Accessible Positional Tracking for Mobile VR

Hassan Karaouni and Aashna Mago
Stanford EE 267: Virtual Reality, Instructors: Gordon Wetzstein and Robert Konrad

Abstract

Positional tracking at the lowest end of VR is crucial to showing mass markets how powerful and far-reaching a tool virtual reality technology will be. The following outlines the method and difficulties of accelerometer based navigation as well as the promise of step detection and RSSI sensors for future work.

1 Introduction

The prospect of democratizing the VR space and making the technology more widely accessible is very exciting. Low-cost mobile positional tracking can substantially improve the quality of VR experiences available to the average person. Mobile positional tracking will also enable non-enthusiasts to create and experience a more diverse range of applications.

We opted to begin approaching this problem through accelerometer-based tracking for three reasons. First, we wanted to add positional tracking as a feature without introducing cumbersome or expensive new hardware. Our aim is to implement mobile positional tracking using sensors that are already on the phone as much as possible and where needed, make use of devices that are already in the average home or that can be easily and cheaply mailed with a Google Cardboard headset.

Second of all, we wanted to understand the details of why accelerometers alone are inadequate for implementing high-fidelity positional tracking. It's common knowledge and oft-discussed that double integration of accelerometer data isn't "good enough," but we wanted to understand what that meant and qualify the that statement. How bad is the data? For how long is it reliable?

Lastly, we wanted to approach the problem from the vantage point of the Stanford EE267 course and best take advantage off the knowledge and skills we've gained in this course. Just having implemented IMU sensor fusion using the Arduino and a Mattel headset, we felt primed to take the next step and tackle positional tracking with similar hardware.

A low-cost solution to the mobile positional tracking problem is important to our team, so we particularly aimed to explore positional tracking through use of a constrained model or additional sensors that are relatively light and affordable, like Estimote Beacons, to mitigate issues like accelerometer noise and the resultant drift.

2 Related Work

Accelerometer-based positional tracking is a frequently considered approach to the problem we're tackling, especially by newcomers to the space. We identified a number of blogs and posts on forums and Stack Overflow from individuals who have previously considered or attempted accelerometer-based tracking, but we were surprised to find that there were many inconsistencies in the advice given online. There was no consolidated or well-documented approach that conclusively outlined the advantages and shortcomings of relying on accelerometer data, which reinforced our motivation to systematically test this approach, identifying and quantifying its limitations and attempting to overcoming them through additional sensors.

A paper published by Lavalle et al. at Oculus VR in 2014 for IEEE outlines Oculus' approach to orientation tracking in VR and gave us a sense of how tracking is being approached in industry and some of the barriers to positional tracking that have already been identified. We watched a Google Tech Talk by David Sachs entitled "Sensor Fusion on Android Devices: A Revolution in Motion Processing," which gave us insight into Android sensors and the provided API as well as how we might handle sensor fusion.

We also read several papers related to the topic of sensor fusion for mobile positional tracking, including "Ambulatory Position and Orientation Tracking Fusing Magnetic and Inertial Sensing" by Roetenberg et al., another IEEE paper which focused on using a complementary Kalman filter to combat integration drift from inertial sensors. "Implementing Positioning Algorithms Using Accelerometers" by Seifert & Camacho was very useful in helping us hone our algorithm for determining position from accelerometer readings and adding filtering mechanisms for noise. The most impactful paper that we came across and utilized was "Indoor Positioning using Sensor-fusion in Android Devices" by Shala & Rodriguez, which surveys multiple sensors on an Android phone and works to determine the optimal sensor fusion algorithm for positional tracking using these onboard sensors. We designed our methods based on those described in this paper in an attempt to replicate and improve upon their findings.

3 Approach

We decided to do our work on an Android mobile phone for several reasons. First of all, we began this project with accessibility and applicability in mind, so we chose a device that was Cardboard compatible and commonly owned by non-enthusiasts. Additionally, we noticed that

Android had an API that included an assortment of documented virtual sensors, such as linear acceleration and step detection, that we could take advantage of.

As mentioned above, we decided to closely follow the Shala & Rodriguez paper since they had implemented and assessed tracking with many of the same sensors that we planned to evaluate. Our aim was to repeat experiments that Shala & Rodriguez had performed in addition to introducing new external devices like Estimote beacon stickers and, time permitting, trying slightly modified approaches to sensor fusion.

We planned to begin our work by implementing a rough and likely unusable version of positional tracking using linear acceleration data. We expected this to have drift, but wanted to use it as a starting point from which we could work to add more precision and accuracy. Beyond that, we wanted to explore constraining accelerometer-based tracking with step detection and WiFi RSSI (signal strength). One of the major issues with double integrating accelerometer data for positional tracking is that readings never really return to zero because of noise — we were eager to try using other sensors to find a "ground truth" to return to periodically and determine the duration of time for which the accelerometer could serve as a reliable source of position (between ground truth check-ins).

3.1 Inertial Navigation System Methods

Our first step in implementing the accelerometer-based tracking was to calculate inertial acceleration of a ViewMaster headset using an external IMU and Arduino. We utilized the complementary filter we had previously written for determining orientation using accelerometer and gyroscope data. From there, we referred to the CHRobotics approach to filtering out the influence of gravity and getting inertial acceleration (unlike the CHRobotics source, our coordinate system had the y-axis pointing up, x-axis to the right, and z-axis coming out of the page). We used the equation below, which is pulled from CHRobotics' site, to rotate the raw accelerometer data (a_m) from the body frame to the inertial frame (a_I), subtract the gravity vector, and then rotate back into the body frame.

$$\mathbf{a}_I = R_B^I \mathbf{a}_m + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

Figure 1. Equation for rotating acceleration vector from body frame to inertial frame and subtracting influence of gravity to get inertial acceleration. From CHRobotics website.

We then switched over to working with the virtual linear accelerometer on the Android phone, which internally handles subtracting the gravity vector, in the hopes of having more reliable sensors. Once we were working with the Android phone, we did our tests as a native Android app for Cardboard, using the Cardboard demo app from Google as a virtual world in which to test our tracking. Throughout our work, we printed raw sensor data to the phone screen to debug. While working with the linear accelerometer, we output data from the gravity virtual sensor and accelerometer physical sensor to make sure that the readings were aligned with what we expected.

Our first attempt to implement tracking on Android consisted of a basic algorithm using double integration of accelerometer data over varying time steps depending on when changes in acceleration were detected. We later tried many potential solutions to filtering out and mitigating errors from accelerometer sensor noise. We performed a calibration step when first launching the app where the phone was left stationary for 1000 readings and the average noise detected along each axis was used as an offset for the remainder of the readings.

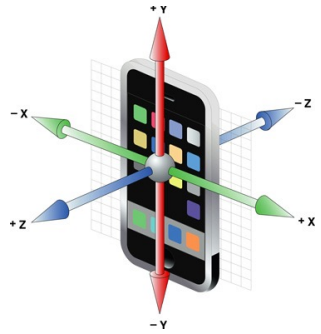

Figure 2. Coordinate system on Android phone, which has the axes rotated compared to the OpenGL convention that we used in the virtual world. From Stack Overflow: http://stackoverflow.com/questions/11175599/how-to-measure-the-tilt-of-the-phone-in-xy-plane-using-accelerometer-in-android

In Oculus' paper on head-tracking, the authors discuss and justify the use of kinematic constraints in their model for a seated user. We similarly decided to constrain our model based on realistic movements. We added a threshold value for acceleration. If acceleration along any axis was below this threshold for multiple consecutive readings, it was determined to be noise, and we reset accelerometer and velocity to zero. In addition, we later tried utilizing step detection as part of the same concept of kinematic constraints.

After reading the positional tracking paper by Seifert & Camacho, we used the trapezoidal method of linear approximation to reduce error in our double integration from acceleration to position. We also tried keep the time between samples constant as suggested in the paper. Lastly, we attempted to average accelerometer values and get a "moving window" of values to reduce the effect of noise on our calculations.

3.2 Step Detection

The challenge of positional tracking has been widely

researched across multiple fields. We were thus eager to consider approaches which, even if ineffective for most fields, might be uniquely effective for virtual reality. One such approach was positional tracking based on step detection. The concept of step detection is to use an accelerometer to detect steps taken by a user. The number of steps taken can be combined with a pre-calibrated average step length to translate the user in virtual space.

Past efforts at step detection have shown reasonable accuracy but demonstrate the importance of the pre-calibration phase [1]. For example, the same pre-calibrated average step length may not be the same across users. Furthermore, the use of step detection for positional tracking suffers if a given user varies step length for any reason after calibration. This is a common problem for users in the real world as users may walk at varying speeds to accomplish different objectives. Yet this also provides a potentially unique opportunity for the virtual reality space because current users of head-mounted displays may not display the same range of step types as users in the real world.

Multiple approaches to step detection have been taken in the past [1][3] and a very common, lightweight approach is to simply use the magnitude of acceleration to detect if a step has been taken.

$$Magnitude_{Acc} = \sqrt{Acc_x^2 + Acc_y^2 + Acc_z^2}$$

Conceptually, the approach guesses that the user has taken a step if a large amount of acceleration occurs for any reason. Shala & Rodriguez implemented such an approach with success. Specifically, it was found that a step tends to coincide with an acceleration magnitude above 12.5 m/s^2 (with acceleration due to Earth's gravity included in the accelerometer readings). However, it was also noted that a single step could detect multiple spikes in acceleration magnitude and it was thus crucial to enforce a threshold of at least 350 ms between steps.

Rather than implement a step detection algorithm, the final approach determined was to use the built-in step detection sensors of Android phones. Android's step detector sensor would be utilized to detect each time a user took a step. The use of a step counter based on the phone was appealing because, if possible, the technique would represent no additional cost to users of mobile virtual reality applications.

4 Analysis

The following outlines the results of positional tracking using the accelerometer and a step detection sensor. To summarize, accelerometer data suffers from the issues of decoupling gravity and extensive noise while current step detection sensors have too much lag to be able to solve the problem of positional tracking alone. A combined solution is promising and outlined in the Discussion and Future Work section.

4.1 Inertial Navigation System Results

Our biggest learning from this project was that accelerometer data alone is, in fact, "not good enough" for the applications we are targeting. As described in our methods, we had originally planned to quickly implement accelerometer-based tracking as a noisy baseline on which to build with additional sensors and constraints. We soon realized that we had grossly underestimated the difficulty of working with such a noisy sensor, and spent the bulk of our time tweaking our double integration algorithm and adding filters to try to make sense between noise and true acceleration values.

When we did our initial tests and obtained inertial acceleration from the standalone IMU and Arduino, we were already able to see how substantial drift and error from an accelerometer can be. In fact, for some time, it was difficult to distinguish between accelerometer noise and an incorrect algorithm for subtracting the gravity vector and extracting inertial acceleration. We saw that these errors were primarily due to the yaw values output by our complementary filter. This was unsurprising, since we know that the accelerometer cannot sense rotation around the y-axis and does not contribute to calculation of yaw. We considered implementing yaw correction on the IMU, but didn't feel this was necessary or important when we switched over to Android.

We have not yet been able to implement rudimentary positional tracking on an Android phone using only the accelerometer. Other than prohibitive drift from the accelerometer due to errors adding up astonishingly quickly, a major issue we faced was the inconsistency in performance of the sensor. We designed an algorithm to filter out noise by assigning conditions to guess when there was no movement. Under these conditions (described in previous section), we reset velocity and acceleration to zero. While the addition of this constraint was our most observably successful to control drift, we couldn't stably set a threshold for the noise level between builds because the noisiness of the accelerometer seemed dependent on so many external factors. Sometimes our threshold ended up so high that no movement was detected at all, and other times, our sensor was so sensitive that the user would shoot far into the distance in the virtual world just by leaning forward slightly in the physical one.

Additionally, it was surprisingly difficult to test our positional tracker since we weren't confident about how scale was measured in the virtual world. This sometimes made it difficult to distinguish between drift and values that were simply too large for the scale of the virtual world and made the user move too far but would make physical sense if they were linearly scaled down. We were unsure when

starting out about whether we needed to utilize a rotation matrix to convert from the body frame (of the phone) to the world frame (the openGL coordinate system convention) or whether remapping the acceleration components would be sufficient. Ultimately, we remapped the axes so that the y-acceleration from the sensor became our world's negative x-acceleration and the phone's x-acceleration became our world's y-acceleration.

The implementation of the linear acceleration sensor on Android doesn't appear to use any sort of complementary filter for orientation when subtracting the gravity vector. When we realized this, we began testing with the raw accelerometer data from the phone as well, with the intention of implementing our own complementary filter later on. We have not yet implemented this, so while testing our positional tracker using accelerometer data, our elimination of the influence of gravity was less than optimal. This "decoupling" from gravity was done simply using the calibration step mentioned in the previous section to determine the average noise level of the sensor along each axis and subtracting this average from all subsequent readings. For the most part, this removed the gravity vector since we kept the phone with its screen perpendicular to the floor (as if in a Google Cardboard headset) while testing. Of course, we weren't always holding the phone perfectly straight, and this wouldn't work in real applications, but we found that it was sufficient for testing and saved us the time of implementing the complementary filter while we focused on getting the higher level accelerometer-based tracking working.

We determined that aside from removing the influence of gravity when using the raw accelerometer, our calibration step didn't make much sense given our sensor and its limitations. Averaging the values measured from the accelerometer when at rest and using the average as an offset would make sense if the sensor had a strong bias that was more or less constant. This would be appropriate for a sensor like a gyroscope. The accelerometer, however, is noisy, and this method is less effective for noise. Focusing on setting a threshold for noise seemed to be a more successful approach and we would like to fine-tune this in future implementations.

4.2 Step Detection Results

Although step detection's potential for positional tracking remain promising in theory, it was found that using step detection from the Android sensor was too inaccurate in practice to warrant the implementation of such a solution.

First and most importantly, it was found that the Android step detection sensor suffers from a large lag in step detection. A number of steps must be taken before the the step detection sensor is able to detect any steps, as shown in the following figure. Although the sensor is very reliable after it starts step detection, a lag of even a few steps would make the sensor unfit for virtual experiences as the resulting positional offset would be on the order of meters.

The second issue with the step detection sensor of the Android phone is that there is a clear predilection for step detection based on the phone being placed in a pocket. When oriented on a head-mounted display, the phone's sensors took slightly longer to start registering steps. Such a result was unexpected but also unsurprising in retrospect. The Android step detection sensor seems optimized for the use case of a phone swinging in a pocket, a situation in which the magnitude of acceleration is far larger than when the phone is fixed to a head-mounted virtual reality viewer.

The following figure provides data for analysis of the step detection sensor in the Android phone:

	Head-Mounted Orientation	Pocket Orientation
Steps Needed Before Detection	7.7	6.3

Figure 3. Steps taken before the Android sensor detected steps accurately. The user took 20 steps in a straight line at approximately 1 m/s, and results are averaged across 20 trials each.

The figure demonstrates the average number of steps taken before the Android sensor could detect steps reliably. In the head-mounted orientation, the phone was placed in the orientation of a phone in a Cardboard viewer. In the pocket orientation, the phone was placed in the user's pockets. Once the sensor began to detect steps in either orientation, the sensor accurately detected every following step. There were no false positives or negatives.

The results demonstrate that, on average, the Android sensor is slightly better at detecting steps for the much more common use case of a phone placed in a pocket. More importantly, the results show that the phone's step sensor is inadequate for independently determining position. Even the lower lag of 6.3 steps before accurate step detection would be unfit for a user in a virtual world.

As a final note, it is important that the step testing was completed at approximately 1 m/s. Although this may not be a suitable assumption in most use positional tracking cases, it is fitting for the current virtual reality ecosystem in which users are not moving at high speeds.

5 Discussion and Future Work

The attempts at using phone-based inertial navigation and step detection for positional tracking not only clarified the advantages and disadvantages of each method, but also provided insight on the types of solutions which are promising for future work.

Use of accelerometers to aid in positional tracking remains promising due to the ubiquity and low cost of accelerometers, but the work also supports that low-cost accelerometers alone are far from sufficient for the

challenge. The work supported the conclusions of prior research: attempts to use a low-cost accelerometer suffer from issues related to gravitational force, sensor noise and the resulting inaccuracy of double integration. Without the use of costly orientation sensors, attempts to decouple gravity from accelerometer readings will be inaccurate. Prior work even suggests that orientation errors when decoupling gravity from the accelerometer may dwarf the errors caused by the accelerometer itself [4]. The inaccuracy of decoupling gravity combined with the noise inherent to low-cost accelerometers strongly support the common conclusion that positional tracking will not be achievable with the method of solely using an accelerometer. We now feel a strong understanding of the limitations of accelerometers and, although we were unable to use the accelerometer for basic tracking in this project, we are excited to add in more sensors to solve the problem in future work.

Although the attempt to solely use an accelerometer for positional tracking failed, the attempt informed directions for how future work may solve the problem. First, it is useful to know when the user of the tracking system is and is not moving. A step detection sensor could be very useful for such a purpose. For instance, if it is detected that the user is not moving using a step detection algorithm, the negatives effects of noisy accelerometer data could be reduced. Velocity and acceleration could be reduced to zero when a user is at rest and recalculated as soon as a significant acceleration is detected.

Unfortunately, current phones are far from reliable step detection sensors and would require a number of changes. The easiest would be to optimize existing algorithms for head-mounted phone orientations instead of the pocket orientations which have been most relevant in the past. Yet more importantly, phone step detection sensors would need to be able to detect steps more quickly. The delay of multiple steps shown in the Analysis section make existing step detection sensors in phones unfit for substantially contributing to positional tracking. Future step detection sensors must reduce the lag in recording steps, or be used as only a partial contribution to the positional tracking problem. One potential solution may be to place sensors on a user's shoes and stream data to a phone, but the resulting system may not be low-cost. Until better step detection sensors are realized, attempts to use phone-based step detection sensors for positional tracking should only expect the sensors to have a minor role in tracking.

A promising direction for future work would be the combination of inertial navigation, step detection sensors, and received signal strength indicator (RSSI) methods into a composite positional tracking system. RSSI is a measurement of the power present in a signal received by a phone. Prior work has shown that using WiFi RSSI alone is too noisy for exact positioning [2], but it is promising to consider how the method could be combined with step detection to provide ground truth to inertial navigation systems. Future work should consider the possibility that WiFi RSSI could provide a reasonable estimate of if a user is moving while a step detection sensor is still calibrating to a user's steps. Then, when the step detection sensor has moved past the point of lag, the WiFi contribution could be weighted less than the step detection sensors to the overall system. Research on WiFi RSSI shows that one problem with the approach is the need to pre-calibrate a space [2] and the fluctuations of building-based WiFi signals. Future work should thus also consider the use of more proximal signals, such as Estimote beacons or other low-cost systems.

References

1] YUN, X., BACHMANN, E., MOORE, H., AND CALUSDIAN, J. 2007. Self-contained Positional Tracking of Human Movement Using Small Inertial/Magnetic Sensor Modules. In *IEEE International Conference on Robotics and Automation*. IEEE Xplore.

[2] SHALA, U. AND RODRIGUEZ, A. 2011. Indoor Positioning using Sensor-fusion in Android Devices. Kristianstad University. School of Health and Society. Department Computer Science.

[3] PAN, M. AND LIN, H. 2015. A Step Counting Algorithm for Smartphone Users: Design and Implementation. In *IEEE Sensors Journal*.

[4] CH ROBOTICS. Using Accelerometers to Estimate Position and Velocity. Link to reference: http://www.chrobotics.com/library/accel-position-velocity

[5] LAVALLE, S. M., YERSHOVA, A., KATSEV, M., AND ANTONOV, M. 2014. Head tracking for the Oculus Rift. In *IEEE International Conference on Robotics And Automation*. IEEE Xplore.

[6] SEIFERT, K. AND CAMACHO, O. 2007. Implementing Positioning Algorithms using Accelerometers. From NXP Semiconductors.

[7] SACHS, D. 2010. Sensor Fusion on Android Devices: A Revolution in Motion Processing. From Google Tech Talks.

[8] ROETENBERG, D., SLYCKE, P. J., AND VELTINK, P. H. 2007. Ambulatory Position and Orientation Tracking Fusing Magnetic and Inertial Sensing. From *IEEE Transactions on Biomedical Engineering 54:5*.