POWerNAV
Map Data in Augmented Reality for Better Pedestrian Navigation

Department of Computer and Information Science, University of Pennsylvania,
Undergraduate Senior Design 2015-2016

Anthony Hsieh  Michael Li  Darren Yin

ABSTRACT
Interpreting a map in real time, whether it be on paper or a smartphone, is often confusing and time-consuming. POWerNAV (Pedestrian Overlay onto World of Navigational Augmented View) is designed to improve the pedestrian navigation process by overlaying data from a map, like building names and streets, onto the user view via augmented reality (AR). The overlay integrates map data with the real world, removing the need to consult a separate map.

The Epson Moverio BT-200 smartglasses, which run Android 4.0.4, serve as the main development hardware. GPS and sensor data is used to detect user location and head orientation. Data is imported from Google Maps to create a model of the world, which serves as the basis for overlays on buildings, streets, and points of interest. Virtual objects like destination beacons and animated path arrows are also supported. To demonstrate the viability of POWerNAV as a platform for more complex, content-rich use, a Penn campus tour is included with the system.

It is evident that AR-based navigation has many uses beyond a campus tour that can be developed further, like in the indoor and automobile domains. However, current hardware suffers from relatively poor performance and needs to advance more. The most prominent challenges of developing POWerNAV involve grappling with imperfect sensors. The user’s location is approximated to the nearest pathway or road; a complex sensor fusion of the accelerometer, gyroscope, and compass, as well as computer vision analysis, is needed to ensure proper alignment of the overlay. The barrier to entry for designing a seamless AR experience will decrease as location and orientation sensors continue to improve.

INTRODUCTION / PROBLEM STATEMENT
Currently, pedestrians navigate the world using their smartphones using map services like Google Maps. However, this second screen creates two different worlds that a user has to process, aligning the map from their smartphones to the real world. Additionally, the map services applications on our phones usually identifies buildings as a single coordinate, instead of a volume of space. We tackle these problems of inconvenience using augmented reality (AR) technology, where the second screen lies on top of the real world, already aligned. To do this, we used the Epson Moverio BT-200 smartglasses (Figure 1). This hardware runs Android 4.0.4’s tablet OS and overlays the view onto the real world (see Figures 2 and 3). The application that we built tries to make a seamless experience for a pedestrian navigating through the world. We specifically looked at displaying buildings, streets, and point of interests located in and around the University of Pennsylvania’s campus. Using AR also allows us to create creative applications, such as a tour that leads a tourist around an area, just by looking at the real world through the AR overlay.
To demonstrate this idea, we also built a tour around the University of Pennsylvania's campus with beacons, directions, and audio explanations.

![Figure 2: The Moverio's home screen, running Android 4.0.4.](image)

**Figure 2:** The Moverio’s home screen, running Android 4.0.4.

![Figure 3: A representation of how large the overlay screen actually appears when wearing the smartglasses.](image)

**Figure 3:** A representation of how large the overlay screen actually appears when wearing the smartglasses.

**APPRAOCH**

We organized development into three distinct modules. Each module corresponds with a single column feeding into the “OpenGL Overlay” component in Figure 4.

(i) **Virtual Eye and Display Engine**

The overlay display is powered by OpenGL, a low-level graphics library that is well-supported on Android. It can efficiently render triangles in a three-dimensional space. POWerNAV only requires three inputs to its OpenGL engine to work properly: a list of 3D coordinates to draw objects, virtual eye position, and virtual eye orientation. We draw all the map information that we store, which includes buildings, streets, and building names. We use an external text rendering library[1] to convert text into an OpenGL-compatible format. For a detailed discussion on how we store map information, see the next section, “Model World from Google Maps Data.”

After drawing a frame, the resulting scene is visually simplistic. Different colors and opacities are supported, but not any more complex features, like texture and lighting. This is good enough for POWerNAV, because it is an augmented reality system, not a virtual reality system. We simply want to indicate where a user should direct their attention in the real world; basic colors and shapes serve this purpose well. Modeling complex objects is not relevant.

We obtain the virtual eye position using a GPS receiver. Latitude and longitude are converted into 3D coordinates in our local system, which places the compass carving at 37th and Locust at (0,0,0). We assume a standard user height of 1.75m. Because the Moverio is not licensed to use Google Play Services, which provides enhanced location detection by incorporating triangulation analysis on wireless signals, we designed a separate Android application on a smartphone to get better location data. This application sends enhanced location data to the Moverio through a Bluetooth connection. The smartphone is carried on the user's body while using POWerNAV.

Virtual eye orientation is calculated using data from the Moverio’s accelerometer, gyroscope, and compass. A full orientation consists of three values which correspond to the three Euler angles: yaw, pitch, and roll. The Android library provides system calls which can convert accelerometer and compass data into a full orientation, which is then used by OpenGL to render the 3D model world onto a 2D screen. The gyroscope, which finely measures accelerations along any one of the three orientation values, is used to smooth updates to the orientation.

The smoothing process, known as “sensor fusion,” essentially institutes complementary filters on the raw orientation (accelerometer & compass) and the gyroscope.
These filters ensure that the eye orientation is updated smoothly using gyroscopic data, while using raw orientation to ensure that gyroscopic drift does not cause misalignment of the overlay. We incorporated work from two researchers, Paul Lawitzki[2] and Alexander Pacha[3] to develop a working sensor fusion solution.

Noise and inaccuracy in all sensor data proved to be a formidable challenge to providing a stable user experience. While sensor fusion was invaluable in mostly stabilizing the overlay, there were other issues which we documented and attempted to overcome. This is explained in great detail in the next section, “Results / Measurements.”

Because POWerNAV must exactly match what the user is seeing at any given time, it is highly sensitive to location and orientation inaccuracies. One source of inaccuracy comes from the GPS receiver, which suffers from rather unpredictable inaccuracy, especially during user movement. Standard filtering techniques were not effective enough in dealing with this issue. Our solution involves assuming that the user will always stay on pedestrian pathways, and locking the virtual eye location onto the nearest street. This “street lock” algorithm converts the system of pathways and streets into a 2D grid filled with line segments, and moves the eye to the nearest street as calculated by perpendicular intersection distance. This functionality is similar to how GPS in an automobile works. (Details on street modeling can be found in the below section, “Model World from Google Maps Data.”)

Additionally, the compass sensor data is more noisy than other sensor data, and occasionally suffers considerable inaccuracy.
We designed a “compass recenter” algorithm that tracks user heading based on movement along a street. Every straight street or walkway has an algebraic “slope” which can be converted into a compass direction. This calculated heading is used to correct the heading as reported by the compass sensor. It is important to note that this only works if the user is facing forward.

(ii) Model World from Google Maps Data

In order to build the model world to overlay on top of the real world, we take advantage of the ability to create one’s own maps with Google Maps. After creating a map with the buildings, streets, and point of interests we wanted, labeled appropriately, we export the map data into a kml file. We built a parser that reads in the kml file, stores all of the different object types into hash maps. Point of interests contain a single coordinate representing where they are located. Buildings contain a list of coordinates in counterclockwise direction, representing the base of the buildings. A user can adjust the height of individual buildings if the map is properly created and labeled. Streets contain the two endpoints. If the street is not just a straight line, we break down streets into straight line segments to make it easier for the pathfinding algorithm described later.

After importing all of the data, we draw all of the objects using OpenGL. See the previous section, “Virtual Eye and Display Engine,” for details.

(iii) Tour Experience

As a use case to demonstrate the capabilities of our application, we developed a tour experience to allow users to have a guided tour throughout the campus of the University of Pennsylvania. This allowed all features that we envisioned to be realized in the experience: virtual buildings and streets overlaid over their real life counterparts, display information about the buildings, using our pathfinding algorithm to suggest routes, and virtual objects such as arrows to show the suggested route and beacons to virtually show the destination.

The tour we developed is a preset queue of GPS coordinates which will take the user to points of interests to points of interest that a person would typically see on their visit to the university. We modeled some elements of it after the tours the university gives to prospective students and their families. Once the tour began, it would send the coordinate to our pathfinding algorithm which used the A* search algorithm given the destination and the user’s current GPS location. This would identify the efficient route to the destination and then draw chevrons on the ground to direct the user along the calculated route and a beacon that could be seen in the distance to visually identify the final destination. To do this, we first create nodes at all intersections between streets. Since we break streets that are not a straight line into pieces of line segments, we also create nodes at points where these line segments connect. Since the user’s current GPS location may not necessarily be on a street, we create the starting node by finding the closest point to a street. We do the same for the end node with the final destination. Since every node can be defined by two unique streets, we define neighbors of a node as one that share a street with each other. The heuristic function we chose is the Euclidean distance between the current node to the final destination. Once we find the path, we display the path on the ground as described earlier.

Once a user is close enough to their destination, we can detect the proximity the user is to the beacon and once detected will erase the current beacon and route, get the next destination on the tour, and send it to our pathfinding algorithm again.

To further augment the user’s experience, text and audio can be played at any time along the tour. In our implementation, building names were drawn on top of the virtual buildings and their sizes were proportionally larger the closer the user was to the building. Hence closer buildings could easily be read, while buildings further away wouldn’t have text
shown to distract the user. In the tour, audio was played at each destination when we detected that a user had arrived to their destination. This is used to enhance the experience with a virtual tour guide who provided information about the point of interest and particular buildings at the destination.

![Figure 5: The POWerNAV overlay in default mode. The black background interferes as little as possible with light from the real world.](image)

![Figure 6: The POWerNAV overlay with an activated tour. Note the destination beacon and chevron arrows on the ground.](image)

**RESULTS / MEASUREMENTS**

POWerNAV is a functional and accurate system that properly tracks user location and head orientation, displays the model world over the real world, and provides a working tour of the Penn campus, complete with virtual chevron arrows on the ground directing the user to the next destination. Figures 5 and 6 provide some sample screenshots of the POWerNAV overlay in action. However, there are several things that need improvement before a system like POWerNAV can be considered marketable to consumers by performing reliably. What follows is a discussion of a few of the most prominent performance shortcomings. (We discuss the implementation of some of our solutions to these shortcomings in the previous section, “Approach.”)

**(i) System Performance**

Alignment of the overlay tracks user head movement well, but tends to become rather inaccurate as the user is walking. One noticeable effect is the jostling of the Moverio headset that occurs when a user’s foot makes contact with the ground during walking. While this sudden minor shock should theoretically match the real world, the Moverio smartglasses are bulky and tend to rest rather loosely on the user’s head, causing the overlay to be jumpy. The smoothing effect of sensor fusion and gyroscope measurements lessen the impact of this effect.

While the street lock algorithm is necessary to even out lateral jumps in GPS data that would cause the overlay to be seriously misaligned, it does present an interesting problem at street intersections. If the user is locked to the wrong street, the overlay will also be misaligned. To combat this issue, we implemented a buffer that essentially does not switch streets even if a different one is closer, until a few GPS data points have come in on the new street. The results are good, but since we do not support curved pathways, there is potential lag when the path the user is on changes direction, especially when the GPS update rate slows.

If the user rotates his or her head quickly, the overlay may overshoot, due to integration errors from gyroscopic angular velocity measurements. The raw orientation from the accelerometer and compass slowly corrects this offset, but we were sure to use relatively slow head movements when recording demo videos.

We did some experimentation on GPS and sensor data accuracy, which is detailed in the next section, “Sensor Accuracy Measurements.” In summary, the GPS is relatively accurate when the receiver is still, but can move off by a considerable amount when the user is in motion. Accelerometer and gyroscope
data is acceptable; compass data suffers unpredictable accuracy shifts, especially upon application startup, but is usually consistent enough to be used in detecting changes in orientation yaw, as long as it is offset properly.

(ii) Sensor Accuracy Measurements

The sensors used in the Moverio are no different from the sensors traditionally found in smartphones: a gyroscope, magnetometer, accelerometer, and GPS. The first three sensors are utilized in conjunction with each other through a process known as sensor fusion in which to detect the orientation of the device as accurately as possible. Each sensor is known to have some errors and weaknesses and hence the goal of the sensor fusion is to eliminate some uncertainties by using the three in conjunction rather than individually, playing to the strengths of each device.

Collecting data about how accurate and precise these sensors was important to development by noting if the sensors on the Moverio had behavior similar or dissimilar to our Android phones. This allowed the team to identify whether certain issues with the sensor fusion were due to the inherent uncertainties within the sensors or due to our software implementation.

(a) GPS (while user is still)

We found that the Moverio GPS system had difficulty in obtaining a satellite lock and would at times take upwards of an hour to connect. We overcame this limitation by creating a companion application that would connect an android phone by Bluetooth to the Moverio and provide GPS data from the phone.

The phone models used were a Motorola Moto X (2nd Gen) model XT1049 and Samsung Galaxy S4 model SCH-I545Z. A set of trials were conducted when the user was walking with the phone and set of trials where the phone was placed on a table. This allowed us to perform circular error probable (CEP) and twice the distance root mean square (2DRMS) analysis of the GPS for the non-moving data to determine the precision of the GPS of systems used in the market. The data collected while walking was visualized[4] to measure how accurate the GPS was compared to the actual path taken.

CEP and 2DRMS analysis[5] is traditionally utilized to measure the precision of ballistics. CEP is defined as the radius of a circle, centered about the mean, whose boundary is expected to contain the landing points of 50% of all data collected. 2DRMS is also defined as the radius of a circle, centered about the mean, but its boundary is expected to contain 95% of all data collected. These values were calculated from longitude and latitude data by converting them into the Universal Transverse Mercator (UTM) coordinate system. This brought the data into a 2-dimensional Cartesian coordinate system.

The UTM Easting and UTM Northing values were used as x and y on the Cartesian coordinate system respectively. CEP is calculated by finding the standard deviation of the x and y, which can be done as follows:

$$
\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}
$$

Where \(\sigma\) is standard deviation, \(N\) is the total number of values, \(x_i\) a specific element of the data, \(\mu\) is the mean of the data.

CEP can then be calculated as follows:

$$
CEP = 0.59(\sigma_x + \sigma_y)
$$

Where \(\sigma_x\) is the standard deviation of x and \(\sigma_y\) is the standard deviation of y.

In our case, the CEP has a radius less than 1 meter. 2DRMS was calculated as follows:

$$
2DRMS = 2\sqrt{\sigma_x^2 + \sigma_y^2}
$$

Where \(\sigma_x\) is the standard deviation of x and \(\sigma_y\) is the standard deviation of y.

From our calculations, 2DRMS had a radius less roughly around 2 meters. From this data, we can confirm that GPS systems on
modern smartphones are very precise, where almost all GPS coordinates landed within a 2-meter radius from the mean of all the GPS coordinates collected.

<table>
<thead>
<tr>
<th>Phone</th>
<th>CEP in meters</th>
<th>2DRMS in meters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung trial 1</td>
<td>0.8643</td>
<td>3.6507</td>
</tr>
<tr>
<td>Samsung trial 2</td>
<td>0.7963</td>
<td>1.7727</td>
</tr>
<tr>
<td>Samsung trial 3</td>
<td>0.6644</td>
<td>2.1591</td>
</tr>
<tr>
<td>Motorola trial 1</td>
<td>0.7287</td>
<td>2.1468</td>
</tr>
<tr>
<td>Motorola trial 2</td>
<td>0.7742</td>
<td>2.4437</td>
</tr>
<tr>
<td>Motorola trial 3</td>
<td>1.7785</td>
<td>13.1283</td>
</tr>
</tbody>
</table>

Figure 7: CEP and 2DRMS calculations for each still trial on both phones.

Results from Motorola trial 3 were considered as an outlier and ignored for our analysis. Our raw data and calculations can be found in Appendix A.

(b) GPS (while user is walking)

The visualized GPS data allowed us to qualitatively identify the maximum distance the GPS error could be. At times we found them to be 10 or 20 meters off the actual path taken in the trial. Though the GPS sensor is found to be precise, it can be inaccurate at times. This directly led to our decision to lock the virtual eye of our app to roads to prevent the disorientation that inaccurate GPS data would cause, which would hinder the user’s augmented reality experience.

Links to the raw data, detailed calculations, and more visualized data can be found in Appendix A.

Figure 8: Visualization of collected GPS data.

Figure 9: Actual path taken.

Figure 10: Variances found for each sensor for each trial, still and walk.

Figure 11: Phone cube orientation before walking.
(c) Sensor Fusion: Gyroscope, Magnetometer, and Accelerometer

Each sensor’s value was recorded when the device was still on a table and when a user was walking while wearing the device. The variances of each sensor’s data collected was calculated to measure the uncertainty inherent in the sensors, on our still trials, and the uncertainty we would expect when a user is walking with the device on. Generally, we found that the sensors were accurate while still, but had a non-insignificant amount of variance when walking.

A link to the raw data can be found in Appendix A.

Qualitative tests were also performed by utilizing Alexander Pacha’s sensor fusion application[3] that demonstrated different sensor fusion algorithms with a cube that changes orientation using said algorithms. This was tested by having a user walk in a straight line with both the Moverio and phone with the app on for a minute. Before and after images of the orientation of the cube were analyzed between the Moverio and phones to determine if there were any differences between the sensors in the Moverio and the phone. Unfortunately, it turned out that both devices similarly suffered from compass inaccuracy; even though the device faced the same direction at the end of the walk as at the start, the cube had rotated a bit. We were unable to detect any large differences between the two devices from this analysis. This gave evidence that sensor fusion algorithms on phones could be used on the Moverio and that the sensors on the Moverio are comparable to those found in modern smartphones in their use. It also demonstrated that sensor inaccuracies were not being caused by our implementation.

More analyzed data can be found in Appendix A.
(d) Magnetometer Heading Stability

Understanding uncertainty in the stability of the magnetometer was important to determine the accuracy of the heading, or direction, the Moverio is facing at any one time. Any issues with stability, would directly limit the ability to seamlessly link the overlaid virtual world show in the Moverio’s viewport to the real world. In our sensor fusion algorithm, we utilize an arctangent of the pitch and roll to calculate an angle which informs the system what heading the user is facing. We collected data again with both the phone and Moverio undergoing the same motion and then compared the ratio to determine the stability of the Moverio and how stable it was compared to the stability of the phone.

Four separate trials were undergone in which the user was standing or walking and was holding their head still or slowly rotating it left and right. The ratio of the pitch and roll was graphed for all trials for each device and compared. Generally, there was little difference in the data between the walking and standing trials. There was a significant amount of instability at times when the devices were not rotated. When undergoing side-to-side rotation, the ratio for both devices undergoes a sine wave. The overall waves match, though the Moverio seems to have a higher peaks and lower valleys than that of phones, though this may be due to the way the Moverio measures the magnetometer values in comparison to the phone.

Graphs of all the data can be found in Appendix A.

ETHICAL / PRIVACY CONSIDERATIONS

Because POWerNAV provides a personalized experience, it analyzes a lot of data associated with the user. While none of this data is particularly sensitive, it does reveal certain types of information beyond public Google Maps data, which the user may not find desirable.

POWerNAV needs location data to properly determine virtual eye location, and also data on the user’s head orientation (i.e. where the user is looking and for how long) to determine virtual eye orientation. The data is constantly used to update these values and can be pieced together to determine what objects the user looks at, the user’s daily routine, and where the user has been. This type of location and behavior tracking is actually already very common among technology companies; Apple and Google have built in location history into their smartphone products, which is then used for a variety of purposes, including targeted advertising. The user’s “visual history” can be analyzed in a similar way.

User privacy is currently safe because POWerNAV only uses the data it collects to properly adjust the virtual eye; the data is then discarded. If this were to change, the user should be notified in an easy-to-understand privacy policy, as well as be given the opportunity to opt out of data collection. Even if the information is stored, it should be treated
with the same level of security as found at major
technology companies.

DISCUSSION

(i) Demonstration of AR’s Applicability

We have created a working application
that demonstrates that AR has a place in
navigation to provide a better experience. We
also created a fully functional campus tour
system that works well with our AR application.

(ii) Hardware Limitations

However, as seen in the results and the
appendix, we hit hardware limitations that
prevented us from creating the seamless
experience that we hoped to achieve for our
project. When walking, the sensors on the
Moverio had too much noise to keep the
orientation perfectly, and we showed that even
the state-of-the-art sensor fusion algorithms
cannot compensate for this noise. However, we
were able to learn about the different sensor
fusion algorithms out there, and the app works
very well if the user’s body stays still. We also
implemented a user-friendly way to refocus the
overlay correctly to overcome the hardware
limitations. Additionally, the noise in the GPS
data from the phones also forced us to lock the
view of the user onto the nearest street.
However, this assumption turns out to work well
in most cases.

(iii) Potential Further Development

If we had the time and resources, we
would like to try our application using better AR
devices, such as the Microsoft HoloLens, to see
if we can overcome the hardware limitations of
the Moverio BT-200 smartglasses. Additionally,
we would like to look at alternatives to using
sensor fusion algorithms to figure out the
orientation of the overlay. For example, an idea is
to track the user’s footsteps to know that the
user is moving, and then using assumptions
about the user to estimate how far the user has
traveled. We would also like to test our
application with test users, and compare how
quickly they can navigate with our application
versus a map application on a smartphone.

One notably absent feature from
POWERNAV is any usage of the camera to
perform computer vision analysis. We ultimately
decided that due to the expected use case –
outside, with few restrictions on movement – we
could not effectively improve the system with
basic computer analysis. However, we did do a
little research with OpenCV.

Optical flow is a measurement of the
velocity of pixels in a series of frames as
captured from a camera. As mentioned in the
“Virtual Eye and Display Engine” section, the
compass recenter algorithm only works when
the user is facing forward. If a user is facing
forward and moving forward, the optical flow
should measure expansion – that is, pixels move
away from the center of the screen. We
designed an OpenCV test application that, for a
set of pixels within each camera frame,
multiplies the velocity vector by the position
vector, with respect to the center of the image.
This value is positive when measuring expansive
movement, and non-positive otherwise. See
Figure 17.

In testing the application outside, we
found that the sensitivity of this method was not
properly tuned to a standard use case, where the
user’s head is always moving around slightly, and
distant objects only expand in the view very
slightly. We did not have the time to further
develop this OpenCV algorithm, but with more

Figure 17: The view is moving left, so right side is moving away
from the center, and the left side is moving towards the center.
The number represents the sum of all dot products, which was to
be used as an index in determining how likely it is that the user is
moving straight forward.
time we would like to fine-tune it to an outside environment.

There is so much potential in AR navigation that we did not get a chance to try yet, such as displaying general information about buildings or the ratings of restaurants from Yelp as the user gets close to them. There are also many potential applications in the indoor and automobile domains; for example, a system like POWerNAV can be used to provide an enhanced museum tour, or the overlay can be integrated into a car windshield to provide map data to an automobile driver, lowering the accident rate due to the driver focusing on a separate map instead of the road. We just touched the surface of what can be done, and we are excited to see future advancement in this area.

ACKNOWLEDGEMENTS AND CREDITS

We would like to thank our project advisor, Stephen H. Lane, for his assistance in the brainstorming and prototyping stages.

We developed on the Android platform, and also used the OpenGL ES 2.0 and OpenCV libraries. Additionally, public data from GPS and Google Maps is a fundamental part of POWerNAV.

We incorporated open-source code from the following sources: Sample Android code at developer.android.com (Apache 2.0 license); the Nobile font (SIL Open Font License 1.1); Texample2 text display library for OpenGL (CC0 1.0 public domain license); sensor fusion from Paul Lawitzki (MIT license); sensor fusion from Alexander Pacha (MIT license).

LINKS
POWerNAV repository: github.com/hsieha/CIS400Project
[1]: github.com/d3alek/Texample2
[2]: codeproject.com/Articles/729759/Android-Sensor-Fusion-Tutorial
[3]: bitbucket.org/apacha/sensor-fusion-demo
[4]: blog.oplopanax.ca/2012/11/calculating-gps-accuracy/
[5]: dtic.mil/dtic/tr/fulltext/u2/a199190.pdf
Appendix A

GPS Still Data

Link to data:
https://docs.google.com/spreadsheets/d/159IVzBiLNln_R3hGN36cK9IE5eSaNkfGi2UsCut7-Mk/edit?usp=sharing

<table>
<thead>
<tr>
<th>Phone</th>
<th>Lat Variance</th>
<th>Long Variance</th>
<th>Lat Stdev</th>
<th>Long Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung trial 1</td>
<td>2.60E-10</td>
<td>1.72E-11</td>
<td>1.61E-05</td>
<td>4.14E-06</td>
</tr>
<tr>
<td>Samsung trial 2</td>
<td>5.16E-11</td>
<td>2.06E-11</td>
<td>7.18E-06</td>
<td>4.54E-06</td>
</tr>
<tr>
<td>Samsung trial 3</td>
<td>9.11E-11</td>
<td>5.91E-12</td>
<td>9.54E-06</td>
<td>2.43E-06</td>
</tr>
<tr>
<td>Motorola trial 1</td>
<td>4.38E-11</td>
<td>8.40E-11</td>
<td>6.62E-06</td>
<td>9.17E-06</td>
</tr>
<tr>
<td>Motorola trial 2</td>
<td>5.10E-11</td>
<td>1.18E-10</td>
<td>7.14E-06</td>
<td>1.09E-05</td>
</tr>
<tr>
<td>Motorola trial 3</td>
<td>1.05E-09</td>
<td>4.12E-09</td>
<td>3.25E-05</td>
<td>6.42E-05</td>
</tr>
</tbody>
</table>

Figure A-1: Variance and standard deviation of each trial of GPS data collected.

<table>
<thead>
<tr>
<th>Phone</th>
<th>Avg UTM Easting</th>
<th>Avg UTM Northing</th>
<th>UTM Easting Stdev</th>
<th>UTM Northing Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung trial 1</td>
<td>2.60E-10</td>
<td>1.72E-11</td>
<td>1.61E-05</td>
<td>4.14E-06</td>
</tr>
<tr>
<td>Samsung trial 2</td>
<td>5.16E-11</td>
<td>2.06E-11</td>
<td>7.18E-06</td>
<td>4.54E-06</td>
</tr>
<tr>
<td>Samsung trial 3</td>
<td>9.11E-11</td>
<td>5.91E-12</td>
<td>9.54E-06</td>
<td>2.43E-06</td>
</tr>
<tr>
<td>Motorola trial 1</td>
<td>4.38E-11</td>
<td>8.40E-11</td>
<td>6.62E-06</td>
<td>9.17E-06</td>
</tr>
<tr>
<td>Motorola trial 2</td>
<td>5.10E-11</td>
<td>1.18E-10</td>
<td>7.14E-06</td>
<td>1.09E-05</td>
</tr>
<tr>
<td>Motorola trial 3</td>
<td>1.05E-09</td>
<td>4.12E-09</td>
<td>3.25E-05</td>
<td>6.42E-05</td>
</tr>
</tbody>
</table>

Figure A-2: UTM Easting and Northing conversion averages and standard deviation.
Figure A-3: Scatterplot of GPS positions of Samsung Trial 1.

Figure A-4: Scatterplot of GPS positions of Samsung Trial 2.
Figure A-5: Scatterplot of GPS positions of Samsung Trial 3.
Figure A-6: Scatterplot of GPS positions of Motorola Trial 1.

Figure A-7: Scatterplot of GPS positions of Motorola Trial 2.
Figure A-8: Scatterplot of GPS positions of Motorola Trial 3.
GPS Walking Data Visualization

Trial 1 - Motorola

Figure A-9: Visualization of collected GPS data. Color of the line represents the altitude and can be ignored.

Figure A-10: Actual path taken. Red marker is the start, and blue marker is the end.
Figure A-11: Visualization of collected GPS data. Color of the line represents the altitude and can be ignored.

Figure A-12: Actual path taken. Red marker is the start, and blue marker is the end.
Trial 3 - Samsung Phone

Figure A-13: Visualization of collected GPS data. Color of the line represents the altitude and can be ignored.

Figure A-14: Actual path taken. Red marker is the start, and blue marker is the end.
Trial 4: Samsung Phone

Figure A-15: Visualization of collected GPS data. Color of the line represents the altitude and can be ignored.

Figure A-16: Actual path taken. Red marker is the start, and blue marker is the end.
Trial 5: Samsung Phone

Figure A-17: Visualization of collected GPS data. Color of the line represents the altitude and can be ignored.

Figure A-18: Actual path taken. Red marker is the start, and blue marker is the end.
Sensor Fusion: Gyroscope, Magnetometer, Accelerometer

Link to raw data:
https://docs.google.com/spreadsheets/d/1qOfyw1K5vlQHRUuIqW5umsWY2IiYdBnt_enV72C_6ls/edit?usp=sharing

Magnetometer Heading Stability

Link to raw data:
https://docs.google.com/spreadsheets/d/1WMmH6Z3s385NS6hZmqN_GIpn4MFNdkYlaRbs4SFEKFE/edit?usp=sharing

Trial 1 - Still, no rotation

Figure A-19: Magnetometer heading ratios of Motorola phone, still with no rotation.
Figure A-20: Magnetometer heading ratios of Moverio, still with no rotation.

**Trial 2 - Walking, no rotation**

Figure A-21: Magnetometer heading ratios of Motorola phone, walking with no rotation.
Figure A-22: Magnetometer heading ratios of Moverio, walking with no rotation.

**Trial 3 - Still, side-to-side rotation**

Figure A-23: Magnetometer heading ratios of Motorola phone, still with side-to-side rotation.
Figure A-24: Magnetometer heading ratios of Moverio, still with side-to-side rotation.

Trial 4 - Walking, side-to-side rotation - April 10th

Figure A-25: Magnetometer heading ratios of Motorola phone, walking with side-to-side rotation.
Figure A-26: Magnetometer heading ratios of Moverio, walking with side-to-side rotation.