Predicting Short-Range Displacements From Sensor Data

Maurice Shih and Jun-Ting Hsieh

1 Introduction

With the advent of Internet of Things (IoT) and mobile devices, a key question is, "how much can we learn about a user by the data from smart devices?" There are many interesting topics in this domain, such as gesture recognition, about assisting users with additional functions.

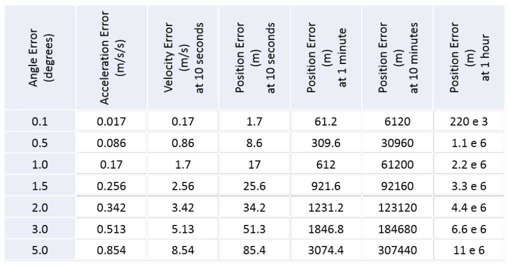
In this project, we are aiming to construct a model to determine distance travelled by a worn wearable in a short-range displacement by sensor data. We define a short-range displacement (gesture) as a natural motion of a stationary user's arm. This typically takes less than a second of travel time. Many wearables can capture motion events from many different sensors. The most basic sensor is the accelerometer, which measures the acceleration felt by a device in three coordinates. The goal of this project is to predict the distance travelled during a gesture using data from the accelerometer.

2 Related Work

Work has been done on identifying motion gestures based with accelerometers, but little done with identitfiy/classifying gestures that take a short amount of time (e.g. a few seconds).The goal is to used the acclerometer data to identify the distance travelled. This is especially hard since the orientation is not possible to determine, since there is no gyroscope.

One straightforward approach would be double integration, but this has mixed results. As one lab noticed, "This isn't because the accelerometers themselves are poor, but because the orientation of the sensor must be known with a high degree of accuracy so that gravity measurements can be distinguished from the physical acceleration of the sensor. Even small errors in the orientation estimate will produce extremely high errors in the measured acceleration, which translate into even larger errors in the velocity and position estimates. "[1]

As you can see in the table below, if the orientation of the device is off by 0.1 degrees, the after 10 seconds, double integration can me 1.7 meters off.



Angle Error (degrees)	Acceleration Error (m/s/s)	Velocity Error (m/s) at 10 seconds	Position Error (m) at 10 seconds	Position Error (m) at 1 minute	Position Error (m) at 10 minutes	Position Error (m) at 1 hour
0.1	0.017	0.17	1.7	61.2	6120	220 e 3
0.5	0.086	0.86	8.6	309.6	30960	1.1 e 6
1.0	0.17	1.7	17	612	61200	2.2 e 6
1.5	0.256	2.56	25.6	921.6	92160	3.3 e 6
2.0	0.342	3.42	34.2	1231.2	123120	4.4 e 6
3.0	0.513	5.13	51.3	1846.8	184680	6.6 e 6
5.0	0.854	8.54	85.4	3074.4	307440	11 e 6

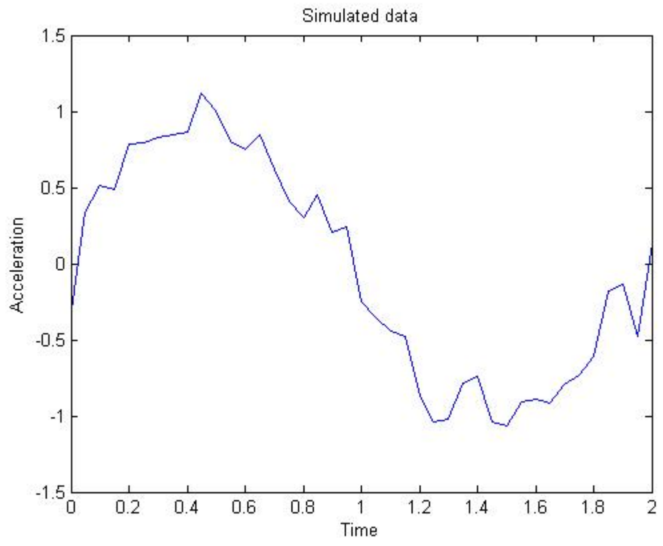
3 Preliminary Tests

From the discussion above, we know that using double integral to calculate distance will result in a huge error. However, there is no guarantee that using a machine learning model will have a better result. In fact, given a set of acceleration data $a_1, ..., a_n$ with a constant time step Δt, the result of double integral is just a linear combination of the data points,

$$D = \sum_{i=1}^{n} v_i \Delta t = \Delta t^2 \sum_{i=1}^{n} \sum_{j=1}^{i} a_j = \Delta t^2 \big(n a_1 + (n-1) a_2 + ... + a_n \big)$$

Thus, double integral is a linear model, so it is possible that machine learning models such as linear regression will do no better than using double integral. In order to test if this is the case, we generated simulated data and compared the performances of double integral and linear regression with only the acceleration data as features.

For a simple hand gesture, the acceleration data is very close to a perfect sine curve, and the errors from the sensor and hand orientation can be modeled as Gaussian noise. Therefore, we simulated our data using the equation $a(t) = A\sin(\omega t) + \epsilon$, where A is a random number between 1 and 2, and ϵ is noise drawn from a Gaussian distribution. Since a hand gesture usually takes 1 or 2 seconds, we simulated the data over 2 seconds with a sampling rate of 0.5 second per data point. Below is the graph of a simulated example,



The actual distance can be calculated using double integral on the perfect sine curve without noise,

$$D = \iint_0^T A\sin(\omega t)dt^2 = \frac{A}{\omega}(T - \frac{1}{\omega}\sin(\omega T))$$

We generated 1000 training examples and 1000 test examples for linear regression, and we used the test examples to evaluate both models. For double integral, the average percent error is 6.5%. For linear regression, the average percent error is 2.5%. This shows that even though both models make predictions by linear combinations of the data, linear regression performs much better. It is able to "learn" the fact that the hand gesture is similar to a sine curve and make more accurate predictions based on that. This also implies that non-linear models are likely to perform even better than linear regression. Therefore, we will use both linear and non-linear regression (neural networks) to train on the real data.

4 Procedure

An app for the pebble watch and android phone was created to measure and store the acceleration caused by the short displacements. We also created an iPad app to measure the distance travelled during short displacements. The procedure of each gesture is as followed:

1. Have the middle finger touch the iPad.
2. Triggers the pebble to start recording data.
3. Make the gesture on the iPad.
4. Finally, touch a button on the phone to stop the recording.

This method assumes that the distance travelled by the middle finger is very similar to the distance travelled by the wrist (pebble). Thus, the motion of the finger and the wrist need to be parallel during the gesture. In order to increase the accuracy of this, a split was used to limit the motion of the wrist.

Due to the limitations of the pebble, the data from the pebble is first sent to the phone, and then manually transferred to a computer.

5 Data Processing

For each gesture, we have the acceleration data of all 3 directions x, y, and z. The y axis is the direction that the hand moves, so the y data look like a sine wave, and the x and z data are roughly constant throughout the gesture. If the data don't look like this, we will disgard it.

Since the pebble starts recording data before the gesture starts and doesn't stop right after it ends, we have to manually select the data corresponding to the gesture. The gesture part of the data (in the y direction) looks like a sine curve and the rest is roughly zero with some noise, so we plot out the data and manually remove the irrelevant parts of the data.

In total, we collected over 220 gesture data from the pebble and iPad.

6 Results and Discussion

We did some initial testing with how double integration and linear regression did on predicting the distance travelled.

Figure 6.1: Double integration vs Actual

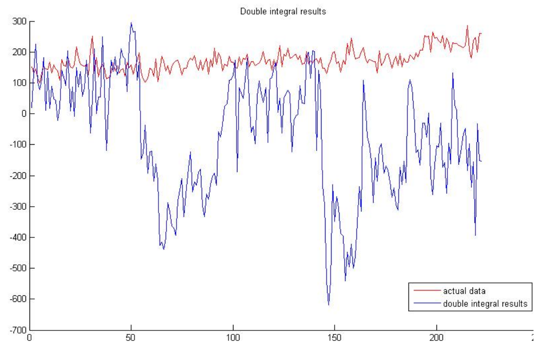
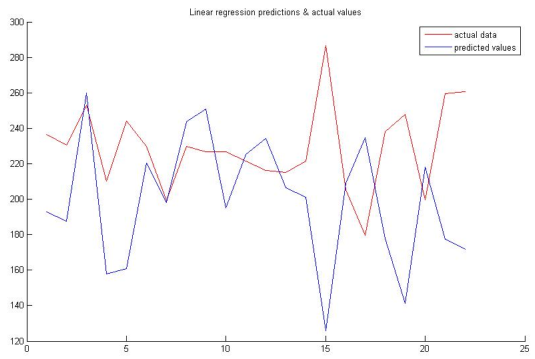
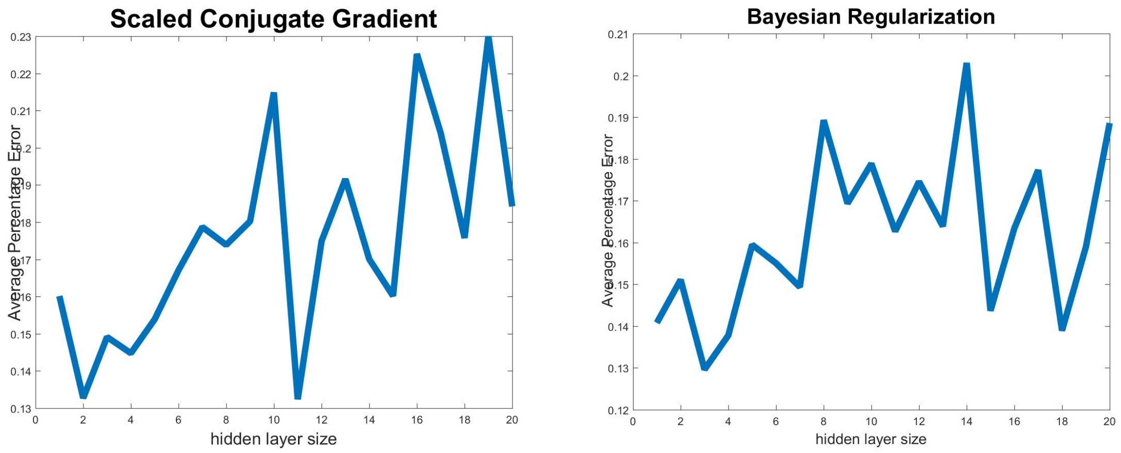


Figure 6.2: Linear Regression vs Actual

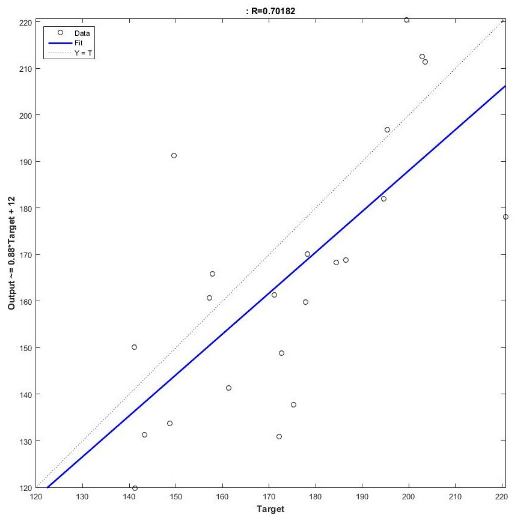


As expected, we need some sort of non-linear regression model, so we turn to neutral networks. The two models we used were scaled conjugate gradient, which runs fast and Bayesian regularization, which runs a lot slower but tend to do better with smaller and nosier data. One key factor that we tested with these two methods was the hidden layer size, or the number of neurons to use. The two figures below are our results.

Figure 6.3: After training, the average error for test case



We fine tuned the Bayesian Regularization model with the smallest percentage error and obtained a correlation between the output vs actually to be 0.70182. Below we have a figure.



7 Conclusion

Relying only on the accelerometer only to determine displacement, traditional methods of double integration and linear regression do not fit well. Neural networks that were implemented were fine tuned to have average errors of around 13 percent. An observation that was made was that the hidden layer sizes of 5 or less tended to yield much higher accuracy in prediction. More data collection to understand the data may lead to increased accuracy.

Potential applications are virtual keyboards, authentication as well as malicious possibilities such as picking up a users keystrokes. This work, with the inclusion of a finger tap classifier (when a finger taps, based on accelerometer data, the classified determines which finger tapped) that was done in a former project by the authors, has the potential to capture users keyboard inputs.

8 Future Work

We hope to fine tune the model with more data in order to understand more characteristics of the data. This includes the variation of motion for different individuals. We also hope to generalize our work with smart devices that offer more sensor data, such as gyroscope. We are currently looking at using apple watch, which can record both accelerometer and gyroscope data. With gyroscope, we can construct yaw, pitch, and roll data, which can hopefully increase the accuracy of our model.

References

[1] Using Accelerometers to Estimate Position and Velocity, http://www.chrobotics.com/library/accel-position-velocity

Prop

Predicting Short-Range Displacements From Sensor Data With the advent of the Internet of Things (IoT) and mobile devices, we can capture gesture and motion events from accelerometer and gyroscope data. The goal of this project is to develop a method to predict short-range displacements of hand gestures using these data. The current method of using double integrations of acceleration data to apply simple Newtonian models is very inaccurate. In this project we apply machine learning techniques to outperform existing models that predict short range displacements. The bulk of the project will be determining a method to calculate ground truths of displacements over short time spans. Our preliminary idea is to track gesture motions by analyzing frame per frame changes of image recordings (at least 10 frames/sec). We hope to start by measuring these changes in one dimension. After this, we hope to expand to the xy-plane of two dimensions. Once we create a method to calculate ground truths, the next step is to find a way to extract meaningful features. Initial feature selections will be based on accelerometer and gyroscope data. This is an example of supervised learning. An interesting result might be whether motions of different people vary significantly. After collecting the training data, we can apply various machine learning techniques to construct a model for predicting displacements. In order to determine the accuracy of our results, we will collect test examples and check if our model closely predicts the displacements of most test examples. For example, using techniques for time series data may prove to be more accurate. We will continue modifying our feature extractions until our model is true. In terms of timeline, we hope to research and develop a technique for calculating ground truths in two weeks. In the consequent week(s), we will collect data and begin feature selection. If time permits, we will investigate the differences in the gesture movements for different people. This particular project stems from a larger project that a member of this team is part of.

Milestone

As a refresher, our project is to develop a method to predict short-range displacements of hand gestures using smart devices. We plan on using apple watches (they will be available to use on 11/17), but for preliminary testing we are using the pebble smartwatch. With this device, the only data we can collect is the accelerometer data. So our objective is to map these accelerometer values to features, and to map these features to the distance that was travelled by the watch. The bulk of our work was to develop a system to measure data accurately and reliable.

Our first goal is to determine how much the watch actually traveled, in order to have a ground truth to test how accurate a model we generated would be. Out initial approach was to have a camera that would record how the watched moved, and then calculate the number of pixels that moved. However, this proved to be too complex (and we would have to scale 1 pixel to how much distance it actually is, eg 1inches). So our next approach (our current) is to measure how much a finger moves during that gesture. Our thought is that if the wrist and finger stay straight (our gestures are only fractions of a second) during the gesture, then the distance travelled should be close to the distance traveled by the smartwatch. In order to do this, we had a wrist brace to limit the movement of the wrist.

We then build an ipad app that can track the movement of a finger to estimate how much the smart watch moved. We also build a pebble app and android app to measure and store the accelerometer that the pebble smartwatch experiences. When we get the apple watch, we will be migrating to using that to collect the accelerometer data. We initially wanted yaw, pitch, and roll, but Apple has prevented 3rd party developers from accessing that until the 2.0 of the apple watch iOS.

After building the apps, we started collecting data. We iterated through various methods to be able to segment the data better. We want to be able to, from just looking at the accelerometer data, figure out when a user is moving his hand for the gesture. This is a challenge that we will have to solve for applications. But for now, we tested different start positions for the hand and found that having a finger placed on the ipad screen from the start eliminated a lot of error. We found that the accelerometer data had the x-axis points moving the most in a shape that is similar a sine or cosine shape (you accelerate you hand, then slow it down). Now we move to feature extraction. The biggest problem is that the length of accelerometer data varies in size for each gesture, so we cannot have them be the features (even if this worked it wouldn't be optimal). However, we are still trying to figure out what it the best features we can extract, since the "fatness" or "height" of the accelerometer data shape (the part the data that looks like the hump of a sine wave) increasing correlates to more distance travelled. So we construct a base model just to test how much tweaking is required. We made our model based on least squares regression, but we need to have invertible matrices of features. We did this by creating a square matrix of unifomly selected accelerometer points. We ran leave-one-out cross validation model to construct the best model. After doing so, we had a very significant error margin. We expected this to be very off, but we are looking into feature selection that is meaningful to the data, such as how constant/not-constant was the acceleration, duration of the acceleration, max acceleration, etc.