Title

Bachelor of Science Thesis

Michele Zoncheddu

University of Pisa

Department of Computer Science

March 2020

Contents

1 Introduction

2 Background

3 Technologies

4 Development

4.1 Cache

...

4.2 Motion capture

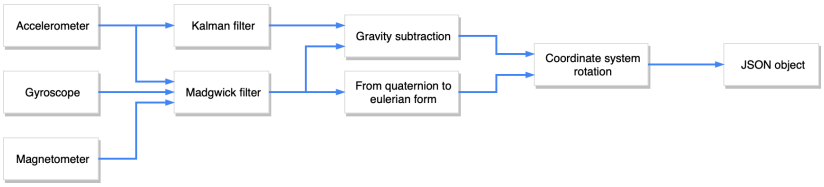

Figure 1: Data fusion schema.

For a better training of the neural network, the main problem was to achieve the cleanest data from the sensors. Previously, the data collected were:

- yaw, pitch and roll angles;
- accelerations in spherical coordinates system;
- raw gyroscope data;
- a velocity approximation.

Through preliminary tests has been discovered that acceleration values included gravity; thus it was necessary a more complex training data collection, due to the lack of orientation invariance (in order to obtain such invariance it would have been necessary to collect every pattern with the toy rotated in any possible angle).

Nevertheless, gravity-free acceleration data were computed by the device (with the help of the Madgwick's fusion algorithm output) and sent to the server but not stored in the database, only used for the 3D visual representation, both in training and session recording phase. It was decided to keep such data and analyze it to see if it was reliable over time.

Some testing showed quickly that there was an error in the yaw angle calculation: the more the toy was rotated (regardless of the axis), the more the yaw diverged. Such error was initially imputed to the Madgwick algorithm implementation, more precisely to the convergence of its error-correction gradient descent algorithm.

Unfortunately, different algorithm implementations and different values of the algorithm's gain parameter (the magnitude β of the gyroscope measurement error[1] [CITE]) have no given better results over time, and worse, sometimes ghost accelerations were found.

During the tests, it was noticed that recordings of the same pattern with different device orientations didn't split the acceleration differently among axes, revealing that the reference system was integral with the Earth's center instead of the device's sensor center. To align it back, the acceleration vector has been rotated using a 3D rotation matrix. The 3 rotation angles are given by the Madgwick algorithm in the quaternion form, then converted to the eulerian one.

A *yaw* is a CCW rotation of α on the z-axis. The rotation matrix is

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Note that the upper left values compose a 2D rotation matrix, and the coordinates on the third dimension (around whom the rotation happens) are left unchanged. The same applies to the other two matrices, but with the second and the first dimension.

A *pitch* is a CCW rotation of β on the y-axis. The rotation matrix is

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & \cos\beta \end{pmatrix}$$

[1] Increasing β leads to faster bias corrections and higher sensitiveness to lateral accelerations.

A *roll* is a CCW rotation of γ on the x-axis. The rotation matrix is

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & sin\gamma & \cos\gamma \end{pmatrix}$$

$$R(\alpha, \beta, \gamma) = R_z(\alpha)R_y(\beta)R_x(\gamma) =$$

$$\begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}$$

It is important to note that $R(\alpha, \beta, \gamma)$ performs the roll first, then the pitch, and finally the yaw. If the order of these operations is changed, a different rotation matrix would result [CITE].

Given an acceleration vector $\vec{a}$, the rotated one is $\vec{a}_r = R(\alpha, \beta, \gamma)\vec{a}$. The matrix-vector multiplication doesn't need specific optimizations in the code because the dimensions at stake are too small for a meaningful speed or space improvement, and however the C++ code is compiled by `GCC` with the `-Os` flag, that includes the loop unrolling optimization [CITE].

At this point, both problems were solved: the coordinate system was integral with the toy, and the yaw angle was calculated correctly.

Once achieved a reliable gravity-filtered acceleration approximation, it was possible to do some fine tuning. Calibration, hard and soft iron.

The velocity approximation was excluded by the training set because...

4.3 Delay in real-time data plotting

...

5 Conclusions