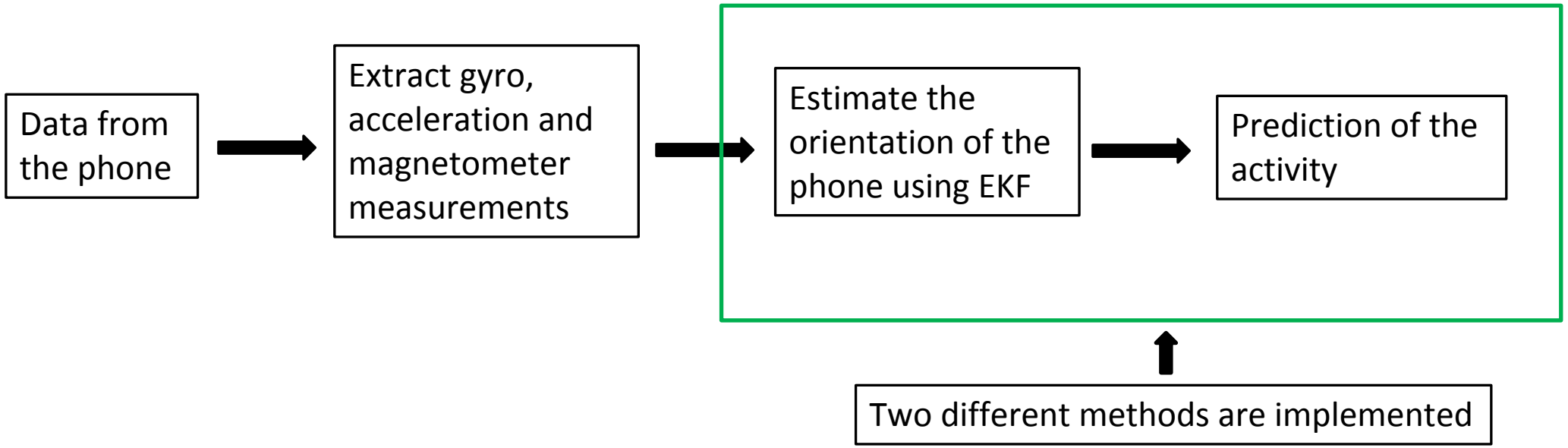
Activity Recognition

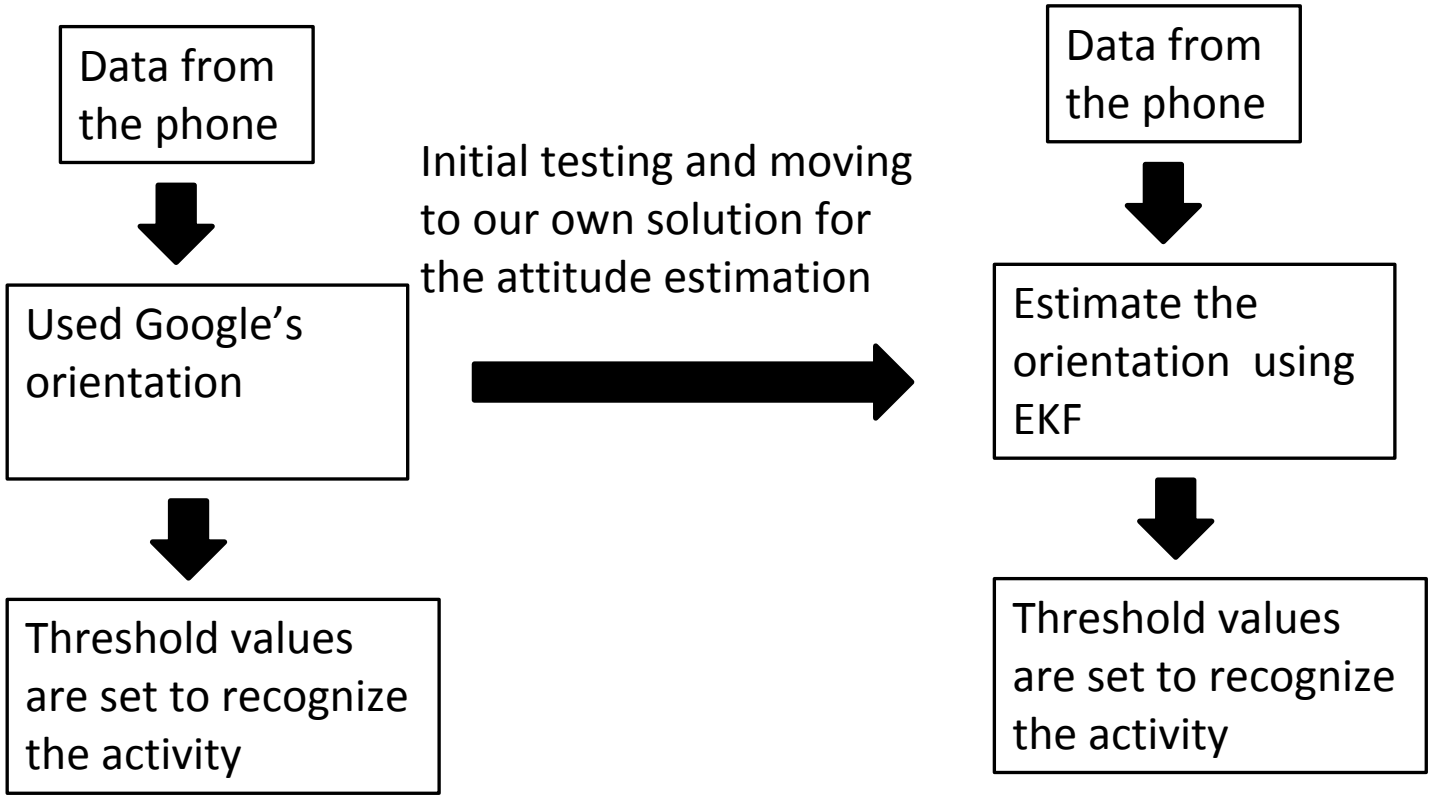
DOMINIK FAY, PEDRO ROQUE, SANDIPAN DAS, VANDANA NARRI, XIAO CHEN

Objective

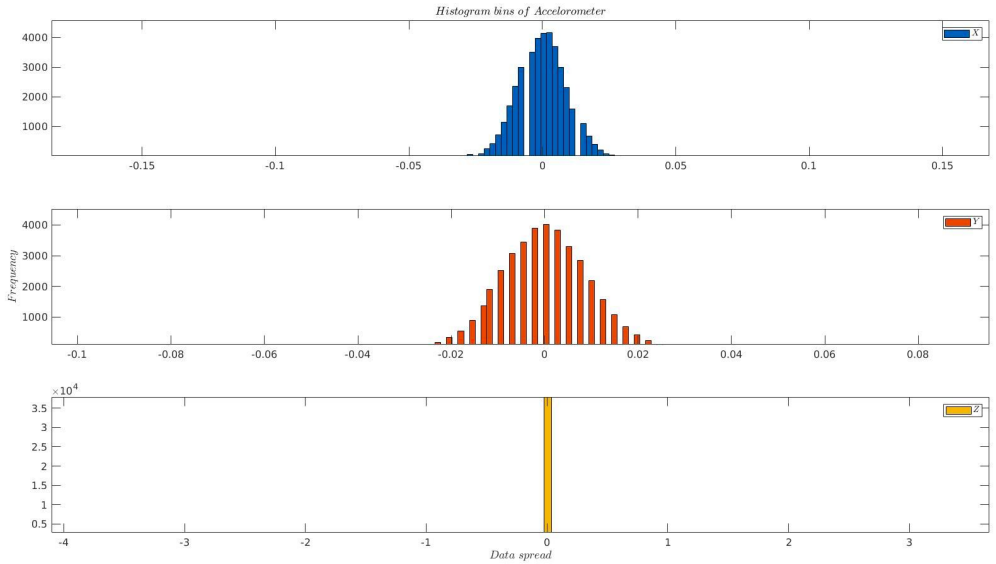
- To implement an “app” that does activity recognition (standing still, walking and running) based on the data from the phone’s sensors.



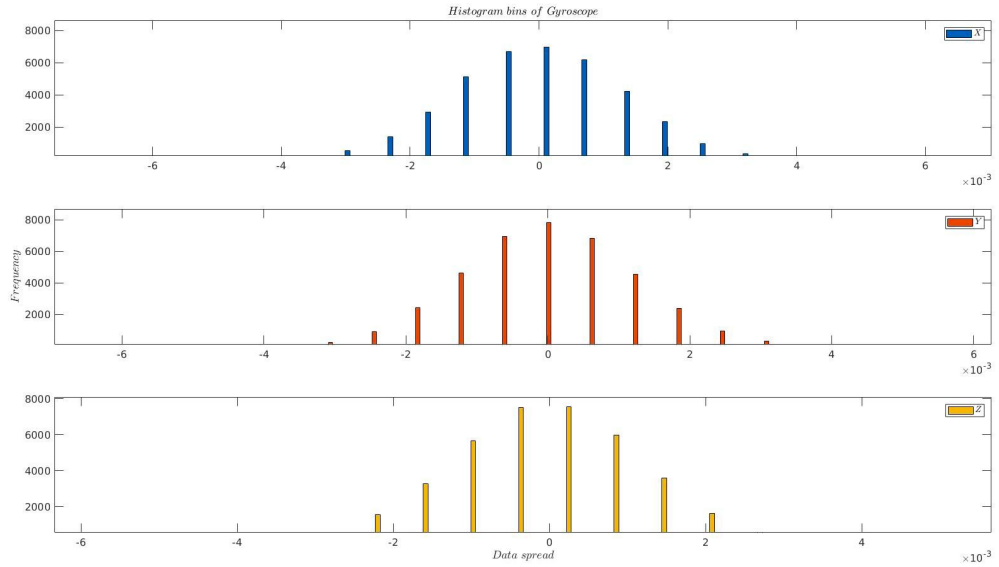
Method 1


Data from the phone
Used Google's orientation
Threshold values are set to recognize the activity
Initial testing and moving to our own solution for the attitude estimation
Data from the phone
Estimate the orientation using EKF
Threshold values are set to recognize the activity

Bias Calculation (acc, gyro)

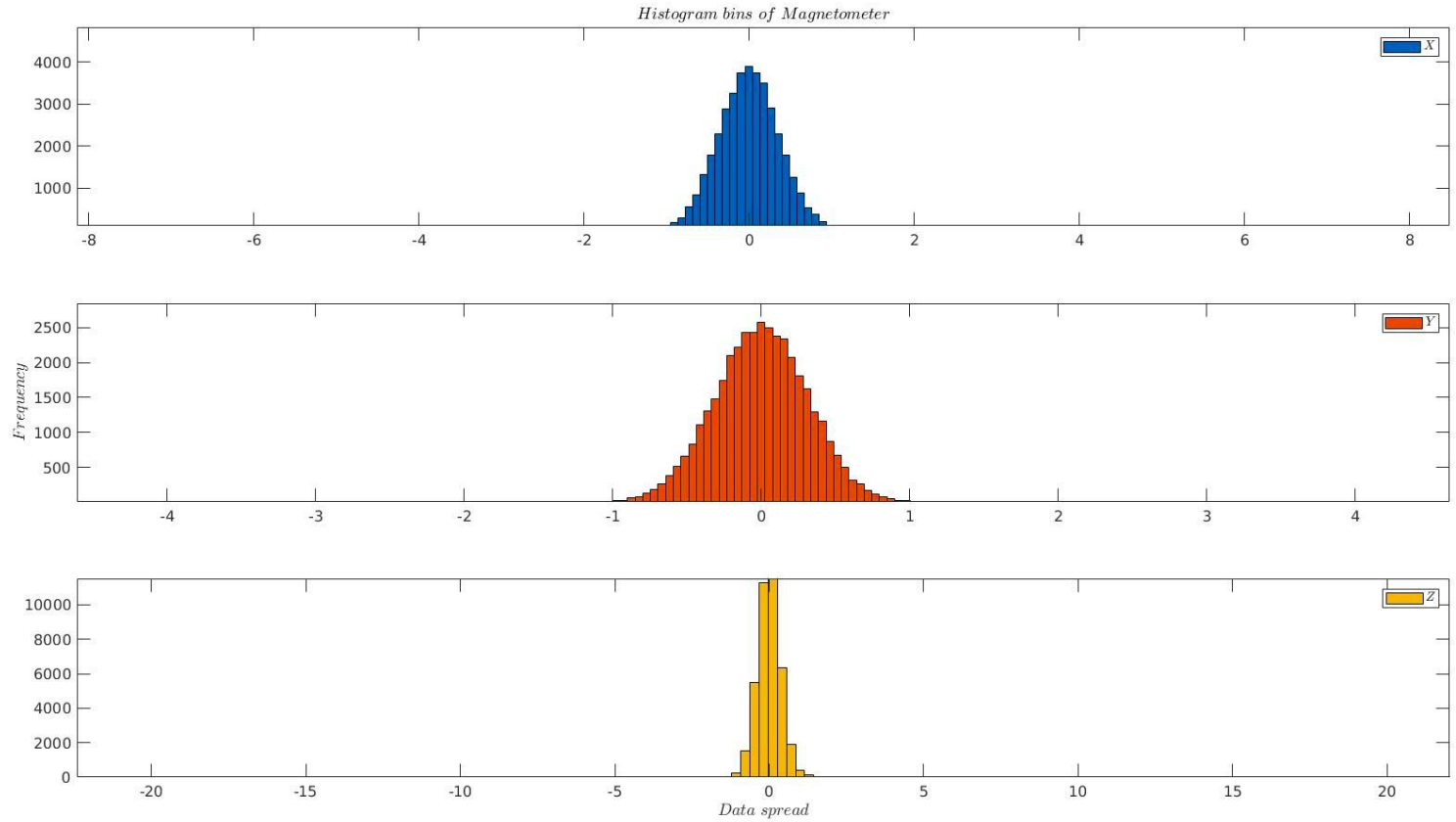


Accelerometer bias calculation



Gyro bias calculation

Bias Calculation (mag)



Orientation

Nonlinear sensor model based on quaternion representation:

$$x_{k+1} = f(x_k, u_k) \quad where\ x_k = q_k \in R^4, and\ u_k = \omega_k \in R^3$$

State transition is modeled as

$$q_{k+1} = \left(\cos\left(\frac{T||\omega_k||}{2}\right) I_4 + \frac{1}{||\omega_k||} \sin\left(\frac{T||\omega_k||}{2}\right) S(\omega_k) \right) q_k$$

$$\tilde{\omega}_k = \omega_k - \delta\omega_k + w_k, \qquad where\ w_k\ is\ white\ noise$$

Extended Kalman Filter

The predicted next state can be approximated as

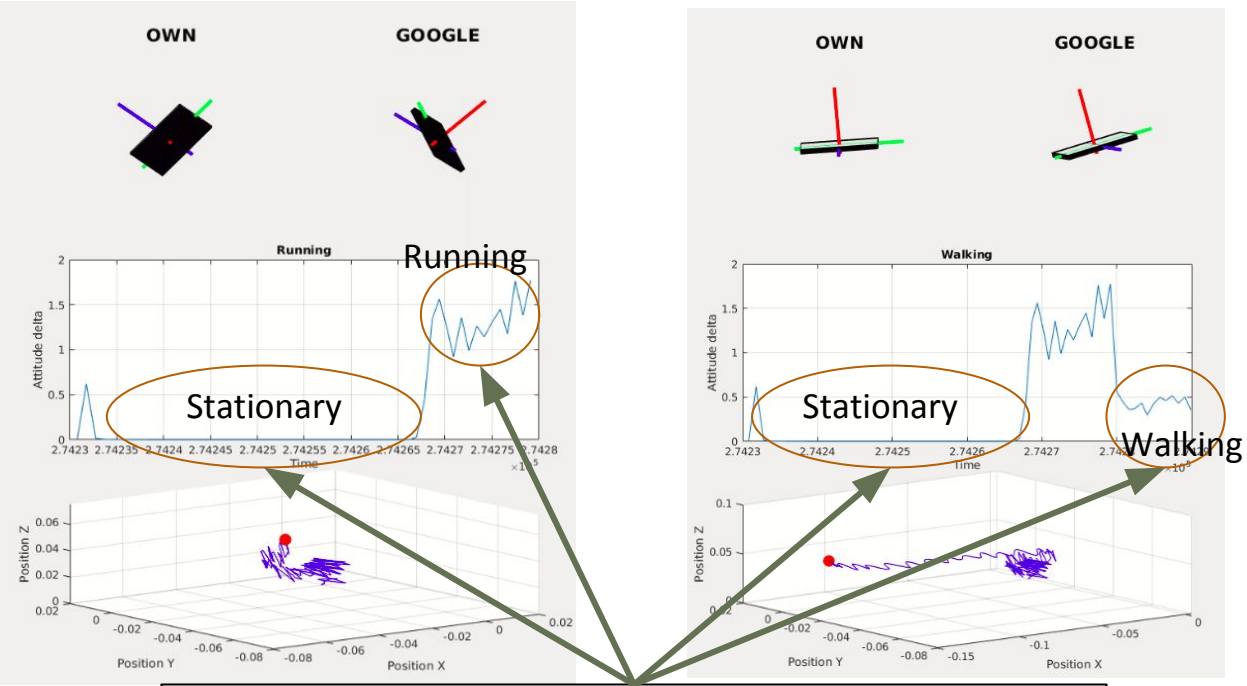
$$q_{k+1} \approx \underbrace{\left(I + \frac{1}{2}S(\omega_k)T\right)}_{F} q_k + \underbrace{\frac{T}{2}\bar{S}(q_k)}_{G} w_k$$

Accelerometer equation: $y_k^a = Q^T(q_k)(g^0 + F_k) + e_k^a$ $g^0: nominal\ gravity\ vector$

Magnetometer equation: $y_k^m = Q^T(q_k)m^0 + e_k^m$ $m^0: earth\ magnetic\ field, m^0 = \begin{pmatrix} 0 & \sqrt{m_x^2 + m_y^2} & m_z \end{pmatrix}^T$

Update the state estimate using $H_k^a = \frac{\partial y_k^a}{\partial q_k}$ and $H_k^m = \frac{\partial y_k^m}{\partial q_k}$

Results

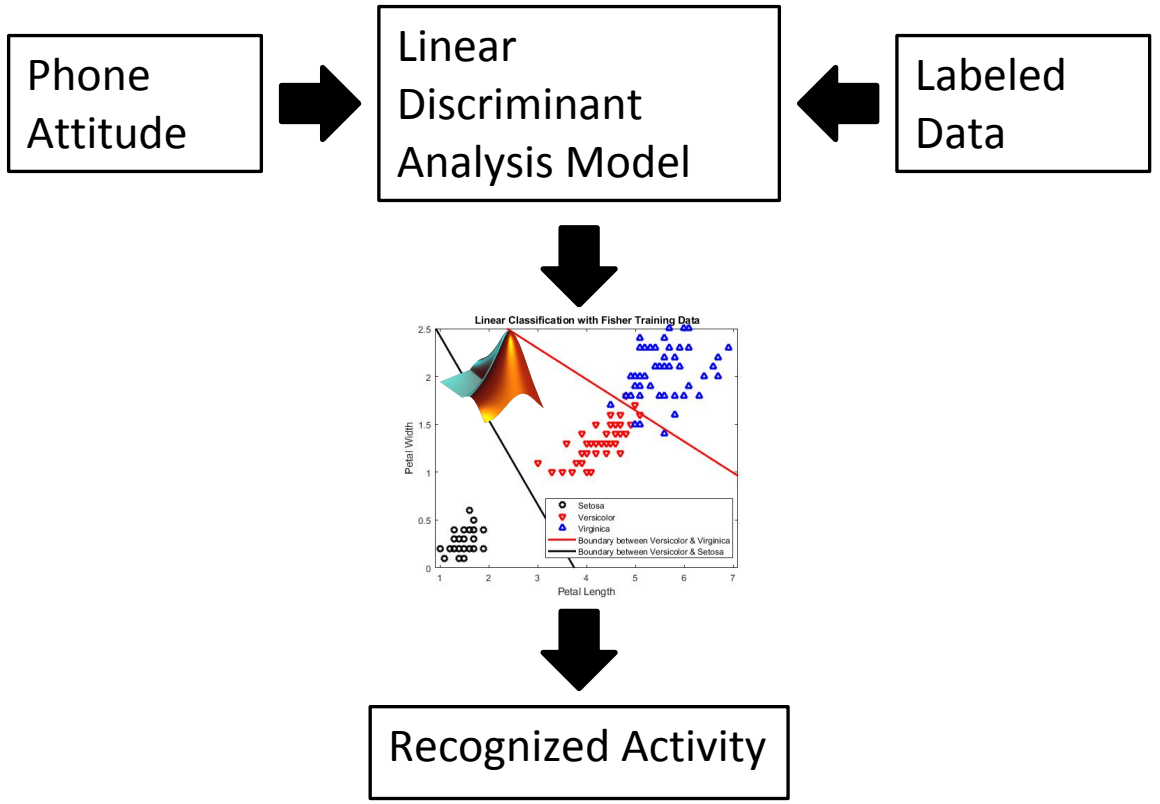


Filtering using Majority voting in a sliding window using the averaged gradient of the quaternions and applying empirically calculated thresholds.

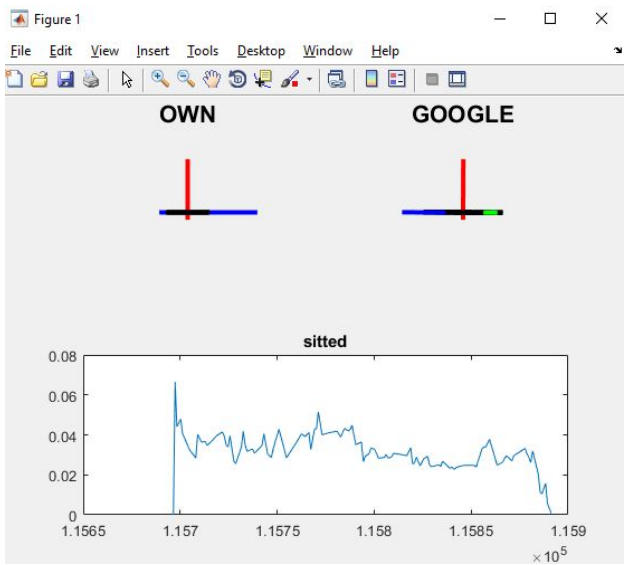
Method 2: Linear Discriminant Analysis

- Machine Learning method for supervised learning tasks
- Reduce the problem dimensionality to fit the data into a lower--dimensionality classification

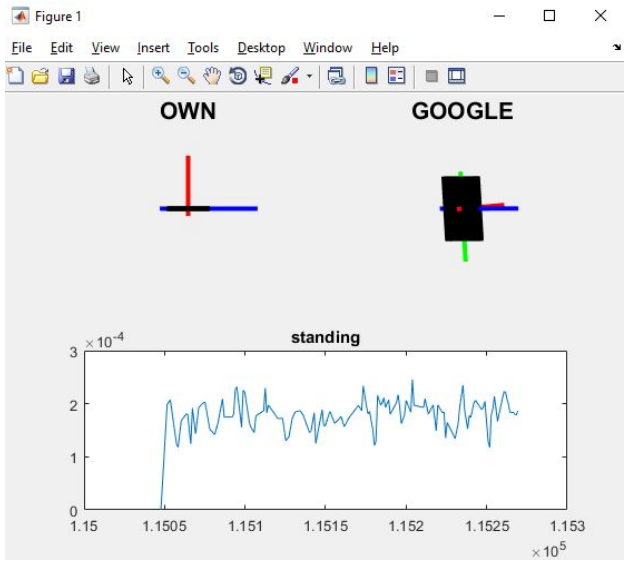
- Recorded data:
  - **sitting:** approx. 1000 points
  - **standing:** approx. 2000 points
  - **walking:** approx. 4000 points
  - **running**: … it's coronavirus time, I can't run in my house...
- Each datapoint consists of:
  - orientation
  - linear and angular velocities
  - linear acceleration



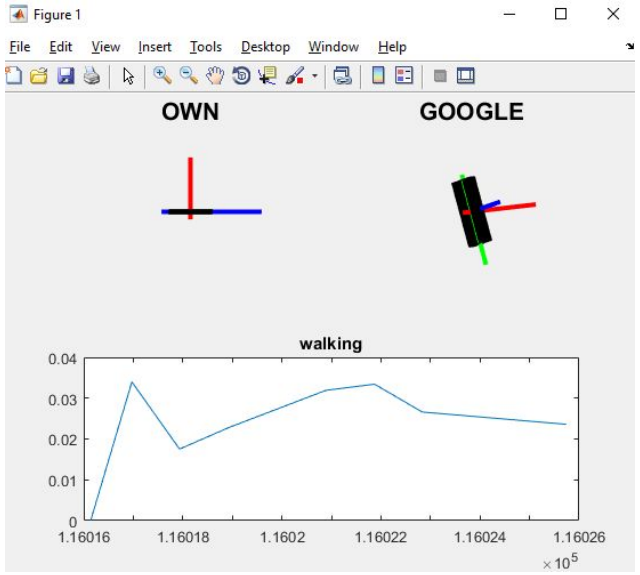
Results: Accuracy



Sitting: 81% correct



Standing: 83% correct



Walking: 89% correct

Conclusion and Future Work

- Method 1: Quaternion calculation with
  - Performance at par with Google's orientation calculation;
  - Also added position estimate and attitude delta view in real time;
  - Some drift while static due to integration of acceleration bias;
- Future work:
  - Extended to use camera with pose estimation. Camera features can be used as constraints to the attitude propagation to get more better estimates by tracking KLT features;
  - Real time performance can be attained by formulating the problem as factor graphs and replacing it with the filtering step. The solution would be more robust for pose estimation given some visual features are present in the environment for tracking;

- Method 2: LDA
  - Accuracy was above 80% with unfiltered data;
  - More data available should increase reliability;
  - Can be extended into quadratic models
- Future work:
  - LDA can be improved with filtered data (remove jitter noise)
  - Collect more data for the existing / future classes