Optimizing IMU Based Gesture Classification

Paa Adu
Stanford University
Department of Electrical Engineering
paa@stanford.edu

Stefan Bran-Melendez
Stanford University
Department of Electrical Engineering
sbran@stanford.edu

Abstract

In this project, we built a gesture classifier for 6 different boxing movements. We collected a number of 1 second time series for each of these movements, ran them through different filters, and then used different combinations of the filtered data as features in three different Machine Learning models. We then evaluated the accuracy for each distinct set of features, run through each of the three models.

1. Overview and Motivation

Modern Virtual Reality systems often rely on controllers that are disruptive to the fully immersive experience they seek to create. A lot of modern VR systems, such as the HTC Vive, rely on large hand-held controllers that disrupt the immersive experience. Additionally, a lot of existing VR systems rely on cameras that track movements, which restrict the user to stay within a particular field of view. We envision a VR system where your body becomes a seamless extension of the system itself, where you can use your limbs and your own movements to control the world you're seeing, and where you are not restricted to a particular space, or field of view.

Figure 1. The HTC Vive controllers

Our goal is to investigate how we can create the most accurate gesture classifier with nothing more than an IMU attached to a hand, to create a fully immersive experience, where the user can use their hand as a controller for the environment. In order to do this, we wanted to investigate which data filtering methods, and which Machine Learning models would be most successful in classifying hand gestures. To this end, we decided to use a series of gestures, in this case in the application of boxing, to determine what methods in data filtering and machine learning yield the most accurate classification results. For each of these gestures, we collected raw IMU accelerometer, gyroscope, and magnetometer data, filtered it according to three different algorithms, and then used this processed data to train and test three different machine learning algorithms.

Figure 2. An example of boxing in VR

2. Related Work

A substantial amount of work has gone into the fields of Computer Vision, with commercial hardware such as the Xbox Kinect and the Ninendo Wii able to recognize certain movements or gestures. Tools such as OpenCV have made it easier to make advancements in the field of Computer Vision, giving rise to applications like these. However, in the context of gesture classification using IMU data, less work has been done. One exemplary attempt at using the hand as a controller is the use of a Hidden Markov Model (HMM), which uses both an IMU and an EMG sensor [7]

IMUs are known to be fairly noisy. A lot of research has been done with regard to improving IMU performance us-

ing filtering and sensor fusion [1][3][5], but not much work has been done with regard to IMU based gesture classification

3. Method

In our experiment, we attempted to optimize gesture classification accuracy by using different filtering algorithms, as well as different machine learning models. First, we captured approximately 100 one-second-long time series IMU data (accelerometer, gyroscope, magnetometer) for 6 different gestures. We then ran this data through three different data filtering algorithms: Complementary Filter, Kalman Filter, and the Savitzky-Golay Filter. We then used different combinations of the filtered and non filtered data as sets of features, that were then used to train three different machine learning models to classify our data. We evaluated the performance of three different machine learning models in order to classify our data. These models were logistic regression, support vector machines (SVM), and multilayer perceptron (MLP). These models were programmed in Python using the scikit-learn package.

3.1. Dataset

The raw dataset was composed of time series accelerometer, gyroscope, and magnetometer values gathered with an InvenSense 9250 Inertial Measurement Unit (IMU). The IMU was attached to a user's hand and was used to record punching movements. 6 different types of movements were recorded: jab, hook, block, uppercut, overhead punch, and guard stance.

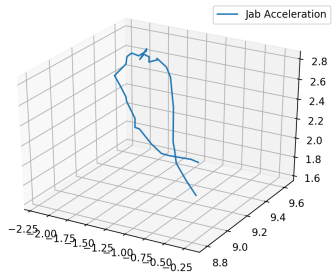

Figure 3. A plot of a sample of acceleration data gathered during a jab.

Each sample in the data set was composed of 1 second of IMU data during which once of the punches was recorded. Data was sampled from the IMU at a frequency of 770Hz. In total 619 punches were manually recorded. These punches were mostly distributed evenly between the 6 different types of movements.

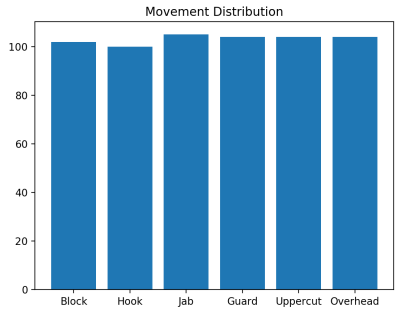

Figure 4. A plot of the distribution of movements.

Once all of the raw IMU data was gathered, we preprocessed the data by applying one of the four filters in order to generate different features for our models: complementary filter, Kalman filter, Savitzky-Golay Filter, no filter.

3.2. The Complementary Filter

The first, and simplest, filter that we implemented was the complementary filter. What the complementary filter does is it acts as a high-pass and a low-pass filter simultaneously, specifically applying the high-pass filter to the gyroscope reading, and the low-pass filter to the to the accelerometer readings. The motivation behind the complementary filter is that the values read from the gyroscope data tend to drift in the long term, but are very accurate in the short term. The accelerometer, on the other hand, is unreliable in the short term due to its sensitivity to motion and noise, but is accurate in the long term due to lack of drift. We can use the properties of the two measurement units, and fuse the two to get a reliable roll angle. Our weighted fusion of these filtered estimates would look as follows:

$$\theta^{(t)} = \alpha(\theta^{(t-1)} + \tilde{\omega}\Delta t) + (1-\alpha)(atan2(\tilde{a_x}, \tilde{a_y}))$$

Where α is a factor that weighs how much of the gyroscope data we use vs how much of the accelerometer data we use. We filtered all of our data through the complementary filter, and used this as one of the sets of features in experimenting which combination of features, and which model would yield the highest classification accuracy [9].

3.3. The Kalman Filter

Next, we implemented the Kalman filter, which is more complicated than the Complementary filter, but yields more accurate data. The Kalman filter uses a feedback mechanism, which estimates the angle of our IMU at the next time step based on previous angles, then takes a physical measurement, and then updates what we call a priori estimate for the next time step. In summary, the Kalman Filter can be thought of as a predictor-corrector algorithm, which uses

data about the error of previous estimates, as well as data about the previous measurements and previous estimates, to take a measurement and 'correct' it based on this previous information. [8] The Kalman Filter can be split into two states: for every measurement, it has a prediction state, as well as a correction state.

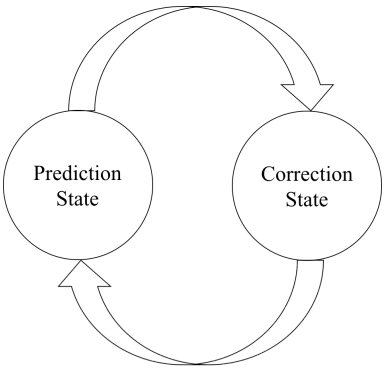

Figure 5. State Machine for Kalman Filtering

Mathematically, the Prediction State would be represented as follows:

$$\hat{x_k}^- = A\hat{x_{k-1}} + Bu_k$$

$$P_k{}^- = AP_{k-1}A^T + Q$$

where:

$\hat{x_k}^-$ is our non-corrected priori estimate,

x_{k-1} is our priori estimate from one step ago

A relates our previous state to our current state (in our case, we assumed a linear relationship, so

$$A = \begin{bmatrix} 1 & -timestep \\ 0 & 1 \end{bmatrix}$$

Similarly, B relates our sensor input to the model, in this case we had

$B = \begin{bmatrix} timestep & 0 \end{bmatrix}$

$P_k{}^-$ is our posteri error covariance matrix,

P_{k-1} is our posteri error covariance matrix from one step ago

Q is our process noise covariance matrix, which we estimated to be:

$$Q = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.003 \end{bmatrix}$$

The Correction State behaves according to the following equations:

$$K_k = P_k{}^- H^T (HP_k{}^- H^T + R)^{-1}$$
$$\hat{x_k} = \hat{x_k}^- + K_k(z_k - H\hat{x_k}^-)$$
$$P_k = (I - K_k H)P_k{}^-$$

where:

K is the Kalman Gain

H relates the current state to the measurement

R is the measurement noise covariance

z_k is the experimental value of the angle we compute

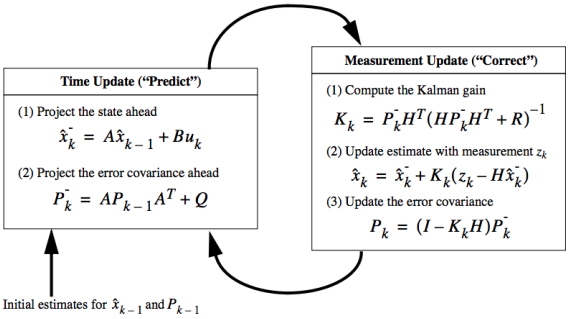

Figure 6. Kalman Error Correction Process [8]

For the detailed implementation of the Kalman filter, please see our code. We used the Kalman filter on two separate occasions: once for the raw roll angles, and once incorporating the gyroscope data.

3.4. Savitzky-Golay Filter

The last filtering algorithm we used is the Savitzky-Golay Filter. This algorithm performs a Least-Square Polynomial approximation, where for a particular window, the Savitzky-Golay Filter calculates the polynomial coefficients that minimize the residual (i.e. approximation error). [6] The approximated polynomial would have the following form:

$$p(n) = \sum_{n=0}^{N} a_k n^k$$

Where we are finding a polynomial p(n) where we are attempting to minimize the mean-squared-approximation error:

$$Error = \sum_{n=-M}^{M} (p(n) - x(n))^2$$

We ran our raw roll data through this filter and used it as another set of features when training our algorithms.

4. Experiments

Data was tested on three different machine learning models.

4.1. Logistic Regression

We ran a regularized logistic regression algorithm that minimized the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(exp(-y_i(X_i^T w + c)) + 1)$$

In order to make predictions over six movements present in the dataset, we trained six different logistic regression models rather than a single multinomial logistic regression model. For each movement, we created a model that predicted whether an example was or wasn't the given movement. Probability estimates were calculated for each movement and the highest scoring class was chosen to be the final prediction[4].

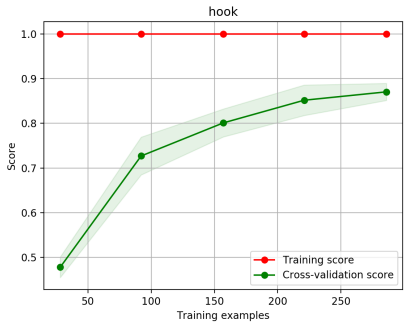

Figure 7. A plot of the learning curve for the hook vs. not hook logistic regression classifier.

4.2. Support Vector Machine

The SVM was another algorithm that was tested for classification. SVMs have been shown to be effective in cases where the number of dimensions is greater than the number of samples (much like our application)[2].

After evaluating the performance of linear, polynomial, and rbf kernels, we found that using a third degree polynomial kernel yielded the best accuracy. Since SVMs do not directly provide probability estimates, we calculated them using five-fold cross-validation.

Similar to logistic regression, in order to make predictions over the six movements present in the dataset, we trained six different SVM models, and picked the highest scoring model for prediction.

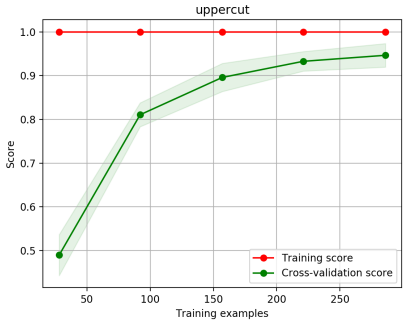

Figure 8. A plot of the learning curve for the hook vs. not hook logistic regression classifier.

4.3. Multilayer Perceptron

A Multilayer Perceptron was used for multi-class classification[10]. The rectified linear unit function (ReLU) was selected as the activation function. The selected MLP had three hidden layers that each had 30 neurons. The input layer was sized to match the number of input features for each of the filtering algorithms. The output layer had 6 neurons.

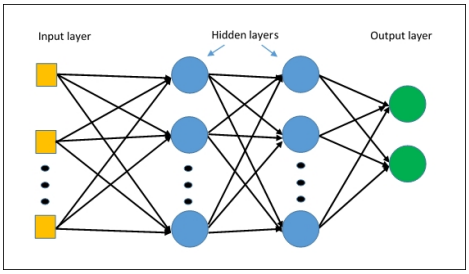

Figure 9. Sample MLP architecture

5. Results

Overall, the MLP proved to be the most accurate model in classifying the 6 boxing movements. The best classification accuracy that was achieved was using the MLP with the Raw IMU data and acceleration data generated by the Kalman filter as input features. With this setup, 93.1% accuracy was achieved when classifying over the 6 different types of movements.

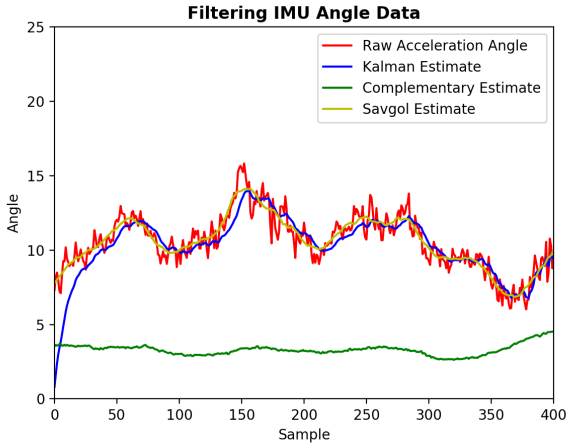

Figure 10. Filter Comparison for Kalman, Complementary, and Savitzky-Goyal Algorithms. Savitzky-Golay yielded the smoothest data, close to the Kalman estimate.

As we can see in the figure above, we obtained relatively good results with our filtering algorithms. The Complementary Filter smoothed the data perhaps too much, which resulted in poorer distinction when it came to classifying with our models. The Kalman and the Savitzky-Golay filters, however, were rather successful in removing a lot of the variance and noise that we notice in the original signal, in red.

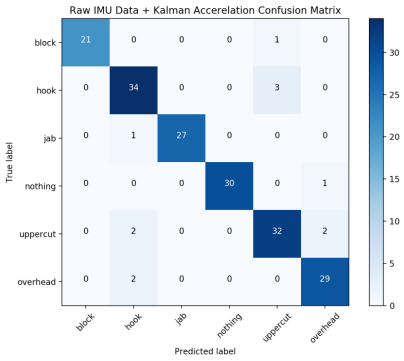

Figure 11. The hook was the prediction with the most errors. The hook is a similar motion to the uppercut and overhead punch.

Ultimately, it is possible to improve classification accuracy by including preprocessed (filtered) data over what can be achieved using only raw IMU data. In our case, the addition of Kalman filtered data to the raw IMU data improved classification accuracy by 1.6%. Although this is a only a slight improvement, it is possible that applications other than boxing could see larger improvements when using filtered data.

	Logistic	SVM	MLP
Raw IMU Data	78.8%	78.4%	91.5%
Savitzky-Golay Filtered Data	80.5%	78.4%	88.8%
Kalman Acc. Data	42.7%	59.5%	59.6%
Kalman Acc. + Gyro. Data	39.5%	56.2%	59.9%
Complementary Data	44.3%	43.2%	57.1%
Raw + Kalman Acc. Data	79.5%	77.8%	93.1%
Raw + Kalman Acc & Gyr. Data	83.2%	80.5%	92.0%
Raw + Complementary Data	80.0%	77.8%	92.8%

Table 1. A table summarizing the results of running the three models with different sets of input data.

6. Future Work

6.1. Improvement 1: Wearable IMU

One of the goals of our project is to create a foundation for seamless interaction with a virtual world, where we're not restricted to a particular field of view. For the purposes of our project, we held an IMU in our hand, as we recorded data for all the gestures that we wanted to train our models with, and that we wanted to classify. This gives rise to two issues:

1. We still have a wired connection from the IMU to the laptop

2. We had to hold an IMU in our hand for the gesture classification process.

One of the ways in which we'd like to extend and improve this project, would be to create a sort of wearable IMU, that can wirelessly transmit either the classified gesture based on the movement, or transmit the raw data for processing separately. Ideally, we would have a wearable with at least an accelerometer and gyroscope on it. Additionally, we'd need a way to transmit this data back to a sort of base station. This could be done through perhaps a Bluetooth-enabled Microcontroller, or through a 3G enabled Microcontroller, such as the Particle Electron.

6.2. Improvement 2: Multiple IMUs

We gathered all of our punch data using an IMU held in the right hand. In the future, we would like to use two IMUs, one for each each hand. Since hand movements greatly differ between the left and right hands depending on the punch that is thrown, it is possible that using IMU data from two hands can further improve classification accuracy. In addition, it is important to know which hand threw a certain punch when rending graphics in a gaming environment.

6.3. Improvement 3: Real-time classification

Our currently experimental setup is only able to classify on a prerecorded dataset. We would like to add support for real-time classification of gestures in a simple 3D game.

This could be accomplished by simply streaming the last second of IMU data through the trained models. Once a gesture has been recognized, a corresponding animation can be rendered.

7. Conclusion

Accurate gesture classification has the potential to greatly improve immersion in VR experiences. We have demonstrated that using an IMU in conjunction with machine learning algorithms can be a successful approach towards accurately classifying gestures. Preprocessing IMU data with filtering algorithms has the potential to slightly increase accuracy (depending on the application). However, as we noticed, the increase in accuracy was minimal - our model was accurate enough without the complex data filtering that we performed. Therefore, we realize that our model for gesture classification could easily be implemented on a microcontroller, as we don't even need the heavy computing power for complex filters such as the Kalman filter - all we really need is the classification, using our already trained models. We believe that spending more time researching and tuning machine learning models can lead to achieving state-of-the art classification accuracy over a complex set of gestures.

8. Acknowledgements

We would like to thank the staff of EE267 for offering a in-depth and interesting course. We especially enjoyed the poster session and the feedback we received.

References

[1] E. Bostanci. Motion model transitions in GPS-IMU sensor fusion for user tracking in augmented reality. *CoRR*, abs/1512.02758, 2015.

[2] S. Datta, S. Nag, S. S. Mullick, and S. Das. Diversifying support vector machines for boosting using kernel perturbation: Applications to class imbalance and small disjuncts. *CoRR*, abs/1712.08493, 2017.

[3] M. Kok, J. D. Hol, and T. B. Schön. Using inertial sensors for position and orientation estimation. *CoRR*, abs/1704.06053, 2017.

[4] W. Li, H. Liu, P. Yang, and W. Xie. Supporting regularized logistic regression privately and efficiently. *CoRR*, abs/1510.00095, 2015.

[5] K. Nirmal, A. G. Sreejith, J. Mathew, M. Sarpotdar, A. Suresh, A. Prakash, M. Safonova, and J. Murthy. Noise modeling and analysis of an IMU-based attitude sensor: improvement of performance by filtering and sensor fusion. *ArXiv e-prints*, Aug. 2016.

[6] R. W. Schafer. What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, 28(4):111–117, July 2011.

[7] S. O. Shin, D. Kim, and Y. H. Seo. Controlling mobile robot using imu and emg sensor-based gesture recognition. In *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 554–557, Nov 2014.

[8] G. Welch and G. Bishop. An introduction to the kalman filter.

[9] G. Wetzstein. Course notes: 3-dof orientation tracking with imus, Sep 2017.

[10] R. Zhang, W. Li, and T. Mo. Review of deep learning. *CoRR*, abs/1804.01653, 2018.