Rendering for Positional Tracking

Heather Shen
Stanford University
Department of Electrical Engineering
hcshen@stanford.edu

Jason Liu
Stanford University
Department of Computer Science
liujas00@stanford.edu

Abstract

Current virtual reality (VR) models rely heavily on head tracking to relay a users change in motion. However, to provide more realistic experiences or any augmented reality experience, VR environments must incorporate position tracking. In this paper, we implement rendering with position tracking to test the fused experience of using both position and orientation tracking. With IR beacons and photodiodes, we use the Levenberg-Marquardt algorithm to estimate user position and translate that into virtual space. Although there were several limitations with hardware and the IR beacons, users generally perceived the combined system to be more seamless to interact with.

Keywords: virtual reality, position tracking, orientation tracking, Levenberg-Marquardt, sensor fusion

1. Introduction

Within the past few years, the virtual reality (VR) market has exploded and is expected to expand more than sevenfold within five years.[1] With cheaper, better phone displays and more accurate sensors, VR is becoming increasingly feasible and affordable. However, VR environments are still not completely immersive: many current VR models have not yet implemented positional tracking and instead can only track the movement of the head. Position-tracking VR is increasingly important for realistic experiences as well as any augmented reality experience.[3] Rendering with position-tracking provides increased opportunities for the user the interact with the virtual world: for example, the user viewpoint can more realistically reflect actions such as jumping, ducking, or leaning forward.

Even more pressing is the motion sickness often associated with VR. Motion sickness is produced by conflicting inputs from visual, vestibular and somatosensory afferents.[4] When the head-mounted display doesnt move with the users body, it produces ambiguous information about self-motion, which can lead to motion sickness and other problems.

Thus, to create a more immersive environment and to address the potential for motion sickness, we plan to integrate position tracking with rendering for more realistic motion in virtual space. Using the Levenberg-Marquardt algorithm to estimate the users position, we can perform sensor fusion to combine position and orientation tracking. A VR demo will be prepared to test the combined tracking to gauge the potential of this algorithm while identifying potential shortcomings.

2. Related Work

Currently, positional tracking is divided into two different problems. In inside-out positional tracking, the camera/sensor is located on the HMD, and no external devices are needed for tracking. In outside-in positional tracking, external cameras and sensors are required, which constrains the tracking experience to a specific area. The former problem is currently being researched more, as it as more impactful applications to the field of VR/AR. Last year, in 2016, at a Google press event, Johnny Lee of Project Tango stated that inside-out positional tracking had been solved for the smartphone; the current limitations were that it was too computationally expensive and the phones would get too hot.[2]

Our current lectures demonstrate a camera problem where, based on the number of unknowns, having four photodiodes on a plane makes it possible to fully determine a solution for the camera matrix. Raharijaona et. al, proposed a different model that involves three photodiodes being placed on different sides of a small cube. Using this setup, and the knowledge that sensors were all 90° apart, they were able to determine the azimuth and elevation of flickering IR LEDs. While their setup involved tracking remote markers, one could imagine this setup being adapted so that the cube localizes its own position.[7] [6] In 2008, Thomas Petersen of Aalborg University performed a study in which he compared various 2D-3D pose estimation methods.[5] The LM algorithm we use is an extension of the Gauss-Newton method, which can also be used for pose estimation, but is less robust. PosIt is a method published by DeMenthon

et. al in 1995 that finds a pose of an object from an image. The method is split in 2 where the first part approximates the perspective projection and finds the rotation matrix and translation vector from a linear set of equations. The second part iteratively uses a scale factor for each point to enhance the found orthographic projection and then repeats on the new points until some convergence. Another method, called CamPoseCalib (CPC) is used to estimate the relative rotation and translation of an object from an initial pose to a new pose. It used the LM algorithm in its optimization, and only requires 3 points of correspondence to work.

2.1. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt Algorithm is an iterative solution takes an optimization approach towards the problem of pose estimation. It goes by the following:

```
x = rand()
for k=1 to max_iter
    f = eval_objective(x)
    J = eval_jacobian(x)
    x = x+inv(J*J+lambda*diag(J*J))*J*(b-f)
```

In the above, x is our pose estimation, f is the error we get when we project our estimation back into two-dimensional space, and J is the Jacobian matrix.

In our case, we split the projection of x into two steps. First, we map it to a homography as shown in Figure 1. Then we mapt the homography to two-dimensional estimates as shown in Figure 2.

$$
\begin{aligned}
h_1 &= \cos(\theta_y)\cos(\theta_z) - \sin(\theta_x)\sin(\theta_y)\sin(\theta_z) \\
h_2 &= -\cos(\theta_x)\sin(\theta_z) \\
h_3 &= t_x \\
h_4 &= \cos(\theta_y)\sin(\theta_z) + \sin(\theta_x)\sin(\theta_y)\cos(\theta_z) \\
h_5 &= \cos(\theta_x)\cos(\theta_z) \\
h_6 &= t_y \\
h_7 &= \cos(\theta_x)\sin(\theta_y) \\
h_8 &= -\sin(\theta_x) \\
h_9 &= -t_z
\end{aligned}
$$

Figure 1. Converting LM estimate to homography

$$
f(h) = \begin{pmatrix} f_1(h) \\ f_2(h) \\ \vdots \\ f_7(h) \\ f_8(h) \end{pmatrix} = \begin{pmatrix} \frac{h_1 p_{1,x}^{3D,o} + h_2 p_{1,y}^{3D,o} + h_3}{h_7 p_{1,x}^{3D,o} + h_8 p_{1,y}^{3D,o} + h_9} \\ \frac{h_4 p_{1,x}^{3D,o} + h_5 p_{1,y}^{3D,o} + h_6}{h_7 p_{1,x}^{3D,o} + h_8 p_{1,y}^{3D,o} + h_9} \\ \vdots \\ \frac{h_1 p_{4,x}^{3D,o} + h_2 p_{4,y}^{3D,o} + h_3}{h_7 p_{4,x}^{3D,o} + h_8 p_{4,y}^{3D,o} + h_9} \\ \frac{h_4 p_{4,x}^{3D,o} + h_5 p_{4,y}^{3D,o} + h_6}{h_7 p_{4,x}^{3D,o} + h_8 p_{4,y}^{3D,o} + h_9} \end{pmatrix}
$$

Figure 2. Converting homography to 2-D estimate

2.2. Quaternions

Quaternions are a four-dimensional number system that can be used to describe three-dimensional rotations. We show the relationship between quaternions and euler angles (roll, pitch, yaw) in Figure 3 and Figure 4.

$$
\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \text{asin}(2(q_0 q_2 - q_3 q_1)) \\ \text{atan2}(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}
$$

Figure 3. Converting Quaternion angles to Euler angles

3. Methods

To provide an original reference position, the algorithm we use performs a calibration step before rendering begins. During this period, the user remains as still as possible and the algorithm calculates the average position and variance using the LM algorithm. Thus, we can set the IR beacon as the origin of the rendered scene and support translation from the original position using the translation values from the positional tracking. Also, by calculating the variance, the algorithm accounts for some of the noise and unintended motion inherent in positional tracking. Although we

$$\mathbf{q}_{\mathbf{IB}} = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix}$$
$$= \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}$$

Figure 4. Converting Euler angles to Quaternion angles

considered using LM angular calculations to define the initial viewer rotation relative to the IR beacon, we ultimately decided against this addition to the algorithm because the viewer must remain in the line of sight of the beacon regardless. We reasoned that the original viewer position should be directly facing our virtual object to best reflect the idea of line of sight towards the beacon.

We perform another calibration step on the Inertial Measurement Unit (IMU) data. During this phase, we set the IMU down and let it collect measurements for a given number of steps. From that, we calculate a bias on the gyroscope outputs across the three axes. We use this bias to zero-mean the gyroscope measurements to calculate our self-rotation. In addition, we have the Arduino set up such that if a key is pressed, the quaternion representing our orientation will be reset.

After the two calibration steps, we programmed our Arduino to output both positional and quaternion calculations. If the Arduino is unable to complete the positional calculation (ie because of occlusion of photodiodes), then our Arduino outputs quaternion data instead of both the position estimate and quaternion data. This way, we always output information and increase information throughput, since the orientation data can be calculated independently of the IR beacons.

A Node server streams the data from the Arduino, which is then fed to a THREE.JS rendering scene. We use the positional estimates from the LM output for our viewer position matrix, and we use the quaternion data from our IMU to calculate our viewer rotation matrix. Combining these two sources of orientation helps constrain orientation and reduce noise.

4. Evaluation

The Levenberg-Marquardt algorithm is not the only algorithm that can be used to solve for position. One can also calculate this by using the homography matrix. Previously we performed a test on the residual between the predicted value and actual location using both formulas. The Homography approach resulted in residuals of around 10^{-13}, while the Levenberg-Marquardt approach had at best an error of 10^{-8}. From this, we would conclude that the Homography matrix is a better candidate for calculating position. However, this is all due to the fact that the Homography approach makes no guarantees on the other values it calculates being valid rotations, while this is so in our LM approach. While we did not end up using the rotational output of the LM algorithm in our paper, this would be the next step, and motivates why we chose to use the LM algorithm in the end.

	Lambda = 0.001		**Lambda = 0.1**		**Lambda = 1.0**	
	Avg residual	Residual of output	Avg residual	Residual of output	Avg residual	Residual of output
kMaxIter = 1	0.08496	0.012902	0.18579	0.019567	0.14953	0.045782
kMaxIter = 10	0.031299	8.2409e-08	0.026544	6.2342e-05	0.024092	0.0001269
kMaxIter = 25	0.010593	2.1346e-08	0.013997	4.3736e-05	0.0090956	0.00013417

Figure 5. Residuals over different combinations of lambda and number of iterations

The Levenberg-Marquardt algorithm has two hyperparameters that can drastically change the accuracy of ones results; the number of iterations and a lambda damping constant. Below, we have a table that details an experiment we ran across different combinations of these values. From the results we got, we determined that a lambda value of 10^{-3} and an 25 iterations per calculation gave us the most accurate result. When put to practice, however, we noticed that having the LM algorithm run for 25 iterations per calculation was too computationally expensive for our poor Teensy. When rendering the output, it took half a second between positional updates, which was a jarring experience for the headset wearer. We found a compromise between accuracy and latency by setting the number of iterations per calculation to 10.

Because our project is a dynamic experience, it is hard to fully capture the output without relaxing some constraints. Figures 6 and 7 show a simulation of what the rendering scene would look like after calibration and lowering of the device, respectively.

5. Discussion

After evaluating our algorithm and receiving user feedback, we identified several benefits and limitations. In terms of benefits, the sensor fusion provides a more seam-

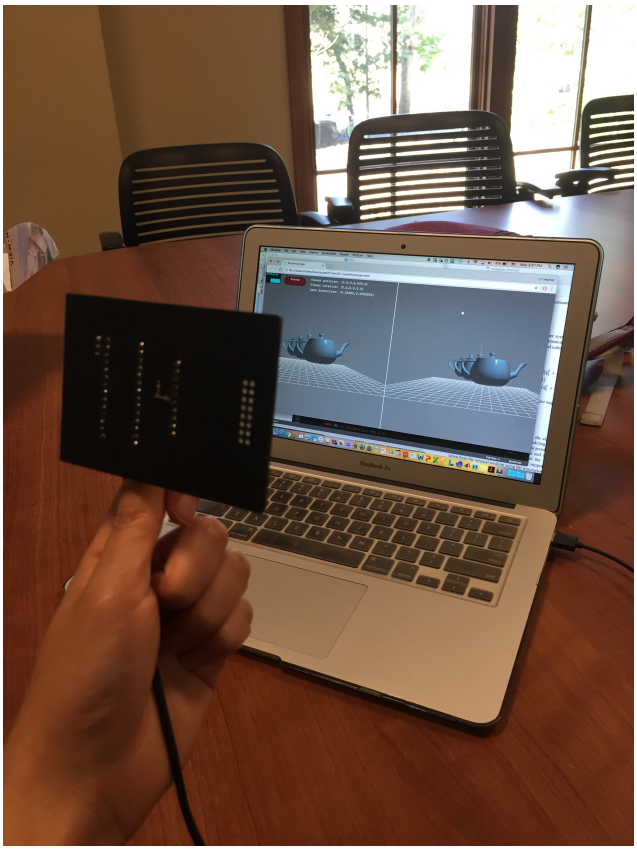

Figure 6. What stero rendering looks like after calibration.

less user experience because even when the LM algorithm cannot perform position estimation (ie obstruction, inability to update, etc), quaternion data is still streamed. Thus, the users orientation is tracked regardless of which direction they turn. Furthermore, with the initial calibrations tep, the user begins in a defined viewer position in the world space. Specifying the original viewer position is helpful, especially in environments where gaze and focus direction are important. Another benefit of this algorithm was that by calculating the variance of the position estimates, the noise or unintended motion of the user can be minimized. Scaling the change in motion with our variance reduces overly jarring or unexpected motion.

However, through user testing, we also identified several limitations of our design. In particular, the restrictions using the IR beacons were especially troublesome. For example, there must be a line of sight between the two IR beacons as well as between the photodiodes and the beacon. The former is difficult to establish with multiple people in a room, as we discovered during the demo session. The latter issue prevents continuous updates for our position estimates. Occlusion of the photodiodes by either the users position or by hardware limitations prevents the LM algorithm from working. This severely limits the authenticity of a users experience as they are unable to turn away from the IR beacon and must always be facing towards the beacon. Furthermore, the wire connection between the Teensy and computer restricted user mobility.

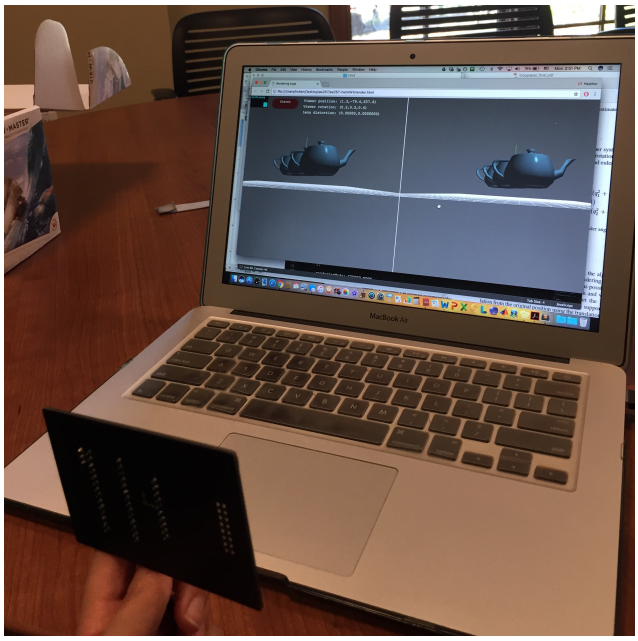

Figure 7. What stereo rendering looks like when the board is lowered and slightly tilted upwards.

For future work, we hope to address several of these limitations. For example, using a bluetooth connection to stream data would allow a wider range of user motion without the restriction of a wire connection. Another direction to explore is trying to read signals from multiple IR beacons. With this technology, position tracking can be more significant since the user is not limited to keeping within the line of sight of a single IR beacon and instead can move more naturally (as long as it is within line of sight of another beacon). We would also explore using Kalman filtering as a method for inferring the position from the IMU data whenever the photodiodes fail to update their position.

6. Conclusion and Future Work

Ultimately, this project helped emphasize the importance of positional tracking, but made us realize the many limitations and difficulties surrounding it. Particularly regarding the IR beacons, maintaining line of sight is impractical in crowded rooms, but we look forward to seeing how industry and research will begin addressing these issues.

Many thanks to the EE267 teaching staff, especially Keenan Molner, for their help and guidance throughout the process and to Gordon Wetzstein for his instruction of EE267.

References

[1] http://www.businesskorea.co.kr/english/news/ict/14165-hurdles-vr-popularization-samsung-lg-compete-vr-display-development-race/.

[2] https://uploadvr.com/inside-out-google-solve-tracking/.

[3] D. Bowman and R. McMahan. Virtual reality: How much immersion is enough? *Computer*, 40(7):36–43, 7 2007.

[4] N. Owen, A. G. Leadbetter, and L. Yardley. Relationship between postural control and motion sickness in healthy subjects. *Brain research bulletin*, 47(5):471–474, 1998.

[5] T. Petersen. A comparison of 2d-3d pose estimation methods. *Aalborg University, Aalborgb*, 2008.

[6] T. Pintaric and H. Kaufmann. Affordable infrared-optical pose-tracking for virtual and augmented reality.

[7] T. Raharijaona, P. Mignon, R. Juston, L. Kerhuel, and S. Viollet. Hypercube: A small lensless position sensing device for the tracking of flickering infrared leds. *Sensors*, 15(7):16484–16502, 2015.