Android App to Classify Activity from Smartphone Accelerometer Data

Anirban Majumder, 17EE35017

Course: Programmable and Embedded Systems (EE60098)

Department of Electrical Engineering
Indian Institute of Technology Kharagpur

Supervisor: Professor Aurobinda Routray

September 17, 2021

Abstract

The aim of this project is to develop an Android application to classify human activity, such as standing still, walking, running, ascending or descending stairs, using accelerometer data collected from the inbuilt sensors of an Android smartphone.

1 Objectives

The objective of this project is to develop an Android app for the following operations -

1. Collect raw accelerometer data from a smartphone's inbuilt sensors.
2. Remove Gaussian sensor noise from the raw data using a Kalman Filter.
3. Using the filtered data, classify human activity into five classes, namely -
   - i Standing Still
   - ii Walking
   - iii Running
   - iv Ascending Stairs
   - v Descending Stairs

2 Methodology

This section provides a brief overview of the various aspects of this project along with the methodology and formal definitions.

2.1 Kalman Filter

The Kalman Filter is a Linear Quadratic Gaussian Estimator used to estimate true system states when the original measurements are corrupted by Gaussian noise and other inaccuracies. Since smartphone sensors such as the 3-axis accelerometer are prone to significant amounts of sensor noise, the Kalman Filter is an ideal algorithm to adaptively estimate true acceleration values.

Consider an unregulated, open-loop discrete time system defined by the following state space equations -

$$\begin{aligned} x_{k+1} &= \Phi_k x_k + \omega_k \\ y_k &= H x_k + \nu_k \end{aligned} \tag{1}$$

where ω_k and ν_k are the process and sensor noise at the k-th sample, Φ_k is the state evolution matrix at the k-th instant, and H is the measurement matrix.

Consider Q to be the covariance matrix of ω_k, R to be the covariance matrix of ν_k, K_k to be the Kalman gain at the k-th instant, $\hat{x}_k$ to be the corresponding estimated state, and P_k be the error covariance matrix.
Then the Kalman Filter can be expressed as the following set of iterative equations -

$$\begin{aligned} K_k &= P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - H\hat{x}_{k|k-1}) \\ P_{k|k} &= (I - K_kH)P_{k|k-1} \\ \hat{x}_{k+1|k} &= \Phi_k\hat{x}_{k|k} \\ P_{k+1|k} &= \Phi_kP_{k|k}\Phi_k^T + Q \end{aligned} \tag{2}$$

The above equations are modified versions of the ones found in [1]. The system state vector is defined as $x_k = [d_x, d_y, d_z, v_x, v_y, v_z, a_x, a_y, a_z]_k^T$, which consists of the displacements, velocities and accelerations of the phone in 3-dimensions.

For our application, we use the constant acceleration model. Let h_k be the k-th sampling period, then this model can be represented as follows -

$$\Phi_k = \begin{bmatrix} 1 & 0 & 0 & h_k & 0 & 0 & h_k^2/2 & 0 & 0 \\ 0 & 1 & 0 & 0 & h_k & 0 & 0 & h_k^2/2 & 0 \\ 0 & 0 & 1 & 0 & 0 & h_k & 0 & 0 & h_k^2/2 \\ 0 & 0 & 0 & 1 & 0 & 0 & h_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & h_k & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & h_k \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

2.2 Classification

The Kalman Filter used can estimate the displacement, velocity and acceleration of the smartphone in three axes. Therefore, the logical choice for classifying motion would be to use all of the estimated states. However there are multiple obstacles to this approach.
Displacement and velocity cannot be used since the constant gravitational acceleration causes these quantities to constantly change with time, even when the

phone is at rest. Using a high pass filter to remove the effect of gravitation would work in theory, but in practice, even the filtered acceleration data exhibited small amounts of skewed sensor noise, which resulted in the displacement and velocity components drifting with time. Thus only acceleration states were considered for classification.
Another problem encountered was the variable tilt of the device. While this can be corrected using gyroscope data, not all devices have a gyroscope. Therefore, to make the classification immune to tilt, only the two-norm of the filtered acceleration vector was used.

A simple histogram matching classifier was successfully deployed. A window of Kalman filtered absolute acceleration data was maintained in real-time. At each sampling instant, the normalized histogram of this window was computed and compared with standard pre-calculated discrete probability distributions of each activity. For comparing two distributions, the asymmetric discrete Kullback-Leibler Divergence [2] was used, given by -

$$KLDiv(p, q) = \sum_{x} p(x) log(\frac{p(x)}{q(x)}) \tag{4}$$

A smaller KL Divergence indicates that the two distributions are more similar and vice versa. Therefore, the standard distribution which has the smallest KL Divergence with the obtained histogram, is the correct category for classification.

3 Algorithms

During each instant where the sensor is sampled, the filtered state is estimated using a Kalman Filter as shown in Alg 1. The two-norm of a window of estimated acceleration values are stored in a queue of fixed length and updated at each sampling interval.

In the classification stage, first the normalized histogram of the window of acceleration values is computed following Alg 2. Next, the Kullback Leibler Divergence of this histogram with all pre-calculated discrete probability distributions is computed following Alg 3. The activity corresponding to the probability distribution which has least divergence from the current histogram is displayed as the current activity of the smartphone user. The overall algorithm is depicted in Alg 4.

Algorithm 1 Kalman Filtering

Initialize $P = 5I_9$, $Q = I_9$, $R = 0.1I_3$;
Initialize H, $\hat{x} = [0, 0, 0]$;
Initialize a queue : $window$, $prevTime =$ **time()**;
for each measurement y **do**
    $h_k \leftarrow$ **time()** $- prevTime$;
    Compute Φ from h_k;
    $K \leftarrow PH^T(HPH^T + R)^{-1}$; ▷ Compute Kalman Gain
    $\hat{x} \leftarrow \hat{x} + K(y - H\hat{x})$; ▷ Update Estimate
    $P \leftarrow P - KHP$; ▷ Compute Posterior Error Covariance
    $\hat{x} \leftarrow \Phi\hat{x}$; $P = \Phi P\Phi^T + Q$; ▷ Project Ahead
    $window$**.enqueue(**$||H\hat{x}||_2$**)**; $window$**.dequeue()**; ▷ Update $window$
    $prevTime \leftarrow$ **time()**
end for

Algorithm 2 Computation of Normalized Histogram

Require: Array of acceleration values : $window$, Max acceleration : ub, Min acceleration : lb, Number of bins : B;
Ensure: Normalized histogram of array : ρ
assert $\max(window) \leq ub$ & $\min(window) \geq lb$;
Initialize ρ with all zeros;
for $i = 1$ **to** length($window$) **do**
    $idx \leftarrow$ (int)$\frac{(window[i]-lb)(B-1)}{ub-lb}$;
    $\rho[idx] \leftarrow \rho[idx] + 1.0/\text{length}(window)$
end for
return ρ

Algorithm 3 Computation of Kullback Leibler Divergence

Require: First probability distribution : ρ_1, Second probability distribution : ρ_2;
Ensure: Kullback Leibler Divergence : $KLDiv$;
Initialize $KLDiv \leftarrow 0$;
Let $eps \rightarrow 0$;
assert length(ρ_1) = length(ρ_2);
for $i = 1$ **to** length(ρ_1) **do**
    $KLDiv \leftarrow KLDiv + \rho_1[i]log(\frac{\rho_1[i]+eps}{\rho_2[i]+eps})$
end for
return $KLDiv$

Algorithm 4 Activity Classification

Initialize standard distributions $\rho = \{\rho_1, \rho_2, ..., \rho_5\}$ corresponding to each activity;
Initialize a distribution p_{cur} corresponding to the current activity;
Initialize *activity* $\leftarrow$ *unknown*
Get queue : *window* from the Kalman Filter;
for each sampling instant **do**
    $\rho_{cur} \leftarrow$ **NormalizedHistogram(***window***)**;
    Find Kullback Leibler Divergence of ρ_{cur} with $\rho_i \forall i = 1, 2, ..., 5$;
    Find ρ_k corresponding to the least KL Divergence;
    activity $\leftarrow$ activity corresponding to ρ_k;
end for

4 Results

Filtered acceleration data, for all activities, was sampled at a frequency of 25 Hz. Kalman Filtering was done on the device itself. An illustration of the tracking for data collected while going down a flight of stairs can be found in Fig 1.

Probability distributions of the acceleration values were computed offline on MATLAB using the ksdensity function at 101 equi-spaced bins ranging from $0m/s^2$ to $40m/s^2$. It was observed, that for each activity, the distributions had their mean at $9.8m/s^2$ which is the acceleration due to gravity, but exhibited different shapes and variances. For instance, running yielded a much more spread out distribution in comparison to walking. The distributions computed by ksdensity are shown in Fig 2.

After the standard distributions were copied to the device, classification using Alg 4 was done in real-time over a window of roughly 2 seconds, with fairly accurate results. A few images of real-time classification can be seen in Fig 3.

5 Conclusion

Kalman filtering proved to be an effective means of mitigating sensor noise from acceleration data. Classification using histogram matching proved to be very successful as an easy to implement and efficient technique for human activity classification.

The only issue with this technique was the fact that classification requires a fully

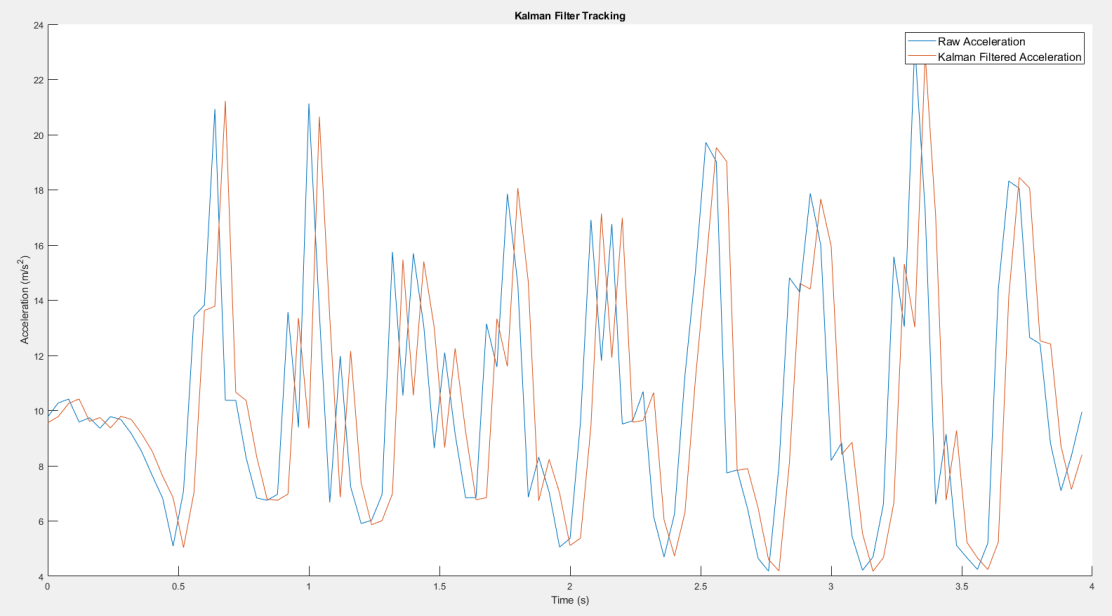


Figure 1: Kalman Filter Tracking

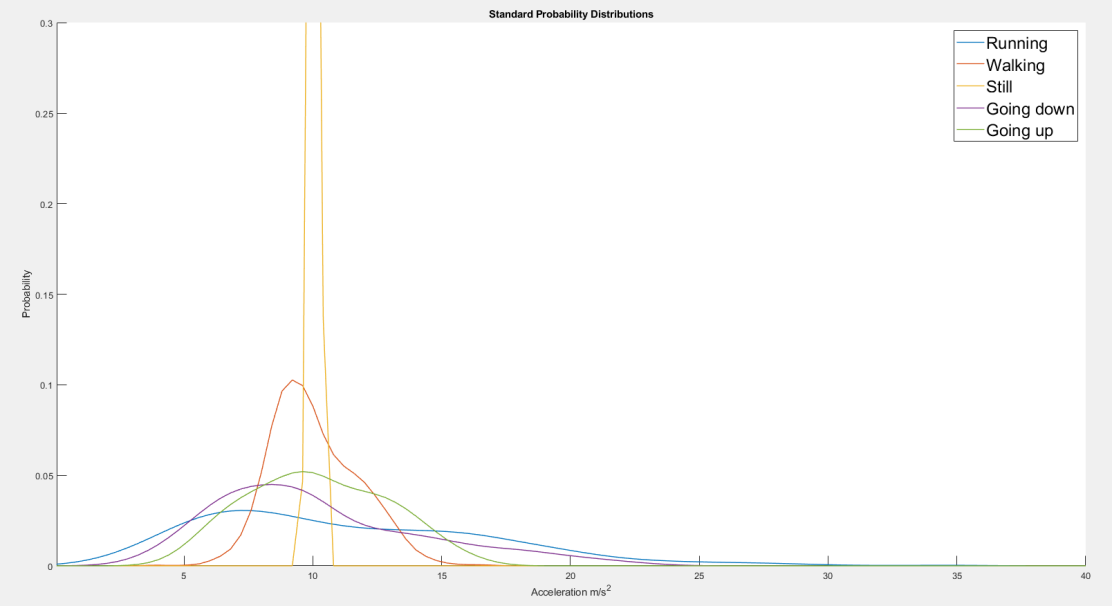
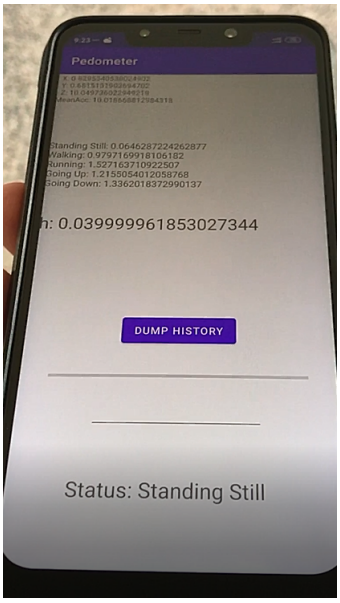
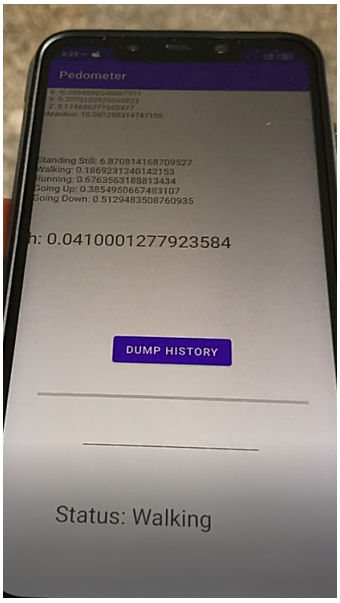


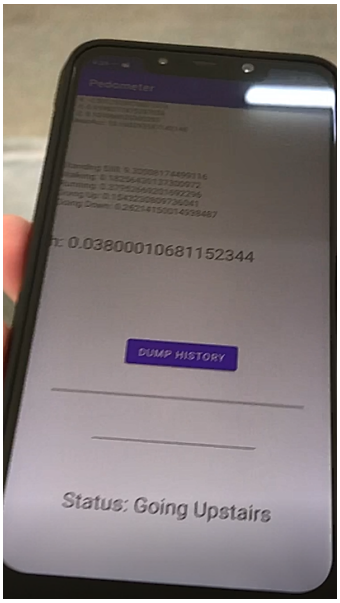
Figure 2: Standard Probability Distributions



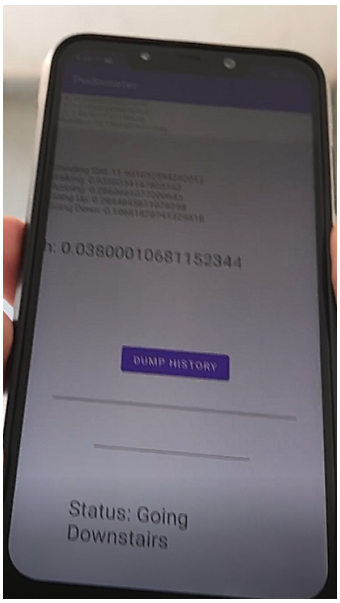
(a) Standing Still



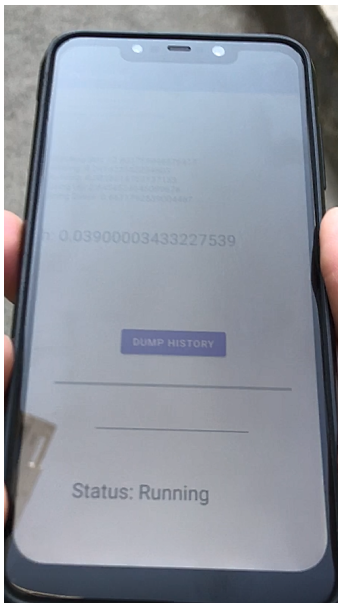
(b) Walking



(c) Ascending Stairs



(d) Descending Stairs



(e) Running

Figure 3: Classification screenshots

populated window of acceleration data. Therefore, sometimes, when a person is transitioning from one activity to another, the algorithm would misclassify due to the fact that the acceleration window contains data from two or more distinct activities. For example, if a person were to suddenly stop whilst running, the classifier output would transition from running → walking → standing still, over a period of 2 seconds (the time it takes to fully refresh the acceleration window data at the given sampling rate).

Future work could involve implementing more sophisticated classifiers such as Support Vector Machines, Neural Networks or Sequence Clustering.

References

[1] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 911–918, 2009.

[2] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, 03 1951.