Virtual & Augmented Reality Report

cvhm34

This report tries to provide evidence-testing details for each question and the investigation answers for Problem 6. The program has finished all the implementations except Problem 5.2. Due to the rendering complexity, the fully sequenced recorded video did not show the physics involved.

1 Problem 2

We first converted the rotational rate to radians/sec by following Equation 1.

$$Angular\ velocity\ (rad/s) = \frac{Rotational\ rate\ (degrees/s) \times \pi}{180} \tag{1}$$

Then we normalized the magnitude of both accelerometer and magnetometer values by using Equation 2, and when the NAN occurs, we will return the original value. Figure 1 shows the processed result.

$$normalized = \frac{a}{a_x^2 + a_y^2 + a_z^2} \tag{2}$$

	time	gyroscope.X	gyroscope.Y	gyroscope.Z	accelerometer.X	accelerometer.Y	accelerometer.Z	magnetometer.X	magnetometer.Y	magnetometer.Z
0	0.000000	-0.016362	-0.021817	0.015272	0.019439	-0.053333	0.998388	0.422961	0.062953	-0.903959
1	0.003906	-0.016362	-0.021817	0.015272	0.019439	-0.053333	0.998388	0.422961	0.062953	-0.903959
2	0.007812	-0.016362	-0.032725	0.004363	0.019042	-0.067868	0.997513	0.422961	0.062953	-0.903959
3	0.011719	-0.022907	-0.034907	0.002182	0.027108	-0.033024	0.999087	0.426619	0.081448	-0.900757
4	0.015625	-0.030543	-0.037088	0.000000	0.003438	-0.032906	0.999453	0.426619	0.081448	-0.900757

Figure 1: Processed result for 2.2

2 Problem 3

We implemented dead reckoning with the accelerometer and complementary filter by investigating different alpha values. We began with 0.5, giving equal weight to the gyroscope and accelerometer estimations. Figure 2 shows the result; when rolling starts, the yawing also changes significantly, especially since the yaw angle changes frequently from 4s to 8s. After that, when yawing started at 10s, the other angles had steady changes. The maximum angle was 0.6 radians, and the minimum was -0.6 radians. However, based on the record, the maximum angle should be around 90 degrees or approximately 1.5 radians. So, we tried decreasing the alpha value to 0.3, as shown in Figure 3. The shape of the figure did not change, but the maximum angle reached increased, reaching around 1.0. We continued to choose alpha around 0.10, and the yaw's highest value reached 1.5. We then continued to decrease the alpha value to 0.03, and as shown in Figure 4, all the angles could reach 1.5 radians.

3 Problem 5

In this question, based on the air resistance Equation 3, we need to analyse the object's drag coefficient C_d, area A, and air density r. We assumed that the headset object is the Oculus Quest 2; we could continue to assume

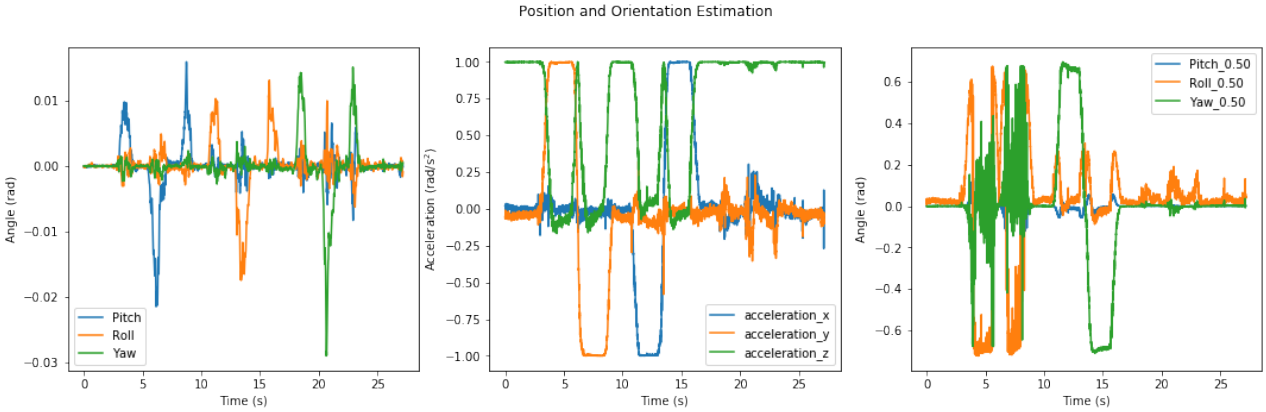

Figure 2: Gyroscope, accelerometer and fused orientation with 0.50

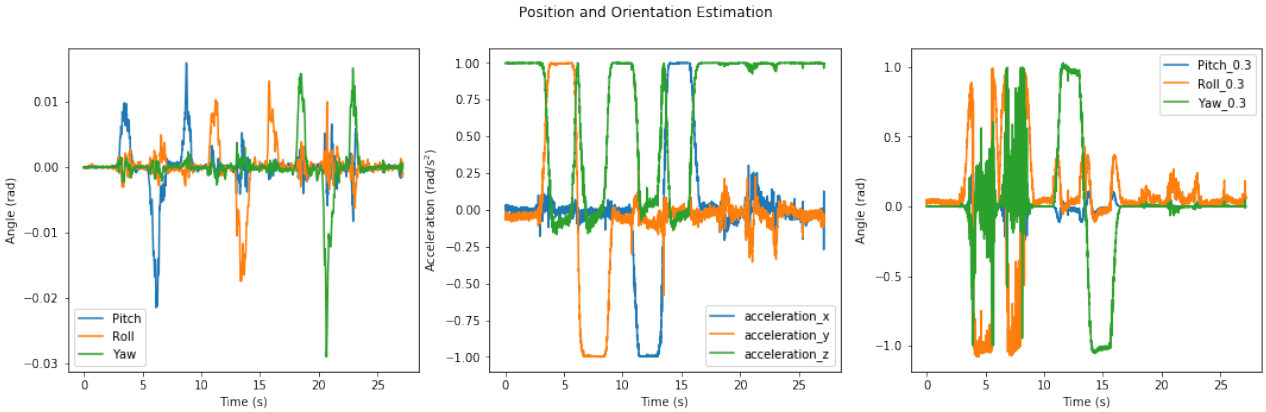

Figure 3: Gyroscope, accelerometer and fused orientation with 0.30

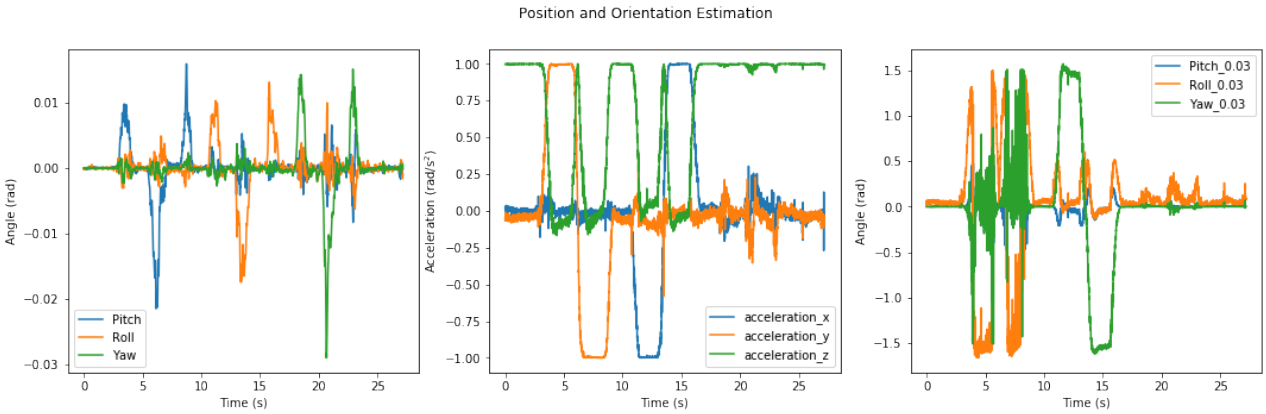

Figure 4: Gyroscope, accelerometer and fused orientation with 0.03

that it is the half sphere based on the shape of the Oculus Quest 2 and the 3D model. So that we can assume the C_d as 0.42 by checking from [1], and then we found that Oculus Quest 2 has a length of 0.1915 metres and width of 0.102 metres [1], we assumed the headset surface is the rectangle so that the A can be calculated

[1] $https://www.gsmarena.com/oculus_quest\_2\_review-news-46255.php$

as 0.019533 m^2. Regarding air resistance, we assumed the object was falling from the 5km sky. Based on the game engine, we assumed the air resistance is 0.683 [2].

$$D = 0.5 * C_d * r * A * V^2 \tag{3}$$

We continued to look for the object's weight, and we took gravity as 9.81, and as we were building the simple physic engine, we assumed the mass would not change when velocity went up. We calculated the weight by using Equation 4, and then after we got the total force, we then calculated the acceleration by using Newton's second law in Equation 5.

$$W = mg \tag{4}$$

$$a = \frac{W - D}{m} \tag{5}$$

After we got the acceleration, we continued to calculate the current velocity by using Equation 6 and position by using Equation 7; we would assume the initial velocity was 0 and the initial position was 0.

$$v = v_0 + a * dt \tag{6}$$

$$x = x_0 + v * dt \tag{7}$$

Figure 5 shows the falling object simulation; the translation will add it to the Z-axis and Y-axis; it is done by experiments.

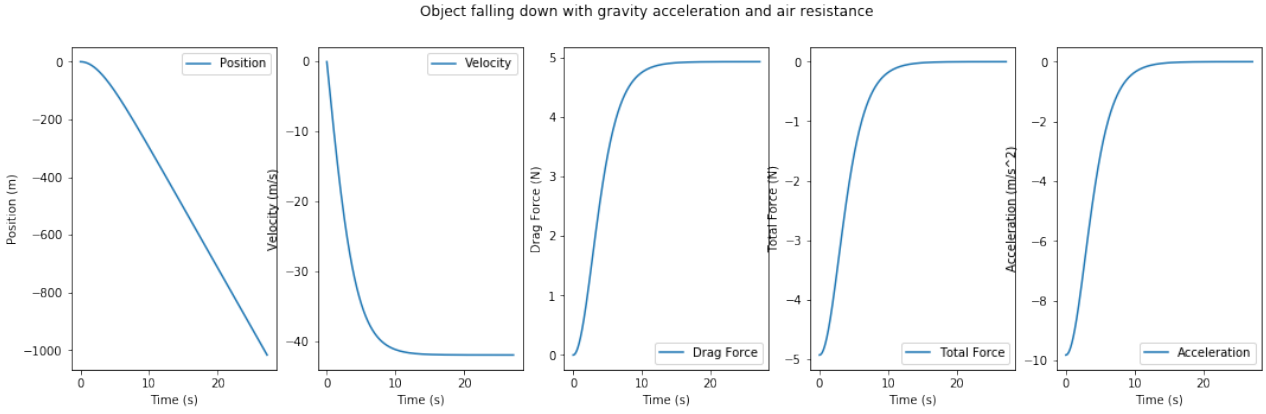

Figure 5: Object falling with gravity and air resistance

4 Problem 6

4.1 Demonstrating tracking and Transformation functions

Figure 6 shows how the camera rotates by following the translation matrix and how it tracks the object. The video would be able to show more smooth results.

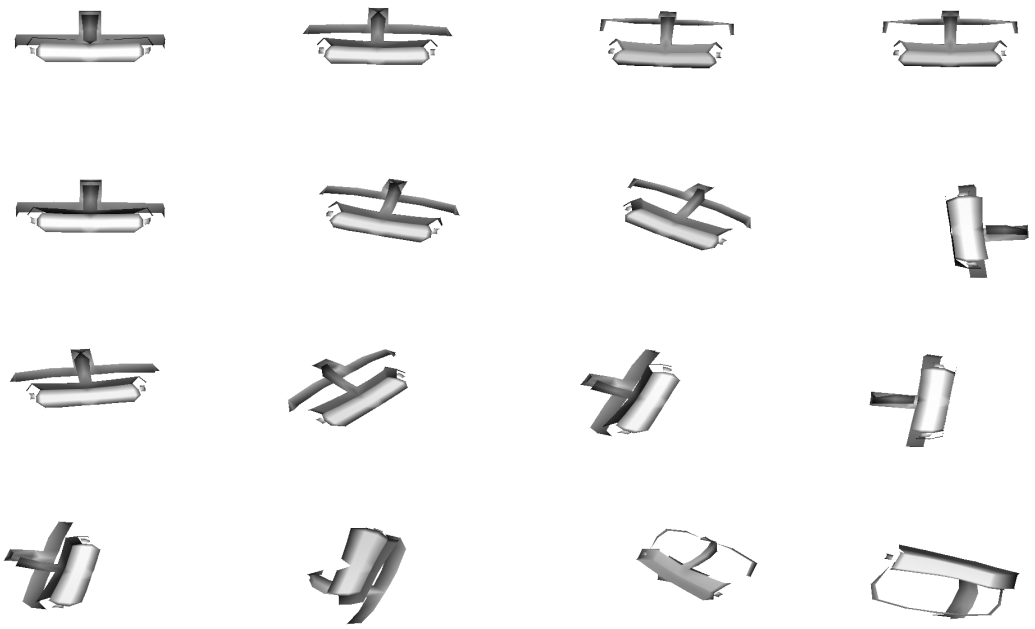

Figure 6: Tracking and translation object

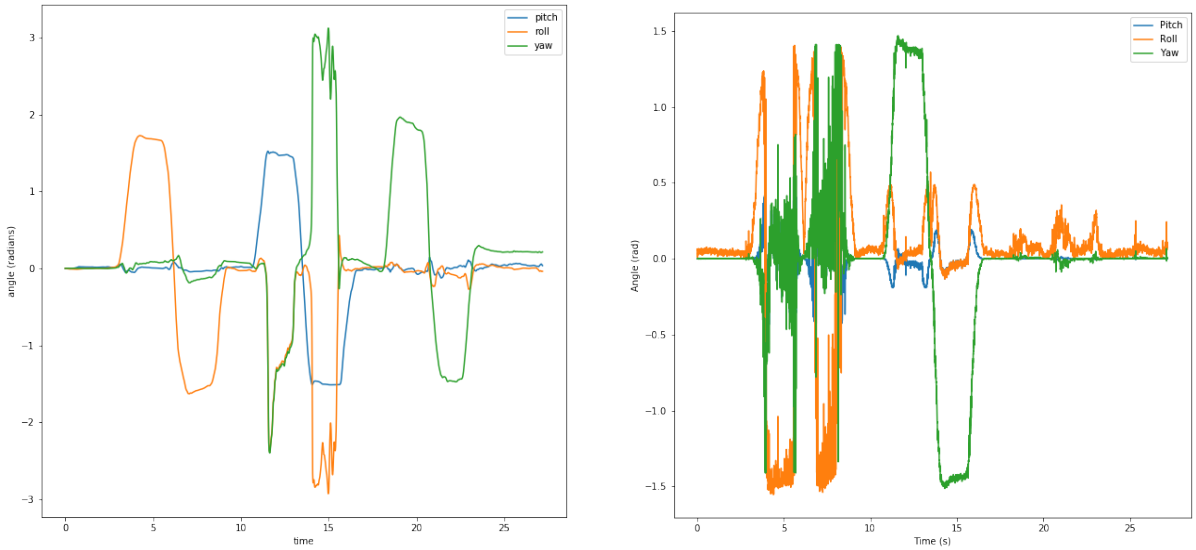

Figure 7: Angle for the simple dead reckoning (left) vs with accelerometer (right)

4.2 Simple Dead Reckoning vs with the Accelerometer

4.2.1 Simple Dead Reckoning

Simple dead reckoning is easy to implement by knowing the previous position and the expected gravity value (for gravity-based tilt correction). Figure 7 shows simple dead and dead reckoning with accelerometer data results showing that some details may match, as seen in the first few seconds. In addition, the general shape is similar, and we can also use it as part of the VR movement computation to speed up the rendering process. However, there are some disadvantages; as mentioned before, the simple dead reckoning relies on the previous

[2]Reference: $https://steamcommunity.com/sharedfiles/filedetails/?id = 2942123847$

position and gravity value; it predicts the movement speed as linear increases or decreases; for example, the pitch angle can reach 3 radians and is unsuitable in real-world scenarios. Also, it relied on the previous orientation, precisely the initial orientation, which would lead to a more significant error when we estimate the incorrect initial orientation.

4.2.2 Dead Reckoning with Accelerometer

We used the accelerometer values to increase the accuracy by making better position estimations. As seen in Figure 7, the accelerometer value helps to modify the angle to improve the position accuracy. It also doesn't rely too much on previous orientations, as the accelerometer would adjust if there were a small mistake. The calculation process is also not linear, and the given value is more suitable than the simple dead reckoning. However, there are some disadvantages. Firstly, it is sensitive to the noises from the accelerometer. Therefore, adjusting the balance between gyroscope and accelerometer estimations adds more challenges and takes time to tune. Secondly, it requires more computational time; for example, a simple dead reckoning would spend 1.2s, but an accelerometer would spend 2s.

4.2.3 Performance comparison

Simple dead reckoning relies on an object's previous position and velocity to estimate its current position. The calculations are straightforward, involving basic arithmetic operations, as shown in Figure 8. However, dead reckoning with the accelerometer requires more complex computations, such as tilt correction, translation from acceleration to the global frame, finding the angle between the up vector and the acceleration vector, and applying a complementary filter. The up vector and the alpha value, which balance the gyroscope and accelerometer estimations, also affect the performance. In general, dead reckoning with the accelerometer performs better but requires more computations than simple dead reckoning. Furthermore, the alpha and up vectors influence the final performance result, which means the performance could worsen if unsuitable values are chosen. The calculation process has been shown in Figure 8.

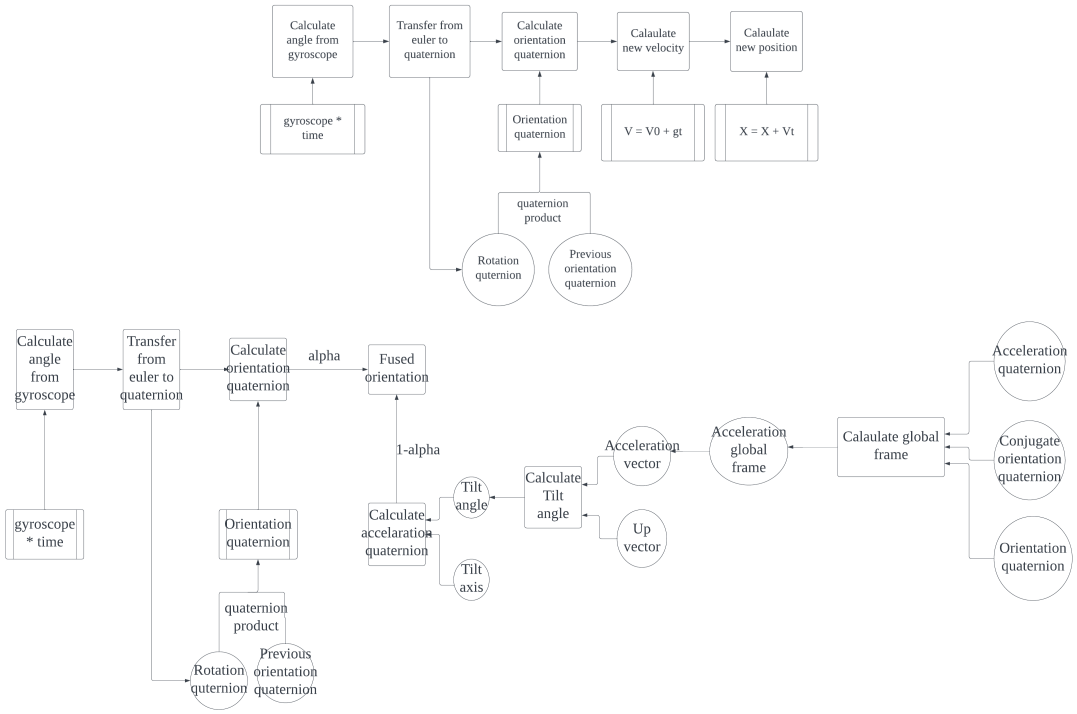

Figure 8: Simple dead reckoning process (Top) and dead reckoning with acceleration (Down)

4.3 Positional Tracking

As demonstrated in this project, it is possible to apply positional tracking based on provided IMU data, and we were using dead reckoning to perform inertial tracking through accelerometers and gyroscopes. This approach can track fast movements and provide high update rates. In general, position tracking aims to provide real-time head position estimates with no delay, but the expected accuracy in this approach may be lower due to drift errors and noise (as seen in Figure 3). There are some limitations to this method. Firstly, estimating the current position takes time as the process involves high computational complexity, so the delay would affect the performance. Secondly, dead reckoning can cause drift due to sensor noise, and therefore it would make errors in the position estimation.

4.4 Collision Detection

In collision detection, we used the sphere as a bounding region. However, the sphere would not be an ideal choice even though it is simple to implement and more accessible to make calculations. If we can improve it, we would assume the headset object is the ellipsoid or a half sphere and by measuring its length, width and height to provide the bounding regions. This will improve the collision detection performance by providing more accurate model data. However, it would affect the rendering process because of the more complex computation process, so we would continue to provide more efficient collision detection algorithms or data structures.

4.5 Distortion Correction

The first option is to find the GPU programming framework, such as the API in NVIDIA [3], then as the benefits of using GPU is it can run on parallel computation to speed up the rendering process, so we can find which part of the process we can make in parallel computing. The second option is from [2], suggesting to use of mesh-based approach correction; the advantage of this approach is to provide convenience on pre-correction and resuing the mesh, speeding up the process. The other option is to make an approximation in which everything between mesh points will be interpolated, which has also been recommended by [2] and has shown better results.

References

[1] Wikimedia Foundation. Drag coefficient. `https://en.wikipedia.org/wiki/Drag_coefficient`, 2023. Accessed: May 8, 2023.

[2] Imagination Technologies. Speeding up GPU barrel distortion correction in mobile VR - Imagination. *Imagination Blog*, 2016. Accessed: May 6, 2023.

[3] $https://docs.nvidia.com/vpi/algo_{l}dc.html$