Attitude Estimation and Synchronization Error Calculation Pipeline

1 Introduction

This document outlines the pipeline for attitude estimation and synchronization error calculation using the Madgwick filter and motion capture (MoCap) data. The pipeline involves several steps, including data synchronization, attitude estimation, and error calculation.

2 Pipeline Overview

The pipeline consists of the following steps:

1. Package structure
2. Data Import and Synchronization
3. Attitude Estimation using the Madgwick Filter
4. Synchronization of Smartphone and MoCap Data
5. Error Calculation and Visualization

3 Package structure

Example `.ipynb` notebooks for attitude estimation and error evaluation are provided. Packages `ahrs` and `mrob` required.

- `tools_ahrs.py` comes from `ahrs` package and is used for plotting.
- `data_processing.py` is used for general operations with data, such as import, synchronization, smoothing, trimming, downsampling and time-shifting.
- `transform.py` is used for transformations from angular velocities to quaternions and back, for usual discrete integrating and differentiating.
- `utils.py` is used for the calculation of errors and some other functions.

- `compare.py` is used for demonstrations. It has comparisons of smartphone (Madgwick filter), Mocap and Android API Game Rotation Vector (GRV) data with each other, and general errors comparison of two arrays of quaternions.

4 Data Import and Synchronization

The first step involves importing data from various sensors (gyroscope, accelerometer, magnetometer) of smartphone, data from MoCap and GRV. Then they are synchronized to a common time base.
By default, all incoming data is being smoothed with Savitzky-Golay, to eliminate some measurement noise.

4.1 Sensors Data Import

4.1.1 Synchronized Sensors Data Import

The function `import_combined_data` reads combined sensors (gyr, acc, magn) data from CSV files and returns timestamps and corresponding sensor measurements. Example:

```
t_base, gyr_sync, acc_sync, magn_sync =
    data_processing.import_combined_data("combined_imu_data.csv")
```

4.1.2 Unsynchronized Sensors Data Import

- Data Import
  The function `import_data` reads sensor data from CSV files and returns timestamps and corresponding sensor measurements. For example, gyroscope data is imported as follows:

  ```
  t_gyr, gyr = data_processing.import_data("gyroscope.csv")
  ```

- Data Synchronization
  The function `sync_data` synchronizes the gyroscope, accelerometer, and magnetometer data to a common time base:

  ```
  t_base, gyr_sync, acc_sync, magn_sync =
      data_processing.sync_data(t_base, t_gyr, gyr, t_acc, acc, t_magn, magn)
  ```

4.2 Game Rotation Vector Data Import

Game Rotation Vector (GRV) is a 3D rotation vector (quaternion), representing the phone's attitude, estimated by Android API. It can be useful as data to compare filter's estimations with.

The function `import_gamerotvec_data` reads GRV from CSV file and returns timestamps and corresponding quaternions of rotation. Example:

```
t_grv, gamerotvec =
    data_processing.import_gamerotvec_data("game_rotation_vector.csv")
```

4.3 MoCap Data Import

The function `import_mocap_data` reads phone's attitudes estimated by Motion Capture system from CSV file and returns timestamps and corresponding quaternions of rotation and position vectors. Example:

```
mocap_t, mocap_q, mocap_T = data_processing.import_mocap_data(mocap_path)
```

5 Attitude Estimation using the Madgwick Filter

The Madgwick filter is used to estimate the attitude (orientation) of the device based on the synchronized sensor data. The filter can operate in two modes: IMU (using gyroscope and accelerometer data) and MARG (using gyroscope, accelerometer, and magnetometer data).

5.1 Madgwick Filter Initialization

The Madgwick filter is initialized with the synchronized sensor data:

```
madgwick_MARG = ahrs.filters.Madgwick(gyr=gyr_sync,
                                      acc=acc_sync,
                                      mag=magn_sync,
                                      frequency=freq)
```

5.2 Quaternion Output

The filter outputs the estimated orientation as a quaternion array:

$$Q = [w, x, y, z] \tag{1}$$

where w is the scalar part and x, y, z are the vector parts of the quaternion.

6 Synchronization of Smartphone and MoCap Data

To compare the smartphone's attitude estimation with the ground truth provided by MoCap, the data from both sources must be synchronized in time and geometry.
For the smartphone's Madgwick filter and smartphone's GRV comparison geometry synchronization is not necessary.

6.1 Time Synchronization

The function `sync_mocap_and_data` synchronizes two datasets in time. It is used for comparison of smartphone's Madgwick and GRV, because they are already in the same reference frame.

```
t_sync, gyr, acc, magn, grv_quats =
    data_processing.sync_mocap_and_data(grv_time, grv_quats,
    sm_time, sm_gyros, sm_accs, sm_magns)
```

The synchronization is done as follows.

1. First, we choose what is going to be goal timestamps. The shortest timestamps sequence is chosen. Then, we are synchronizing one of data sources to another.
2. For IMU data (angular velocities and accelerations) and magnetometer data, function is using the Savitzky-Golay filter and resamples it to mocap timestamps using cubic spline interpolation.
3. For Mocap data, it smooths 1D quaternion data using the Savitzky-Golay filter and resamples it to smartphone timestamps using Spherical Linear Interpolation (SLERP) for accurate quaternion interpolation.

6.2 Geometry Synchronization

For data from different devices, we need both time and geometry synchronization. They are done simultaneously with TwistnSync.
The transformation matrix M is used to align the smartphone and MoCap data geometrically:

$$\omega_{\text{aligned}} = M \cdot \omega_{\text{smartphone}} \tag{2}$$

where $\omega_{\text{smartphone}}$ is the angular velocity from the smartphone, and ω_{aligned} is the aligned angular velocity.

This transformation matrix M, and the time offset t_{offset} between smartphone and MoCap data sequences, is obtained by TwistnSync, launched with datasets of angular velocities. For smartphone, angular velocities are imported from IMU data. For MoCap and GRV, they are obtained from rotations:

$$\omega_t = \frac{d}{dt}\|Ln(R_{t-1}^{-1} \cdot R_t)\| \tag{3}$$

Then, angular velocities are synchronized with TwistnSync.
Finally, we obtain synchronized quaternions of orientations as follows:

$$R_t = R_{t-1} \cdot Exp(\omega_t \cdot dt) \tag{4}$$

7 Error Calculation

The final step involves calculating the error between the smartphone's attitude estimation and the MoCap ground truth. The error is calculated as distance between input data sequences. We calculate APE and RPE.

7.1 Absolute Pose Error

The Absolute Pose Error (APE) is given by the simple difference between the aligned estimated trajectory $\hat{x}^+$ and ground truth x.

$$e_{\text{APE}} = \sqrt{\frac{1}{K}\sum_{k=1}^{K} \|x_k \boxminus \hat{x}_k^+\|_2^2}$$

7.1.1 Rotation distance

The function `calculate_quat_distances` calculates the distance between two sets of quaternions:

$$\text{distance} = \|\ln(R_1 \cdot R_2^{-1})\| \tag{5}$$

where R_1 and R_2 are the rotation matrices corresponding to the quaternions.

7.1.2 Gravity distance

The function `calculate_g_distances` calculates the distance between two sets of gravity vectors directions:

$$\text{distance} = \frac{\|g_1 - g_2\|}{g} \tag{6}$$

where g_1 and g_2 are the default gravity vector g multiplied by rotation matrices corresponding to the quaternions:

$$g_i = R_i \cdot \begin{pmatrix} 0 \\ 0 \\ -9.81 \end{pmatrix} \tag{7}$$

7.2 Relative Pose Error

The Relative Pose Error (RPE) is showing how different are pose increments of estimated trajectory and ground truth. Pose increments on d_i steps are obtained as:

$$\tilde{\mathbf{X}}_r = \mathbf{X}_k \boxminus \mathbf{X}_{k+d_i}$$

And the RPE is:

$$e_{\text{rpe},d_i} = \frac{1}{D_i}\sum_{k=1}^{D_i} \|\tilde{\mathbf{X}}_r \boxminus \hat{\tilde{\mathbf{X}}}_r\|_2$$

One of RPE's advantages is that it is less sensitive to jumps in estimation error due to sampling the trajectory over many smaller segments. This allows for a much fairer comparision of methods for imperfect datasets.

7.2.1 Rotation distance

The function `RPE` calculates the relative distances between two sets of quaternions:

$$\text{RPE} = \| \ln(\tilde{R}_1 \cdot \tilde{R}_2^{-1}) \| \tag{8}$$

where

$$\tilde{R}_i = R_i \cdot R_{i+d}^{-1} \tag{9}$$

where R_1 and R_2 are the rotation matrices corresponding to the quaternions, and d is step size, default is 1.

7.2.2 Gravity distance

The function `RPE_g` calculates the relative distances between two sets of gravity vectors directions:

$$\text{RPE\_g} = \| \ln(\frac{\tilde{g}_1 - \tilde{g}_2^{-1}}{g}) \| \tag{10}$$

where

$$\tilde{g}_i = \boldsymbol{g}_i \cdot \boldsymbol{g}_{i+d}^{-1} \tag{11}$$

and

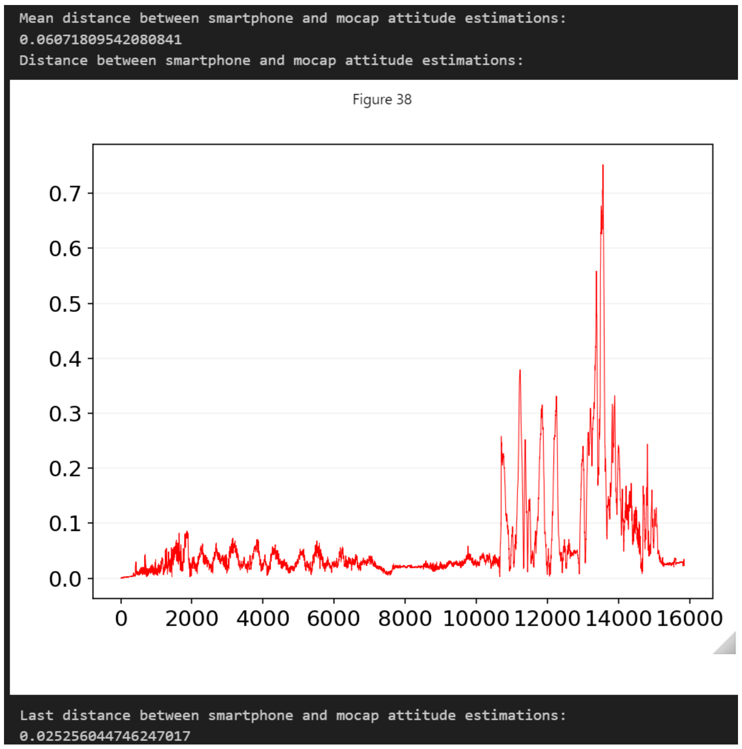
$$\boldsymbol{g}_i = SE3(R = 0, T = g_i) \tag{12}$$

where g_1 and g_2 are the default gravity vector g multiplied by rotation matrices corresponding to the quaternions:

$$g_i = R_i \cdot \begin{pmatrix} 0 \\ 0 \\ -9.81 \end{pmatrix} \tag{13}$$

8 Visualization

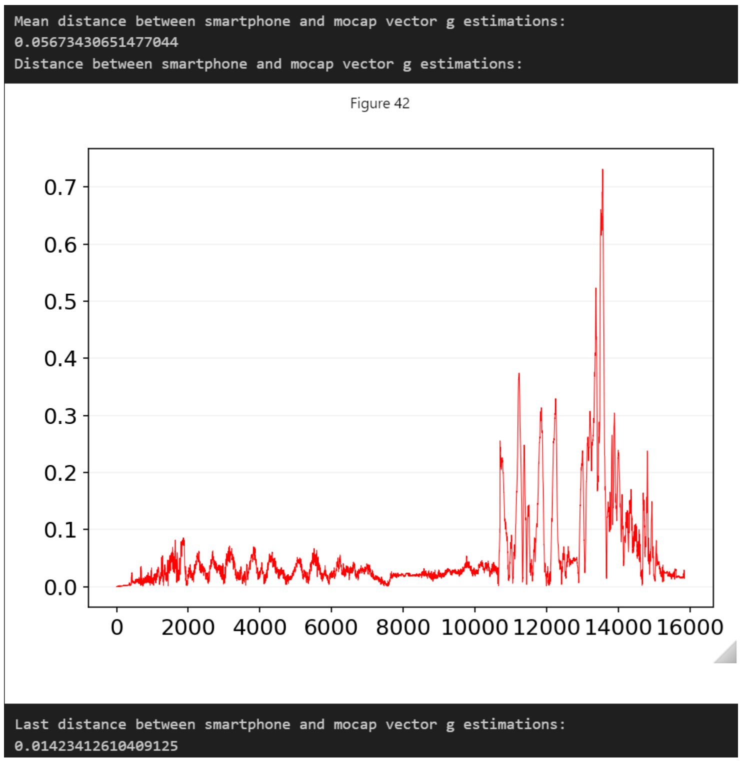
The error is visualized using plots to compare the smartphone and MoCap data:



```
Mean distance between smartphone and mocap attitude estimations:
0.06071809542080841
Distance between smartphone and mocap attitude estimations:
```

```
Last distance between smartphone and mocap attitude estimations:
0.025256044746247017
```

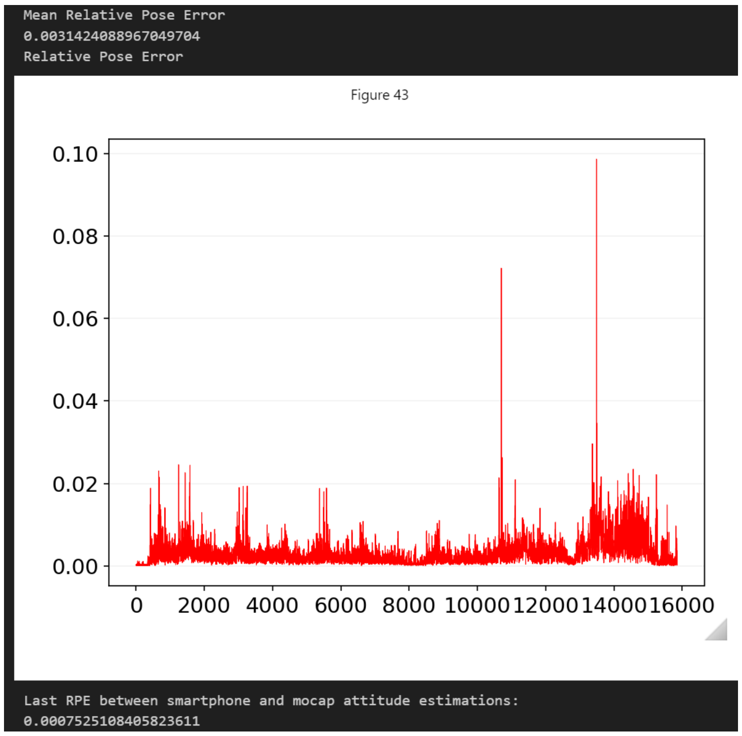
Figure 1: Absolute Pose Error of attitude estimations



```
Mean distance between smartphone and mocap vector g estimations:
0.05673430651477044
Distance between smartphone and mocap vector g estimations:
```

```
Last distance between smartphone and mocap vector g estimations:
0.01423412610409125
```

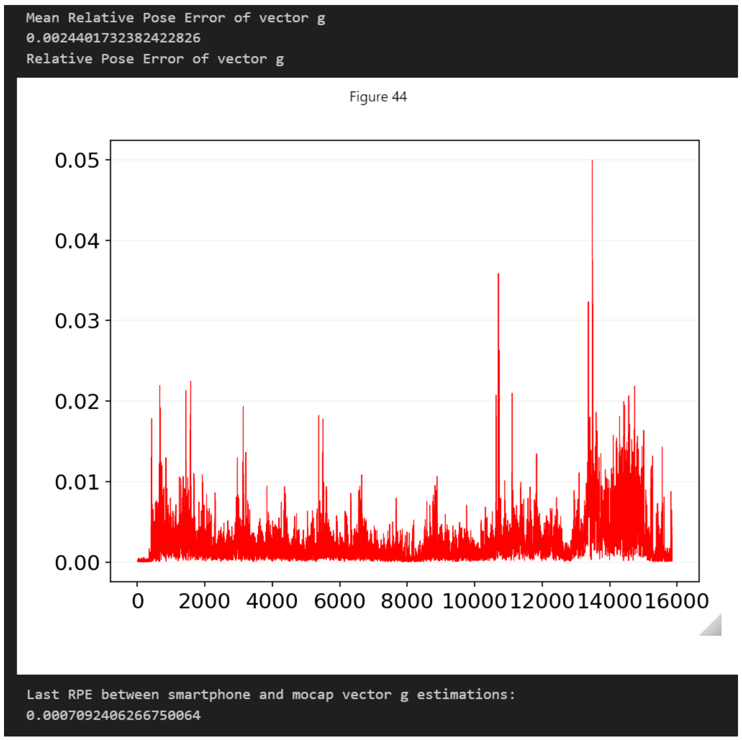
Figure 2: Absolute Pose Error of gravity vector estimations



```
Mean Relative Pose Error
0.0031424088967049704
Relative Pose Error
```

```
Last RPE between smartphone and mocap attitude estimations:
0.0007525108405823611
```

Figure 3: Relative Pose Error of attitude estimations



```
Mean Relative Pose Error of vector g
0.0024401732382422826
Relative Pose Error of vector g
```

```
Last RPE between smartphone and mocap vector g estimations:
0.0007092406266750064
```

Figure 4: Relative Pose Error of gravity vector estimations

9 Conclusion

This pipeline provides a method for estimating device attitude and calculating synchronization errors between smartphone and MoCap data. The Madgwick filter is used for attitude estimation, and the data is synchronized in both time and geometry before error calculation. For evaluation Absolute and Relative Pose Errors are calculated for both rotation matrices of attitude estimations, and estimations of gravity vector.