Virtual Reality Summative

Bradley Mackey

for 15th March 2019

Question Remarks

1. `get_raw_imu_data()` returns the raw data readings from the `.csv` file (given that it is located in the same directory), returning a 2D array of data rows.
   `sanitize_imu_data(data)` cleans the data as specified, returning a 2D array of rows of the modified data.
   `euler_to_qtrn(euler)` computes a quaternion (a, b, c, d) from a given array of Euler angles (θ, ψ, ϕ), the rotations around the x-, y- and z-axes respectively.
   `qtrn_to_euler(qtrn)` computes the Euler angles (θ, ψ, ϕ) for a given quaternion representation (a, b, c, d).
   `qtrn_conj(qtrn)` takes a quaternion (a, b, c, d) and returns its conjugate, $(a, -b, -c, -d)$.
   `qtrn_mult(qtrn_1, qtrn_2)` computes the product of 2 quaternions, returning this product (a, b, c, d).

3. For the smallest values of α_{tilt} (< 0.001), very little drift correction is applied and the headset is able to maintain smooth, albeit slightly misaligned motion after correction. For high values of α_{tilt} (0.1–1), after around 20 seconds I noticed the Euler angle around the x-axis begins to drift by around -25° towards the end of the readings. This is due to the fact that as the IMU is drift corrected, drift correction rotation occurs around the x-y plane only (as z is the 'up' axis, pointing upwards initially according to the accelerometer, shown in Figure 1). As such, a yaw motion around the z-axis is represented by the x (or θ) Euler angle here.

   The action of correcting the tilt can slightly induce a drift into this angle, so high values of α_{tilt} are not preferable, as this effect becomes more exaggerated. Additionally, high values of α_{tilt} can cause very noisy positional readings as each movement is precisely drift corrected. This would cause a very unpleasant viewing experience for a VR user as the viewport abruptly snaps back to a correct position with each IMU reading.

   I found that an α_{tilt} value of around 0.01 resulted in good drift correction without the large amount of noise caused by correction. This is visible in Figure 2. A reduction in the noise could be reduced by taking the average position of the 'up vector' over a number of samples in proximity.

4. Yaw correction effects the Euler angle around the x-axis (θ angle) because of the initial mounted positioning of the IMU, explained above. I found that all values of α_{yaw} greater than 0.0001 were able to correct drift induced into the θ angle by the tilt correction and, as such, I found that $\alpha_{tilt} = 0.001$ allowed a for more stable output in the other two axes, without affecting the correction in the yaw axis. Increasing the value from this point all the way up to $\alpha_{yaw} = 1$ maintained the positioning of the drift correction of the x-axis, but induced more noise into the Euler angle. Therefore, the best angles for the combined tilt and yaw correction I found were $\alpha_{tilt} = 0.001$ and $\alpha_{yaw} = 0.001$. The result of this correction can also been seen in Figure 2.

   When experimenting with a value of α_{yaw} much greater than α_{tilt}, I noticed very unusual (and obviously incorrect) results. This is likely due to the overcompensation of yaw correction while not taking into account the tilt drift error. This is why my α_{yaw} value is very small, as for this to not dominate the correction and produce incorrect results.

Visualisations

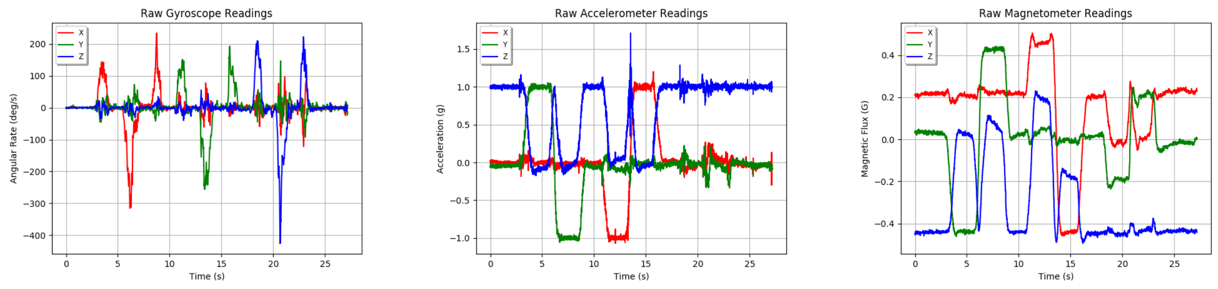

Figure 1: Raw sensor readings from the IMU. Raw accelerometer data (centre) clearly displaying the z-axis as the initial 'up' position (reading of g), so tilt corrections are made towards this axis, yaw corrections are made around this axis.

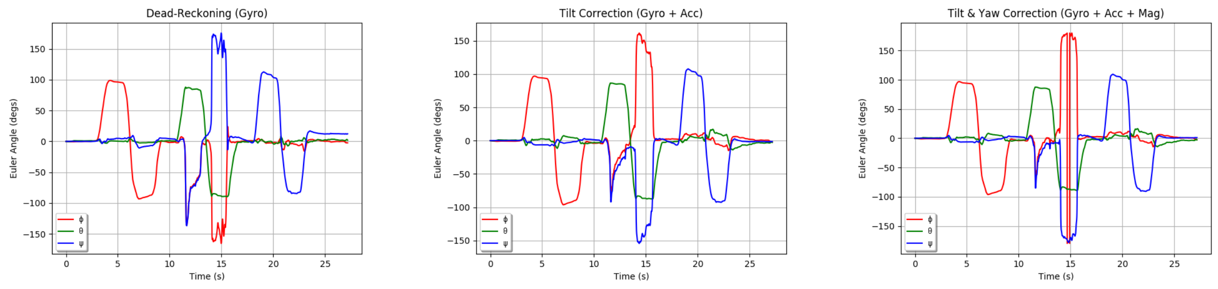

Figure 2: Euler angle readings, with and without various levels of correction. For tilt correction, $\alpha_{tilt} = 0.01$. For tilt & yaw correction, $\alpha_{tilt} = 0.001$, $\alpha_{yaw} = 0.001$. Notice the clear improvements and reduction in axis drift as more levels of error correction are added to the raw dead reckoning data. The clear area of error reduction is in the two axes that are supposed to have 0 angle (these vary as the time progresses) between the times $t \approx 3$ and $t \approx 17$. The erratic angle changes occurring at $t \approx 18$ and $t \approx 20$ is caused by gimbal lock induced by the rotation of the IMU and the fact we are displaying this data as Euler angles.

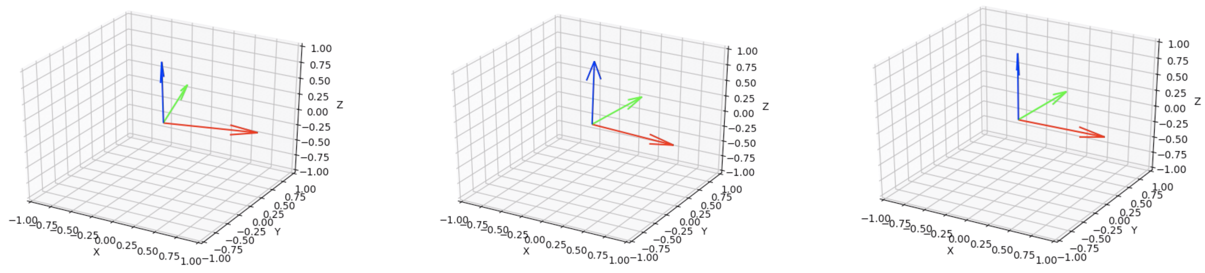

Figure 3: From left to right: dead reckoning, tilt correction, tilt & yaw correction. The blue, green and red vectors correspond to the z, y and x vectors respectively. Vectors shown are the final positions of the IMU after all data points were played back.

Figure 3 shows screenshots from the playback of data for each correction method. It is clear there is less drift as more levels of correction are included in the calculation of the true orientation Data was recorded for each of the correction methods and interactively displayed in 3D using `matplotlib` for Python.

Due to technological constraints, data was recorded at 40x slower than realtime, recorded and played back at realtime, then half speed. With all levels of correction, the micro-judders caused by twitchy movements of the IMU are not able to be corrected, so these slight judders are visible in all animations. The reduced drift is noticeable in both the tilt and tilt & yaw corrected movements though—although the difference between these two is minor. Overall, I found that tilt correction alone was able to dramatically increase the positional accuracy of the raw IMU data, yaw correction only slightly improving this accuracy.