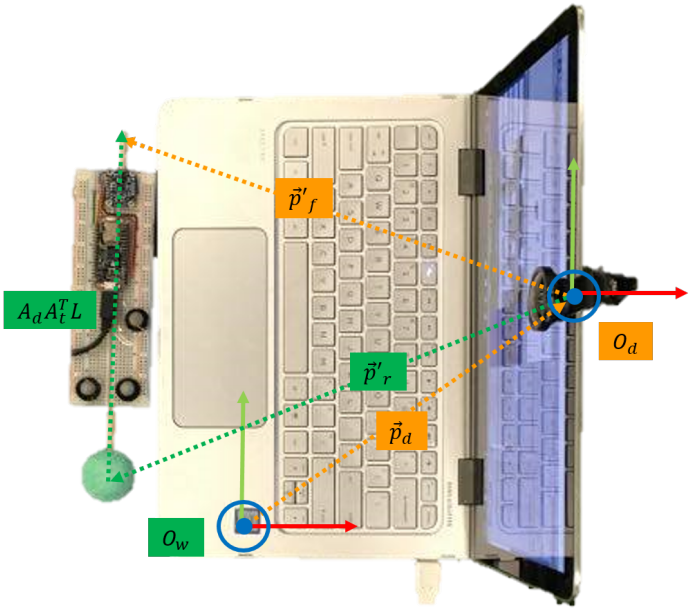

Figure 9: Geometrically equivalent tracking system

The virtual reality application runs on the computer shown in Figure 9. While this virtual reality application is not immersive, it creates a virtual 3D environment displayed to the user on the computer screen, where they can interact with virtual objects. Thus, this geometry is sufficient to prove the usefulness of the 3D Interaction Tool for virtual reality applications.

The spherical tracking target is chosen because its size in each image does not depend on its orientation. Additionally, if the angle subtended by the tracking target in the field of view is small,

then the midpoint of the tracking target in world coordinates is on the same plane as the vanishing points on the edges of the target.

The block diagram of the implemented tracking system is shown in Figure 10.

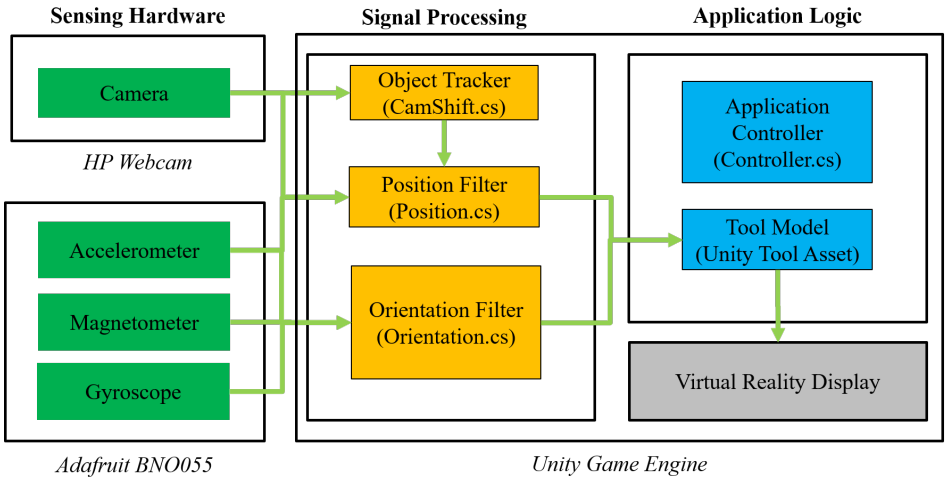

Figure 10: Full system diagram of implementation

The files referenced with the extension 'cs' are included with the code repository of this report at `https://github.com/lgemar/thesis`.

VIII. Orientation Tracking

The orientation of the interaction tool must be estimated to find the span vector, $\vec{p}_l = \vec{p}_f - \vec{p}_r = \mathbf{A}_t \vec{p}'_l$. The naive approach for estimating the orientation of the tool is to start with two known vectors in the reference coordinate system – the gravity vector and the earth's magnetic field vector are good candidates. Then we can find the attitude matrix directly by sensing these two reference vectors in the body coordinates of the device and finding the rotation matrix that maps them back to the reference coordinate frame. While this method is computationally and conceptually simple, the estimation is very sensitive to noise from both the accelerometer and the magnetometer that sense the gravity and magnetic field vectors, respectively. We would like some way of filtering these estimates in real-time. Before investigating a real-time orientation tracking filter, the results of the TRIAD measurement model are presented for comparison.

A. Naive Measurement Model: TRIAD

The TRIAD estimation method finds the attitude matrix A that maps three reference frame vectors $\vec{r}_k$ to their corresponding body frame vectors $\vec{b}_k$ for $k = 1, 2, 3$ [?]. Two observations of reference frame vectors provide sufficient information to compute a right-handed orthonormal triad of vectors in the reference frame. If $\vec{v}_1$ and $\vec{v}_2$ are reference frame vectors for the accelerometer and the magnetometer, respectively so that the gravity vector points down and the magnetometer vector points in the direction of magnetic North in Boston, relative to the world reference frame. The orthonormal vector triad representing the reference frame is given by,

$$\vec{r}_1 = \frac{\vec{v}_1}{\|\vec{v}_1\|}, \qquad \vec{r}_2 = \frac{\vec{v}_1 \times \vec{v}_2}{\|\vec{v}_1 \times \vec{v}_2\|}, \qquad \vec{r}_3 = \vec{r}_1 \times \vec{r}_2$$

Let $\vec{w}_1$ be the measurement of the normalized unit vector from the accelerometer and $\vec{w}_2$ be the measurement of the normalized unit vector from the magnetometer, then the orthonormal vector triads for the reference vectors are given by,

$$\vec{b}_1 = \frac{\vec{w}_1}{\|\vec{w}_1\|}, \qquad \vec{b}_2 = \frac{\vec{w}_1 \times \vec{w}_2}{\|\vec{w}_1 \times \vec{w}_2\|}, \qquad \vec{b}_3 = \vec{w}_1 \times \vec{w}_2$$

If there was no noise in the system, then it would be the case that $\vec{b}_k = A\vec{r}_k$ for $k = 1, 2, 3$. An equivalent way of writing this in matrix form is the following:

$$M_b = AM_r$$

where

$$M_b = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix}$$
$$M_r = \begin{bmatrix} \vec{r}_1 & \vec{r}_2 & \vec{r}_3 \end{bmatrix}$$

Since M_r is an orthogonal matrix, its inverse is equal to its transpose: $M_r^{-1} = M_r^T$. Thus, the solution for the attitude matrix is,

$$A = M_b M_r^T$$

A.1 Error Analysis of TRIAD Measurements

There are three independent maneuvers that are possible with the interaction tool: roll maneuvers, pitch maneuvers, and yaw maneuvers. The error analysis focuses on the pitch maneuver but similar results are obtained for a rotation of the interaction tool in any dimension. Using the TRIAD algorithm, the roll, pitch, and yaw angles are estimated, and these are seen side-by-side with the absolute orientation reference in Figure 11.

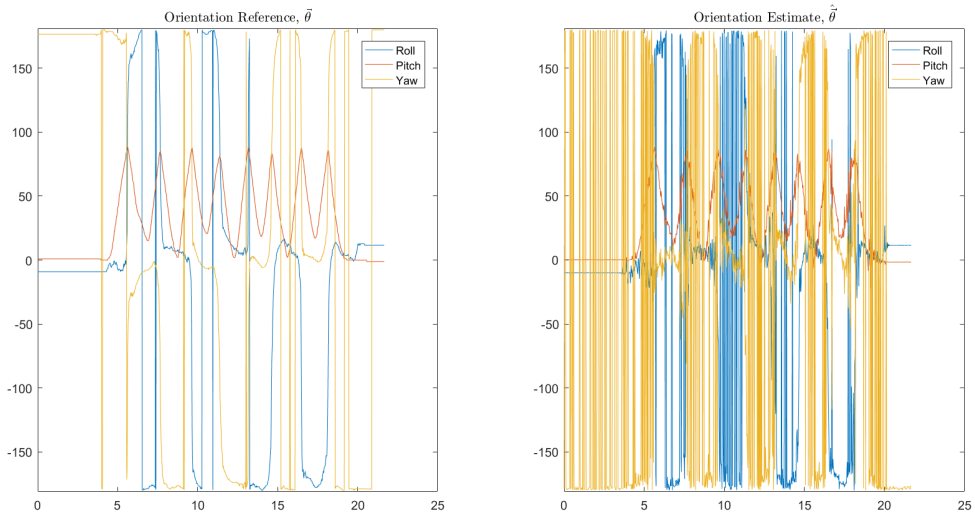

Figure 11: Pitch maneuver reference versus estimates

The error associated with the estimation of roll, pitch, and yaw is shown in 12.

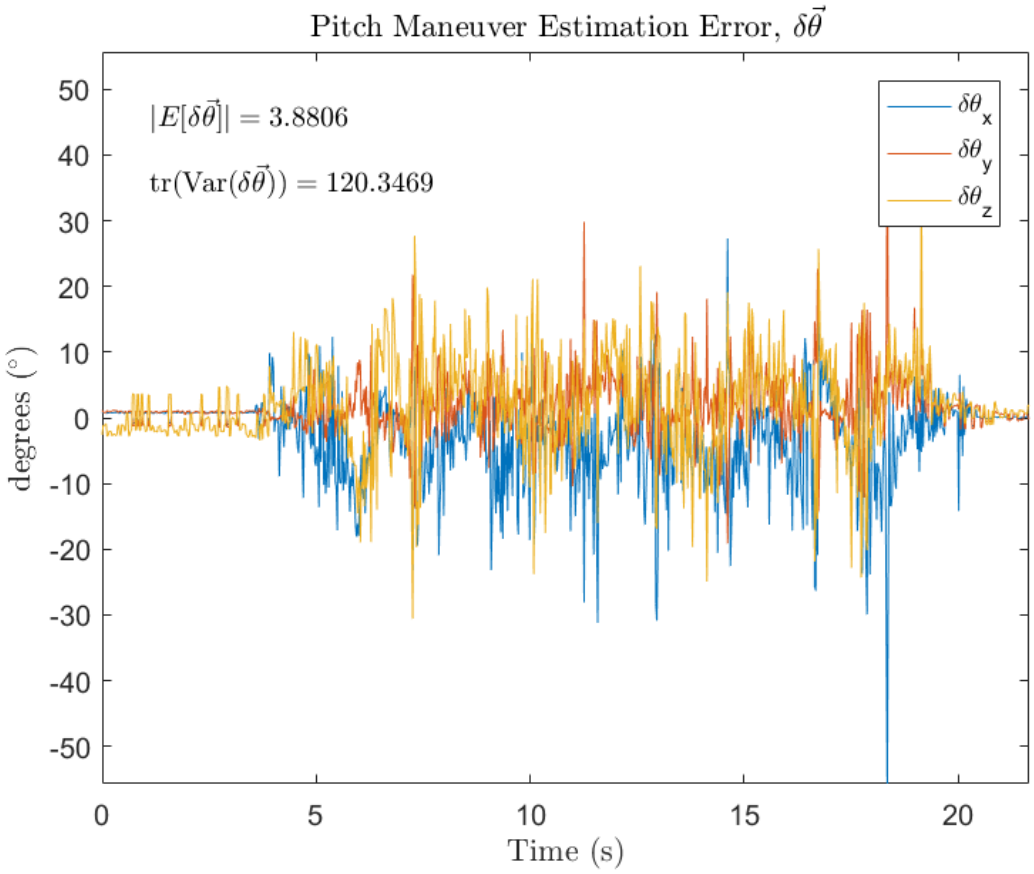

Figure 12: Pitch error for TRIAD estimation

The bias of the measurements is just 4.3° but the variance is more than 125°, corresponding to a root-mean-squared error of 11.2°. If a user notices an angular change of 5°, a deviation of more than 10° is likely to give them a headache. Similar results are obtained when analyzing maneuvers in the roll and yaw dimensions of the rotation. These errors and root-mean-squared errors are summarized in the table of Figure 13.

Bias of angular error, $\delta\vec{\theta}$ (degrees °)			
	Yaw Maneuver	Pitch Maneuver	Roll Maneuver
$E[\delta\vec{\theta}_x]$	-1.3	-2.8	-0.2
$E[\delta\vec{\theta}_y]$	-0.0	1.7	-1.0
$E[\delta\vec{\theta}_z]$	-0.3	2.0	1.3
$\left\|E\left[\delta\vec{\theta}\right]\right\|$	-1.4	3.9	1.6

Variances and RMSE of angular error, $\delta\vec{\theta}$ (degrees °)			
	Yaw Maneuver	Pitch Maneuver	Roll Maneuver
$\text{var}(\delta\vec{\theta}_x)$	$(9.3)^2$	$(7.0)^2$	$(7.9)^2$
$\text{var}(\delta\vec{\theta}_y)$	$(6.6)^2$	$(4.6)^2$	$(9.1)^2$
$\text{var}(\delta\vec{\theta}_z)$	$(14.2)^2$	$(7.1)^2$	$(7.4)^2$
$\sqrt{\text{trace}(\text{cov}(\delta\vec{\theta})}$	18.2	11.0	14.1

Figure 13: TRIAD orientation tracking errors

Notice that the yaw maneuver suffers from the greatest root-mean-squared error (RMSE). The yaw maneuver is particularly difficult to track due to the unreliability of the magnetometer sensor and this will be a theme throughout the remainder of the error analysis.

B. Improving the Orientation Estimation

The error of the TRIAD estimation routine is 2-3x times the targets presented in 3. Therefore, we would like to find a way to decrease the variance of the orientation estimation without increasing the latency of the system. To do this we need a real-time estimator. Any optimal real-time estimator for this system should incorporate all the information we know about the system. Note that the TRIAD method used only the accelerometer and magnetometer information from the inertial sensor and neglected the data from the gyroscope.

Is there a way that we can use the data from the gyroscope to improve the orientation estimation? In fact there is! But it requires that we related the gyroscope measurements to the dynamics of rotation. This is not a simple problem for three reasons. First, rotations are non-linear and therefore the dynamics are non-linear. Second, any 3-dimensional parameterization of the rotations for purposes of writing down a non-linear model contains a singularity and we must use a higher dimensional parameterization of the rotation [**?**]. Singularities in the three-dimensional representation of rotations are not problematic for small rotations. However, they are problematic for the 3D Interaction Tool because the user of the interaction tool is free to move the device into any orientation. This means that the tool can be held in any orientation, and the changes in the orientation are potentially extremely large. Thus, singularities are inevitable.

The final challenge with writing down the dynamics of the rotation is that while a higher dimensional representation of the rotation is required to prevent singularities – in our case we choose a four-dimensional representation – the angular error occurs in two dimensions. Therefore, linearizing the error in the rotation around any point involves reducing the dimensionality of the state representation of the rotation around that point. All of this and more is discussed in the following sections, but first there are some mathematical preliminaries about the particular representation of orientation chosen here: the quaternion.

C. Mathematical Representation of Orientation

The orientation of the interaction tool relative to the world coordinate frame can be represented in a number of ways. The two representations that will be important here are the attitude matrix and the attitude quaternion. First, the attitude of the orientation tool can be described by the attitude matrix

A. The attitude matrix typically maps a vector in the reference frame to a vector in the body frame [**?**]. This mapping is given by the relation,

$$\mathbf{b} = \mathbf{Ar}$$

Second, the attitude of the interaction tool can be represented as a quaternion. A rotation in three-dimensions is equivalent to an axis of rotation and an angle of rotation. If $\hat{n}$ is the axis of a rotation and θ is the angle of the rotation, then the quaternion $\overline{q}$ represents the attitude:

$$\overline{q} = \begin{bmatrix} \vec{q} \\ q_4 \end{bmatrix}$$

with $\vec{q} = \hat{n} \sin \frac{\theta}{2}$ and $q_4 = \cos \frac{\theta}{2}$. The quaternion must have length 1: $\overline{q}^T \overline{q} = 1$. One way of looking at this normalization constraint is that it reduces the degrees of freedom of the quaternion from four to three.
The attitude matrix and the quaternion representing attitude are related by the following equation, where A is a function of $\overline{q}$ [**?**]:

$$A(\overline{q}) = (q_4^2 - \|\vec{q}\|^2) I_{3x3} + 2\vec{q}\vec{q}^T - 2q_4[\vec{q}\times]$$

where $[\vec{q}\times]$ is the skew-symmetric cross product matrix,

$$[\vec{q}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}$$

The relationship between A and $\overline{q}$ can also be expressed as the product of two matrices [**?**]:

$$A(\overline{q}) = \Xi(\overline{q})^T \Psi(\overline{q})$$

with

$$\Xi(\overline{q}) = \begin{bmatrix} q_4 \mathbf{I_{3x3}} + [\vec{\mathbf{q}}\times] \\ -\vec{q}^T \end{bmatrix}$$

$$\Psi(\overline{q}) = \begin{bmatrix} q_4 \mathbf{I_{3x3}} - [\vec{\mathbf{q}}\times] \\ -\vec{q}^T \end{bmatrix}$$

D. Dynamic Measurement Model

The rate of change of the attitude can be written in the form of a general state space model:

$$\dot{\vec{x}} = A(\vec{x}, t)\vec{x} + B(\vec{x}, t)\vec{u}$$

The kinematic equation for the change in attitude as a function of the angular rate can be written as a rate of change of the attitude matrix or a rate of change in the quaternion [**?**]. The angular velocity vector is given by $\vec{\omega}$. The rate of change of the attitude matrix is,

$$\dot{A}(t) = [\vec{\omega}\times]A(t)$$

In this equation, $[\vec{\omega}\times]$ is an instance of the state transition matrix $A(\vec{x}, t)$ from the general schema for a state space model. The kinematic relation for the rate of change of the quaternion can be written in the following two ways:

$$\dot{\overline{q}} = \frac{1}{2}\Omega(\vec{\omega})\overline{q}$$

$$\dot{\bar{q}} = \frac{1}{2}\Xi(\bar{q})\vec{\omega}$$

where the first relation is a state space model with state transition matrix $A(\vec{\omega}) = \Omega(\omega)$, where the state transition matrix is a function of the angular rate vector $\vec{\omega}$:

$$\Omega(\vec{\omega}) = \begin{bmatrix} -[\vec{\omega}\times] & \vec{w} \\ -\vec{\omega}^T & 0 \end{bmatrix}$$

The second kinematic equation for the rate of change of the quaternion is an instance of the schema for the state space model where the angular rate $\vec{\omega}$ is treated as the input $\vec{u}$ to the system and the input matrix $B(\bar{q}) = \Xi(\bar{q})$ is a function of the current attitude $\bar{q}$.

There are two good reasons to use the quaternion representation of attitude: first, the rate of change of the quaternion is linearly related to the state of the quaternion; and second, successive rotations are computed by taking the product of quaternions [**?**]. The following identity holds of the composition of quaternions [**?**]:

$$A(\bar{q}_1)A(\bar{q}_2) = A(\bar{q}_1 \otimes \bar{q}_2)$$

with the operation $A(\bar{q})$ finding the attitude matrix A that corresponds with the quaternion $\bar{q}$. Following the notation in [**?**], the product of the quaternion is linear in each of the quaternions $\bar{q}_1$ and $\bar{q}_2$:

$$\bar{q}_1 \otimes \bar{q}_2 = \begin{bmatrix} \Psi(\bar{q}_1) & \bar{q}_1 \end{bmatrix} \bar{q}_2 = \begin{bmatrix} \Xi(\bar{q}_1) & \bar{q}_1 \end{bmatrix} \bar{q}_2$$

The inverse of the quaternion is found by negating the imaginary component of the quaternion:

$$\bar{q}^{-1} = \begin{bmatrix} -\vec{q} \\ q_4 \end{bmatrix}$$

D.1 Filter Design

How does the orientation of the interaction tool change from one time step to another? To discretize the dynamic model of rotations, we must know how the orientation of the tool will change between time steps. The change in orientation of the hand-held tool over time must be dependent on the dynamics of wrist movement. The peak velocity for human wrist movements is on the order of $10^2 \frac{^\circ}{s}$ for fast and accurate wrist rotations [**?**]. We can assume that users of the interaction tool will pursue accurate movements from one orientation to another and not move the tool wildly. Therefore, the change in the orientation of the tool between time steps for maximum angular velocity movements will be on the order of 2°, given a sample rate of 50 Hz. This rotation is so small as to be imperceptible to most users [**?**]. Given these assumptions, we can assume that the orientation of the tool is more or less fixed between samples and we can treat changes to the orientation as solely due to small deviations – essentially noise.

The small deviations in the angular velocity between samples can be approximated by integrating the angular velocity over the time step. The integrated angular velocity can be treated as a small deviation to the quaternion at time step k:

$$\delta\bar{\mathbf{q}}_t = \begin{bmatrix} \frac{1}{2}\vec{\omega}\Delta t \\ 1 \end{bmatrix}$$

Thus, given this small deviation we can predict the orientation at time step k given the orientation at step $k-1$:

$$\bar{\mathbf{q}}_k^- = \delta\bar{\mathbf{q}}_t \otimes \bar{\mathbf{q}}_{k-1} = \begin{bmatrix} \Xi(\bar{\mathbf{q}}_{k-1}) & \bar{\mathbf{q}}_{k-1} \end{bmatrix} \delta\bar{\mathbf{q}}_t = \bar{\mathbf{q}}_{k-1} + \Xi(\bar{\mathbf{q}}_k)(\frac{1}{2}\vec{\omega}\Delta t)$$

Now this is our discrete-time dynamic model! The state transition matrix $F = I_{3x3}$ and the input matrix G is parameterized by $\overline{\mathbf{q}}_k$ so that $G = \Xi(\overline{\mathbf{q}}_k)$.

In order to be able to integrate this orientation prediction with our measurements, we must also approximate the evolution of the covariance between time steps. Here the assumption that the orientation is fixed comes in handy for the covariance of the estimation does not change if the body is in a fixed configuration. However, the input from the gyroscope adds uncertainty. Let Q be the process noise in the gyroscope input, then the predicted covariance at time k given the covariance at step $k-1$ is

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

How do these predicted quantities relate to our measurements? Ultimately we want to be able to compute the innovation, the difference between the predicted measurements and the actual measurements. Once we multiply the innovation by the Kalman gain, we will know how to filter our prediction to produce a better estimate of the orientation. So how do we predict what will be measured?

We expect that the measured gravity vector and magnetic field vector will simply be the gravity reference vector rotated into the current body coordinates of the interaction tool. Similarly, the expected measurement for the magnetic field vector depends on the tool's current orientation. Thus, we estimate that what we will measure is

$$\hat{z}_k = \begin{bmatrix} A(\overline{\mathbf{q}}_k)\vec{g} \\ A(\overline{\mathbf{q}}_k)\vec{b} \end{bmatrix}$$

Now we can compute the innovation μ as the difference between what we predicted we would measure and what we actually measure:

$$\mu = z_k - \hat{z}_k$$

Multiplying the innovation by the gain of the filter, K, we compute the update residual

$$\delta\overline{\mathbf{q}}_t^+ = K\mu$$

This update residual is given as an error in the rotation of the interaction tool. Therefore instead of adding it to the predicted quaternion directly, we compose the two, as we do for rotations

$$\overline{\mathbf{q}}_k^+ = \delta\overline{\mathbf{q}}_t^+ \otimes \overline{\mathbf{q}}_k^-$$

This is the estimated orientation for time step k.

There is some subtlety to finding the filter gain K. The Kalman filter equations for the extended Kalman Filter give K as

$$K = P^- H_k (H_k P^- H_k^T + R)^{-1}$$

but this requires that we know both the measurement noise, R, and the measurement sensitivity matrix H_k. While the measurement noise is initialized with reasonable values based on the expected noise of the sensors, the measurement sensitivity matrix changes on each time step because the errors in the measurements depend on the current orientation. It turns out that if $\hat{z}_k = h(\overline{\mathbf{q}}_k)$ then the measurement sensitivity is

$$H_k = \frac{\partial h}{\partial \delta\overline{\mathbf{q}}_t} = \begin{bmatrix} 2A(\overline{\mathbf{q}}_k)\vec{g}^\times \\ 2A(\overline{\mathbf{q}}_k)\vec{b}^\times \end{bmatrix}$$

where $\vec{g}^\times$ is the cross product matrix. This measurement sensitivity matrix is the linearization of the rotation around the current orientation. This result will be derived in the later section on the overall performance analysis of the tracking system.

D.2 Filter Initialization

The initial state of the orientation is estimated using the TRIAD approach. The rotation matrix output of the TRIAD approach is converted to a quaternion to initialize the dynamic model. In order to initialize the covariance, an initial uncertainty in the orientation of 10 degrees is assumed.

The Kalman filter assumes process noise and measurement noise in the dynamics. The process noise for the dynamic model of rotations assumes that the body is fixed at each time step, so the noise in the dynamics must come from noise on the gyroscope input. The root-mean squared noise density of gyroscope on the Adafruit BNO055 given in the data sheet as $0.014\frac{\frac{\circ}{s}}{\sqrt{\text{Hz}}}$ [**?**]. To find the mean squared noise, this value is squared and multiplied by bandwidth of the signal, taken here to be the sample rate, 47 Hz. This yields a value of approximately $(0.1)^2\frac{\circ}{s}$. Converting to radians, the mean squared noise of the gyroscope is $2.8\text{e}{-6}\frac{rad}{s}^2$. Thus, the process noise Q is initialized as

$$Q = \begin{bmatrix} 2.8\text{e}{-6} & 0 & 0 \\ 0 & 2.8\text{e}{-6} & 0 \\ 0 & 0 & 2.8\text{e}{-6} \end{bmatrix}$$

Similarly, the measurement noise is initialized with noise values for the accelerometer and magnetometer listed in the datasheet [**?**]. Again, squaring the noise densities and multiplying by the bandwidths yields mean squared errors of $1.1\text{e}{-6}g^2$ and $17\mu T^2$ for the accelerometer and the magnetometer, respectively. The six dimensional measurement noise matrix, R, is initialized as

$$R = \begin{bmatrix} 1.1\text{e}{-6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.1\text{e}{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1\text{e}{-6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 17 & 0 & 0 \\ 0 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 17 \end{bmatrix}$$

D.3 Filter Tuning

The Kalman filter assumes that the measurement noise and process noise are Gaussian. However, this assumption rarely holds, so the noise parameters of the filter must be tuned to achieve better performance of the filter. The measurement noise parameters is particularly important. Increasing the measurement noise parameter, represented by R in the Kalman filter equations, decreases the Kalman gain.

Intuitively, a larger value for the measurement noise means that the variability of the measurements is high, so the filter is less responsive to measurements and more responsive to the model. Conversely, if the measurement noise parameter decreases, the Kalman gain increases. The graph in figure 14 shows how the Kalman filter responds to a step input in the roll for different values of R, the measurement gain.

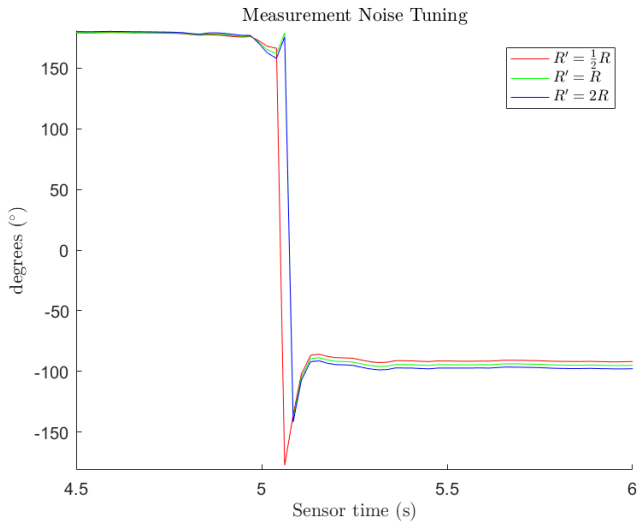

Figure 14: Tuning the measurement noise parameter of the Kalman filter

Lower values of R lead to a faster step response but more overshoot because the Kalman gain is high. Higher values of R have a slower step response but less overshoot because the Kalman gain is lower.

D.4 Filter Performance

The performance of the extended Kalman Filter model for the orientation is summarized in the charts in 15.

Bias of angular error, $\delta\vec{\theta}$ (degrees °)			
	Yaw Maneuver	Pitch Maneuver	Roll Maneuver
$E[\delta\vec{\theta}_x]$	1.0	0.2	-1.8
$E[\delta\vec{\theta}_y]$	2.7	0.9	-0.4
$E[\delta\vec{\theta}_z]$	-0.1	1.4	0.7
$\left\|E\left[\delta\vec{\theta}\right]\right\|$	-2.9	1.6	2.0

Variances and RMSE of angular error, $\delta\vec{\theta}$ (degrees °)			
	Yaw Maneuver	Pitch Maneuver	Roll Maneuver
$\text{var}(\delta\vec{\theta}_x)$	$(3.6)^2$	$(1.2)^2$	$(3.5)^2$
$\text{var}(\delta\vec{\theta}_y)$	$(2.9)^2$	$(1.9)^2$	$(1.9)^2$
$\text{var}(\delta\vec{\theta}_z)$	$(8.8)^2$	$(2.5)^2$	$(2.2)^2$
$\sqrt{\text{trace}(\text{cov}(\delta\vec{\theta})}$	9.9	3.3	4.5

Figure 15: Kalman filter orientation tracking errors

The comparison in the precision of these results to the TRIAD estimation algorithm is shown in 16.

Comparison of RMSE for TRIAD and EKF (degrees °)			
	Yaw Maneuver	Pitch Maneuver	Roll Maneuver
TRIAD	18.2	11.2	14.1
EKF	9.9	3.3	4.5

Figure 16: Kalman filter and TRIAD root-mean-squared error comparison

Note that the angular error target of 5° rmse is achieved in each maneuver direction except for yaw. As the chart shows, the Kalman Filter reduces the mean-squared-error of the orientation estimates by a factor of more than four versus the naive TRIAD estimation method.

IX. Position Tracking

The output of the position tracking filter is the position of the rear of the interaction tool $\vec{p}_r$. Tracking this position with low variance is important to produce an accurate estimation for the position of the front of the interaction tool.

A. Measurement Model

The following is a model of a single frame from the camera. In this frame, shown in image Figure 17 the red circular object represents the rear of the interaction tool. The spherical object of the rear of the interaction tool is called the tracking target. The width of the target along the x dimension of the image is s_u.

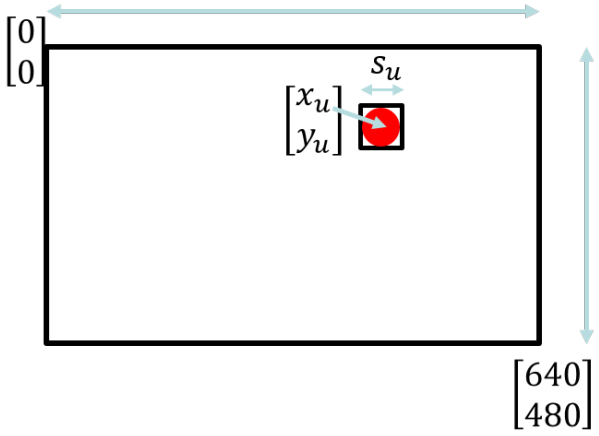

Figure 17: Computer vision tracking model

The midpoint of the target in the ideal undistorted image is $\vec{p}_u = \begin{bmatrix} x_u & y_u \end{bmatrix}^T$. The center of the undistorted image is c_x in the x direction and c_y in the y direction. The focal lengths in the x and y directions are f_x and f_y, respectively. These parameters are found through the camera calibration routine, explained in detail below. The measurements of the center of the target and its width are related to the hidden state $\vec{p}'_r = \begin{bmatrix} x_s & y_s & z_s \end{bmatrix}^T$ in the following way:

$$\vec{\mathbf{y}} = h(\vec{\mathbf{x}}) = \begin{bmatrix} x_u \\ y_u \\ s_u \end{bmatrix} = \begin{bmatrix} \frac{f_x x_s}{z_s} + c_x \\ \frac{f_y y_s}{z_s} + c_y \\ \frac{s_c}{z_s \sqrt{\frac{1}{f_x^2} + \frac{1}{f_y^2}}} \end{bmatrix}$$

These relationships are derived by using a pin-hole camera model as shown in [**?**].

A.1 Camera Calibration

The measurement model requires knowledge of the intrinsic camera parameters, f_x, f_y, c_x, c_y, and the radial and tangential distortion factors. There are two steps to translating a point from camera coordinates to image coordinates [**?**]. First, the pin-hole camera model shows how to find the ideal image coordinates of an object, (x_u, y_u) based on the position of the object in camera coordinates, $\vec{p}'_r = (x, y, z)$:

$$x_u = f_x \frac{x}{z}$$
$$y_u = f_y \frac{y}{z}$$

Second, the radial and tangential distortion relations show how the actual image coordinates, (x_u, y_u), relate to the image coordinates on the actual or distorted image plane, (x_d, y_d). The radial distortion correction equation is as follows:

$$x_u = x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_u = y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

and the tangential distortion equation is given by,

$$x_u = x_d + (2p_1 xy + p_2(r^2 + 2x^2))$$
$$y_u = y_d + (p_1(r^2 + 2y^2) + 2p_2 xy)$$

Thus, the five distortion parameters are summarized in a vector D:

$$D = \begin{bmatrix} k_1 & k_2 & p_1 p_2 & k_3 \end{bmatrix}$$

The calibration parameters are discovered by using an open source implementation provided by OpenCV. The camera collects a series of images that contain a checkerboard pattern. The calibration computes the transformation from the image coordinates of the checkerboard to its real world coordinates. The calibration images look like the ones in figure 1 below.

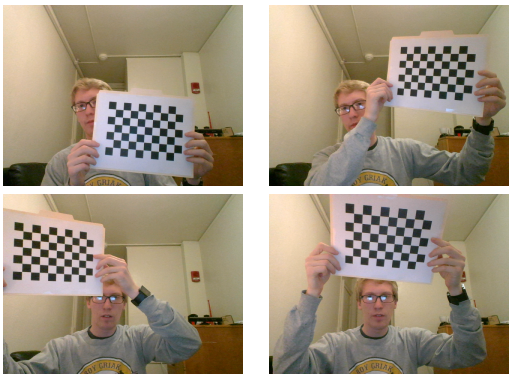

Figure 18: Camera calibration routine

A.2 Object Tracking

The signal processing layer contains an object tracker to find the rear of the interaction tool in each frame. Since a robust estimate of the object size in each image is required, its edges must be estimated as exactly as possible. One simple way to estimate the edges of the spherical tracking target is to segment the tracking target from the background by using color. The color of the object is used determine which pixels belong to it and which do not. Additionally, the color-based object tracking must happen in real time. To meet the demands of real-time, color-based object tracking, the CamShift algorithm was chosen [**?**].

A.3 Measurements of Tool Maneuvers

The possible maneuvers of the interaction tool are classified into two categories: maneuvers in the image plane and those in the depth dimension. For a user of the 3D Interaction Tool, side-to-side or up-and-down movements of the interaction tool map to motions in the image plane of the tracking camera, while movements of the tool towards and away from their body map to motions in the depth dimension. More precisely, the maneuvers in the image plane are movements of the interaction tool that occur perpendicular to the viewing angle of the camera.

The convention established for the purposes of testing is that the image plane is the yz-plane of the coordinate system and the depth dimension maps to the negative x-dimension. Maneuvers in the x-dimension encompass the movements of the tool towards and away from the tracking camera. Figure 19 compares the actual motion of the interaction tool to the estimates made by the tracking system.

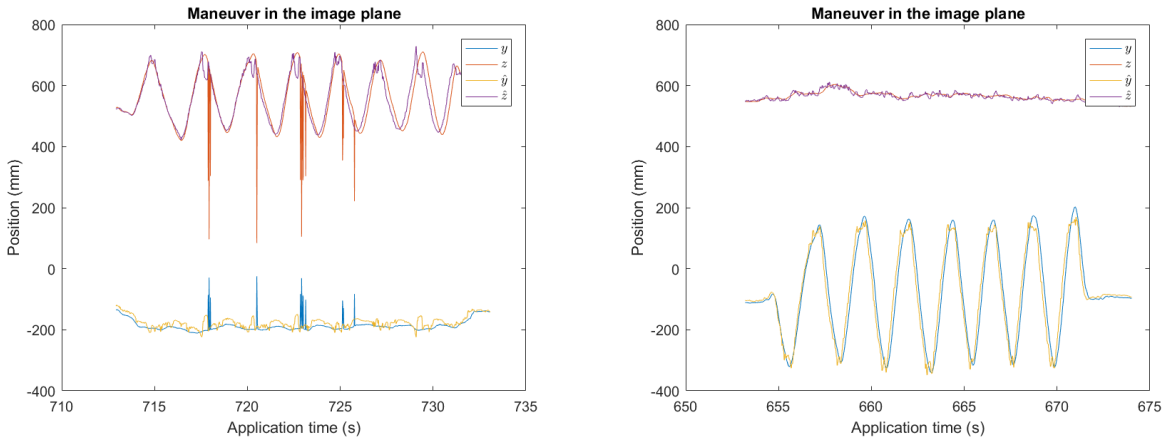

Figure 19: Vertical and horizontal motion in the image plane

In 19, the traces on the left correspond with vertical maneuvers of the tool and the traces on the right correspond to horizontal movements of the tool. In both graphs it is apparent that the tracking system is very accurate with low bias. The estimates of $\hat{y}$ and $\hat{z}$ map almost exactly to the true values.

The second set of maneuvers are those in the depth dimension. Figure 20 shows a trace of the actual depth x and the estimated depth $\hat{x}$.

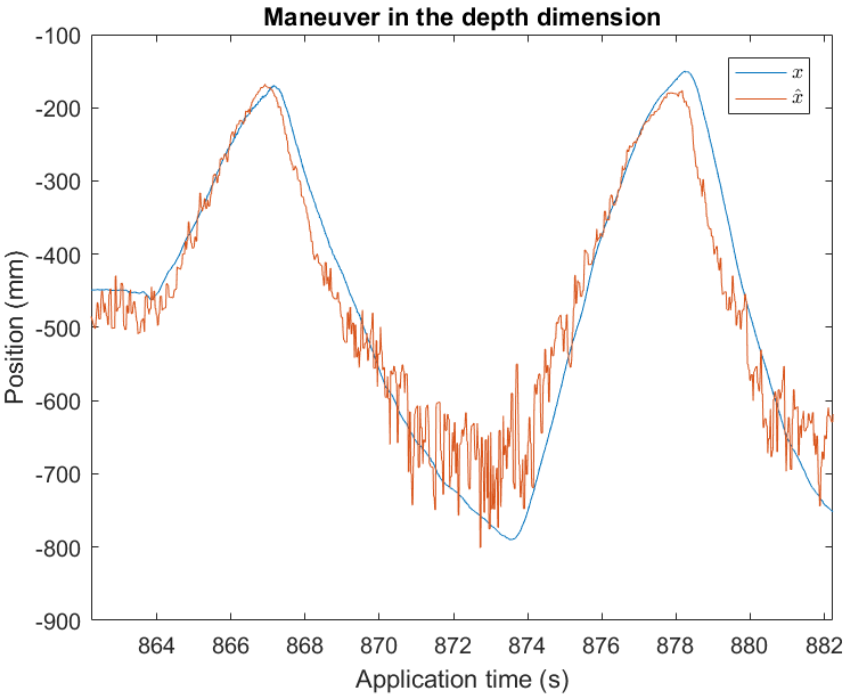

Figure 20: Motion of the interaction tool in the depth dimension

The natural conclusion from the traces in 20 is that the depth tracking variance depend on how far away the interaction tool is from the tracking camera.

B. Error Analysis

B.1 Motion in the Image Plane

The precision of the tracking system depends on the direction of the tool maneuvers. This is evident from analyzing the covariance of the error in the y and z directions for vertical and horizontal maneuvers. Figure 21 shows the error ellipse associated with motion in the image plane. On the left is the error ellispse associated with vertical motion and on the right is the error associated with horizontal motion.

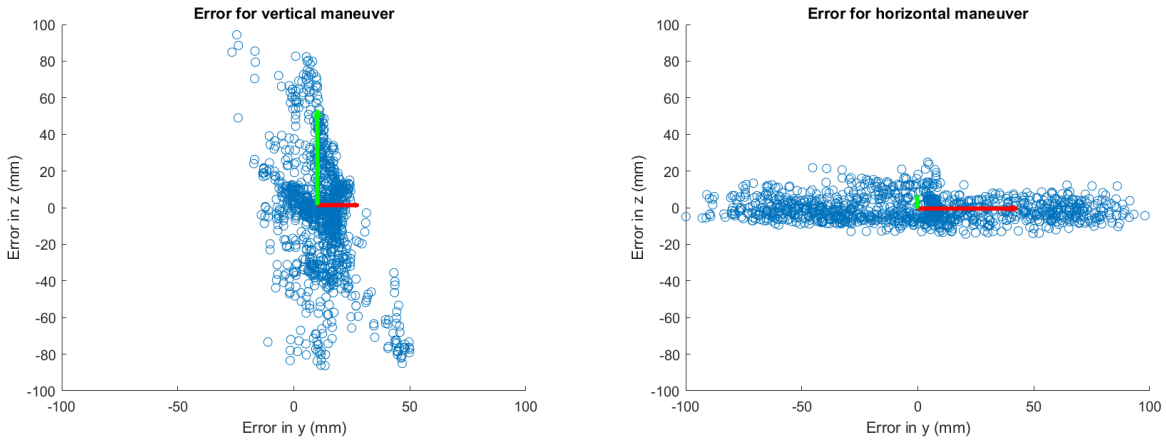

Figure 21: Tool tracking errors in the yz-plane

In 21 two arrows are placed with their origins at the mean of the yz-error distribution and their length

corresponding to the square root of the variance along the direction they point. It is clear from the magnitudes of these arrows that their is a breakdown of precision of the tracking system along the direction of motion for the 3D Interaction Tool.

What explains the higher variance along the direction of motion for the interaction tool? There are essentially two sources of errror for tracking the target relative to the camera. First, there is error that comes from the camera calibration routine in the form of error of intrinsic parameters of the camera – errors in fx, fy, for example. Second, there is error that comes from finding the centroid of the object in each image frame. Broadly, we could classify the calibration errors as systemic and the image processing errors as dynamic. Since the variance of the tool tracking increases in the direction of motion of the object, it is not likely that this error comes from a systemic problem with the calibration routine. Therefore, we can attribute the descrease in precision to error in the image segmentation routine.

Why does motion along a particular direction increase the error in the image segmentation? When the object is moving, its boundaries become blurred along the direction of motion in each frame. These blurred edges increase the uncertainty about the location of the object along the direction of motion. Therefore, when the object is moving along a particular direction, the uncertainty of its location in the direction of motion increases, but quickly drops again when the motion of the object halts.

B.2 Error Analysis of Motion in the Depth Dimension

The distribution of errors is larger for depth-tracking than tracking in the image plane. This result is displayed graphically in Figure 22.

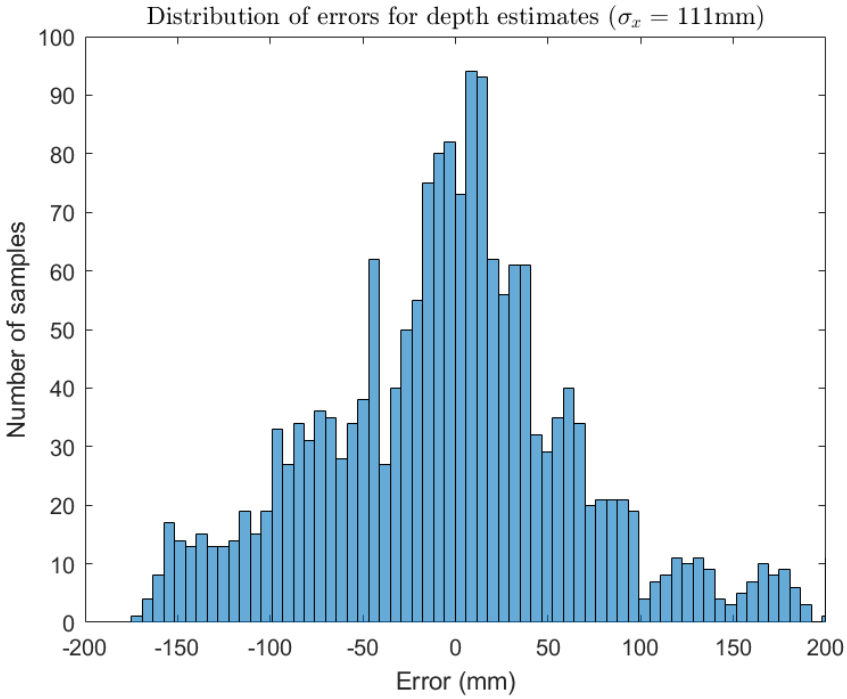

Figure 22: Distribution of depth-tracking errors

While the bias of the tracking errors tends to zero, the variance is much larger than for the depth-tracking in the image plane.

B.3 Summary of measurement errors

The naive tracking model above has errors for the three maneuvers as summarized by the tables below in Figure **??**.

Bias of position error, $\delta\vec{p}$ (mm)			
	Horizontal	Vertical	Depth
$E[\delta\vec{p}_x]$	-0.6	-8.9	-1.2
$E[\delta\vec{p}_y]$	-0.3	1.1	-4.4
$E[\delta\vec{p}_z]$	-0.5	7.0	2.5
$\lvert E[\delta\vec{p}]\rvert$	-0.8	11.5	5.2

Variances and RMSE of position error, $\delta\vec{p}$ (mm)			
	Horizontal	Vertical	Depth
$\text{var}(\delta\vec{p}_x)$	$(35)^2$	$(59)^2$	$(111)^2$
$\text{var}(\delta\vec{p}_y)$	$(43)^2$	$(18)^2$	$(12)^2$
$\text{var}(\delta\vec{p}_z)$	$(6.3)^2$	$(52)^2$	$(15)^2$
$\sqrt{\text{trace}(\text{cov}(\delta\vec{p}))}$	56	81	113

Figure 23: Bias and variance for position measurements

Note from the tables that the direction that is not the depth dimension or the dimension of motion has very small estimation error relative to the other two, which is exactly what we saw in the graphs presented above.

C. Dynamic Measurement Model

Since the user's movement of the interaction tool is voluntary, no assumptions can be made about its overall trajectory. However, human-motion obeys minimum-jerk constraints [**?**]. Since the jerk of the tool's motion is small, it can be modeled as a white-noise process. The dynamical model for the tool's motion along the x axis is estimated by the discrete time state-space model

$$\vec{x}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} (\Delta t)^2/2 \\ \Delta t \end{bmatrix} a_x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_k$$

where w_k is the jerk and a_x is the acceleration of the interaction tool in the x-dimension. Additionally, the motion of the tool is assumed to be coordinate uncoupled, since the user is free to move the tool in any direction. Therefore this same model can be extended to motion along the y and z axes as well.

Using this dynamic model for integrating the position estimates from the camera and the acceleration measurements from the accelerometer, slight improvements in the position estimation are obtained. These are summarized in the tables of Figure **??**:

Bias of position error, $\delta\vec{p}$ (mm)			
	Horizontal	Vertical	Depth
$E[\delta\vec{p}_x]$	-0.5	-9.5	-1.2
$E[\delta\vec{p}_y]$	-0.3	1.1	-4.4
$E[\delta\vec{p}_z]$	-0.5	7.0	2.5
$\lvert E[\delta\vec{p}]\rvert$	-0.8	11.9	5.2

Variances and RMSE of position error, $\delta\vec{p}$ (mm)			
	Horizontal	Vertical	Depth
$\text{var}(\delta\vec{p}_x)$	$(32)^2$	$(47)^2$	$(118)^2$
$\text{var}(\delta\vec{p}_y)$	$(43)^2$	$(17)^2$	$(12)^2$
$\text{var}(\delta\vec{p}_z)$	$(5.9)^2$	$(52)^2$	$(15)^2$
$\sqrt{\text{trace}(\text{cov}(\delta\vec{p})}$	54	72	120

Figure 24: Bias and variance for filtered position estimates

X. Performance Analysis

This section summarizes the performance of the tracking system implementation according to the performance design goals.

A. Robustness

No quantitative metrics were made for the robustness of the tracking system. In order to make the object tracking system workable, so that the CamShift algorithm could find the position of the tool rear in each frame, a Kalman filter was implemented to predict the next location of the bounding box of the interaction tool, following the method of [**?**].

B. Latency Calculation

The latency calculation goes as follows. We differentiate the VICON reference position signal to find a reference acceleration signal. Second, we compute the time shift Δs between the the acceleration readings from the accelerometer and those from the VICON tracking system. Third, we calculate the time shift Δe between the position estimates of the filter and the VICON reference clock. Finally, we find the latency, Δt by taking the difference between the times of the sensor measurements and the position estimates. The autocorrelation graphs are shown in Figure **??**

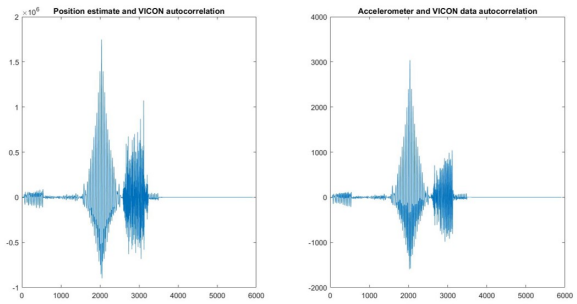

Figure 25: Autocorrelations for latency calculation

The result of the latency calculation shows an overall latency of 22ms, including latency in the system itself. This latency corresponds to the frame update rate of the camera.

C. Estimation Error

Given the tool model $\hat{\vec{p}}_f = \hat{\vec{p}}_r + \hat{\vec{p}}_l$, the overall error covariance $\mathbf{\Sigma}_{pf}$ is a function of the error due to the position estimation and the error to the orientation estimation. Since the estimation of $\vec{p}_r$ and $\vec{p}_l$ occur independently, it is reasonable to assume that the errors in the estimation of these vectors is not correlated. Thus, the covariances of the errors for these two vectors simply add to yield the covariance of the error for the tool front:

$$\mathbf{\Sigma}_{pf} = \mathbf{\Sigma}_{pr} + \mathbf{\Sigma}_l \tag{1}$$

D. Covariance Transformations

Here are three properties of covariance transformations, useful for analysing the performance of the tracking system as a whole:

(1) Given two independent random variables X, Y, and their sum $Z = X + Y$ how does the covariance of Z Σ_Z relate to the covariances Σ_X and Σ_Y of X and Y, respectively? In this case the covariances simply sum so that

$$\Sigma_Z = \Sigma_X + \Sigma_Y$$

(2) Given two random variables X and Y related by the linear operator T so that $Y = TX$ how does the covariance of Y relate to the covariance of X? If Σ_X is the covariance of X and Σ_Y is the covariance of Y, then

$$\Sigma_Y = T\Sigma_X T^T$$

(3) Given two random variables X and Y, and the non-linear relationship that $Y = f(X)$, how does the covariance of Y change with the covariance of X? As a first-order approximation, the randomness of Y can be approximated as a linear function of X [**?**]:

$$Y - \mu_Y = T(X - \mu_X) \tag{2}$$

Now, given the linear relationship, $Y - \mu_Y = T(X - \mu_X)$, the covariance of Y is related to the covariance of X by

$$\Sigma_Y = T\Sigma_X T^T$$

using property (2) of the covariance transformation under a linear operator.

E. Estimation Error for $\hat{\vec{p}}_l$: Finding position error due to orientation error

How do errors in the orientation estimation lead to errors the tool position estimation? In the tool model, the position of the front of the tool is estimated from three pieces of information: the position of the rear of the tool $\vec{p}_r$, the orientation of the tool $\mathbf{A}_t$, and the tool length in body coordinates $\vec{p}'_l$. This relationship is described by the equation

$$\vec{p}_f = \vec{p}_r + \vec{p}_l = \vec{p}_r + \mathbf{A}_t^T \vec{p}'_l$$

There is no error in the measurement of $\vec{p}'_l$; that is a well known parameter of the interaction tool. The error in $\vec{p}_l$ comes from the estimation of the rotation matrix from body coordinates to world coordinates.

The attitude matrix $\mathbf{A}_t$ is parameterized by the quaternion $\overline{\mathbf{q}}_t$ estimated from the EKF so that $\vec{p}_l = \mathbf{A}_t(\overline{\mathbf{q}}_t)^T \vec{p}'_l$. This expression is helpful because it describes the tool span vector $\vec{p}_l$ as a function of $\overline{\mathbf{q}}_t$. The relationship between $\vec{p}_l$ and $\overline{\mathbf{q}}_t$ must be linearlized in order to understand how errors in the estimation of $\overline{\mathbf{q}}_t$ translate to errors in $\vec{p}_l$.

Let the small deviations in the orientaiton be represented by the error quaternion $\delta\overline{\mathbf{q}}_t$ so that $\vec{p}_l = \mathbf{A}(\delta\overline{\mathbf{q}}_t)\mathbf{A}_t(\overline{\mathbf{q}}_t)^T \vec{p}'_l$, where $\mathbf{A}(\delta\overline{\mathbf{q}}_t)$ is the attitude matrix parameterized by the orientation error. Then, the goal here is to linearize the relationship between $\vec{p}_l$ and $\delta\overline{\mathbf{q}}_t$, and to accomplish we find the

matrix first-order derivative of $\vec{p}_l$ with respect to the three dimensional representation of the error quaternion $\delta\vec{q}$:

$$\frac{\partial \vec{p}_l}{\partial \delta\vec{q}} = \frac{\partial}{\partial \delta\vec{q}} \mathbf{A}(\delta\overline{\mathbf{q}}_t)\mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'_l$$

Now simplify $\mathbf{A}(\delta\overline{\mathbf{q}}_t)$ using the assumptions that $\delta q_4 \approx 1$ and $|\delta\vec{q}| \approx 0$, both of which will be true for small errors:

$$\mathbf{A}(\delta\overline{\mathbf{q}}_t) = (|\delta q_4|^2 - |\delta\vec{q}|)I_{3x3} + 2\delta\vec{q}\delta\vec{q}^T - 2\delta q_4 \left[\ \delta\vec{q}^{\times}\ \right] = I_{3x3} - 2\left[\ \delta\vec{q}^{\times}\ \right]$$

Pplugging in this expression for $\mathbf{A}(\delta\overline{\mathbf{q}}_t)$:

$$\frac{\partial \vec{p}_l}{\partial \delta\vec{q}} = \frac{\partial}{\partial \delta\vec{q}}(I_{3x3} - 2\left[\ \delta\vec{q}^{\times}\ \right])\mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'_l$$

Note that $\mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'_l$ is a constant vector – in fact it is simply the estimate for $\vec{p}_l$!

$$\frac{\partial \vec{p}_l}{\partial \delta\vec{q}} = \frac{\partial}{\partial \delta\vec{q}}(I_{3x3}\vec{p}_l - 2\left[\ \delta\vec{q}^{\times}\ \right]\vec{p}_l) = -2\frac{\partial}{\partial \delta\vec{q}}\left[\ \delta\vec{q}^{\times}\ \right]\vec{p}_l$$

The quantity $\left[\ \delta\vec{q}^{\times}\ \right]\vec{p}_l$ is simply $\delta\vec{q} \times \vec{p}_l = -\vec{p}_l \times \delta\vec{q}$. Therefore,

$$\frac{\partial \vec{p}_l}{\partial \delta\vec{q}} = -2\frac{\partial}{\partial \delta\vec{q}}(-\vec{p}_l \times \delta\vec{q}) = 2\left[\ \vec{p}_l{}^{\times}\ \right]\frac{\partial}{\partial \delta\vec{q}}\delta\vec{q} = 2\left[\ \vec{p}_l{}^{\times}\ \right]$$

Now we can write deviations in $\vec{p}_l$ as a linear function of the quaternion error or angular error ($\delta\vec{q} = \frac{1}{2}\delta\vec{\theta}$):

$$\delta\vec{p}_l = 2\left[\ \mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'^{\times}_l\ \right]\delta\vec{q} = \left[\ \mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'^{\times}_l\ \right]\delta\vec{\theta}$$

This is written in the form of Equation **??**. Thus, if $\mathbf{\Sigma}_{\theta t}$ is the covariance of the estimation of the tool's orientation, then the covariance $\mathbf{\Sigma}_l$ of $\vec{p}_l$ is

$$\mathbf{\Sigma}_l = \left[\ \mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'^{\times}_l\ \right]\mathbf{\Sigma}_{\theta t}\left[\ \mathbf{A}_t(\overline{\mathbf{q}}_t)^T\vec{p}'^{\times}_l\ \right]^T = \mathbf{A}_t(\overline{\mathbf{q}}_t)^T\left[\ \vec{p}'^{\times}_l\ \right]\mathbf{\Sigma}_{\theta t}\left[\ \vec{p}'^{\times}_l\ \right]^T\mathbf{A}_t(\overline{\mathbf{q}}_t) \tag{3}$$

F. Overall performance

The trace of the covariance matrices are invariant to linear transformations, thus we can derive the expression that

$$\mathrm{tr}(\mathbf{\Sigma}_{pf}) = \mathrm{tr}(\mathbf{\Sigma}_{pr}) + \mathrm{tr}(\mathbf{\Sigma}_l) = \mathrm{tr}(\mathbf{\Sigma}'_{pr}) + l^2\mathrm{tr}(\mathbf{\Sigma}_{\theta t})$$

Equivalently,

$$\mathrm{MSE}(\vec{p}_f) = \mathrm{MSE}(\vec{p}'_r) + l^2\mathrm{MSE}(\hat{\vec{\theta}})$$

Thus the overall error of the estimation of the position of the front of the tool in world coordinates depends on the position estimation error and the square of the length of the tool times the orientation estimation error. Using the error estimates derived in the sections on position and orientation tracking, this result is summarized in Figure **??**

Overall Tracking Performance: Error in the estimation of $\vec{p}_f$ (mm)	
$RMSE(\hat{\vec{p}}'_r - \vec{p}_r)$	≈ 100
$\sqrt{l^2 MSE(\hat{\vec{\theta}}_t - \vec{\theta}_t)}$	$\sqrt{(120)^2(0.08)^2} \approx 10$
$RMSE(\hat{\vec{p}}_f - \vec{p}_f)$	≈ 100.5

Figure 26: Overall tracking error given position and orientation tracking errors

This chart illuminates the following result: the position tracking errors dominate, with root-mean-squared error of only 1cm coming from the errors in the orientation tracking.

XI. Conclusion

Revisiting the design objectives in 3, the table in Figure ?? captures the progress made on the performance of the tracking system. In green are those specifications that were achieved, and in orange are the specifications towards with progress was made.

Performance		
Objective	Target	Achieved
RMSE (Position)	<5cm	$\approx$ 10.1cm
RMSE (Orientation)	<5°	$\approx$ 4.5 °
Latency	<60ms	22ms
Robustness	Robust to tool velocity, tool occlusions	Improvements in image segmentation at high tool velocities

Figure 27: Performance design goals, revisited

The specifications for the latency and orientation error were achieved so that the lag of the system is not noticable and the estimation errors are imperceptible to the user. The root-mean-squared error for the position estimates was about twice that which was specified. These position errors were largely due to errors in the depth dimension of the tracking camera and errors due to tool velocity.

Usability		
Objective	Target	Achieved
Portability	The interaction tool tracking system should be easily translatable between uses	Portable interaction tool, usable with desktop computer
Natural form factor	Easy to learn and provides high degree of control	Functional breadboard form factor usable for tool maneuvers
Large tracking volume	$\approx 1m^3$	Viewpoint of camera, limited tracking volume at close distances
Low cost	<$20	Total cost $160, but easily decreased for future prototypes

Figure 28: Performance design goals, revisited

Figure ?? revisits the design objectives in 2. Portability and form factor requirements were met. The cost specification is easy to meet in future prototypes. The sensors used for the intial prototype were particularly expensive because they came with on-board filtering that was useful for prototyping. However, now that the tracking algorithms are implemented, lower-part costs could be purchased for any future prototypes.

The tracking volume objective can be met by increasing the field of view of the camera. Some experiments were done in this configuration, but they introduce a large bias into the position estimates due to errors in the calibration of the distortion parameters associated with widening the field of view of the lens.

XII. Future Work

References

[1] J. Verhage, "Goldman Sachs Has Four Charts Showing the Huge Potential in Virtual and Augmented Reality." http://www.bloomberg.com/news/articles/2016-01-13/goldman-sachs-has-four-charts-showing-the-huge-potential-in-virtual-and-augmented-reality, 2016. Accessed: 2016-03-19.

[2] A. Robertson, "Valve's virtual reality headset is great, but its controllers are the real story." http://www.theverge.com/2015/3/4/8150653/valve-steam-controller-vive-vr-gdc-2015, 2015. Accessed: 2016-03-01.

[3] E. Frederiksen, "Oculus Touch will be your virtual hands." http://www.technobuffalo.com/2015/06/11/oculus-touch-will-be-your-virtual-hands/, 2015. Accessed: 2016-03-01.

[4] J. Constine, "Oculus Previews Oculus Touch Handheld Motion-Tracking Haptic Controllers." http://techcrunch.com/2015/06/11/oculus-touch/, 2015. Accessed: 2016-03-01.

[5] B. Russell, "Oculus Touch delayed into second half of 2016." http://www.technobuffalo.com/2015/12/31/oculus-touch-delayed-2016/, 2015. Accessed: 2016-03-01.

[6] C. Hand, "A survey of 3d interaction techniques," in *Computer graphics forum*, vol. 16, pp. 269–281, Wiley Online Library, 1997.

[7] M. F. Deering, "Holosketch: A virtual reality sketching/animation tool," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 2, no. 3, pp. 220–238, 1995.

[8] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr, "Cavepainting: a fully immersive 3d artistic medium and interactive experience," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 85–93, ACM, 2001.

[9] R. Wang, S. Paris, and J. Popović, "6d hands: markerless hand-tracking for computer aided design," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 549–558, ACM, 2011.

[10] J. P. Rolland, L. Davis, and Y. Baillot, "A survey of tracking technology for virtual environments," *Fundamentals of wearable computers and augmented reality*, vol. 1, pp. 67–112, 2001.

[11] M. Ribo, A. Pinz, and A. L. Fuhrmann, "A new optical tracking system for virtual and augmented reality applications," in *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, vol. 3, pp. 1932–1936, IEEE, 2001.

[12] G. Welch and E. Foxlin, "Motion tracking survey," *IEEE Computer graphics and Applications*, pp. 24–38, 2002.

[13] "Virtual reality just got real." http://www.samsung.com/us/explore/gear-vr/, 2016. Accessed: 2016-03-25.

[14] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell, "Usability analysis of 3d rotation techniques," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 1–10, ACM, 1997.

[15] E. M. Kolasinski, "Simulator sickness in virtual environments.," tech. rep., DTIC Document, 1995.

[16] "Oculus Best Practices: Tracking." https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_app_tracking/, 2016. Accessed: 2015-12-07.

[17] M. D. Shuster, "Deterministic three-axis attitude determination," *Journal of Astronautical Sciences*, vol. 52, no. 3, pp. 405–419, 2004.

[18] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM review*, vol. 6, no. 4, pp. 422–430, 1964.

[19] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods," *Journal of guidance, control, and dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[20] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[21] D. S. Hoffman and P. L. Strick, "Step-tracking movements of the wrist in humans. i. kinematic analysis," *The Journal of neuroscience*, vol. 6, no. 11, pp. 3309–3318, 1986.

[22] B. Sensortec, "BNO055: Intelligent 9-axis absolute orientation sensor." `https://www.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf`, 2014. Accessed: 2016-01-28.

[23] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 323–344, 1987.

[24] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*, pp. 214–219, IEEE, 1998.

[25] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[26] "An object tracking project using camshift and Kalman Filter based on OpenCV." `http://www.samsung.com/us/explore/gear-vr/`, 2016. Accessed: 2016-03-15.

[27] R. M. Haralick, "Propagating covariance in computer vision," *International journal of pattern recognition and artificial intelligence*, vol. 10, no. 05, pp. 561–572, 1996.