Virtual Reality Assignment

1 Initial Processing

Quaternions, XYZ angles and other data are stored as Numpy Arrays for speed and to take advantage of native np libraries but managed as tuples inside of functions for simplicity. More details on the software can be found in function comments and at the start of the code. Default options can be run by entering 'python main.py' from the command line with the input data in the same directory, which runs the main method after function initialisation using 'main()'. Alterable parameters are highlighted where appropriate and additional functions are available at the bottom of the code, used during testing phases as explained in sections 2 and 3. The following is a summary of all primary functions used during position calculation.

Function Name	***Input***	***Output***
quaternion_product	quaternion_a, quaternion_b	quaternion_c
euler_to_quaternion	euler angles = (x,y,z)	quaternion = (w,x,y,z)
quaternion_to_euler	quaternion = (w,x,y,z)	euler angles = (x,y,z)
quaternion_conjugate	quaternion_a	quaternion_b
convert_rotation_deg_rad	gyroscope in deg/s	gyroscope in rad/s
axis_angle_to_quaternion	tilt-axis, theta	quaternion rotation
normalize_data	data	normalized data
calculate_position	time, gyroscope	position estimate
calculate_tilt_correction	time, gyroscope, accelerometer, alpha_tilt	position estimate
calculate_yaw_correction	time, gyroscope, accelerometer, magnetometer, alpha_tilt, alpha_yaw	position estimate
read_data	Data Filename	time, gyroscope, accelerometer, magnetometer
plot_data	Plots a single input data source into a figure	
plot_position	Plots a position for a specified correction level	
plot_all_positions	Plots all position estimations in a single figure	
animated_plots	Accepts a speed parameter, loops estimated IMU movement	
main	When run, collects and processes data, estimates all positions, plots and animates	

Table 1: The Primary Functions of the Program

2 Tilt Drift Correction

Incorporating accelerometer data into the dead reckoning allows for drift compensation, bringing the estimated IMU position closer to its true value. The calculation first takes an average accelerometer reading over all currently viewed results to out smooth correction, an alpha value representing the weighting being place on rotation. The table below shows the last tri-axial Euler angles recorded and an absolute calculation for each vector. The IMU is assumed to end at (0,0,0), so the closet to 0 the net position vector, the more appropriate the alpha value. Functions were created for iteratively searching for the best alpha value, including incrementally iterating over a fixed range of values and a binary search for smaller precisions. All such functions have been included at the end of the code. As can be seen, if alpha is not appropriate, it can increase the 'abs' value and worsen results. However, the overall effect of drift compensation is significant when comparing the optimal value of 0.05531 to that of 0 in table, 25.434 and 2.963 'abs' respectively. It was found that alpha values of order 10^{-2} were generally most appropriate.

Tilt Alpha	***X***	***Y***	***Z***	***abs***
0	15.812	-8.412	-1.21	25.434
1	-17.835	-1.764	7.691	27.291
0.1	-6.542	-1.222	0.738	8.502
0.01	9.963	-7.025	-0.558	17.546
0.001	14.963	-8.252	-1.113	24.328
0.0001	15.723	-8.395	-1.2	25.318
0.00001	15.803	-8.41	-1.209	25.422
0.05	0.835	-3.151	0.091	4.076
0.05531	-0.004	-2.813	0.146	2.963

Table 2: Effect of Tilt compensation on XYZ degree position values

Yaw Alpha	***X***	***Y***	***Z***	***abs***
1	6.475	3.611	-172.55	182.638
0.1	-1.434	18.874	-114.67	134.976
0.01	-2.57	24.34	-3.299	30.209
0.001	-2.916	36.419	42.216	81.55
0.0001	-0.301	-1.932	-1.382	3.615
0.00001	-0.031	-2.728	0.032	2.79
1e-06	-0.007	-2.805	0.135	2.946
1.2e-05	-0.036	-2.71	0.008	2.755
1.272e-05	-0.038	-2.704	-0.0	2.742

Table 3: Effect of Yaw compensation on XYZ degree position values

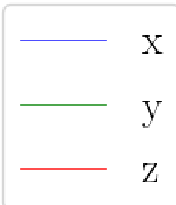
3 Yaw Drift Correction

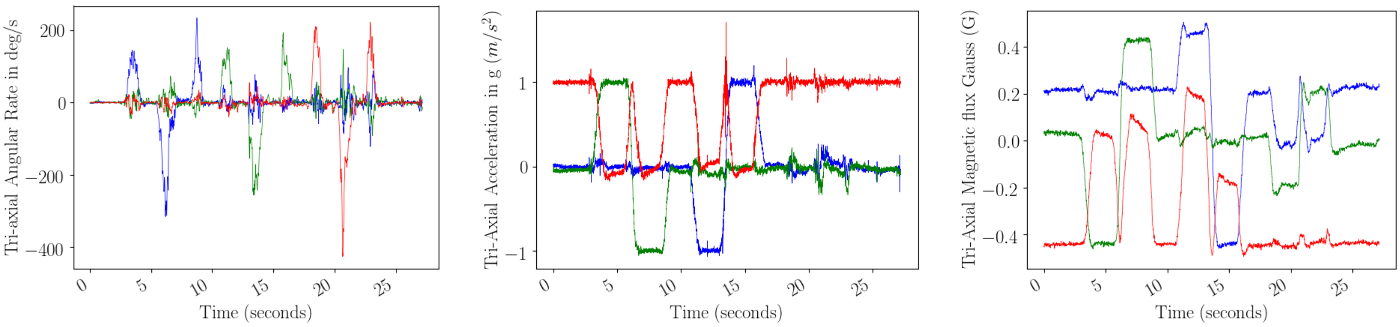
Additionally, using the magnetometer readings can produce a further improvement to the final XYZ readings, by estimating the yaw drift and correcting for it in the final position calculation. The process therefore follows on from tilt correction and gyroscope integration to produce a further improvement to position readings. The value of alpha for tilt compensation was be kept constant at the most effective value found: 0.05531. Magnetometer readings were intially averaged to smooth results, however this resulted in worse correction data, so has been removed. As can be seen in table 3 compared to 2, the effectiveness of yaw correction is less significant when compared to tilt correction, which appears to have a more extreme impact. However, improvements to position estimations are seen for alpha values to the order of 10^{-5}. A similar process for optimising the value was implemented, using the search methods followed by trial and error, similar to that in section 2, optimising the alpha yaw value at 0.00001272.

4 Displaying the Data

4.1 Input Dataset

Figure: Consistent colour coding for XYZ angle values during plotting

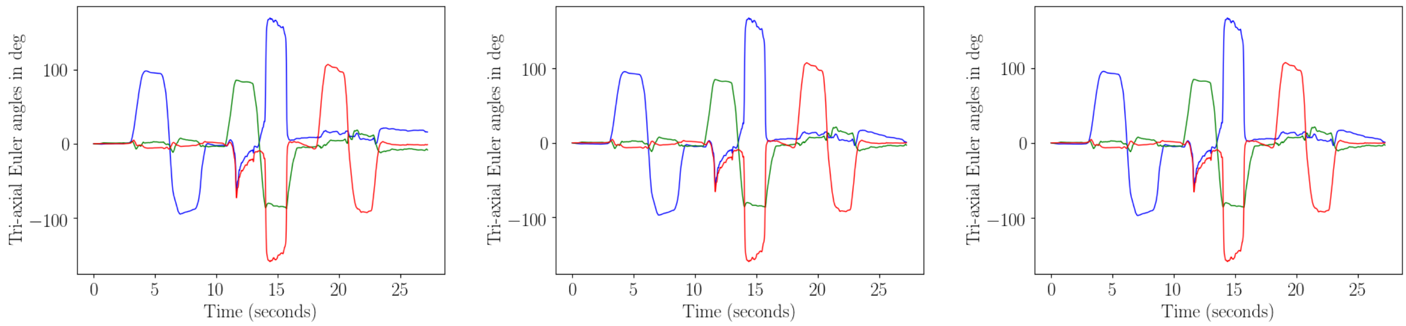




(a) Gyroscope Readings (b) Accelerometer Readings (c) Magnetometer Readings

Figure 1: Input Data Set as a function of time

4.2 Tri-axial Euler Angles



(a) Position (b) Position with Tilt Correction (c) Position with Tilt and Yaw Correction

Figure 2: Tri-axial Euler Angles in degrees, for each position estimation as a function time

4.3 Animated Plotting of IMU rotation

Running the main application will default, render the animation in real-time on a repeated cycle at normal speed. Closing the window will then render the animation at half speed. The function call takes the speed as a parameter and skips a prescribed number of frames/data points proportional to the requested speed. This value however can be altered manually if a more powerful computer is available for rendering at a smoother frame rate. As can be seen, the stability of each method varies, with integration of all methods producing the best results as long as alpha values are optimised. The impact of each correction is met with diminishing returns however, with tilt correction being significantly more impactful then yaw correction, suggestively due to the lower alpha values appropriate for yaw drift. As each plot is viewed relative to the initialised grey axis, as time plays out, the movement moves further away from the initial calibration. The methods remain accurate to within 5 degrees by the full playback, just enough for a VR headset, however without calibration stability will continue to drop off.

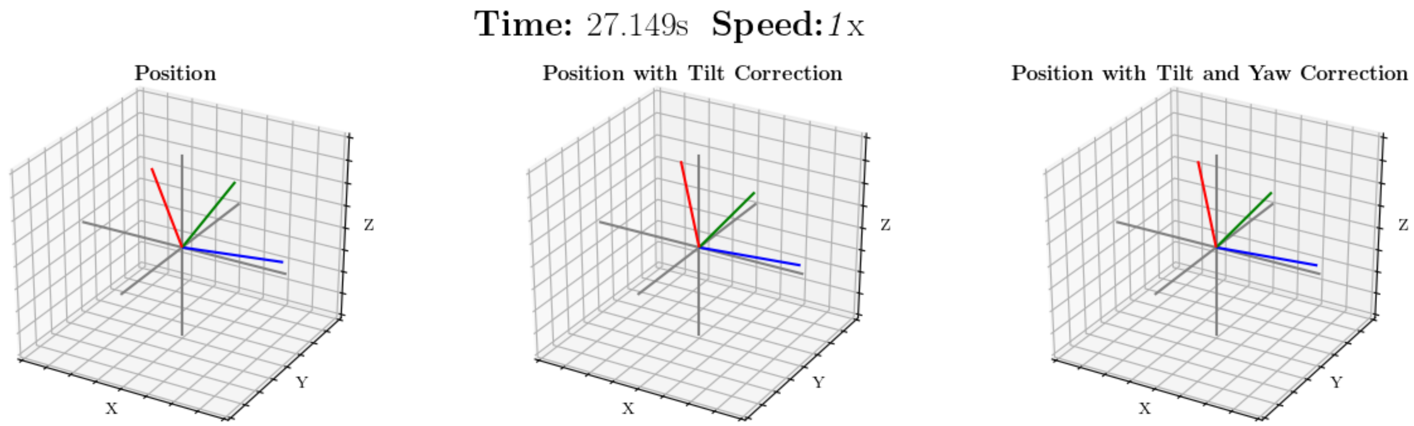


Figure 3: Final screenshot of animated sequence for XYZ, IMU positions using each method

References

Conversion between quaternions and Euler angles (n.d.), `https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles`. [Online; accessed 27-Feburary-2019].

LaValle, S. M., Yershova, A., Katsev, M. & Antonov, M. (2014), Head tracking for the oculus rift, *in* '2014 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 187–194.

Madgwick, S. (2010), 'An efficient orientation filter for inertial and inertial/magnetic sensor arrays', *Report x-io and University of Bristol (UK)* **25**, 113–118.