**Walk or Run?**

**Term Project**

**Austin Whitney, awhitney**

**Joshua Malmberg, jmalmber**

**Mandeepa Mashhura, mmashhur**

Simon Fraser University

CMPT 353 : D100 - Data Science

**Introduction**

Activity detection is one useful metric that might be provided by fitness analysis software. One potential application of activity detection is in sports referee evaluation. In many sports, a good referee must run continuously throughout a match to keep close to the action and make accurate calls. However, many referees of amateur sports do not run and keep up with the players, leading to poor judgements. Measuring the amount of time a referee spends running during a match would allow amateur sports clubs to evaluate a major aspect of referee performance. Another possible application of this metric is in physical education. In many high school physical education classes, students are sent off-campus to run unsupervised. Measuring the time students spend running would allow teachers to assign accurate grades based on participation and effort. In this project, we aim to develop a robust data science tool that classifies whether a person is running or walking based on cell phone sensor data.

**Data Collection**

The data was collected by two subjects using smartphone applications. Two applications were used: Physics Toolbox Sensor Suite on Android[[1]](#footnote-1) and INSERT HERE on iPhone[[2]](#footnote-2). Using these applications, 3-dimensional linear acceleration and 3-dimensional rotational velocity data were recorded. The frequency of data observations varied between the different cellphones used to record, ranging from 50Hz to 500Hz. Subjects attached the cellphone to either their ankle or wrist, fastening it to avoid movement between their body and the measuring device. This is depicted in Figure 2 below. Data was collected for both left and right ankles and wrists. In total, approximately two hours of usable walking and running data was collected.

INSERT FIGURE HERE

**Data Treatment**

The treatment of the data consists of three stages: extract-transform-load, data cleaning, and feature extraction.

1. **Extract-Transform-Load**

Both phone applications output the sensor data as .csv files; however, the format used differed slightly between the two:

|  |  |  |
| --- | --- | --- |
|  | **iPhone Application** | **Android Application** |
| **Fieldnames** | Timestamp, ax, ay, az, gx, gy, gz | time, ax, ay, az, wx, wy, wz |
| **Record Delimiter** | “\n” | “,\n” |
| **Date Format** | YYYY-mm-dd HH:MM:SS.ffffff | HH:MM:SS.fff |

The data obtained was initially manually organized into directories based on recording parameters: application used, sensor position (wrist vs. ankle, left vs. right), and activity (run vs. walk). In total, there were 16 such directories, representing each unique configuration of parameters.

The program etl.py takes as arguments input and output directories. For each .csv file in the input directory etl.py does the following:

* Transforms the file into a canonical format,
* Converts the time string into a float indicating seconds since recording started,
* Adds columns to indicate the recording parameters,
* Writes the file to the output directory.

A Makefile and build targets were specified using this program, enabling ETL operations to be performed on different subsets of the original data. This approach scales with little to no modification and conveniently produces different training sets, allowing for performance comparisons with different subsets of the data.

1. **Data Cleaning**

The data cleaning process consists of four stages:

1. **Aggregating Observations with Identical Timestamps**

Due to high sampling frequency and limited precision in the time values provided by the data collection applications, some of the .csv data files contain adjacent records with identical time fields. This interferes with tools used later in the data pipeline. To address this issue, records with identical timestamps were combined by taking the mean. The time values provided by the data collection applications are precise to 1/1000th of a second. Therefore, aggregation will not appreciably reduce the resolution of the dataset.

1. **Removing Discontinuities**

As mentioned previously, the data collection applications used often stopped recording unpredictably. These discontinuities will interfere with the transformation of linear acceleration data performed in the next stage, as well frequency analysis performed later in the data pipeline.

Discontinuities in a data frame were detected using the following procedure:

1. Compute a series diff indicating the difference in time value between rows.
2. Apply the function scipy.stats.zscore() to the series diff and create a new series discont with 1’s in each position where the z-score is above a certain threshold.
3. Take the cumulative sum down the series discont and add the resultant series cont\_groups to the data frame.

Continuous sections of the data frame will have the same value in series cont\_groups. Thus, iteratively filtering the data frame by value of cont\_groups will produce slices of the data frame without discontinuities. Each of these subsets is then written to a separate .csv file, except those with relatively few records, which are discarded. Using a for loop here is justified because a high threshold is used for the z-score, guaranteeing that the number of discontinuities is small compared to the size of the data frame.

1. **Transforming Linear Acceleration Data to a Common Reference Frame[[3]](#footnote-3)**

Cellphones sensors record acceleration relative to the axis of the phone. The phone may accelerate uniformly in a single direction, however if it rotates as it moves the acceleration recorded by the phone will change directions.

To remove the effects of rotations from the acceleration data, the rotational velocity data was integrated using scipy.integrate.cumtrapz() to calculate the angular orientation of the phone at each time step, relative to the initial angular orientation. The linear acceleration at each timestep was then transformed using the 3-dimensional rotation matrices:

When multiplied with the acceleration vector , these matrix transformations rotate the acceleration vector degrees about the x-axis and degrees about the y-axis. These linear transformations were computed component wise using functions numpy.cos() and np.sin(). The effect of this transformation algorithm on a tailored data stream is depicted in Figure 3 below.

INSERT FIGURE HERE

During data collection, it was ensured that the y-axis of the phone was parallel to the subject’s ankle or wrist. Thus, rotation about the y-axis corresponds to rotation of a subject’s leg or forearm while running. While running, rotation of the subject’s leg primarily occurs when the subject turns. Thus, not rotating about the y-axis equates to ignoring turns and interpreting the data as if the subject ran in a straight line, ensuring consistency in the data. Furthermore, since most people do not rotate their wrists or tilt their torso sideways while running, this effect is ignored.

1. **Filtering**

A low-pass Butterworth filter with frequency threshold INSERT HERE was applied to the transformed linear acceleration data in each data stream. Since people run with stride frequency <5Hz[[4]](#footnote-4), this filter removes high-frequency noise from the data without affecting the meaningful acceleration data. The effect of this filtering on a data stream is depicted in Figure 4 below.

INSERT FIGURE HERE

1. **Feature Construction**

Discussion of FFT and values used to train ML model here.

**Data Analysis**

Discussion of ML model implementation here.

**Model Performance**

**Limitations**

One major obstacle to the data collection process was inconsistency in the behaviour of the applications used. Often the applications would stop making observations shortly after the subject began walking/running and only resume when the subject opened the app to stop the recording. This is likely due to the phone operating system not allowing the application to continue after the phone screen shut off. In other instances, the phone continued recording after the phone screen shut off, albeit inconsistently. This produced a data stream with 1-2 second gaps spread throughout it. Several hours of both walking and running data was recorded before these obstacles were identified. Due to these obstacles, much of this data was unusable.

Another obstacle to the data collection process was the difficulty to start the recording after the subject secured their phone in place. This resulted in several seconds of stationary observations at the beginning of each data stream, creating noise in the data stream since these stationary intervals were marked either as walking or running. This noise was also present anytime the subject stopped briefly walking or running briefly during the data collection.

**Conclusion**

**Project Experience Summary**

**Joshua Malmberg, 301348298**

1. https://www.vieyrasoftware.net/physics-toolbox-sensor-suite [↑](#footnote-ref-1)
2. INSERT HERE LINK TO IPHONE APP [↑](#footnote-ref-2)
3. The algorithm used in this step was derived from an algorithm presented in the master’s dissertation of Maria Yousefian. The full dissertation can be found at: http://summit.sfu.ca/item/17204 [↑](#footnote-ref-3)
4. M. Kale, C. Acikada, “Effects of stride length and frequency training on acceleration kinematic, and jumping performances.,” *Sports Science Review*, vol. 25, no. 3-4, pp. 243-260, doi: 10.1515/ssr-2016-0013. [↑](#footnote-ref-4)