# RECOLIFT: AN ANDROID WEAR FITNESS TRACKER FOR STRENGTH TRAINING

*Draft of May 13, 2015 at 03 : 19*

BY

DARIO ARANGUIZ

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the Undergraduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor Romit Roy Choudhury

# ABSTRACT

Despite the plethora of fitness trackers on the market, few monitor signals other than number of steps and heart rate. With the increasing mainstream acceptance of general-purpose smartwatches however, we have the capability to track more complex activities. We propose RecoLift, an Android-based system to track exercises and repetitions in weight training and bodyweight training activities based on the work of Morris et al. Our goal is to provide a system which provides feedback to the user in an autonomous, online fashion, harnessing both smartwatch and smartphone sensors. This system is separated into three key phases: *segmentation*, during which we use the periodicity of the signals to determine if an exercise is being performed, *recognition*, which calculates signal features to determine which exercise is being performed, and *counting*, which uses periodicity to calculate the number of repetitions in a set. Our evaluations trained on a single user show perfect exercise recognition, with 80% of repetitions on average being within 1 repetition of the true count.

*To my mother, for her love and support.*

# ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Romit Roy Choudhury, for taking me under his wing and showing me how to break into the mobile space. Without his influence, this thesis would not have been possible.

I would also like to thank Morris et al. for their outstanding paper (REFERENCE) which my thesis is based on. Their work enabled me to showcase the skills I have acquired over the duration of my undergraduate studies.

Finally, I would like to thank Kashev Dalmia for his continual support. When I needed a gym buddy to keep me company, he was always there.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| PCA | Principal Component Analysis |
| DTW | Dynamic Time Warping |
| IMU | Inertial Measurement Unit |
| KNN | K-Nearest Neighbors |
| SVM | Support Vector Machine |
| MEMS | Micro-Electro-Mechanical Systems |
| ZOH | Zero-Order Hold |
| IIR | Infinite Impulse Response |
| CUSUM | Cumulative Sum |
| RMS | Root Mean Square |

# CHAPTER 1

# INTRODUCTION

If we were to peruse the headlines on a technology news website such as *The Verge* or *ArsTechnica*, a vast majority of the posted articles would revolve around the new Apple Watch. Google has been pushing their new Android Wear platform for over a year now, and Pebble has been working for even longer, with their original Kickstarter launching on April 11, 2012 [1]. At least in the eyes of these device manufacturers, smartwatches comprise the next wave in mobile computing.

Occurring concurrently is the recent public interest in personal analytics. Fitbit and Jawbone have become the two frontrunners in this space. Their product lines of fitness trackers primarily record data related to general purpose fitness, such as resting heart rate, number of steps taken per day, and calories burned. Both Google and Apple have also taken to this space, incorporating fitness tracking in their own wearable devices by including an optical heartrate on the undersides of their watches and various MEMS sensors onboard the watches themselves. They have also opened up new APIs to allow developers free access to their personal data stream [2, 3], enabling such applications as runner route tracking and sleep tracking [4, 5]. One space has remained relatively empty of fitness tracking applications however, and that is strength training.

Strength training comprises of three distinct disciplines: weightlifting, powerlifting, and bodybuilding. Weightlifting involves two lifts only, the clean and jerk and the snatch. These two lifts are the only lifts among strength training exercises that are tested at the Olympics [6]. Both of these lifts are highly technical and not often performed by beginners, with the exception of CrossFit, a new lifting paradigm which starts beginners on high-repetition Olympic lifts. Powerlifting focuses on three lifts only, bench press, squat, and deadlift. Like weightlifting, the primary goal of powerlifting is to maximize the weight lifting among the three lifts. The main distinction, aside from the difference in lifts, is powerlifters tend to focus on raw strength, whereas weightlifters focus on speed. Finally, bodybuilding is significantly different than both weightlifting and powerlifting. Strength does not matter in bodybuilding, and as such, bodybuilders focus on a plethora of smaller, isolated lifts to improve their physique. It is not uncommon for a bodybuilder to spend two hours in the gym performing 30 or 40 sets at eight repetitions per set. This can be problematic, as gym goers often neglect to record their lifts, leading to confusion during the next session in the gym.

To this end, we have created an application based on the work of Morris et al. in *RecoFit* [7] which tracks exercises without user intervention using the commonly available Android Wear platform.

# CHAPTER 2

# LITERATURE REVIEW

As mentioned, this work is based on *RecoFit* by Morris et al. [7], which uses the intuition that exercise is distinctly periodic and can be well-discerned from non-exercise. They achieve 86% segmentation accuracy and 98% recognition accuracy using a custom-built arm-worn device which samples at 50Hz. Additionally, their computation is done on local desktop machine, eliminating the need for energy optimizations beyond the sampling rate of the IMU. Our solution uses readily-available hardware which has been gaining traction in the consumer space [8, 9] to expand the possible userbase. We also consider battery life optimizations on the Android devices while still maintaining comparable levels of accuracy, allowing a user to spend an hour at the gym and continue their day without smartwatch or smartphone recharging. Finally, because we utilize the Android smartwatch, we can make stronger assumptions about placement of the smartwatch. This enables us to perform classification on a higher-dimensional dataset.

Pernek et al. [10] propose an algorithm to count the number of repetitions of an exercise using DTW, a dynamic programming technique which allows for comparison between two non-temporally aligned signals by calculating a mapping which minimally warps and shifts one signal onto another. To differentiate between exercise and non-exercise, Pernek et al. utilize a thresholding algorithm which triggers when the device's accelerometer signal peaks approach the magnitude of the peaks in their prerecorded dataset. Their method performs very well with regard to repetition count, although their solution is not entirely autonomous during operation, requiring input from the user at the beginning of each exercise.

Seeger et al. [11] describe a system which utilizes a network of embedded wearable sensors across the body to compute high-dimensional features for exercise classification. Equipping a user with an accelerometer above the right knee, a heart rate sensor, an accelerometer attached to a weight lifting glove, and a chest strap, this system is able to highly accurately detect and count exercise. However, this system is suboptimal due to the infrastructure required. A user attending an incredibly upscale gym may have access to these sensors, but the average user would not. Wearing so many sensors would also obstruct the user during lifting, which could cause both damage to the sensors and discomfort to the user.

Muehlbauer et al. [12] follows a similar pattern to Morris et al. [7] and our own solution by dividing the task into three phases, segmentation, recognition, and counting. Autocorrelation analysis is used during segmentation to determine of a user is performing an exercise. After determining that an exercise is being performed, a number of features are calculated such as mean and standard deviation. These are passed into a KNN classifier which comprises the recognition phase. Lastly, counting is perfored using simple peak counting. Muehlbauer et al. performs well, with 85% segmentation accuracy and 94% recognition accuracy, although their segmentation thresholds are based off heuristics, and they do not address online performance.

To summarize, the discussed related systems often fall short in one key category. For our solution, we aim to track exercises using no additional hardware beyond the smartwatch, without user input during the exercise session, and in an online fashion, all while maintaining high accuracy rates.

# CHAPTER 3

# GENERAL METHODOLOGY

## 3.1 Problems

There exist a number of problems when doing strength training-based fitness tracking. The first is the issue of *separating exercise from non-exercise*. As with most problems in machine learning, exercise is easily discerned from non-exercise when viewed with the naked eye. This separability breaks down when viewing exercise from the perspective of the onboard sensors. Performing a repetition on bench press may look similar to grabbing a water bottle from a gym bag, for example. Additionally, common non-exercise movements such as pacing about or drinking from the water fountain may be accidentally misconstrued as exercise when looking at time series data.

We have a key intuition however to separate exercise from non-exercise, and that is periodicity. Non-exercise tends to be very aperiodic, such as in Figure 3.1, while exercise (especially high-repetition bodybuilder-esque exercise) tends to be very periodic, such as in Figure 3.2. This can be easily exploited then when performing classification. If a certain subset of the signal is highly periodic, it is likely to be an exercise of some kind. However, there is one activity users do in the gym that is both periodic and non-exercise, and that is walking. This necessitates the use of machine learning, as there is no ubiquitous heuristic which can determine whether a signal is periodic

Figure 3.1: Single axis accelerometer values while idling

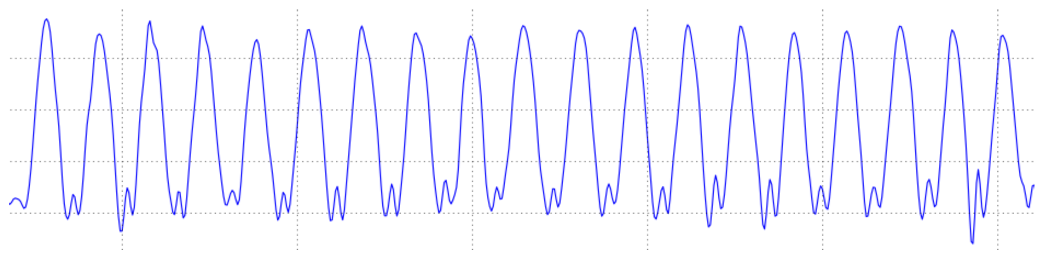

Figure 3.2: Single axis accelerometer values while performing curls

as well as if a signal represents walking versus other exercises. Fortunately, features can easily be extracted and passed into an SVM which accomplishes both of these tasks.

The second issue we face is the *wide variability in form*. We make the assumption that users of this application will have a fundamental understanding of the lifts they perform, although sensor data will inevitably vary depending on physiological features such as height, arm length, and leg length, regardless of how well they perform the repetition. It is possible that a DTW algorithm could allay this, but we choose to simply train our classifiers over a large dataset. By choosing five to ten athletes to provide training data, we can reasonably cover most variations in form.

Another issue we have is *properly counting*. For an application like a pedometer, counting repetitions becomes almost a trivial task. A heavy low-pass filter can be applied to the signal, and then the number of steps simply becomes the number of peaks. This is due to the relative steadiness with which we walk. Generally speaking, people's gaits do not change much from step to step. A user may walk at a different speed in their house than going when they go to class, but on the whole, walking is a cleanly periodic signal. Counting lifting repetitions however is a much more difficult task, as the bar speed and repetition rate can change from repetition to repetion, as well as from set to set. A user doing a set of ten repetitions or greater may tire by the end, performing much slower and perhaps even failing a lift. A tracking system must be able to account for this. Additionally, the accelerometer traces of an exercise may not be amenable to simple peak counting, such as in Figure 3.3.

Finally, we have a problem of *differentiating between similar exercises*. It may be easy for the human eye to differentiate between a Pendlay row and a bench press, but when looking at the movement itself, both simplify down

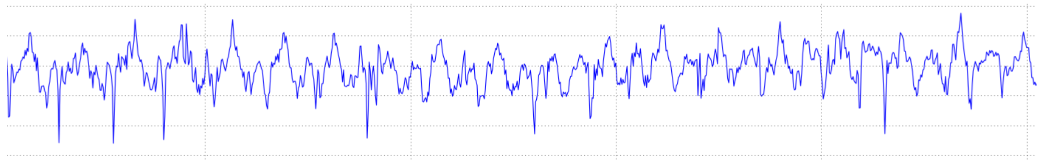

Figure 3.3: Single axis accelerometer values while performing squats

to an explosive upward movement and a controlled downward movement. To this end, we must employ machine learning instead of simple heuristics for lift recognition.

## 3.2 System Design

This leads us to our final system design. Our system comprises of four primary phases: *preprocessing, segmentation, recognition, and counting.*

First is the *Preprocessing* phase. Android does not guarantee a consistent sampling rate on its sensor readings, thus we first evenly resample the data. Next, we pass the data through a low-pass filter to remove noisy high-frequency components which may not be indicative of the actual lift being performed. This data is then stored for the subsequent three phases.

The *Segmentation* phase is the first phase of intensive computation during which we determine whether or not a lift is being performed. Autocorrelation is computed over a sliding window, then features are extracted and passed into an SVM. The output of this classifier is passed to an accumulator, which acts to smooth any jitter from the classifier. If this phase determines that a lift has been performed, we pass the exercise window to the next phase.

Our *Recognition* phase determines which lift is being performed, given that a lift is actually being performed. Similar feature extraction is done as in the segmentation phase, calculating feature sets over a sliding window. These feature sets are passed to an SVM, and the prediction is stored. As we are using a sliding window, our system makes many predictions for a full exercise window. The final result is determined using a majority voting system.

Once the lift has been successfully recognized, we move to the final phase, *Counting*. Our counting algorithm is essentially a sophisticated method of peak counting. First, we find all local maxima in the signal, discounting peaks if they are too close to one another. Secondly, we use autocorrelation to determine the local periodicity of the signal about a peak and remove peaks that are too close, given the assumption that repetitions may vary over a full signal but will stay relatively consistent from a single repetition to another. Lastly, we use a thresholding heuristic to remove peaks that are too low. The remaining number of peaks is then our final repetition count.

## 3.3 Equipment

For testing, we use a first generation Motorola Moto 360 smartwatch running Android Wear v5.0.2. Our smartphone is a Samsung Galaxy S4 running Android v4.4.2. Any combination of smartwatch and smartphone running Android should function with this system, although additional calibration may be needed when working with different smartwatch MEMS sensors. Data collection is performed using the accelerometer and gyroscope on the smartwatch.

# CHAPTER 4

# SYSTEM DESIGN

## 4.1 Preprocessing

### 4.1.1 Resampling

An unfortunate downside to the Android platform is that Android does not guarantee an even sampling rate on its sensors. In lieu of defining a sampling rate, Android allows the developer to request the delay amount between sensor readings. These delays can be *NORMAL, GAME, UI,* or *FASTEST*, going from a 200000$\mu s$ delay down to a zero second delay.

Likewise, although Android does not allow us to define a sampling rate, setting the delay to *FASTEST* results in a sampling rate consistently near 25Hz. To be precise, the accelerometer was measured to have a sampling rate of approximately 24.67Hz on average. This is however an average, thus samples may come early or late, depending on how the Android scheduler prioritizes sensor readings.

Because of this, we resample with a zero-order hold (ZOH). ZOH is very easy to implement, and considering how close our sampling rate is to our desired rate of 25Hz, we maintain a high fidelity signal. Any additional noise introduced due to resampling is effectively filtered out in our next step.

### 4.1.2 Filtering

Repetitions occur generally on the order of 1Hz, and bodybuilding exercises tend not to have many sharp movements due to the threat of injury, thus we decide on a frequency cutoff of 12Hz for our signal. This is conveniently in the middle of our (now resampled) sampling rate, and it will take care of noise added in the previous step.

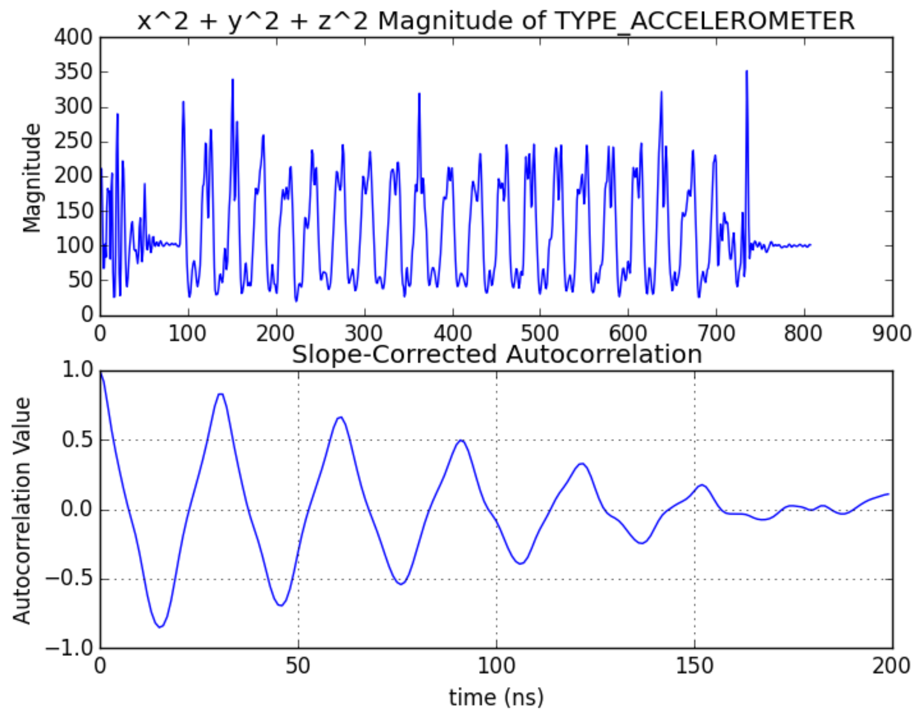


Figure 4.1: Autocorrelation of a periodic signal

Our low-pass filter is implemented using a unity-gain five-tap IIR Butterworth filter generated in MATLAB.

## 4.2 Segmentation

Sensor data is sent in batches from the watch to reduce message overhead and preserve battery life, thus large buffers of sensor data are sent to segmentation at a time.

### 4.2.1 Sliding Window Buffers

First, our data is split into five second buffers using a sliding window with 4.8s overlap. This is done for the purposes of autocorrelation. As mentioned earlier, we expect exercise to be roughly periodic across the entire signal, exhibiting an autocorrelation similar to Figure 4.1, contrasted to aperiodic movement as in Figure 4.2. This is not entirely true however. Anecdotally,

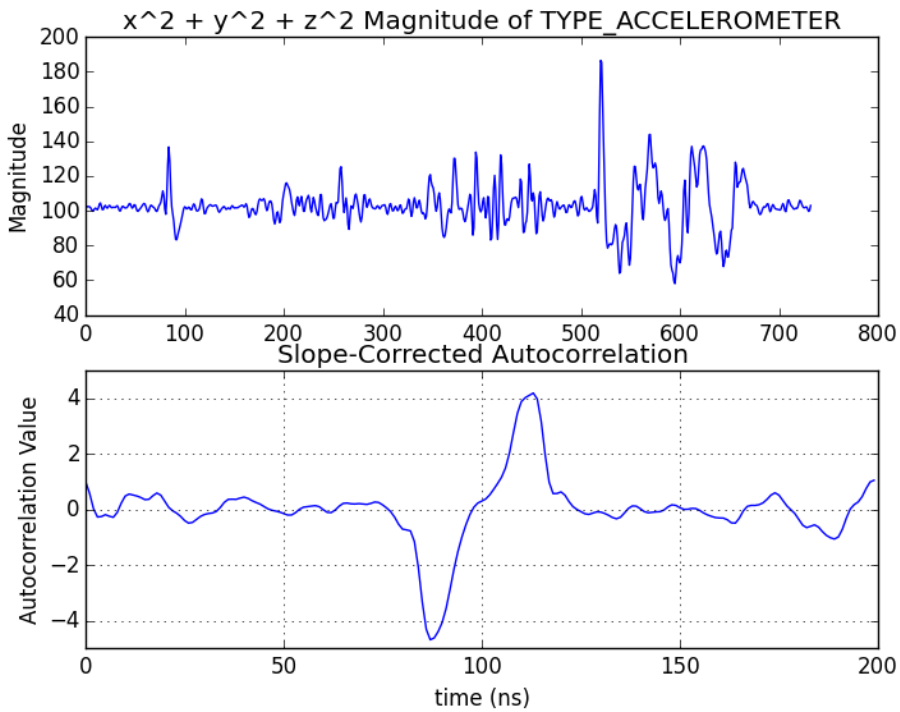


Figure 4.2: Autocorrelation of an aperiodic signal

when performing a set of ten repetitions, the first repetitions will be done better than the final repetitions. Bar speed is directly indicative of the quality of a repetition, thus early repetitions are performed quicker than later repetitions. Computing autocorrelation over the entire signal would not make sense then - concurrent repetitions will likely be similar, but the first is nearly guaranteed to have a different period than the last. Therefore, we compute segmentation over sliding windows.

### 4.2.2 Signal Variants

Our system uses two three-axis sensor sources, the smartwatch's accelerometer and gyroscope. We derive the following signals from each sensor stream:

1. **X-axis:** the X-axis of the accelerometer/gyroscope
2. **Y-axis:** the Y-axis of the accelerometer/gyroscope
3. **Z-axis:** the Z-axis of the accelerometer/gyroscope

4. **Magnitude:** the $\sqrt{x^2 + y^2 + z^2}$ magnitude of the accelerometer/gyroscope

5. **PCA:** the projection of the three-axis data onto its first principal component

This results in ten total signals, five for each sensor. In the past, the orientation of the IMU could not be determined a priori, thus only the axis pointing along the direction of the arm could be used. We can make stronger assumptions however with the use of Android Wear, as a smartwatch has only one possible orientation on the wrist. This allows us to use all three axes as raw signals.

Included in this set are two derived signals, magnitude and PCA. Both signals are attempts at illustrating the primary axis of movement, with magnitude being a crude estimate and the projection onto the raw signal's first principal component being a more refined estimate. In our trials, we have found that computing both the primary projection and the magnitude signals are computationally inexpensive, thus both are used. Further investigation is required to determine if using both is necessary for sufficient classification, although the gain in omitting one is minimal.

### 4.2.3 Feature Selection

From each of the ten sensor streams, we compute 29 features:

1. **Autocorrelation Features:**

   (a) **Total Number of Autocorrelation Peaks:** After computing autocorrelation, the total number of local maxima across the signal is recorded. For exercise, we expect this number to be on the order of two to five. An autocorrelation with no peaks implies idling, and an autocorrelation with too many peaks implies random movement.

   (b) **Number of Prominent Autocorrelation Peaks:** Prominent peaks are determined using a threshold heuristic. If they are relatively isolated from surrounding peaks, and they are also larger in magnitude than surrounding peaks, they are considered prominent. We expect this to be close to the total number of autocorrelation peaks for exercise.

   (c) **Number of Weak Autocorrelation Peaks:** Likewise, weak peaks are determined using the inverse of the strategy above. This should be close to the total number of peaks during non-exercise.
   (d) **Maximum Autocorrelation Peak Value:** A large autocorrelation peak value implies high similarity between the original signal and the delayed version of the signal, indicating periodicity.
   (e) **First Autocorrelation Peak Value:** Likewise, we expect the first peak to be very large for exercise. There is no such expectation for non-exercise.
   (f) **First and Maximum Peak Values Equal:** Finally, the first peak is what we use to determine the period of the signal during exercise. If this first peak is also the largest in the autocorrelated signal, we are likely to be exercising.

2. **Energy Features:**
   (a) **RMS Amplitude:** RMS amplitude is computed for the full window, the first half of the window, and the second half of the window to account for when the window lies on an exercise boundary. Additionally, the CUSUM RMS amplitude is computed as a loose approximation of velocity RMS amplitude.
   (b) **Power Spectrum Bin Magnitudes:** The power spectrum is binned into ten equal width segments and recorded.

3. **Statistical Features:** Similar to RMS, the following features are computed for the full window, the first half of the window, and the second half of the window.
   (a) **Mean**
   (b) **Variance**
   (c) **Standard Deviation**

This results in a total of 291 features, which are passed into a classifier.

### 4.2.4 Classification

These features are passed into a binary SVM trained using athletes familiar with each lift. We then use an accumulator-style voting system to determine when an exercise boundary has been crossed. If the SVM predicts that a lift is occurring, we increment the accumulator. Likewise, if it predicts non-exercise, we decrement the accumulator. Once an accumulator threshold is crossed, we can say with relative certainty that exercise is occuring. The same is done in reverse for computing the end of an exercise boundary.

In practice, we set an accumulator threshold equal to two seconds of activity. Two seconds tends to be the time it takes to complete one full repetition and begin another repetition. This could be increased to reduce false positives, although computing an accurate exercise window boundary is important for counting the number of repetitions. This becomes more difficult as the accumulator threshold increases.

## 4.3 Recognition

The recognition phase follows a similar structure as the segmentation phase. From the segmentation phase, we receive a set of data indices bounding the beginning and end of an exercise window. This window is then broken down into five second buffers with 4.8s overlap. The same set of five signals per sensor source are also calculated, those being the **X, Y, and Z axes, the signal magnitude, and the signal's primary PCA projection**. The main difference however is in the features extracted.

### 4.3.1 Feature Selection

Autocorrelation is useful for estimating whether a signal is periodic as well what its period may be, but we cannot use that information for recognition. Training a classifier which depends on bar speed would be inherently erroneous, as bar speed varies wildly depending on strength, stamina, and lift intensity. To this end, we omit most autocorrelation features from the segmentation phase and focus on other features:

1. **Autocorrelation Features:**

   (a) **Autocorrelation Bins:** Compute the autocorrelation of the window and instead of searching for peaks, compute the magnitude for ten evenly-spaced bins.

2. **Energy Features:**
   (a) **RMS:** The RMS of the full window is computed.
   (b) **Power Spectrum Bin Magnitudes:** Similar to the segmentation phase, the power spectrum is split into ten evenly-spaced bins, and the magnitudes of each are recorded.

3. **Statistical Features:**
   (a) **Mean**
   (b) **Standard Deviation**
   (c) **Kurtosis**
   (d) **Interquartile Range**

This results in a total of 200 features, which are passed into a classifier.

### 4.3.2 Classification

Classification is performed using a multi-class SVM. Predictions are made for each sliding window across the full exercise buffer and aggregated, with a final majority voting system determining which lift was performed during this segment.

## 4.4 Counting

The counting phase is performed over the primary PCA projection of the accelerometer, as this axis contains the majority of the signal's energy by definition. The accelerometer signal is chosen over the gyroscope signal as bar acceleration necessarily changes at the top and bottom of each lift, whereas rotation should not change at the top and bottom when performed correctly for most lifts.

Our counting algorithm can be defined by the following algorithm:

1. (a) Find all local maxima in the signal
   (b) Sort by amplitude
   (c) For each peak, add the index to a set of candidate peaks if the distance to any other peak already in the set is at least MIN_PERIOD away
2. For each candidate peak:
   (a) Compute autocorrelation in a five second window about the peak
   (b) Find the maximum value in the autocorrelation, determining local periodicity P
   (c) Remove peak from candidate set if it is $< 0.75P$ away from any other peak
3. (a) Find the peak at the 40th percentile
   (b) Reject any peaks smaller than half the amplitude of the 40th percentile peak
   (c) Return the total number of remaining peaks as our final count

MIN_PERIOD is defined as the minimum amount of time a particular lift has ever taken to complete. This number is typically around 0.5s.

Figures 3.2 and 3.3 illustrate the difficulties with counting. For an isolation exercise such as a bicep curl, repetitions tend to be clean, and peaks can be easily counted. Contrast this with a compound movement such as a bench press or a squat; it can be difficult to discern repetitions even when manually examining the signal.

### 4.4.1 Algorithm Part 1

Our first step in counting repetitions is to compile an initial list of peaks. We make an assumption that two repetitions could never occur within MIN_PERIOD of each other, and so if two peaks are ever within that range, we assume they correspond to the same repetition. A prime example of this occurring is on Pendlay row; if the repetition is explosive enough, the bar will first accelerate off the floor, then it will experience another spike when it collides with the user's chest.

### 4.4.2 Algorithm Part 2

Next, we further cull the candidate peak set by iterating through the list and determining the local period about each peak. If any surrounding peaks are too close, we remove them from the list. This segment works on the principle of self-similarity between concurrent repetitions.

Speaking generally, Part 2 is a more liberal implementation of Part 1. In Part 1, we assume that the user could be lifting extremely quickly and allow for peaks that indicate rapid repetitions. In Part 2, we relax this assumption and compute what rate the user is actually lifting at, removing peaks that are notably faster.

### 4.4.3 Algorithm Part 3

At this point, we have a well-formed candidate peak set, although there could still be peaks remaining that correspond to noise. For example, if a user stops at the top of a squat to catch his or her breath, there could be a peak when they readjust their position. This would not be removed using the autocorrelation method, thus it would be prudent to make one final pass through our candidate peak list to remove small peaks.

Sorting our list by peak value, a seemingly accurate threshold would be at half the 40th percentile peak value. Few true peaks are less than that, so we can safely remove them.

The size of the candidate peak set is then returned as our total number of repetitions.

# CHAPTER 5

# RESULTS

## 5.1 Methods

Our accuracy can be characterized by three factors:

1. **Segmentation**: Does the system properly detect *when* a lift is being performed? Additionally, does the system ignore all idle motion?

2. **Recognition**: Does the system properly determine *which* lift is being performed?

3. **Counting**: Does the system count the number of repetitions correctly?

To test our system, we trained our classifiers on five lifts: low bar squat, overhead press, bench press, Pendlay row, and barbell curl. Testing comprised of performing 5 sets of each lift at 8 repetitions each, totaling 40 repetitions for each lift. An empty barbell weighing 45 lbs was used.

| Lift | Exact | Within 1 | Within 2 |
|---|---|---|---|
| Low Bar Squat | 80 % | 80 % | 100 % |
| Overhead Press | 40 % | 100 % | 100 % |
| Bench Press | 40 % | 40 % | 80 % |
| Pendlay Row | 40 % | 80 % | 100 % |
| Curl | 20 % | 100 % | 100 % |
| Overall | 44 % | 80 % | 96 % |

Figure 5.1: Difference in reported repetition count versus actual repetition count

## 5.2 Final Results

We achieved 100% accuracy both on segmentation and recognition. Every set performed was noted and classified correctly, and no misclassification of idle movement occurred. The watch was given to a different user at the gym for a single set, however, and that set was misclassified. Perfect consistency with one user and variable accuracy with another user implies a lack of training data; in the future, our dataset will be trained on more than a single user.

Recognition achieved permissible levels of accuracy, with 80% of sets being reported within one repetition of their true count.

Considering this system was physically implemented, we can make inferences regarding battery life. The Moto 360 smartwatch used in this system lost approximately 25% of its battery life during our one hour testing session, and the Galaxy S4 smartphone used in this system lost approximately 15% of its battery life. Further user study will determine whether this is permissible or not; conserving battery life is an important factor for many in the mobile space, and a two hour gym session draining 50% of the watch's battery may not be acceptable. Future iterations of Android Wear devices should allay battery life concerns, however.

# CHAPTER 6

# CONCLUSION

An unsatisfactorily resolved issue inherent with the design of the system is dependence on bar speed. Many of the features computed for our SVM classifiers deal directly or indirectly with bar speed, which can vary greatly from set to set and user to user. Currently, our approach deals with this issue by drastically increasing the dataset and attempting to replicate every possible bar speed in the training data. Not only is this approach inelegant, but in its current state, it is insufficient. Comparison to a baseline repetition using DTW may solve the problem, but that has yet to be explored.

Another issue involves the core intuition that concurrent repetitions will be similar. When performing high-intensity lifts, form often breaks down and users end up "grinding" through the repetition, meaning that users pause partway through the lift and slowly push/pull the bar up. This violates our intuition, and in our current system, there is no way to deal with this. As such, our system is recommended only for bodybuilding-esque lifting, not low-repetition weightlifting or powerlifting.

Despite these shortcomings, our system performs remarkably well. Segmentation and recognition achieve consistently accurate results, and repetition counting can be improved by using a larger training dataset. We forsee our system being readily usable in the near future.

# REFERENCES

[1] "Pebble: E-paper watch for iphone and android," May 2015. [Online]. Available: https://www.kickstarter.com/projects/597507018/pebble-e-paper-watch-for-iphone-and-android

[2] "Google fit," May 2015. [Online]. Available: https://developers.google.com/fit/

[3] "Health. an entirely new way to use your health and fitness information," May 2015. [Online]. Available: https://www.apple.com/ios/whats-new/health/

[4] "Running and cycling gps tracker, performance analytics, maps, clubs and competition — strava," May 2015. [Online]. Available: https://www.strava.com/

[5] U. Team, "Sleep as android," May 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.urbandroid.sleep

[6] "Weightlifting," May 2015. [Online]. Available: http://www.olympic.org/weightlifting

[7] D. Morris, T. S. Saponas, A. Guillory, and I. Kelner, "Recofit: Using a wearable sensor to find, recognize, and count repetitive exercises," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557116 pp. 3225–3234.

[8] P. Elmer-DeWitt, "How many apple watches were sold worldwide last weekend?" 2015. [Online]. Available: http://fortune.com/2015/04/13/how-many-apple-watches-were-sold-worldwide-on-friday/

[9] D. Kessler, "More than 720,000 android wear devices were shipped in 2014, led by the moto 360," Feb. 2015. [Online]. Available: http://www.androidcentral.com/more-720000-android-wear-devices-were-sold-2014-lead-moto-360

[10] I. Pernek, K. A. Hummel, and P. Kokol, "Exercise repetition detection for resistance training based on smartphones," *Personal Ubiquitous Comput.*, vol. 17, no. 4, pp. 771–782, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1007/s00779-012-0626-y

[11] C. Seeger, A. Buchmann, and K. Van Laerhoven, "myhealthassistant: A phone-based body sensor network that captures the wearer's exercises throughout the day," in *Proceedings of the 6th International Conference on Body Area Networks*, ser. BodyNets '11. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2318776.2318778 pp. 1–7.

[12] M. Muehlbauer, G. Bahle, and P. Lukowicz, "What can an arm holster worn smart phone do for activity recognition?" in *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*, ser. ISWC '11. Washington, DC, USA: IEEE Computer Society, 2011. [Online]. Available: http://dx.doi.org/10.1109/ISWC.2011.23 pp. 79–82.