Inferring Pitch and Roll using Accelerometer and its applications

Category: Athletics and Sensing Devices
SUNet ID: 06512865
Name: Saravana Kumar Rathinam and Aditya Saravana [1]

Introduction

Device orientation can be inferred using accelerometer signals. Using this inferred tilt can improve accuracy of Human Activity Detection (HAR) tasks at lower power consumption. Yaw, Pitch and Roll are used to describe an objects orientation with respect to the world frame of reference. Usually this information is obtained from the IMU by using a combination of accelerometer, magnetometer and gyroscope. Accelerometer is a low power device but Gyroscope and Magnetometer can consume 10x the power of an accelerometer on mobile devices [1].

The accelerometer values are specified in multiples of g. So say the device is lying flat on a table with screen pointing up the values for a_x, a_y would be 0 and a_z would be -1.0. Since $+Z$ axis points in the direction opposite to center of earth. Also the Pitch θ is the rotation around X axis and Roll ϕ is the rotation around Y axis. Together pitch and roll are used to infer tilt of the object. If device tilt can be inferred using accelerometer signals alone, they can be use as features for running models on low power devices.

Related Work

The NXP AN3461 [7] manual demonstrates how simple trigonometric formula can be used to determine orientation of a microcontroller device. Some of these methods are also found in online documentation of Smart phone providers [2][1]. The state of the art attention based models [5] use a fusion of accelerometer, gyroscope and magnetometer data. Sensor fusion is needed to compensate for drift in Gyroscope measurements and noisy accelerometer samples [10].

HAR detection systems that have power limitations only use accelerometer [11]. Applications running on low power devices thus choose not to use Gyroscope signals [11]. Device orientation can be a useful feature in human activity recognition (HAR) tasks and some advances in HAR accuracy have been based on sensor fusion [5].

Dataset and Features

The Human Activity Recognition using Smart Phone data set [3] our test data set to validate our hypothesis. The dataset contains movement of 30 volunteers in the age of 19-48 years. The activity performed was walking, walking upstairs, walking downstairs, sitting, standing and laying. The data was collected using a smartphone placed on the waist of the participant. The data include accelerometer and gyroscope data captured at 50Hz. The dataset also included derived 531 features. These features were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise [3]. The features also included Jerk signals, frequency domain signals using FFT of time series, mean, std-dev, energy, kurtosis and angle between vectors.

We also collected data using a custom data collector app [9]. The data collected was raw accelerometer a_x, a_y and a_z collected at 100Hz. Motion data from the watch that includes pitch, yaw, roll, quaternions and rotation matrix was also collected but not used for the project. The θ and ϕ vectors were derived using the raw data as well as the absolute value $\sqrt{a_x^2 + a_y^2 + a_z^2}$, computed velocity vector $[v_x, v_y, v_z]$ and computed displacement $[d_x, d_y, d_z]$.

[1] Aditya Saravana is the authors son and is in 5^{th} grade at Easterbrook Discovery School, San Jose. He is helping his father with an Apple WatchOS App to collect data and build fun ML workout App for the models produced.

Methods

In order to validate our hypothesis that pitch and roll (tilt) can be inferred, we needed a large data set. The custom data collection had about 200 sessions of users performing different action types was not sufficient. So we used the HAR data set from [3].

The second step was to now train a model using the inferred tilt signal on the custom data we collected. This model was then released as part of a Watch OS App. As of writing this report the App is still in review at the App Store [8].

Deriving Pitch and Roll

When the device is stationary the pitch and roll can be computed with simple trigonometric equations. Say a device that was lying flat on the ground is rotated by θ degrees along the Y axis, we can compute θ (roll) as follows [7]: $\theta = \tan^{-1}(\frac{a_z}{a_x})$. Similarly, the pitch can be computed as $\phi = -\tan^{-1}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right)$

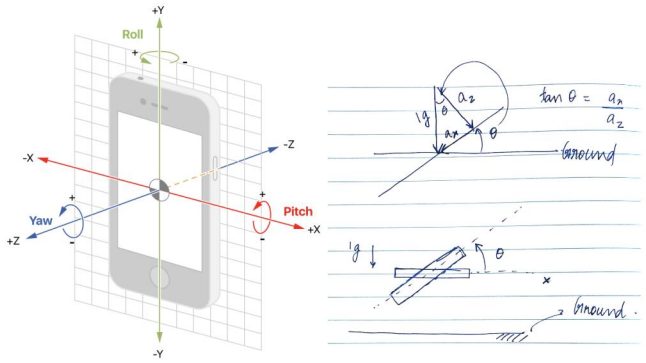


Figure 1: Schematic of θ and ϕ calculations and device orientation

These results are valid for stationary objects, when the object is in motion there is noise added by angular momentum in the rotation and inherent noise associated with reading the accelerometer. The measured linear acceleration can be represented as $\tilde{a} = a^{(g)} + a(l) + \eta_{acc}$ [10], where $\eta_{acc} \sim N(0, \sigma^2_{acc})$. For a stationary object the measured value is just the acceleration due to gravity and noise $\tilde{a} = a^{(g)} + \eta$.

Gyroscope measures angular velocity $\tilde{\omega}$. The measured angular velocity can be modeled as $\tilde{\omega} = \omega + b + \eta_{gyro}$, where ω is the true angular velocity, b is the bias and η_{gyro} the additive zero mean Gaussian noise $\eta_{gyro} \sim N(0, \sigma^2_{gyro})$. To derive pitch and roll from Gyro we would need to integrate the angular velocity. This can be done using Taylor series expansion [4]. As seen below.

Using Taylor expansion for deriving sensor readings

The Taylor series is a series expansion of a function about a point. A one dimensional Taylor expansion of an infinitely differentiable function $f(x)$ about $x = 0$ is given by:

$$f(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f(0)}{2!}x^2 + \frac{f'(0)}{3!}x^3 + \cdots = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k$$

where $(f^{(k)}(0))$ is the (k)th derivative of (f) evaluated at 0. Using this we can derive the following using simple substitution:

$$\phi(t + \Delta t) \approx \phi(t) + \tilde{\omega_1}(t)\Delta t + \frac{1}{2}(\tilde{\omega_1}(t) - \tilde{\omega_1}(t - \Delta t))\Delta t$$

$$\theta(t+\Delta t) \approx \theta(t) + \tilde{\omega_2}(t)\Delta t + \frac{1}{2}(\tilde{\omega_2}(t) - \tilde{\omega_2}(t-\Delta t))\Delta t$$

Where $\tilde{\omega} = [\tilde{\omega_0}, \tilde{\omega_1}, \tilde{\omega_2}]^T$ is the angular velocity vector measured by the gyroscope. Similarly velocity and displacement can be approximated using the following:

$$v(t+\Delta t) \approx v(t) + a(t)\Delta t + \frac{1}{2}(a(t)$$

$$d(t+\Delta t) \approx d(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + \frac{1}{3!}(a(t) - a(t-\Delta t))\Delta t^2$$

Both the $v(t)$ and $(d(t)$ are subject to heavy drift because of cascading error terms. This makes these equations difficult to use for accurate calculation of these values over time. However they can act as a good input signal for deep learning models to pick up a velocity and displacement hint.

This drift in ϕ and θ can be corrected if we have both gyroscope and accelerometer available. Using a fusion method [10]:

$$\theta(t+\Delta t) \approx \alpha \left(\theta(t) + \tilde{\omega_2}(t)\Delta t + \frac{1}{2}(\tilde{\omega_2}(t) - \tilde{\omega_2}(t-\Delta t))\Delta t\right) + (1-\alpha)\left(\tan^{-1}(\frac{a_z(t)}{a_x(t)})\right)$$

Where $\alpha \in]0, 1[$ is a scaling factor. This has the added benefit of removing drift from gyroscope integration and reducing high frequency noise from accelerometer.

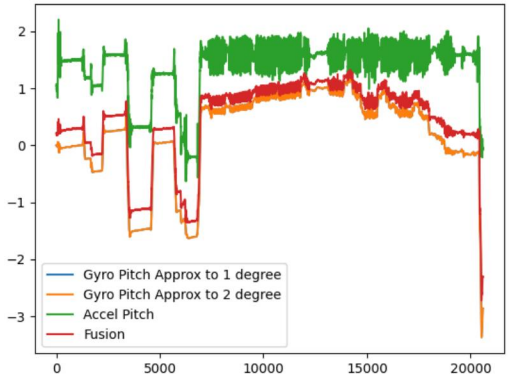


Figure 2: Plot of θ inferred values using above method. The data sample is from the [3] dataset. Note that the gyro pitch approximated to first degree and second degree overlap, indicating the error term is small.

Data collection

A custom data collector was used to obtain accelerometer and gyro data from Apple Watch. The data collected was labelled for different activity - jump, dodge right, dodge left and duck. Here are the screen shots of the app's workflow [9]:

Figure 3: Workflow of collector.

Experiments

The first experiment was to deduce if inferred tilt served as a useful signal for improving model accuracy.

1. Trained a simple CNN model that used all **561** features of the HAPT database.
2. Trained the same CNN model architecture with all Gyro related features removed from HAPT database. Reducing the number of features to **348**.
3. Trained the same CNN model architecture with all Gyro related features removed and added the inferred θ and ϕ. With just two additional features taking the feature count to **350**.

The CNN architecture used was $[conv2d \rightarrow maxpool2d \rightarrow conv2d \rightarrow maxpool2d \rightarrow flatten \rightarrow dropout(0.5) \rightarrow dense]$. The training was carried out for 100 epochs, with *categorical cross-entropy* as the loss function (multi class classification). The optimizer used was *adam*.

Using the result of the first experiment the following models were trained for the workout collector App:

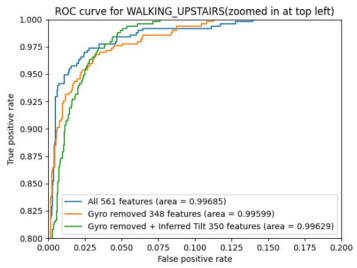
1. **Simple CNN** - $[conv2d(relu) \rightarrow maxpool2d \rightarrow flatten \rightarrow dropout(0.5) \rightarrow dense(numclasses, softmax)]$
2. **MLP** with dropout - $[dense(relu) \rightarrow dropout(0.15) \rightarrow dense(relu) \rightarrow dropout(0.15) \rightarrow flattendense(numclasses, soft$
3. **Logistic Regression** with regularization $[flatten \rightarrow flattendense(numclasses, softmax, l1(0.0), l2(0.1))]$
4. **CNN feeding into LSTM** with dropout - $[conv2d \rightarrow lstmbatchnormalization \rightarrow dense(numclasses, softmax)]$
5. **Large CNN** with dropout - $[(conv2d(relu) \rightarrow maxpool2d \rightarrow)^2 \rightarrow flatten \rightarrow (dense \rightarrow batchnormalization \rightarrow activation(tanh) \rightarrow dropout(0.5))^3 \rightarrow dense(numclasses, softmax)]$

Dropout layers and regularization was used where possible to reduce overfitting. Batch normalization layers were used to stabilize the training and reducing the number of epochs needed. A train/test split of 0.6/0.4 was used with data augumentation using additive zero mean gaussian noise with variable σ tuned as a hyper-parameter. The model hyperparameters were tuned using Bayesian Optimization package [6].

Each session of the training set was divided into equal sized frames. The frame length (called context length) was tuned as a hyper-parameter. Wrap around padding was applied at the beginning and end of each session to make it's length a multiple of context length. ROC curves were obtained on the framed sessions. A total of 706 models were trained.

Results

The two additional inferred tilt features improved the accuracy of the CNN model. The ROC curves for the different classes also show a slight improvement when the inferred tilt features are used:



Experiment	Loss	Accuracy
Baseline	0.1855	0.9484
No Gyro Features	0.2708	0.9026
No Gyro Features + Inferred Tilt	0.2439	**0.9040**

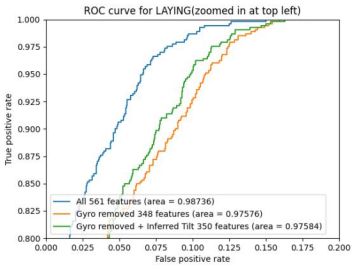


Figure 4: ROC Curves of different featurization schemes

The ROC curves for frame level accuracy of different models showed interesting results. The MLP model seems to have a good accuracy in all classes. However the deep CNN model had the best AUC in DODGERIGHT and DODGELEFT. The DUCK class was the most hard to predict. Probably because it was similar to the JUMP class.

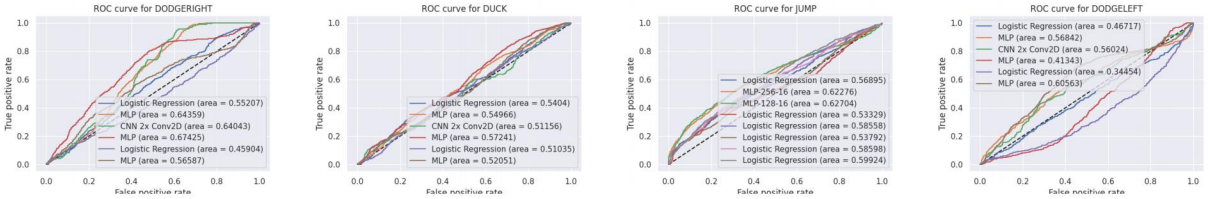


Figure 5: ROC Curves of frame level accuracy of different model architectures

Activity Detection App

The best model from the above training exercise was used in a fun workout game that detects human movements - jump, dodge left, dodge right, and duck. Here are some screenshots from the game. This will be available for free on the Watch App Store.

Figure 6: Workflow of activity detection app.

Discussion and Conclusion

The experiment results suggest that inferred tilt is a useful signal in training motion classifiers. The accuracy of the motion classifier built for the ML workout app was much lower than expected. This may be related to the lack of sufficient training samples. Also larger number of features (similar to the 541 features in HAPT [3] database could help boost accuracy. Even though the frame accuracy was low, the per session accuracy of the end to end system (given a prior) seemed high while using the game. The policy used on the app to have at-least one frame have a probability score of > 0.5 helped improve the App's accuracy.

Contributions

The team:

Saravana Kumar Rathinam designed the experiments, methodology, and trained the models. He experimented with several different modeling approaches and narrowed down on a few. He also helped with the server setup for uploading the data from the collector and participated in some data collection sessions. He is an aspiring ML engineer and hopes someday he will completely understand the EM algorithm.

Aditya Saravana (my son) programmed, designed, debugged, and streamlined both the collector and main AI Workout app. He also collected 130+ sessions of motion data, designed the app landing page, wrote the instructions for the collector's GitHub repository, and as of writing this report, is in the process of publishing the main app to the WatchOS App Store. Aditya is graduating from his elementary school this year and looks forward to join the group of middle schoolers in the Fall.

Also, a special thanks to **Aditya**'s friend **Igor Ambartcumov** for helping us collect some sessions and **Gautam Saravana** for running around and jumping with his brother to keep him motivated.

Figure 7: Outdoor data collection in Progress

References

[1] Android Developer Documentation. *Motion Sensors — Andriod Development*. URL: https://developer.android.com/guide/topics/sensors/sensors_motion?authuser=2#java. (accessed: 04.07.2021).

[2] Apple Developer Documentation. *Understanding reference frames and device attitude*. URL: https://developer.apple.com/documentation/coremotion/getting_processed_device-motion_data/understanding_reference_frames_and_device_attitude. (accessed: 04.07.2021).

[3] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[4] Maria Girardi. *Taylor/Maclaurin Polynomials*. URL: https://people.math.sc.edu/girardi/m142/handouts/10sTaylorPolySeries.pdf. (accessed: 06.03.2021).

[5] Haojie Ma et al. "AttnSense: Multi-level Attention Mechanism For Multimodal Human Activity Recognition." In: *IJCAI*. 2019, pp. 3109–3115.

[6] Fernando Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014. URL: https://github.com/fmfn/BayesianOptimization. (accessed: 06.03.2021).

[7] NXP. *NXP AN3461 Application Notes*. URL: https://www.nxp.com/docs/en/application-note/AN3461.pdf. (accessed: 04.07.2021).

[8] Aditya Saravana. *AI Workout Watch App App*. URL: https://devknight-studios.herokuapp.com/DevKnight_Studios/AIWorkoutSite/index.html. (accessed: 06.03.2021).

[9] Aditya Saravana. *Apple Watch Data Collector App*. URL: https://github.com/aditya298/AIWorkout-Collector-App. (accessed: 05.07.2021).

[10] Gordon Wetzstein. *Inertial Measurement Units - EE267 Class notes*. URL: https://stanford.edu/class/ee267/lectures/lecture9.pdf. (accessed: 06.03.2021).

[11] Shiwen Zhao et al. "Raise to Speak: An Accurate, Low-power Detector for Activating Voice Assistants on Smartwatches". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2736–2744.