Insight

A virtual reality project

By

Brian Spence
Svyatoslav Markeyev

Senior Project

Computer Engineering Department
California Polytechnic State University
San Luis Obispo
June 2013
# 1. Table of Contents

List of Figures .................................................................................................................... 6
List of Tables ......................................................................................................................... 7
Acknowledgements............................................................................................................. 7
Abstract ............................................................................................................................... 7

1. Introduction ........................................................................................................................................ 8
   1.1 Motivation ................................................................................................................................. 8
   1.2 Overview ................................................................................................................................... 8

2. Background ..................................................................................................................................... 8
   2.1 Oculus Rift .............................................................................................................................. 8
   2.2 UDK ......................................................................................................................................... 9
   2.3 Real Time Kinematics Library ............................................................................................... 9
       2.3.1 Real Time Kinematics ..................................................................................................... 9
       2.3.2 RTKLIB Software ............................................................................................................ 9

3. Requirements ............................................................................................................................ 10
   3.1 Desired Features and Justifications ..................................................................................... 10
   3.2 Engineering Specifications ................................................................................................. 10

4. Design ......................................................................................................................................... 11
   4.1 Overview .................................................................................................................................. 11
   4.2 UDK ........................................................................................................................................ 13
       4.2.1 Overview .......................................................................................................................... 13
       4.2.2 UDK TCP server ............................................................................................................. 13
       4.2.3 Commands ...................................................................................................................... 14
   4.3 IMU .......................................................................................................................................... 14
       4.3.1 Overview .......................................................................................................................... 14
       4.3.2 Hardware .......................................................................................................................... 15
       4.3.3 Software ........................................................................................................................... 15
   4.4 GPS .......................................................................................................................................... 15
       4.4.1 Overview .......................................................................................................................... 15
       4.4.2 RTK Library ...................................................................................................................... 16
4.4.3 Antenna and Receiver .................................................................................. 16
4.4.4 Base station and Raspberry Pi ................................................................. 16
4.5 Middleware .................................................................................................... 16
  4.5.1 Overview ................................................................................................. 16
  4.5.2 Filtering .................................................................................................... 16
4.6 Rift .................................................................................................................. 17
5.  Construction and Problems Encountered ...................................................... 17
  5.1 Overview ...................................................................................................... 17
  5.2 GPS .............................................................................................................. 17
  5.3 IMU ............................................................................................................... 18
  5.4 UDK ............................................................................................................. 18
6.  Development and Regular Use ...................................................................... 19
  6.1 Powering the Oculus Rift ............................................................................ 19
    6.1.1 Power Requirements ............................................................................ 19
    6.1.2 Hardware Modifications ....................................................................... 19
  6.2 Project Source Code and Dependencies ..................................................... 20
  6.3 Setting up the Environment ....................................................................... 21
    6.3.1 Windows .............................................................................................. 21
    6.3.2 Java Sun ............................................................................................... 21
    6.3.3 Oculus Rift ........................................................................................... 21
    6.3.4 YEI Threespace Sensor Drivers ............................................................ 21
  6.4 Setting up the Production Environment ..................................................... 21
    6.4.1 Install Packaged UDK .......................................................................... 21
    6.4.2 Install Insight Project ........................................................................... 21
  6.5 Setting up the Development Environment .................................................. 22
    6.5.1 Integrated development environments ............................................... 22
    6.5.2 Middleware .......................................................................................... 22
    6.5.3 Unreal Development Kit ........................................................................ 22
    6.5.4 IMU Data Com ...................................................................................... 22
    6.5.5 UDK Assistant ...................................................................................... 23
6.6 Building RTKLIB Components .................................................................. 23
  6.6.1 Windows ............................................................................................... 23
# Table of Contents

## 7. User Guide

- **7.6 How to Launch the Project** .......................................................... 31

## 8. Developer Guide

- **8.3 API** .................................................................................................... 33
  - **8.3.1 API calls using ApiObject** ............................................................. 34
- **8.2 Middleware** ......................................................................................... 31
  - **8.2.1 Components** ................................................................................. 31
  - **8.2.2 Filters** ............................................................................................ 32
  - **8.2.3 IP Receive-Transmit Interface** ....................................................... 32
  - **8.2.4 Webserver** .................................................................................... 33
  - **8.2.5 SimpleProcessor** ......................................................................... 33
  - **8.2.6 Logging** ......................................................................................... 33
- **8.1 This Guide** ......................................................................................... 31

## 6. User Guide

- **6.6.2 Getting RTKLIB** ............................................................................ 23
- **6.6.3 Compiling** ..................................................................................... 23
- **6.6.4 Use** ................................................................................................. 24

## 5. Navigation

- **5.3 Log Terminal and Controls Screen** .................................................. 30
- **5.4 Live Graphs Screen** ......................................................................... 30
- **5.5 RTKRCV Terminal Screen** ............................................................... 30

## 4. RTKRCV Configuration

- **4.2 RTKRCV Configuration** .................................................................. 25

## 3. Middleware Configuration

- **3.2.1 Components** ................................................................................. 27
- **3.2.2 IP Receive-Transmit Interface** ....................................................... 27

## 2. Setting up RTK on Backpack

- **2.7 Setting up a RTK Base Station** ....................................................... 24
  - **2.7.1 Base Station Location** ................................................................. 24
  - **2.7.2 Finding Device Number** ............................................................. 27
  - **2.7.3 RTKRCV Configuration** ............................................................ 27
  - **2.7.4 Poor GPS Signal** ......................................................................... 26

## 1. Setting up a RTK Base Station

- **1.1 Setting up a RTK Base Station** ....................................................... 24
  - **1.1.1 Base Station Location** ................................................................. 24
  - **1.1.2 Running str2str** ........................................................................ 24
  - **1.1.3 Attaining a GPS Fix** ................................................................... 25
  - **1.1.4 Poor GPS Signal** ........................................................................ 26
10.1 Conclusion........................................................................................................................................... 45
10.2 Future Work........................................................................................................................................ 46

11. Works Cited............................................................................................................................................. 47

12. Appendices............................................................................................................................................. 48
   A. Parts List and Costs ............................................................................................................................ 48
   B. Web GUI ............................................................................................................................................ 50
   C. NMEA-183 Message Format .............................................................................................................. 54
   D. Filter Results .................................................................................................................................... 55

List of Figures
Figure 1 - Design Overview ....................................................................................................................... 11
Figure 2 - Backpack Hardware Layout....................................................................................................... 12
Figure 3 - Base Station Hardware Layout .................................................................................................. 13
Figure 4 - Screenshot of the Unreal Engine ................................................................................................ 13
Figure 5 - UDK Map editor ....................................................................................................................... 14
Figure 6 - Base Station ............................................................................................................................... 17
Figure 7 – Oculus Rift Circuit Board .......................................................................................................... 20
Figure 8 – RTKNAVI Configuration ......................................................................................................... 26
Figure 9 – RTKNAVI PPP Fix .................................................................................................................... 26
Figure 10 – Satellite Signal Strength ........................................................................................................ 27
Figure 11 – UDK coordinates Head Rotation Test 1 .................................................................................. 41
Figure 12 – UDK coordinates Rectangle Test 1 ......................................................................................... 41
Figure 13 – UDK coordinates Rectangle Test 2 ......................................................................................... 42
Figure 14 – UDK coordinates Rectangle Test 3 ......................................................................................... 42
Figure 15 – UDK coordinates Circle Test 1 ............................................................................................... 43
Figure 16 – UDK coordinates Circle Test 2 .............................................................................................. 44
Figure 17 – UDK coordinates Circle Test 3 .............................................................................................. 44
Figure 18 – UDK Velocity Circle Test 1 ..................................................................................................... 45
Figure 19 – Web GUI Log Terminal and Controls Screen ......................................................................... 50
Figure 20 – Web GUI Log Live Graphs Screen ......................................................................................... 51
Figure 21 – Web GUI RTKRCV Terminal Screen ...................................................................................... 52
Figure 22 – Finding Serial Port Number ..................................................................................................... 53
Figure 23 – Low Pass Factor 50 .................................................................................................................. 55
Figure 24 – Low Pass Factor 100 ................................................................................................................ 55
Figure 25 – Low Pass Factor 250 ................................................................................................................ 56
Figure 26 – Low Pass Factor 500 ................................................................................................................ 56
Figure 27 – Low Pass Factor 1000 ............................................................................................................. 57
**List of Tables**

Table 1 - Features .................................................................................................................. 10
Table 2 - Specifications ........................................................................................................... 10
Table 3 – Project Archives ..................................................................................................... 20
Table 4 – RTKRCV Parameters ............................................................................................. 28
Table 5 – Web GUI Configuration Parameters ...................................................................... 28
Table 6 – GPS Configuration Parameters ............................................................................. 28
Table 7 – IMU Configuration Parameters .............................................................................. 28
Table 8 – UDK Configuration Parameters ............................................................................ 29
Table 9 – Filter Configuration Parameters ........................................................................... 29
Table 10 – Log Terminal and Controls Screen ....................................................................... 30
Table 11 – Log Terminal and Controls Screen Buttons ........................................................... 30
Table 12 – API Status Calls .................................................................................................. 34
Table 13 – API Command Calls ............................................................................................ 34
Table 14 - Source Files .......................................................................................................... 35
Table 15 - Web GUI Libraries ............................................................................................... 36
Table 16 - Source Files .......................................................................................................... 36
Table 17 – Orientation Parameters ........................................................................................ 36
Table 18 – Direction Parameters ........................................................................................... 37
Table 19 - Optional Command Line Arguments .................................................................... 38
Table 20 - Heading Data Packet Parts ................................................................................... 38
Table 21 - Available Commands ............................................................................................ 38
Table 22 - Source Files .......................................................................................................... 39
Table 23 – Test Settings ......................................................................................................... 39
Table 24 - Parts and Costs ..................................................................................................... 48

**Acknowledgements**

We would like to acknowledge the following people for their help with this project. Without them this project could not have been completed.

Jon Oliver – Senior Project Advisor

Ian Stewart – Project concept originator

Michele Bavaro – Employee of onetalent-gnss.com

**Abstract**

This is a project about extending the capabilities of current consumer virtual reality systems. The goal of the project was to extend the capabilities of the Oculus Rift Headset in order to create a system which would allow a user to explore a virtual environment in an immersive way just as they would explore
reality. The system created would allow a user to explore a virtual environment by walking around in a familiar immersive manner. This was accomplished using the Unreal Development Kit, a state of the art GPS system, and industry proven IMU. The user’s movements are tracked as he walks around a large open space and these movements are then processed and fed into the virtual world to move the game avatar. This project, called Insight, will allow a user to explore virtual worlds in an intuitive way that used to be the stuff of science fiction.

1. Introduction

1.1 Motivation

This project started with the desire to develop for the Oculus Rift virtual reality headset after experiencing an impressive demo at QuakeCon 2012. The Rift was shown to be responsive, immersive, and incredibly fun. With the Rift, virtual reality is starting to enter the consumer market. The Rift will be released with support for many current games and developers are scrambling to create more game with support of the headset. This project intends to expand the capabilities of the Oculus Rift (Oculus VR, 2013).

1.2 Overview

This project intends to expand upon the capabilities of the Oculus Rift. The user is tracked as he walks around a large flat open space. His movements control the movement of the avatar in the game. If the user walks forward, the avatar walks forward the same amount. If the user faces his left, the avatar faces its left. The idea is complete control of the avatar’s movement so as to immerse the user in the virtual environment so that the user feels he is really in the virtual world. A custom tracking system is used to track the user’s speed and direction of movement. An IMU is used to calibrate the user’s heading, augmenting the Rift’s sensor suite. These measurements are then filtered and combined to control the virtual game avatar. The complete system is set up to integrate with the Rift headset to further immerse the user in the virtual environment by tracking the users head as well.

2. Background

2.1 Oculus Rift

The Oculus Rift (Oculus VR, 2013) is a new virtual reality headset with impressive capabilities. It is the first product from Oculus, a company founded in 2012 by VR enthusiast Palmer Lucky. The company’s mission is to create immersive virtual reality technology that’s wearable and affordable. After a successful Kickstarter Campaign which raised over 2.4 million dollars, Oculus began development on the Rift development kit. The Rift will have a high field of view, low latency head tracking, and a high resolution display. The company will also release an online developer SDK with support of many popular game engines including Unreal Engine 3.
2.2 UDK
The Unreal Development Kit (UDK, 2013) is a free edition of the Unreal Engine 3 software development kit. Unreal Engine 3 is a popular video game engine used to power many video games. The UDK has been used in video game development, architectural visualization, mobile game development, 3D rendering, digital films, and much more. It is used by video game developers, researchers, television studios, directors, artists, and students.

2.3 Real Time Kinematics Library
This project uses the RTKLIB open source program package (RTKLIB:Overview, 2013). The RTKLIB program package uses the real time kinematic satellite navigation technique to obtain a more accurate fix than a standard GPS navigation system. Real time kinematics works by using two GPS receivers. One receiver is on the mobile unit and the other at a fixed base station. Using two receivers allows for many of the errors seen in standard GPS to be removed. This technique is further described in section 2.3.1. The RTKLIB program package provides programs and libraries for easily incorporating real time kinematics navigation into a project. RTKLIB is described in more detail in section 2.3.2.

2.3.1 Real Time Kinematics
Real Time Kinematics (RTK) is technique for calculating position using differential GPS that results in very high accuracy, in the range of a few centimeters (RTK Fundamentals, 2013). The infrastructure needed for RTK consists of a base station, one or more mobile units, and a communication channel that allows for the base station to transmit information to the mobile units in real time. The base station covers a service area of 10 to 20 kilometers, must have a clear view of the sky, and its location must be well known.

Many of the errors in GPS navigation are constant when the receiver is in a clear sky location. These errors can be cancelled out with two GPS receivers using differential processing. The differential processing technique removes errors caused by factors such as satellite clock bias, the satellite orbital error, the ionospheric delay and the tropospheric delay. RTK goes further than standard differential GPS by using another technique called code-based positioning which uses pseudorandom codes transmitted by the GPS satellites to determine position (Novatel, 2013). The technique is very complicated, but at the basic level, the mobile unit determines the number of radio carrier wave cycles between it and the satellite and then multiplies that number by the carrier wave length to get its distance to the satellite. Using the differential GPS and pseudorange techniques, RTK can provide highly accurate positioning information.

2.3.2 RTKLIB Software
The RTKLIB open source programming package provides many helpful tools and libraries for satellite navigation (RTKLIB:Overview, 2013). The libraries are written in ANSI C and will compile and run in both Windows and Unix. The provided tools only have Windows support. RTKLIB has many features and support tools, but only a few were needed for this project. The main feature of RTKLIB is support for real time kinematics navigation. The tools provided allow for the use of RTK without needing to understand the theory behind it. The RTKLIB tools support TCP communication which allows for easy integration. The GUI tools allow for easy setup and quick diagnostics and testing. Furthermore, the RTKLIB comes
with support for a variety of GPS receivers found on the market. With the RTKLIB programming package, it is easy to get accurate GPS position using RTK.

3. Requirements

3.1 Desired Features and Justifications
The idea for project Insight is for a person to be able to experience virtual reality in an un-confined space. This system would allow a user to use the system over a large outdoor area. The features show in Table 1 - Features were chose to try and maximize the experience for a user in this context.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Wearable</td>
<td>The system must be small and light enough to be worn by the user. This is to allow the user to walk around a large field wearing the system.</td>
</tr>
<tr>
<td>Integrates with the Oculus Rift Headset</td>
<td>The Oculus Rift is the headset we have available, so the system must be able to integrate and work with the Rift.</td>
</tr>
<tr>
<td>Immersive</td>
<td>The system must be immersive as possible so as to allow a comfortable user experience without any motion sickness or breaking of the illusion of being in the virtual world</td>
</tr>
<tr>
<td>Very large play area</td>
<td>as this will allow a user to explore a large map and differentiates it from current and soon-to-be VR systems</td>
</tr>
</tbody>
</table>

3.2 Engineering Specifications
The specifications shown in the Table 2 - Specifications were written as guide when researching engineering solutions for the project. The specifications are the minimum requirements needed to be met by the project in order to obtain of all the desired features shown in Table 1 - Features.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Justifying Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>All hardware is mountable on or in a backpack</td>
<td>User Wearable</td>
</tr>
<tr>
<td>The total weight of the hardware should be as light as possible, not to exceed 50lbs.</td>
<td>User Wearable</td>
</tr>
<tr>
<td>This system must be capable of running a compatible OS for the Rift Headset</td>
<td>Rift</td>
</tr>
<tr>
<td>This system must have the resources to run the Rift Headset.</td>
<td>Rift</td>
</tr>
<tr>
<td>The system must include an HDMI/ or DVI port.</td>
<td>Rift</td>
</tr>
<tr>
<td>The system must include a USB port.</td>
<td>Rift</td>
</tr>
<tr>
<td>The system must have a low response time to user input, not to exceed 100ms.</td>
<td>Immersive</td>
</tr>
<tr>
<td>Accurate location tracking to within 10cm.</td>
<td>Immersive</td>
</tr>
<tr>
<td>Accurate orientation tracking to with 5 degrees.</td>
<td>Immersive</td>
</tr>
<tr>
<td>User is tracked with 3 degrees of freedom, allowing a full range of motion</td>
<td>Immersive</td>
</tr>
</tbody>
</table>
4. Design

4.1 Overview
The system has 5 principle subsystems: GPS, IMU, UDK, Middleware, and Rift each of which will be described in the following sub-sections. These subsystems work together to create a system capable of allowing the user to walk around a virtual world as they walk around the real world. These systems are designed to be run on a laptop that the user will wear on his back. The systems were implemented using software libraries where possible, so as to limit the time and engineering need to incomplete each module. In order to be modular these systems communicate through the TCP protocol. This allows any subsystem implementation to be easily swapped with another without the needing to create a custom communication protocol.

The GPS and IMU keep track of the user’s movements. This is communicated to the middleware, which filters the data and converts it into a form the UDK engine can understand. The middleware then sends the commands to the UDK to control the avatar.
Figure 2 - Backpack Hardware Layout
4.2 UDK

4.2.1 Overview
The UDK was chosen to create and power the virtual world the user would explore. It was chosen because it is freely available, powerful, contains native Rift support, and allows for easy development. Normally the UDK is used to create video games in which the user controls an avatar using a mouse and keyboard. For this project, the UDK game has been modified to get commands from the Middleware.

4.2.2 UDK TCP server
In order to communicate with the middleware, the game runs a TCP server that awaits commands. When a command is received, it is parsed and then the command is executed on the game server. The parsing is handled by a special C library we created. This is done because the Unreal Script is slow at parsing floats from the string commands.
4.2.3 Commands
The UDK can receive two commands. The first is a rotate command, which tells the game to rotate the character to face a specific heading. It consists of a single number which is the direction to face. The second command tells the avatar to move. It consists of a heading and speed. The heading number tells the game which direction to move the avatar; the speed number tells the game how fast to move the avatar. The avatar will keep moving at this speed and heading until a new command is received. The command format is covered in more detail in Section 8.5 - UDK Game mod.

4.2.3.1 Maps
The UDK comes with a map editor that allows for the creation of new custom maps and the modification of existing maps. This allows for the easy creation of test and demo maps. The UDK editor seen in Figure 5 also allows for the import of 3d content, which can be downloaded for free from various internet sites. This will allow for the addition of rich content in the demo level.

Figure 5 - UDK Map editor

4.3 IMU

4.3.1 Overview
The IMU subsystem is tasked with calibrating the user’s orientation in order to align the Rift’s rotation coordinate space with the GPS’s heading system. A user’s orientation is the heading he is facing, where heading is a 360 degree circle with 0 degrees pointing north and 90 degrees pointing east. The information is used to rotate the game avatar as the user rotates. When the system is starting the value of 0 degrees for heading output by the rift is not north, it is the direction the headset was facing when powered on. Using the IMU we can determine an offset to add the Rift’s heading so that a heading of 0 degrees for the Rift is north. This is done so that the heading given by the GPS system will be on the same coordinate space as the rift. That is to say that when the GPS subsystem reports the user as moving with a heading of 90 degrees or East, this direction is the same direction that the Rift is facing when it reports a heading of 0 degrees. By having both systems on the same coordinate space, this allows for the user to move in one direction while looking in another direction.

The heading that the IMU is the angle the IMU is turned relative to an initial reference point. This reference point can be set in a variety of ways. By default the IMU, will set the direction of magnetic north as the reference point. This can be changed using a tare matrix. A tare matrix is just a matrix of
transformations that will take place on the data before it is reported. The YEI Threespace IMU senses rotation in thee dimension so its tare matrix is a 3x3 matrix. It is possible to set the values of the matrix manually, but it is more convenient to use the build in tare command. The tare command will tell the IMU to use the orientation it is in at the moment as its new reference point. The tare command will modify the tare matrix so that the value reported when the IMU is in that position is 0. Initially, the IMU system was designed to take advantage of the tare command but it was later decided that the IMU’s default behavior of using magnetic north as a reference was desirable.

4.3.2 Hardware
The hardware sensor chosen to do the job was YEI’s 3-Space USB Sensor as referenced in the parts list on page 48. This was chosen because it was a complete attitude heading and reference system, with a fast update rate, built in filtering, a USB connector, and an extensive C library. This sensor has been used in past VR projects with multiple game engines including Unreal 3.

4.3.3 Software
The IMU subsystem application was written in C to take advantage of the C library. The software starts up a TCP server and connects to the IMU device. Once the Middleware connects, it begins to take orientation readings from the IMU and send the readings to the Middleware. If a connection is lost, the program waits for another connection. How often the IMU is read and the data is sent can be controlled via command line arguments. This is further explained in Section 8.6 - IMU Data Com Utility.

4.4 GPS

4.4.1 Overview
The GPS system is tasked with tracking the user’s movement as he or she walks around. Specifically, this system provides information about the user’s velocity, that is the speed the person is walking at, and the user’s heading, that is the direction the person is walking. The system uses the RTKLIB program package in order to take advantage of the RTK (Real Time Kinematics) satellite navigation technique. This technique is preferable to standard GPS navigation because it is able to track the user’s position much more accurately. Standard GPS typically will give a position with an error of up to 2.5 meters. This can potentially mean that standard GPS is unable to detect the user is moving until the user already taken a few steps or that the user is standing still. The RTK technique is able to report user position to an accuracy of a few centimeters. This allows the Middleware to detect when the user takes a step or makes a slight movement. Additionally, the Middleware is able to filter out the slight errors in position so that when the user is standing still, the game avatar also remains still. More information about Real Time Kinematics can be found in section 2.3.1.

The heading tracked by the GPS subsystem is different from the heading tracked by the Rift headset. The Rift tracks the direction the user is facing, whereas the GPS subsystem tracks the direction the user is moving. For example, a person can be facing north while walking backward towards south. In this case, the person’s orientation would be north or 0 degrees, and his GPS heading would be south or 180 degrees.
4.4.2 RTK Library
The RTK Library is used to get speed and heading information from the GPS receiver. It uses the real
time kinematics technology described in section Real Time Kinematics Library. It comes with an
extensive library, as well as precompiled apps. The RTK apps contain all the features needed, allowing
for GPS readings to be taken, formatted to NMEA-183, and then sent through TCP to the Middleware.

4.4.3 Antenna and Receiver
The V.Torch VTGPSIA-3 antenna was chosen for this project. The VTGPSIA-3 has a gain of 28dB, a built in
Low-Noise-Amplifier, and an impedance of 50 Ω (V.Torch Electronics). These characteristics make the
VTGPSIA-3 superior to other antennas in the respective price range. The Yuan-10 receiver was chosen
for its high update rate of up to 20 Hz and the support availability by the RTK library. The Yuan-10 USB
doncle is based off the Skytraq S1315F-RAW GPS chip (Bavaro, 2013).

4.4.4 Base station and Raspberry Pi
A base station is needed by the RTK library. We used a Raspberry Pi (Raspberry Pi, 2013) that sits on the
roof of a friend’s house in San Luis Obispo with an antenna and receiver. It’s a one story house 1.5 miles
from the test field at Cal Poly. This also requires that the user system have internet access. This was
accomplished through the use of a mobile Wi-Fi hotspot created by a smartphone.

4.5 Middleware
4.5.1 Overview
The Middleware’s job is to act as an intermediary between the other subsystems. This allows for more
modular design, because the other subsystems don’t need to know the details of each other’s
operation. One subsystem can be complete redesigned without needing to make any changes to the
others. The Middleware receives data from the GPS and IMU subsystems, filters the data, the converts it
to UDK units and sends it to the UDK. It is also responsible for starting the other processes.

4.5.2 Filtering
The Middleware also handle’s additional filtering of incoming data. The filtering system is set up to be
modular. This allows for the creation of new filters which can then be dropped into the system. The
filter parameters can be easily changed in a properties file. This allows for simple fine tuning during
testing. The following filters are employed in the middleware: a step filter, a floor filter, and low pass
filter.

The step filter works by only allowing the output to change by a set amount. For instance, a step filter
with a step size of 3 would only allow the following outputs: 0, 3, 6, 9... and so on. The step filter rounds
down the input to the nearest step size. This prevents small jumps in output but allows for bigger jumps.
This filter is employed by the middleware to filter the IMU orientation heading. The heading is measured
to 5 decimals places and picks up a lot of noise when measured to this degree. It would be undesirable
for the output to change because of this noise and send an update to the UDK every time the heading
changed by the smallest amount. The step filter allows for the amount of change needed to cause an
update in the heading to be configured.
The floor filter and low pass filter are used in conjunction. The modular design of the Middleware filters allow for the easy linking of different filters, so that the output of one filter is used as the input of another. GPS speed and heading and speed are fed into the low pass filter to filter out any sudden changes. It does this by changing the previous output by the scaled difference of the current input and the previous output. The effectively prevents the output from changing less quickly. This is done because the GPS will occasionally have sudden signal loss or interference which results in sporadic speed changes. The low pass filter will filter out these short sudden speed changes by require a consistent change over time to affect the output of the filter. The floor filter then receives the previous filter’s output and rounds it down to the nearest whole integer. Together these two filters result in smoother movement, filtering out harsh and sudden changes that can break the user’s immersion. Filter results can be seen in Appendix D - Filter Results.

4.6 Rift
Initially the Oculus Rift for which this system was designed to support was scheduled to be released in December 2012. Unfortunately, the Rift was delayed until April 2013. The majority of the system was designed without access to the Rift or the Oculus software development kit. This forced the design of the system to be as modular and easily changeable as possible so that when the Rift was released no large design changes needed to be made to the system in order to support the Rift.

5. 5. Construction and Problems Encountered

5.1 Overview
No major obstacles were encountered during the course of the project, though many smaller ones were overcome. The following documents the issues encountered when developing each subsystem.

5.2 GPS

Figure 6 - Base Station
The development of the GPS subsystem was fraught with problems. Learning how to use the RTK programs was very difficult. The programs must be configured properly. Setting up the RTK base station also proved difficult. Initially, we were trying to set up the base station a few minutes before running test. Additionally, the base station was only a few meters from the user GPS. It took us a while to figure out that the base station should be set up before hand and have its GPS location very accurately identified. However, this led to problems with time delay between the base station and the user GPS.

We also had problems with antennas included with the GPS units. The initial antennas had long cables. These long cables affected added noise to the antenna. To fix this, antennas with shorter cables were purchased. Helical antennas were tested, but found not to have as high a signal strength reception.

We also had problems with the Yuan-10 receivers purchased. The receivers would get out of time sync and start reporting negative pseudo ranges. RTK would disregard these negative pseudo ranges, causing it to stop reporting position. Initially, this was fixed by doing a factory reset of the receivers. This would fix the problem for a few hours of continuous operation. A firmware update was released during the project that would fix this issue. When attempting to update the two receivers, both updates failed and the receivers failed to function. We were forced to purchase two new receivers which came with the updated firmware. The latest working firmware is version 1.7.25.

5.3 IMU
The YEI 3-Space C API library (YEI 3-Space Sensor Software Suite, 2013) made getting data from the IMU very easy. A few small issues were encountered involving tarring the device and the heading system used. Initially, the IMU was only going to be used for relative orientation change. So it didn’t matter if it reported 0 degrees as north or not. But as the project was developed it was realized that in order to have 3 degrees of freedom it was necessary to know the direction the person was facing compared to the GPS heading. This required for us to stop tarring the device and rely on the default behavior of IMU to have 0 degrees as North. However, this required the device to be in a specific orientation. A system needed to be created to manually set the tare matrix to allow for arbitrary orientations. For information on what it means to tare the IMU see section 8.6. This system was not developed before the project was finished. Instead the IMU was mounted to take advantage of its default orientation.

Another issue was found when attempting to mount the IMU on the backpack. If the IMU was mounted to close to the metal frame of the backpack, the compass sensor would fail to find north properly leading to poor performance. When attempting to mount the IMU on the antenna mast, it was found that the mast vibrated when walking which would add noise to the IMU’s orientation data.

5.4 UDK
The changes that needed to be made to the UDK engine were relatively simple. An online tutorial (TCP Link, 2013) was followed to add a TCP server to the game engine. Changing the player avatar to be controlled by TCP commands required few changes to the game code. However, figuring out exactly what changes needed to be made was very difficult. The free version of the UDK does not come with any documentation or support, and the code does not contain any commenting. A method a trial and error was employed to see how changes to the code would affect the game. Once an understanding of
the behavior of the code was achieved, it was straightforward to make changes to allow for the player avatar to be controlled through the TCP server.

6. Development and Regular Use

6.1 Powering the Oculus Rift

6.1.1 Power Requirements
The current manufactured version of the Oculus Rift is powered by a 5 volt DC, 1.5 amp, wall adapter and consumes 3 watts at peak usage. The wall adapter prohibits mobile operation of the unit, however, the Rift can be modified to use USB supplied power via the existing USB control connection.

The USB-IF, Inc., the governing organization of the USB standard, has outlined the minimal power supply of USB 3.0 to be 900mA and 500mA for USB 2.0 (USB-IF Corp, 2011). Hardware manufactures typically provide additional power above and beyond the minimal specification. This allows the Oculus to be safely powered through USB.

6.1.2 Hardware Modifications
The Oculus Rift box circuit board can be accessed by removing the four rubber pads on the base of the unit. The four pads reveal four small Philips screws which when removed allow the case to easily be taken apart. A single wire of at least 30 gauge must be soldered from the power adapter positive tap marked A in Figure 7 – Oculus Rift Circuit Board to the USB +5V VBUS power pin marked B in the same figure.
6.2 Project Source Code and Dependencies

Table 3 – Project Archives lists and describes the packaged archives for this project. The archives contain all source code written for this project as well as all dependencies. The project archives can be requested from Dr. John Oliver.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev_udk.zip</td>
<td>The March 2012 version of the Unreal Development Kit with Oculus Rift support as well source code for the Insight game mod.</td>
</tr>
<tr>
<td>dev_imu.zip</td>
<td>Visual Studio 2010 project and source code files for the IMU Data Com utility.</td>
</tr>
<tr>
<td>dev_udkassistant.zip</td>
<td>Visual Studio 2010 project and source code files for the UDK Assistant dll.</td>
</tr>
<tr>
<td>dev_middleware.zip</td>
<td>Eclipse project file with source code for the Middleware component. All dependencies are included along with the project files.</td>
</tr>
<tr>
<td>prod_insight.zip</td>
<td>A preconfigured package that includes compiled versions of the IMU Data Com, Middleware and RTKLIB.</td>
</tr>
<tr>
<td>prod_udk.exe</td>
<td>A packaged version of the UDK that contains only the required content to run the UDK Game for this project.</td>
</tr>
</tbody>
</table>
6.3 Setting up the Environment
This section describes how to setup and configure the minimal requirements for this project for both development and production purposes. Sections 6.3 and 6.4 provide further details for setting up production and development environments, respectively.

6.3.1 Windows
At present the Unreal Development Kit is only available for the Windows platform. As such, this project requires Microsoft Windows. While most components of the project are cross-platform they are primarily designed and supported on Microsoft Windows versions Windows XP and newer. While not specifically required, it is highly recommended that all Windows updates are applied via the Windows Update manager prior to proceeding in the setup instructions.

6.3.2 Java Sun
The Middleware component of this project is written on top of the Java SE 7 environment. While the Java SE Runtime Environment is sufficient enough to run the Middleware, it is suggested that users install the latest version of the Java Development Kit 1.7 instead. The JDK is available for free on the Oracle website (Oracle, 2013).

6.3.3 Oculus Rift
The Oculus Rift does not require the installation of drivers. The Rift is seen by Windows display drivers as a regular monitor device. To use the Rift, configure the Windows display drivers to clone the primary monitor on to the Rift. Please note that because this procedure varies between Windows versions and display drivers, specific instructions are not included. The intention is to display the same image on the Rift as the laptop monitor.

6.3.4 YEI Threespace Sensor Drivers
The YEI sensor requires drivers to be installed. The driver version 2.0r9 was used in this project. These drivers are available from the YEI Technology website (YEI 3-Space Sensor Software Suite, 2013).

6.4 Setting up the Production Environment

6.4.1 Install Packaged UDK
The packaged installer named prod_udk.exe contains the minimal components of the Unreal Development Kit required to run the Insight project. The default installation path, C:\UDK\Insight, is recommended as the Middleware is pre-configured to expect the UDK at that location.

6.4.2 Install Insight Project
The Insight project requires no special installation procedure. Simply extract the project from prod_insight.zip to a folder. Once extracted proceed to read Section 7 for additional instructions. The remaining sub-sections of this section relate to development environment configuration.
6.5 Setting up the Development Environment

6.5.1 Integrated development environments
The Middleware was written using the Eclipse IDE for Java Developers version 4.2. The provided source code files are packaged along with the Eclipse project folder which allows for project code to be easily imported into the Eclipse workspace. Eclipse is freely available to download from the Eclipse Foundation (Eclipse Foundation, 2013).

The IMU Data Com along with UDK Assistant were written using Microsoft Visual Studio 2010. The Visual Studio project solutions along with source code are packaged together to allow for the easy import of the projects. Microsoft Visual Studio 2010 can be purchased from Microsoft. Alternatively, Microsoft offers a free lightweight version called Visual Studio 2010 Express which can be downloaded from the Microsoft website (Microsoft, 2010).

6.5.2 Middleware
To set up the development environment for the Middleware project perform the following steps.

1. Extract the project archive named dev_middleare.zip to a desired folder.
2. Launch the Eclipse IDE
3. Then click File -> Import
4. Under General select “Existing Projects into Workspace” and click Next.
5. Select the directory to which the project was extracted.
6. Check the “Copy the projects into workspace” checkbox.
7. Click Finish.

6.5.3 Unreal Development Kit
A pre-configured version of the Unreal Development Kit with Oculus Rift support has been included with this project. Simply extract the dev_udk.zip archive to the C:\UDK directory. Developers should familiarize themselves with the UDK official wiki located at EpicGames (Unreal Engine 3 Basics, 2013). Section 8.5 provides additional details for development of the Insight UDK game mod.

6.5.4 IMU Data Com
To set up the development environment for the IMU Data Com project perform the following steps.

1. Extract the project archive named dev_imu.zip and open in Microsoft Visual Studio 2010.
2. Then click Build->Build solution to build the project. An executable file called IMU_DataCom.exe will be created in the Release folder in the project directory.

Additional Notes:

- To run the utility the dynamic linked library file ‘ThreeSpace_API.dll’ must either be in the same directory as the utilities executable file or the path to it specified in the environmental variable Path found in System Properties -> Advanced -> Environmental Variables.
- Ensure that the files ‘ThreeSpace_API.lib’ and ‘ws2_32.lib’ are listed under Additional Dependencies in the Property Pages.
6.5.5 UDK Assistant
The UDK Assistant provides the UDK game mod a C++ interface for performing string to integer conversion. The UDK scripting language does not include such functionality thus requiring the use of the UDK Assistant.

To set up the development environment for the UDK Assistant project perform the following steps.

1. Extract the project archive named `dev_udkassistent.zip` and open in Microsoft Visual Studio 2010.
2. Then click Build->Build solution to build the project. A dll file called Insight.dll will be created in the Release folder in the project directory.
3. The compiled dll must be copied into the C:\UDK\UDK-2013-04\Binaries\Win32\UserCode folder in order for the UDK to load the dll.

6.6 Building RTKLIB Components
The two primary components of RTKLIB which are used in this project are rtkrcv and str2str. The former is a full-fledged program for processing and filtering GPS. The latter only interfaces with a GPS dongle and outputs raw data. Str2str is use in the RTK base station. This section covers how to build each component from source on both Windows and Linux.

6.6.1 Windows
RTKLIB components are runnable in Windows but must be compiled from within Cygwin. Cygwin is a set of Linux programs compiled for Windows. Among the variety of Linux components included in Cygwin are libraries, linkers, and compilers. Cygwin can be downloaded for free from RedHat (RedHat, 2013).

After downloading the installer executable from RedHat proceed with the installation process. Select “Install from Internet” and choose an installation directory and root directory. A prompt will appear with a list of packages to install. The defaults provide all minimal requirements needed to compile and run rtkrcv and str2str. After installation is complete, launch the Cygwin terminal from the desktop.

6.6.2 Getting RTKLIB
The RTKLIB source can be downloaded from rtklib.com. While the current stable version is 2.4.1, it contains bugs which prevent proper operation of rtkrcv. As such, be sure to use version 2.4.2 beta or newer. Download the source zip and extract. Note that by default Cygwin terminal will start in C:\cygwin\home\<your user>\. It is recommended to extract RTKLIB in this directory.

6.6.3 Compiling
If compiling under Windows launch the Cygwin terminal. Change directories into rtklib_2.4.2xx/app. To compile rtkrcv issue the following commands:

```
# cd rtkrcv
# make gcc/rtkrcv
```
Make will produce an rtkrcv executable. To make str2str issue the following commands:

```bash
# cd ../str2str
# make gcc/str2str
```

### 6.6.4 Use
Once str2str and rtkrcv have been compiled they can be copied out of their respective directories. In order for the RTKLIB programs to run properly on Windows, it is required that cygwin1.dll is in the same directory as the executables. The dll can be found inside the Cygwin installation directory under `bin\cygwin1.dll`.

### 7. User Guide
This section provides detailed instruction on how to configure and run the components of this project.

#### 7.1 Setting up a RTK Base Station
The str2str application provided by RTKLIB reads raw data from the USB GPS dongle and converts the data into the standardized RTCM3 format (Eleiche, 2008). The base station itself does not do any filtering or post processing of the GPS data; that is the responsibility of the rtkrcv running on the backpack laptop. Str2str simply allows the GPS data to be accessed by rtkrcv through TCP/IP.

##### 7.1.1 Base Station Location
GPS satellites are not in geo-synchronous orbit meaning they do not stay in the same position in the sky; as such for optimal results the antenna needs to have a full view of the entire sky and not be obstructed by trees, buildings, or landscape. The Attaining a GPS Fix section will cover how to determine if a location is undesirable.

##### 7.1.2 Running str2str
Before running str2str, the GPS device must be identified. In Linux devices are found in the `/dev` directory. The simplest way to identify which tty device maps to the GPS is by unplugging the GPS USB dongle and listing all tty devices using `ls`. Afterwards connect the GPS back and list the devices again. A new addition will be in the list. Typically the GPS is listed as ttyUSB0 or ttyUSB1.

```bash
# ls -l /dev/tty*
```

After the GPS device name has been identified, str2str can be run using the following command:

```bash
# ./str2str -in serial://ttyUSB0:115200:8:n:1:off#stq -out tcpsvr://:5000#rtcm3 -c
data/skytraq_raw_20hz.cmd
```

The above parameters initialize str2str to use ttyUSB0 as the GPS input with a baud rate of 115200, 8 bit, no parity, 1 stop bit, and no flow control. These are the default settings for the Yuan10 GPS module used in this project. A control file must also be specified for additional GPS initialization settings. These settings can be found in the `data` directory of the RTKLIB folder. Lastly, str2str is configured to run a TCP server on port 5000 and output RTCM3 formatted GPS data.
Example output for str2str is:

```
# ./str2str -in serial://ttyUSB0:115200:8:n:1:off#stq -out tcpsvr://:5000#rtcm3 -c
data/skytraq_raw_20hz.cmd

stream server start
2013/05/29 20:42:45 [CW---]  0 B    0 bps
2013/05/29 20:42:50 [CW---] 2902 B  4298 bps (1) waiting...
2013/05/29 20:42:55 [CW---] 5602 B  4298 bps (1) waiting...
```

It is important to note that str2str does not launch as a service but rather as a foreground application. For this reason it must be launched into background. There are two primary ways to accomplish this, GNU Screen and nohup. Screen is available to download on most major Linux distributions using their respective package managers (GNU Operating System, 2010). The benefit to using Screen is that it allows a background terminal to be launched and interacted with. Nohup simply launches a foreground process into the background with lower overhead. Example uses of nohup and screen follow below.

```
# nohup ./str2str -in serial://ttyUSB0:115200:8:n:1:off#stq -out tcpsvr://:5000#rtcm3 -c
data/skytraq_raw_20hz.cmd &

# screen -dm ./str2str -in serial://ttyUSB0:115200:8:n:1:off#stq -out tcpsvr://:5000#rtcm3 -c
data/skytraq_raw_20hz.cmd
```

### 7.1.3 Attaining a GPS Fix

Once str2str is setup on a base station computer it is time to produce an accurate GPS location fix for the base station. The best fix can be obtained by running the procedure for 12 hours in fair to good atmospheric conditions. Generally the best results are produced at night when atmospheric conditions are typically calmer and minimally affected by the Sun.

On a Microsoft Windows machine launch the rtknavi.exe provided by the Middleware in the rtk folder. The application is pre-configured with typical GPS filter settings and options. It is important to note RTKLIB contains sophisticated filtering with many parameters. It is highly recommended that users read the RTKLIB documentation. The documentation contains detailed explanation of all parameters and how they affect fix results (Takasu, 2013).

Once rtknavi is launched proceed to Input Streams (labeled “I”) -> Rover Options (“…”). Fill in the IP address and port of the base station running str2str. Click “OK”, “OK”, and “Start”. Figure 8 – RTKNAVI Configuration can be referenced for additional details.
An initial fix may take a minute. An example view of rtknavi attaining a fix can be seen in Figure 9 – RTKNAVI PPP Fix. If a fix is not acquired within two minutes this indicates an issue attaining a valid lock. Attempt to “Stop” and “Start” the process again. If the issue persists refer to RTKLIB documentation for troubleshooting instructions (Takasu, 2013).

**Figure 9 – RTKNAVI PPP Fix**

### 7.1.4 Poor GPS Signal

Poor signal can be identified by viewing the signal strengths of observed satieties. Figure 10 – Satellite Signal Strength shows an example of signal strengths for satellites. The color green identifies good signal while shades of yellow, orange and red indicate moderate through poor signal with grey indicating
intermittent signal. A desirable base station location should produce green signal strengths above 45dbHz on a minimum of 4 satellites.

![Signal Strength Diagram](image)

**Figure 10 – Satellite Signal Strength**

### 7.2 Setting up RTK on Backpack

#### 7.2.1 Finding Device Number

Before configuring rtkrcv the serial port number of the USB GPS dongle must be found. The port number can be found on Windows 7 by:

1. Click “Start” button
2. Right click “My Computer”
3. Click “Manage”
4. Clicking on “Device Manager” under “System Tools”
5. Clicking the + expand symbol next to “Ports (COM & LPT)”
6. A device named “USB Serial Port” will be in the list.
7. The Port will be labeled to the right in parenthesis.
8. Take note of the number for example COM13 is port 13.

See Figure 22 – Finding Serial Port Number for a visual overview of the steps. When entering the port number in the rtkrcv.conf configuration file described in Subsection 7.2.3 it is important to use the Unix device name format rather than the Windows format. For example if the device serial port is “COM13” the device name used in rtkrcv.conf should be “ttyS12”. The reason the identifier 12 is used rather than 13 is because rtkrcv uses the Unix device numbering scheme which begins at 0 whereas Windows begins at 1. Remember to decrement the device number by 1.

#### 7.2.2 RTKRCV Configuration

A pre-configured rtkrcv configuration file is provided with this project. The configuration file is located inside `middleware/rtk` and is named `rtkrcv.conf`. The important configuration parameters to edit are as follows in Table 4 – RTKRCV Parameters.
Table 4 – RTKRCV Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inpstr1-path</td>
<td>ttyS11:115200:8:n:1:off</td>
<td>The local GPS device. This is the serial port the device is on.</td>
</tr>
<tr>
<td>Inpstr2-path</td>
<td>:@192.168.1.2:5000:/</td>
<td>The base station IP address and port.</td>
</tr>
<tr>
<td>ant2-pos1</td>
<td>35.29561415</td>
<td>The latitude location of the base station in degrees.</td>
</tr>
<tr>
<td>ant2-pos2</td>
<td>-120.67658877</td>
<td>The longitude location of the base station in degrees.</td>
</tr>
<tr>
<td>ant2-pos3</td>
<td>49.94</td>
<td>The height of the base station in meters.</td>
</tr>
</tbody>
</table>

7.3 Middleware Configuration

The Middleware configuration file is located inside the base Middleware folder and is named “middleware.properties”. Notepad is strongly recommended to edit the file as programs such as Wordpad and Microsoft Word leave hidden formatting characters behind.

The middleware properties contains six sections detailed below. The default configuration file enables all components of the Middleware but disables them from automatically launching on startup. Meaning the user must manually start each component from the web GUI.

Table 5 – Web GUI Configuration Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>waitForApi</td>
<td>Boolean</td>
<td>If other components are enabled to start automatically they will wait until a user has opened the web GUI in a browser.</td>
</tr>
<tr>
<td>webPort</td>
<td>Integer</td>
<td>Port the Middleware web GUI is accessible in a browser.</td>
</tr>
<tr>
<td>apiPort</td>
<td>Integer</td>
<td>This is the port the Middleware listens on for commands from the web GUI.</td>
</tr>
<tr>
<td>apiEnabled</td>
<td>Boolean</td>
<td>Enables the web GUI component of the Middleware.</td>
</tr>
<tr>
<td>enableBrowser</td>
<td>Boolean</td>
<td>If web GUI is enabled this launches a web browser automatically.</td>
</tr>
</tbody>
</table>

Table 6 – GPS Configuration Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpsIP</td>
<td>String</td>
<td>IP address of the rtkrcv program’s server. Typically should be 127.0.0.1.</td>
</tr>
<tr>
<td>gpsPort</td>
<td>Integer</td>
<td>Port the rtkrcv server listens on. Configurable in rtk settings.</td>
</tr>
<tr>
<td>enableGps</td>
<td>Boolean</td>
<td>Disables the GPS component of the Middleware.</td>
</tr>
<tr>
<td>startGps</td>
<td>Boolean</td>
<td>Automatically launches rtkrcv program on Middleware startup.</td>
</tr>
<tr>
<td>gpsExec</td>
<td>String</td>
<td>Specifies the location and launch parameters of the rtkrcv program. Please note the application path requires double backslashes.</td>
</tr>
</tbody>
</table>

Table 7 – IMU Configuration Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imuIP</td>
<td>String</td>
<td>IP address of the IMU program’s server. Typically should be 127.0.0.1.</td>
</tr>
<tr>
<td>imuPort</td>
<td>Integer</td>
<td>Port which the IMU program’s server is listening on.</td>
</tr>
<tr>
<td>enableImu</td>
<td>Boolean</td>
<td>Disables the IMU component of the Middleware.</td>
</tr>
<tr>
<td>startImu</td>
<td>Boolean</td>
<td>Automatically launches IMU program on Middleware startup.</td>
</tr>
<tr>
<td>imuExec</td>
<td>String</td>
<td>Specifies the location and launch parameters of the IMU program.</td>
</tr>
</tbody>
</table>
Please note the application path requires double backslashes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>udkIp</td>
<td>String</td>
<td>IP address of the UDK game server. Typically should be 127.0.0.1.</td>
</tr>
<tr>
<td>udkPort</td>
<td>Integer</td>
<td>Port which the UDK game server listens on for commands. This is currently hard coded to be port 9000 inside the UDK project code.</td>
</tr>
<tr>
<td>enableUdk</td>
<td>Boolean</td>
<td>Disables the UDK component of the Middleware.</td>
</tr>
<tr>
<td>startUdk</td>
<td>Boolean</td>
<td>Automatically launches UDK game on Middleware startup.</td>
</tr>
<tr>
<td>udkExec</td>
<td>String</td>
<td>Specifies the location and launch parameters of the UDK game. Please note the application path requires double backslashes.</td>
</tr>
</tbody>
</table>

Table 9 – Filter Configuration Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lowPassFactor</td>
<td>Integer</td>
<td>Controls how aggressive the low pass filter is. A higher value creates a smoother output but adds delay.</td>
</tr>
<tr>
<td>stepSize</td>
<td>Integer</td>
<td>How large steps are in the step filter. This value is in Unreal unites which equates 52uu/meter.</td>
</tr>
<tr>
<td>speedMultiplier</td>
<td>Decimal</td>
<td>Multiplier used to increase actual speed of user. A value of 1.5 would produce a 50% increase.</td>
</tr>
</tbody>
</table>

7.4 Starting the Middleware

The Middleware can be started in two ways. The first is by double clicking on the middleware.jar file. A terminal will not be displayed to indicate the middleware is running. However, if configured, components will begin to launch automatically as specified in the configuration file and if enabled the web graphical user interface will launch a browser window. The second method is by opening a Command Prompt and changing directories to where the middleware.jar file is located and running the following command:

```
java -jar middleware.jar
```

7.5 Web Graphical User Interface

7.5.1 Accessing the Web GUI

The Web GUI can be accessed from either the computer running the Middleware at http://localhost:9004 or from a networked computer at the IP address of the machine running the Middleware. For example http://192.168.1.101:9004. Note that port 9004 correlates to the webPort configuration parameter in Section 7.3.
7.5.2 Screen Navigation
The Web GUI contains three pages referenced as Screens. Users can switch between screens using the left and right arrows located on either side of the page. Figure 19 – Web GUI Log Terminal and Controls Screen shows the arrows circled in red.

7.5.3 Log Terminal and Controls Screen
The initial screen a user sees when connecting to the Web GUI is the Log Terminal and Controls page seen in Figure 19 – Web GUI Log Terminal and Controls Screen. This page allows users to start and stop components of the system through the Middleware. The Log Terminal displays real-time output of the Middleware log to help with debugging and operation.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>The System button is used to start and stop the reading and processing of data. In the provided default configuration the UDK Game, IMU Data Com and rtkrcv do not automatically start when the System is started. It is important to note that those components must be started manually prior to starting the System.</td>
</tr>
<tr>
<td>UDK</td>
<td>Starts and stops the UDK Game and associated communication module.</td>
</tr>
<tr>
<td>IMU</td>
<td>Starts and stops the IMU Data Com and associated communication module.</td>
</tr>
<tr>
<td>IMU Tare</td>
<td>Used to initially align the in game player torso and head to face the same direction as user.</td>
</tr>
<tr>
<td>GPS</td>
<td>Starts and stops the rtkrcv and associated communication module. The rtkrcv can be interacted with through the RTKRCV Terminal Screen. See section 7.5.4 for more details.</td>
</tr>
</tbody>
</table>

7.5.4 Live Graphs Screen
Two real-time graphs are generated by the Live Graphs Screen. The top graph displaying the current raw velocity and filtered velocity in blue and red respectively. The second graph displays the raw heading and filtered heading in blue and red respectively as well. Note that the current implementation of the Middleware does not have heading filtering enabled. Lastly, the signal strength of each satellite in dB is displayed in vertical bars. Refer to Figure 20 – Web GUI Log Live Graphs Screen to see the Live Graphs Screen.

7.5.5 RTKRCV Terminal Screen
The RTKRCV Terminal Screen allows users to directly interact with the rtkrcv application through a web based terminal. For ease of use buttons are also provided for common rtkrcv terminal commands. Table 11 – Log Terminal and Controls Screen outlines the use of each button. Refer to Figure 21 – Web GUI RTKRCV Terminal Screen to see the RTKRCV Terminal Screen.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send</td>
<td>Sends the typed command to the rtkrcv. For detailed commands see the RTKLIB user manual (Takasu, 2013).</td>
</tr>
</tbody>
</table>
## 7.6 How to Launch the Project

This section assumes the components of this project have been successfully installed, configured, and Middleware program launched.

In the Web GUI click “Start” next to IMU. This will launch the IMU Data Com program in the background. Next click “Start” next to GPS. This will start the rtkrcv program and allow users to interact with it through the Web GUI. Navigate to the RTKRCV Terminal Screen in the Web GUI and click “Status.” A print out of the current status of the GPS fix will be displayed in the terminal. Periodically click Status to refresh the information. Wait a moment for a fix to be acquired. When a valid fix is attained a GPS location next to the line containing “pos llh single (deg,m) rover” will appear. For example:

```
    pos llh single (deg,m) rover: 35.29393806,-120.68628558,44.874
```

If a valid fix is not attained after several minute click “Restart.” Once a valid fix is attained proceed by switching to the Log Terminal and Controls screen and clicking “Start” next to UDK. Wait a moment for the UDK game to be launched. Note that ideally the Web GUI should be accessed from a different computer for when the UDK game is launched it starts in full screen mode.

After the UDK game is launched click “Start” next to System in the Web GUI. Lastly have the user wearing the Backpack and Rift face their head the same direction as their torso and click “Tare”. The tare function will align their view with the direction they are facing. The system is now ready to be used.

## 8. Developer Guide

### 8.1 This Guide

This guide is written to familiarize developers to where various components of the system reside and their basic functionality. This is a complement to the existing source code documentation and describes what rather than how.

### 8.2 Middleware

#### 8.2.1 Components

As described in Section 4.1, the Middleware is the core program that reads data from the IMU and GPS, filters the data, and outputs to the UDK game mod. The Middleware also provides administrative control of the entire system through a web based graphical user interface.
8.2.2 Filters
The filter interface IFilter is located in org.insight.middleware.ifilter/IFilter.java. The filter system design allows for filters to be linked sequentially. For example data can be automatically processed through a low-pass filter followed by a floor filter. This is accomplished by the IFilter constructor requiring an IFilter filter input. The input filter is run prior to the newly constructed filter.

StepFilter exampleFilter = new StepFilter(new LowPassFilter(null));

In the above code snippet, when passing data through the exampleFilter StepFilter the LowPassFilter will run prior to the StepFilter. It is important to note the above filters do contain additional parameters not shown in the example.

Filter implementations are located in org.insight.middleware.filter. In total there are eight filters: complementary, floor, high pass, two different low pass filters, a quick and dirty implementation of a complementary filter, a round filter and a step filter.

8.2.3 IP Receive-Transmit Interface
The IP Receive-Transmit Interface or RxTx provides basic functionality to connect to an arbitrary TCP server using a Java TCP client socket. org.insight.middleware.irxtx/ICommunicationIp.java contains the client initialization along with a multitude of send functions to send various Java primitives such as boolean, char, byte, integer, double, float, etcetera. The interface also defines an abstract method named packageData which is invoked when data is received by the TCP client from a server. The purpose of packageData is to allow different modules to implement their own handling of input data. The following subsections outline the different modules which use the interface.

8.2.3.1 IMU RxTx Module
The IMU RxTx Module connects to the IMU Data Com server and listens for server calls. The IMU Data Com sends data using the specification in section 8.5.4. The RxTx module creates a Java DataImuYaw object located in org.insight.middleware.rxtx.imu/DataImuYaw.java. Do note the DataImuYaw converts the IMU sensor’s yaw value of radians into UDK yaw which is an integer of value 0 to 65535.

8.2.3.2 GPS RxTx Module
Rtkrcv outputs GPS data using the NMEA-183 standard. Specifically the RxTx module parses messages of $GPRMC and $GPGSV type into DataGpsGprmc and Satellite objects respectively. See Appendix C for message format. The RxTx module is located in org.insight.middleware.rxtx.gps/CommGpsIp.java.

8.2.3.3 UDK RxTx Module
The UDK RxTx Module connects to the UDK Game mod server and is used to send filtered position and orientation data to the UDK. The message format is specified in Section 8.4. The RxTx module is located in org.insight.middleware.rxtx.udk/CommUdklp.java.
8.2.4 Webserver
The Middleware hosts a basic webserver for serving static HTML content for the web GUI. The webserver is implemented using the Netty Java library. Netty is an asynchronous event-driven network framework that allows the rapid development of network applications (Netty, 2013). The implementation used in this project is based off the sample HTTP server provided by Netty example code (netty/example, 2013). The webserver is located in org.insight.middleware.webserver, HttpStaticFileServer.java and HttpStaticFileServerPipelineFactory.java initialize the server while HttpStaticFileServerHandler.java handles web page requests. Files from the http directory inside the middleware project are served to web browser clients.

8.2.5 SimpleProcessor
The SimpleProcessor can be thought of as the core component of the Middleware that reads heading and orientation data, runs the data through specified filters, and outputs the filtered data to the UDK Game. The processor is located in org.insight.middleware.exec/SimpleProcessor.java. When run, the processor begins by copying data from the dataStore Vector object which the IMU and GPS RxTx modules store received data in to a local buffer named tempStore. The processor then reads data from the buffer chronologically, determines what type of data is stored and applies the appropriate filters on that data object. After the filters are run the filtered value is output to the UDK using the UDK RxTx module.

8.2.6 Logging
The Middleware relies on Apaches log4j Java logging library. A simple extension of the skeleton log appender is located in org.insight.middleware.log4j/Log4jJsAppender.java. The log appender utilizes the server API socket to broadcast log messages to connected web GUI clients. The log4j.properties file located in the middleware root folder outlines the log format.

8.3 API
An application programming interface, API, is a specification of how software components communicate. In this project an API was created in order to allow the web GUI to send control commands to the Middleware. The API leverages the power of the Socket.IO communication library (Rauch, 2013). Socket.IO began as a JavaScript library but now is a leading technology in client-server communication with implementations written in PHP, C, C++, Python, Java, and a variety of other languages. The library allows for bi-directional communication from client to server. Bi-directional communication enables the server to send events that do not correspond to direct client API calls. This functionality allows the Middleware to automatically send data such as log messages and terminal output on its own to the web GUI.

The Socket.IO library uses Events and Objects to communicate from server to client. In this project’s implementation Events are used as API call names and Objects as parameters or return values. In org.insight.middleware.api/ApiObject.java is a java object that contains a single String object with name value. A message is saved in the value variable and the ApiObject is passed by the API. For example a client creates a sysstatus event call with an empty object. The server replies with sysstatus and an
ApiObject which contains a message corresponding to the system’s status. The ApiGraph object and getgraph event function in a similar fashion with ApiGraph containing additional variables.

**8.3.1 API calls using ApiObject**

All command communications are done using the ApiObject. Typically the client does not use the ApiObject when sending a request to the Middleware but the Middleware does reply with an ApiObject.

<table>
<thead>
<tr>
<th>Call Event</th>
<th>Reply Event</th>
<th>Reply Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sysstatus</strong></td>
<td>sysstatus</td>
<td>started / stopped</td>
<td>Returns if system is started or stopped.</td>
</tr>
<tr>
<td><strong>gpsstatus</strong></td>
<td>gpsstatus</td>
<td>started / stopped</td>
<td>Returns if rtkrcv and RxTx are started or stopped.</td>
</tr>
<tr>
<td><strong>imustatus</strong></td>
<td>imustatus</td>
<td>started / stopped</td>
<td>Returns if IMU and RxTx are started or stopped.</td>
</tr>
<tr>
<td><strong>udkstatus</strong></td>
<td>udkstatus</td>
<td>started / stopped</td>
<td>Returns if UDK and RxTx are started or stopped.</td>
</tr>
</tbody>
</table>

An API client issues the “cmd” event with the following call values. The server replies with a different event and corresponding value.

<table>
<thead>
<tr>
<th>Call Value</th>
<th>Reply Event</th>
<th>Reply Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sys_start</strong></td>
<td>middleware_start / Error</td>
<td>success / Error message</td>
<td>Starts system processor.</td>
</tr>
<tr>
<td><strong>sys_stop</strong></td>
<td>middleware_stop</td>
<td>success</td>
<td>Stops system processor.</td>
</tr>
<tr>
<td><strong>udk_start</strong></td>
<td>udkstatus</td>
<td>started</td>
<td>Starts UDK game.</td>
</tr>
<tr>
<td><strong>udk_stop</strong></td>
<td>udkstatus</td>
<td>stopped</td>
<td>Stops UDK game.</td>
</tr>
<tr>
<td><strong>imu_start</strong></td>
<td>imustatus</td>
<td>started</td>
<td>Starts IMU program and RxTx.</td>
</tr>
<tr>
<td><strong>imu_stop</strong></td>
<td>imustatus</td>
<td>stopped</td>
<td>Stops IMU program and RxTx.</td>
</tr>
<tr>
<td><strong>gps_start</strong></td>
<td>gpsstatus</td>
<td>started</td>
<td>Starts rtkrcv and RxTx.</td>
</tr>
<tr>
<td><strong>gps_stop</strong></td>
<td>gpsstatus</td>
<td>stopped</td>
<td>Stops rtkrcv and RxTx.</td>
</tr>
<tr>
<td><strong>do_tare</strong></td>
<td>--</td>
<td>--</td>
<td>Performs tare function on rotation.</td>
</tr>
<tr>
<td><strong>get_sats</strong></td>
<td>get_sats</td>
<td>Satellite Object</td>
<td>When a client makes this call the server replies with a <code>get_sats</code> along with a single satellite object which contains properties about that satellite. The server sends a message back for each satellite.</td>
</tr>
<tr>
<td><strong>Termrtkrcv</strong></td>
<td>--</td>
<td>--</td>
<td>The web GUI is able to send text directly to the terminal through the Termrtkrcv event with ApiObject’s value being the desired text.</td>
</tr>
<tr>
<td><strong>Termrtkrcv</strong></td>
<td>--</td>
<td>Terminal output</td>
<td>The Middleware will automatically output data from the terminal to the API client for display</td>
</tr>
</tbody>
</table>
8.3.2 API calls using ApiGraph
The other type of object which is used by the API is the ApiGraph. The ApiGraph contains speed, filtered speed, timestamp, heading, and filtered heading. The object is located in org.insight.middleware.api/ApiObject.java.

8.3.3 Web GUI
The API portion of the html web GUI is contained inside html/js/api.js. The file contains code for both sending API requests and listening to replies as well as code to manipulate the web GUI html. For example on a successful gps_stop call the web GUI will update the GPS button to indicate the GPS is stopped and can be started again.

8.3.4 Middleware
On the Middleware side, the API is split up in different files. org.insight.middleware/Main.java contains code to initialize the Socket.IO server and lists what Event calls trigger respective Middleware Java functions. org.insight.middleware.api/ApiFunctions.java contains the functions associated with API event calls.

8.4 Web GUI
8.4.1 HTML Overview
The core HTML of the Web GUI resides inside the html/index.html file located in the Middleware folder while the JavaScript logic is in html/js/api.js. The Web GUI user guide in Section 7.5 refers to "Screens." These Screens are contained inside the carousel-inner html DIV tag with each screen being contained in its own item html DIV tag.

8.4.2 JavaScript Overview
As described in Section 8.3.3 the API communication logic resides in js/api.js. The socket.on() callback functions allow the web client to listen for specific API events. For example the following code snippet is called when a log API event is triggered. A trigger occurs when the Middleware API server sends a log event and the event is received by the JavaScript API client.

    socket.on('log', function(data) {
        // processing logic for "log" event
    });

8.4.3 Source Files

<table>
<thead>
<tr>
<th>Source File</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index.html</td>
<td>Contains all html code elements for the Web GUI.</td>
</tr>
<tr>
<td>css/ui.css</td>
<td>Html style and markup.</td>
</tr>
<tr>
<td>js/api.js</td>
<td>Javascript code for connecting to and communicating with Middleware API.</td>
</tr>
</tbody>
</table>
8.4.4 Included Libraries
The Web GUI utilizes a variety of JavaScript libraries. The libraries along with short descriptions and license type are outlined in Table 15 – Web GUI Libraries.

<table>
<thead>
<tr>
<th>Name</th>
<th>License</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bootstrap</td>
<td>Apache v2.0</td>
<td>Bootstrap is a front-end user interface framework for web development. Bootstrap was originally developed by Twitter (Twitter, 2013).</td>
</tr>
<tr>
<td>Bootstrap Progress Bar</td>
<td>Apache v2.0</td>
<td>Bootstrap Progress Bar is an extension of Bootstrap that adds sleek progress and loading bar (Geers, 2013).</td>
</tr>
<tr>
<td>Bootstrap Switch</td>
<td>Apache v2.0</td>
<td>Bootstrap Switch is an extension of Bootstrap that toggleable switches (Larentis, 2013).</td>
</tr>
<tr>
<td>JQuery</td>
<td>MIT</td>
<td>JQuery is a general purpose library that provides an easy to use API for manipulating HTML from within JavaScript (The jQuery Foundation, 2013). JQuery is required by the Bootstrap library.</td>
</tr>
<tr>
<td>Smoothie</td>
<td>MIT</td>
<td>Smoothie is used to produce real-time graphs for streaming data (Walnes &amp; Noakes, 2013).</td>
</tr>
<tr>
<td>Moment.js</td>
<td>MIT</td>
<td>Moment.js is a time keeping and date formatting library (The jQuery Foundation, 2013). This library is required by Smoothie.</td>
</tr>
</tbody>
</table>

8.5 UDK Game mod

8.5.1 File Overview
The UDK Game mod is located in the C:\UDK\UDK-2013-04\Development\Src\Insight\Classes folder.

<table>
<thead>
<tr>
<th>Source File</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsightGame.uc</td>
<td>Specifies what player controller and pawn this mod uses.</td>
</tr>
<tr>
<td>InsightPawn.uc</td>
<td>Used to start a TcpLinkServer server when pawn is created.</td>
</tr>
<tr>
<td>InsightPlayerController.uc</td>
<td>Overrides the UTPlayerController which controls pawn movement.</td>
</tr>
<tr>
<td>TcpLinkServer.uc</td>
<td>Basic TCP/IP socket server.</td>
</tr>
<tr>
<td>TcpLinkServerAcceptor.uc</td>
<td>Parses data from TCP clients and moves pawn.</td>
</tr>
</tbody>
</table>

8.5.2 TCP Server
The TCP server listens for connections from the Middleware UDK RxTx module. The two types of messages the UDK accepts are orientation and direction. Orientation messages are expected in the following format: “0,pitch,roll,yaw\n”. The format of direction messages are: “1,velocity,x,y\n”.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates that this data packet contains information about the orientation of the IMU.</td>
</tr>
<tr>
<td>pitch</td>
<td>Pitch of the IMU. This value is in UDK format where 0 is 0 degrees and 65535 is 360</td>
</tr>
</tbody>
</table>

36
Table 18 – Direction Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates that this data packet contains information about the direction of the GPS.</td>
</tr>
<tr>
<td>velocity</td>
<td>Velocity to apply to the player pawn. This value is in Unreal Unites where 1 meter is 52.5 UU.</td>
</tr>
<tr>
<td>x</td>
<td>The X component of the unit vector heading.</td>
</tr>
<tr>
<td>y</td>
<td>The Y component of the unit vector heading.</td>
</tr>
<tr>
<td>\n</td>
<td>Indicates the end of the message.</td>
</tr>
</tbody>
</table>

8.5.3 Pawn Movement

Before the direction or orientation parameters can be used by the UDK they must be first converted from comma separated value string format into integers. The conversion is not possible in Unreal Script and must be done in C++ using the UDK Assistant dll. The UDK Assistant provides two functions: ConvertRotation and ConvertSpeed. Both functions return an Unreal vector object which consists of three integers of names X, Y, and Z.

Player orientation in game is set using the Pawn.ClientSetRotation() is called with the processed orientation vector. Player speed is set by setting Pawn. GroundSpeed. The GroundSpeed specifies the velocity of the player. To specify the direction of the player Pawn. Acceleration unit vector is set to the processed X and Y components of the unit vector.

8.6 IMU Data Com Utility

8.6.1 Purpose

The purpose of the IMU Data Com Utility is to read heading form the IMU and send that heading to the middleware. The utility connects to a YEI Threespace IMU device and sets up a TCP server. Once a client, in our case the middleware application, connects to the utility, the utility will read heading information from the IMU device and send that data to the client. The following sections describe using the utility in more detail.

8.6.2 CMD line arguments

8.6.2.1 8.5.2.1 Synopsis

Command line usage: IMU_Data_Com.exe [PORT_NUM [WAIT_TIME]]
**8.6.2.2 Description**

Example Use: `IMU_DataCom.exe 500 10000`

This example will cause the utility to listen on port 500 for incoming connections. Once a connection is established, the utility will send the IMU heading data to the connected client every 10,000 microseconds. The optional arguments are further explained in Table 19 - Optional Command Line Arguments.

<table>
<thead>
<tr>
<th>Optional Argument</th>
<th>Explanation</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT_NUM</td>
<td>Used to specify the port number that the utility will use to listen for incoming TCP connections.</td>
<td>9001</td>
</tr>
<tr>
<td>WAIT_TIME</td>
<td>Used to set how often the utility will send data when a client is connected. The wait time is specified in microseconds.</td>
<td>5000</td>
</tr>
</tbody>
</table>

**8.6.3 Heading Data Packet Format**

The utility will send data about the IMU device’s heading to the client. The data is sent in the following format: “Y,Heading\n”. The details of the data packet are further explained in the below table.

<table>
<thead>
<tr>
<th>Data Packet Parts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Indicates that this data packet contains information about the Yaw of the IMU. The IMU can take three rotational measurements: Pitch, Roll, and Yaw. Yaw indicates heading.</td>
</tr>
<tr>
<td>,</td>
<td>The comma character is used to separate the type from the heading value.</td>
</tr>
<tr>
<td>heading</td>
<td>This is a decimal number indicating the heading in radians. The heading can range from –π to π radians.</td>
</tr>
<tr>
<td>\n</td>
<td>The data packet is ended with a new line character.</td>
</tr>
</tbody>
</table>

**8.6.4 8.5.4 Commands**

The client has the options of sending commands to the utility. The commands supported are shown in Table 21 - Available Commands. Any data received by the utility that is not a valid command is ignored.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tare the IMU Device</td>
<td>IMU,TARE</td>
<td>The utility will attempt to tare the IMU device. To tare the IMU means to set the zero reading for the heading to be the current orientation of the IMU. By default the IMU will show a heading of zero when facing magnetic north.</td>
</tr>
</tbody>
</table>

**8.6.5 Source Files**

The IMU Data Com project contains the source files shown in the following table.
Table 22 - Source Files

<table>
<thead>
<tr>
<th>Source File</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMU_Data_Com.cpp</td>
<td>This is the main application source file.</td>
</tr>
<tr>
<td>stdafx.h</td>
<td>Include file for standard system include files and project specific include files.</td>
</tr>
<tr>
<td>yei_threespace_api.h</td>
<td>YEI Threespace C API header file</td>
</tr>
</tbody>
</table>

9. Testing

9.1 Test Plan

9.1.1 Overview
The test plan described below attempts to verify the basic operation of the system while in use by a user. It is important to note the velocity filters applied provide responsiveness in favor of accuracy. This trade off allows the system to quickly attain if a user has stopped rather than how slowly they stopped. Appendix D shows the effects of various filter parameters. The downside to using step filters is they cause inaccuracies in displacement. This happens because the system judges in game displacement by using velocity. Steps are approximations of velocities and the approximations incur errors which are carried over to the errors in displacement. A consistent velocity must be used in order to provide accurate displacements.

9.1.2 Test Environment and Settings
The location of the testing will be the Cal Poly Athletic Fields. The GPS base station is to be located no less than 60 meters away from the closest location of the testing area. The GPS base station will be allowed 10 minutes to attain a location fix. The following filter and filter parameters will be used for testing:

<table>
<thead>
<tr>
<th>Filter/Setting</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor Filter</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Low Pass</td>
<td>Low Pass Factor</td>
<td>250</td>
</tr>
<tr>
<td>Speed Multiplier</td>
<td>--</td>
<td>1.0</td>
</tr>
</tbody>
</table>

9.1.3 Head Rotation Test
The Head Rotation Test is designed to verify that the in game user movement is unaffected by user head rotation. The test is conducted by having a user walk straight at a constant velocity from point A to B, approximately 20 meters away, while panning their head from left to right. The in game movement should not be altered by the head panning action.

9.1.4 Head and Torso Rotation Test
The Head and Torso Rotation Test is designed to verify the torso rotation and head rotation are dependent. The test is conducted by having a user walk along a 20 meter arch while looking at the focal point created by the arc. While walking the user should not adjust their head rotation to mitigate
inconsistencies in the location of the perceived focal point. If by the end of the arc the user still perceives the focal point to be at the same location test is successful.

9.1.5 Rectangle Test
The Square Test is designed to verify general accuracy of the localization system. As described above in Subsection 9.1.1 it is important to note accuracy is dependent on a tester’s ability to maintain a consistent velocity. This test is meant to verify that if a user travels along a square they return to approximately where they started. This test is conducted by having a user walk along a rectangle with sides of length 10 meters and 8 meters; stopping at where they initially started.

9.1.6 Circle Test
The circle test is similar to the Box Test. Instead of a tester walking along the sides of a square he or she is now walking around a circle of radius 6 meters.

9.2 Test Results

9.2.1 Head Rotation
The graph in Figure 11 displays the X and Y coordinates of the tester walking as seen by the UDK game. The X and Y axis are in Unreal Units which equate to 52.5UU per meter. The tester walked straight at a rate of 1m/s with their head turned at 90 degrees to their direction for approximately 2 second. He then turned his head to -90 degrees while continuing to walk straight. This panning motion was continued until reaching the stop location. The graph in Figure 11 illustrates the movement. In general the movement traces a negatively sloped line with no significant changes in heading. The results indicate that head rotation does not affect player heading.
9.2.2 Head and Torso Rotation
The Head and Torso Rotation test resulted in success. The tester maintained a direct line of sight to the focal point of the arc. The test results provided confirmation that the torso and head rotation are dependent and consistent. Further testing must be performed in order to ascertain the consistency of the inner-dependency. This test provided qualitative results rather than quantitate. A more elaborate test must be constructed and conducted to find quantitate results.

9.2.3 Rectangle
The rectangle test was conducted three times. The results appear in Figure 12, Figure 13, and Figure 14. The graphs plot the X and Y position of the player inside the UDK game coordinate system. The tests began at coordinate <-96, -1152>. In tests one and three the user arrives back to the starting position after completing the circuit. However, test two shows drift in the tracking system with the user not arriving back to the origin despite completing the circuit.
9.2.4 Circle Test

The circle test was conducted three times. The results appear in Figure 15, Figure 16, and Figure 17. The graphs plot the X and Y position of the player inside the UDK game coordinate system. The tests began at coordinate <-96, -1152>. A circle with an approximate radius 6 was created using athletic field cones. Unfortunately due to the lack of proper measuring equipment the circle was oval shaped. This error does not diminish the test results as the test was designed to test the system with constant changes in heading. All three tests resulted in the user not arriving back at the origin despite completing the full circuit on the field. The discrepancy in starting and finishing positions can be attributed to the inaccuracies in heading tracking rather than velocity tracking. Figure 18 shows a constant velocity was maintained during Test 1.

![Figure 15 – UDK coordinates Circle Test 1](image-url)
Figure 16 – UDK coordinates Circle Test 2

Figure 17 – UDK coordinates Circle Test 3
10. Conclusion and Future Work

10.1 Conclusion
The result of this project was a system that could allow a user to navigate a virtual world very intuitively by walking around naturally. The best example of this was by a visiting high school student. She was able to use the system to explore a virtual world with very little instruction and no training. All of the subsystems were able to be integrated and are able to communicate when the system is running. The test results also show that the system performs well in most areas and meets the required goals.

From the filter test results in Appendix D it can be shown that the system performs well tracking the velocity of the user. The velocity of the user can be tracked in any direction and the filtering performed by the Middleware results in a smooth and stable velocity. Most importantly, the velocity output by the system responds quickly to user input. When the user stops or starts to move, the game avatar responds quickly enough that no noticeable lag time is seen by the user.

When tracking the heading of the user, that is the direction the user is moving, the system had some issues. The heading was accurate when moving in a straight line for at least several seconds. However,
when the user stops and then moves in a different direction or changes direction while moving, the
heading output by the Middleware is not as accurate. This can be seen in the results of Circle Test in
Section 9.2.4. It takes a few seconds for the heading to update to an accurate number as the user
moves. This is thought to be caused by the fact the heading is determined by averaging the direction of
the last few GPS positions. This would lead to inaccurate headings when changing direction.

The following qualitative characteristics of the system were seen during testing and use. The RTKLIB
subsystem was found to be very accurate when it had a good view of the sky, but if the user is near
obstructions like trees or buildings, the accuracy and reliability is greatly reduced. The result being that
the user is unable to move the avatar, or the avatar moves without user input. However this did result in
an interesting observation, when the sky was obstructed and the avatar was moving without user input
it would occasionally rush towards a wall. This would inevitably cause the users to put up their hands to
brace themselves and scream, only to realize that it is a virtual wall and they are in no danger. This
situation shows how immersive the Insight VR platform coupled with the Oculus Rift can be, users are
able to lose themselves in the virtual world and start to see it as reality.

10.2 Future Work
In the future, the performance issues of the system could be fixed and additional features added. The
main performance issue where accuracy was poor is in tracking the movement heading of the user.
Since the heading is unfiltered and relies entirely on the GPS system, there are areas for improvement.
One idea is to use the IMU to get a heading for the user. This would be accomplished by integrating the
acceleration vector given by the IMU to get a velocity vector. The accuracy of the velocity or magnitude
of the vector are not the desired values, only the heading found by the angle of the velocity vector
would be used. If this heading were found to be reasonably accurate, even if only when starting to move
from a stand still, it could be used to augment the heading found by the GPS resulting in a more
accurate heading of the users movement.

Examples of additional features include the ability to play games, multiplayer support, and hand or tool
tracking. Examples of simple games that could be created using this system are Pac-man, maze and
puzzle solving, and exploration games. The addition of hand or tool tracking would increase the
immersion of the system and allow for new applications. The ability to move your hands in front of your
face and see the virtual avatar do the same would great increase the feeling of immersion in the virtual
world. Tracking a simple foam tool in reality, would allow for a virtual tool by the avatar to be used in an
immersive way. An example might be a virtual paint brush or hammer that is manipulated by the user
through the use of a simple foam version that is being tracked by the system. This would allow for
applications where a user could paint virtual objects or create and destroy virtual blocks using the
hammer tool.

Virtual reality is beginning to enter main stream consumption. With products like the Oculus Rift and
Google Glass being developed, the average consumer will soon have access to amazing virtual and
augmented reality technology. The Insight system could be the foundation for a VR platform that could
be used to create a variety of fun and interesting applications for entertainment, education, training and
therapy.
11. Works Cited


Appendices

A. Parts List and Costs

Table 24 - Parts and Costs

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost per unit</th>
<th>Quantity</th>
<th>Outlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yuan10 GPS module</td>
<td>$ 152.77</td>
<td>2</td>
<td>Onetalent-gnss.com</td>
</tr>
<tr>
<td>Replacement Yuan10 GPS module</td>
<td>$ 76.39</td>
<td>2</td>
<td>Onetalent-gnss.com</td>
</tr>
<tr>
<td>GPS Antenna Magnetic Mount SMA</td>
<td>$ 16.62</td>
<td>2</td>
<td>Sparkfun.com</td>
</tr>
<tr>
<td>Antenna GPS Embedded SMA</td>
<td>$ 11.95</td>
<td>2</td>
<td>Sparkfun.com</td>
</tr>
<tr>
<td>Item</td>
<td>Price</td>
<td>Quantity</td>
<td>Supplier</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>--------</td>
<td>----------</td>
<td>--------------</td>
</tr>
<tr>
<td>YEI Threespace Sensor IMU</td>
<td>$176.29</td>
<td>1</td>
<td>Yostengineering.com</td>
</tr>
<tr>
<td>Raspberry Pi Model B</td>
<td>$51.95</td>
<td>1</td>
<td>Amazon.com</td>
</tr>
<tr>
<td>16GB SDHC Flash Memory Card</td>
<td>$12.98</td>
<td>1</td>
<td>Amazon.com</td>
</tr>
<tr>
<td>PowerGen USB 2.1A 10W AC Wall Charger</td>
<td>$9.99</td>
<td>1</td>
<td>Amazon.com</td>
</tr>
<tr>
<td>Passive PoE Cable Set</td>
<td>$4.95</td>
<td>2</td>
<td>Sparkfun.com</td>
</tr>
<tr>
<td>Tabletop File-Mesh</td>
<td>$16.99</td>
<td>1</td>
<td>Staples</td>
</tr>
<tr>
<td>PVC Elbow</td>
<td>$0.39</td>
<td>1</td>
<td>Home Depot</td>
</tr>
<tr>
<td>PVC Cross</td>
<td>$1.22</td>
<td>1</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Hose Clamp 3/4” x 1-3/4”</td>
<td>$1.15</td>
<td>6</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Hose Clamp 1-3/4” X 2-3/4”</td>
<td>$1.27</td>
<td>4</td>
<td>Home Depot</td>
</tr>
<tr>
<td>1 Gal Empty Metal Paint Can W/Lid</td>
<td>$4.98</td>
<td>1</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Hex Bolts 1/4”X2”</td>
<td>$0.20</td>
<td>2</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Cut Washers 1/4”</td>
<td>$0.11</td>
<td>4</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Hex Nuts 1/4”</td>
<td>$0.06</td>
<td>2</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Lockwasher 1/4”</td>
<td>$0.15</td>
<td>2</td>
<td>Home Depot</td>
</tr>
<tr>
<td>3/4” X 10’ PVC40 PE Pipe</td>
<td>$2.28</td>
<td>1</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Satin Spray Paint</td>
<td>$3.87</td>
<td>1</td>
<td>Home Depot</td>
</tr>
<tr>
<td>Frame Alice GI Used</td>
<td>$38.95</td>
<td>1</td>
<td>SLO Camp N’ Pack</td>
</tr>
<tr>
<td>Oculus Rift</td>
<td>$315.00</td>
<td>1</td>
<td>OculusVR.com</td>
</tr>
<tr>
<td>Lenovo Thinkpad W510</td>
<td>$2000.00</td>
<td>1</td>
<td>Lenovo.com</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>$3173.49</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


B. Web GUI

Figure 19 – Web GUI Log Terminal and Controls Screen
Figure 20 – Web GUI Log Live Graphs Screen
Figure 21 – Web GUI RTKRCV Terminal Screen
Figure 22 – Finding Serial Port Number
C. NMEA-183 Message Format

$GPRMC,hhmss.ss,A,llll.ll,a,yyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh

1 = UTC of position fix
2 = Data status (V=navigation receiver warning)
3 = Latitude of fix
4 = N or S
5 = Longitude of fix
6 = E or W
7 = Speed over ground in knots
8 = Track made good in degrees True
9 = UT date
10 = Magnetic variation degrees (Easterly var. subtracts from true course)
11 = E or W
12 = Checksum

$GPGSV,1,1,13,02,02,213,.,03,.,3,000,.,11,00,121,.,14,13,172,05*67

1 = Total number of messages of this type in this cycle
2 = Message number
3 = Total number of SVs in view
4 = SV PRN number
5 = Elevation in degrees, 90 maximum
6 = Azimuth, degrees from true north, 000 to 359
7 = SNR, 00-99 dB (null when not tracking)
8-11 = Information about second SV, same as field 4-7
12-15= Information about third SV, same as field 4-7
16-19= Information about fourth SV, same as field 4-7
D. Filter Results

Figure 23 – Low Pass Factor 50

Figure 24 – Low Pass Factor 100
Figure 25 – Low Pass Factor 250

Figure 26 – Low Pass Factor 500
Figure 27 – Low Pass Factor 1000

Lowpass Filter Followed by Floor Filter
Low Pass Factor: 1000

Knots/Sec

Time (Seconds)