A Telemonitoring Solution to Long-Distance Running Coaching

Uma Balakrishnan
Carlos Asuncion
Hannah Sarver
Lucas Serven
Eugene Song
Ruzena Bajcsy, Ed.
Daniel Aranki, Ed.
Ali Javey, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-94
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-94.html

May 13, 2016

Acknowledgement

Daniel Aranki, University of California, Berkeley
Professor Ruzena Bajcsy, University of California, Berkeley
Professor Ali Javey, University of California, Berkeley
Dr. David Liebovitz, MD, University of Chicago Medicine

BERKELEY TELE-MONITORING

A Telemonitoring Solution to Long-Distance Running Coaching

Master of Engineering Capstone Report 2016

Uma Balakrishnan

with Carlos Asuncion, Eugene Song, Hannah Sarver, and Lucas Serven

Prepared for Professors Ruzena Bajcsy and Ali Javey

Contents

List of Figures

Introduction

1 Project Overview

Telehealth, which is defined as "the use of medical information exchanged from one site to another via electronic communications to improve a patient's clinical health status" [1], has been shown to reduce healthcare expenses and hospital admissions, and improve patient health. By improving the flow of information between doctors and patients, medical interventions can occur before more critical and expensive care, such as hospitalization, is needed.

The Berkeley Telemonitoring Project is a smartphone based tele-health framework that aims to give doctors, trainers and coaches an easy way to remotely monitor the health and/or performance of their patients or trainees [2]. Using this framework, doctors and coaches can develop Android applications that allow them to monitor and stay connected with their clients even with limited knowledge of programming or app development. Our capstone project will extend this framework, aimed at advancing tele-health, by adding various sensing, analysis and intervention capabilities. We do this by implementing a marathon training application to understand the specific needs of athletes in the context of the larger telemonitoring framework. We are working under the guidance of our faculty advisor, Professor Ruzena Bajcsy and our PhD student advisor, Daniel Aranki who were working with an M.Eng team on a telehealth framework last year, which we are trying to leverage and expand in the context of our specific application. This framework was motivated in part by a 2014 study by Aranki et al on real-time monitoring of patients with chronic heart failure

[3]. David Liebovitz, MD at University of Chicago, Medicine is advising us on understanding the best practices in marathon running and will guide the kind of results we want to arrive at in terms of our framework. The framework is made for Android devices and so is written in Java.

1.1 Telemonitoring and the Long-Distance Training Application

Telemonitoring essentially means the ability to monitor and control some aspect of the environment remotely. In a more general sense, the process of telemonitoring is a cyclic process which involves getting information from the user (sensing), sending this sensed data to a server, analyzing the data received from a number of such users, coming to a conclusion based on this analysis and providing some form of intervention or feedback to the user [Figure 1].

Our Android application is targeted at marathon runners. It collects information about the user's run, analyses this information and tells the runner what he can do to improve his performance in real time.

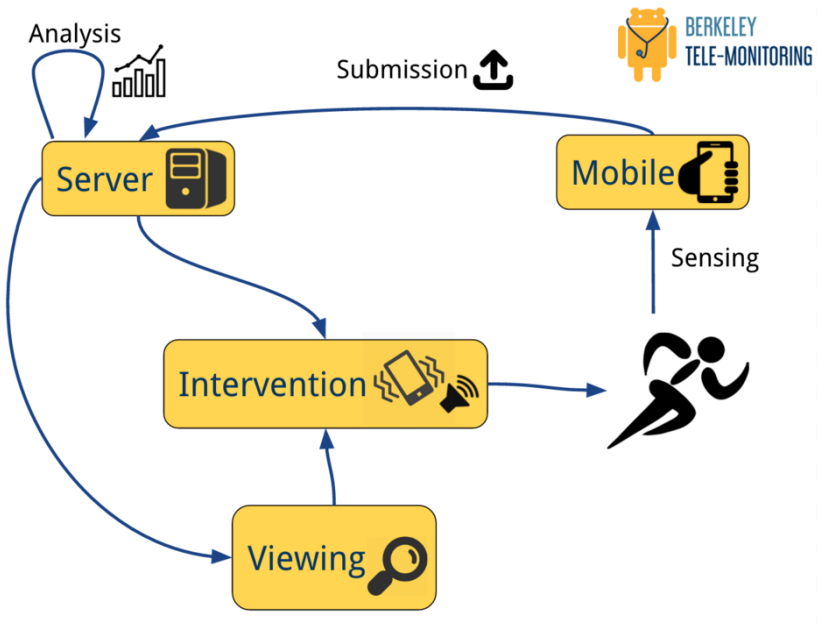

Figure 1: The Telemonitoring Loop

In the context of telemonitoring, as can be visualised in Figure 1, our application consists of the following elements:

- Sensing: The app gathers information about the runner including their speed, cadence (steps per minute), heart-rate, gender, age, weight, height and leg length. This information is sent to a server which performs the analysis.
- Analysis: The collected information is analysed based on our performance model, developed using several machine learning tools. Based on the results of the analysis, we estimate a target speed for the runner that is relayed back to their smartphone.
- Intervention: If the runner is running significantly above or below the target speed, the app will use audio or vibrations to communicate to them that they should be running faster or slower. This may be done for example, by changing the music appropriately so that the runner can adjust their speed to keep up with the rate of the music beats.

1.2 Work Breakdown Structure

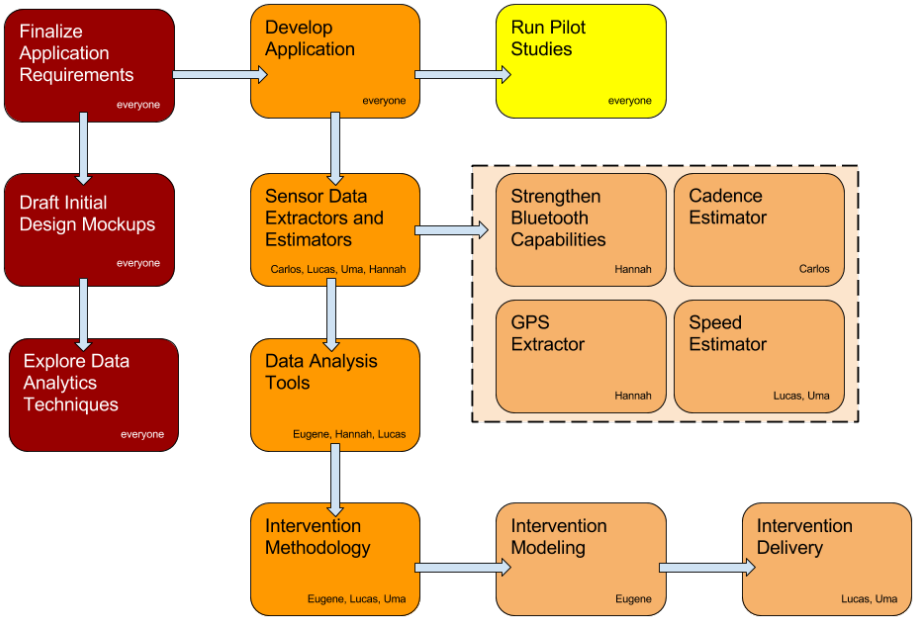

Figure 2: Work distribution between team members

A rough distribution of the project related task are illustrated in Figure 2. Many tasks

were shared by several members of the team including the design and implementation of the application and some of the analysis tools. Eugene Song's role largely dealt with the preparing the performance model and working on developing the Android application. Hannah Sarver worked on implementing the GPS extractors and on improving the framework's Bluetooth stack for connecting with an external heart-rate monitor. Lucas Serven, Carlos Asuncion and I worked on estimating running data (speed and cadence). All members of the team also contributed to data collection and preparing the proposal for the Institutional Review Board (IRB) for the purpose of conducting pilot studies.

Chapter 1

Individual Technical Contributions

1.1 Overview

This paper deals with a few different aspects of our project. First, it details the implementation of a prototype for our running speed estimation technique and initial results. Second, it talks about our data collection process for the purpose of training and validating our speed and cadence estimators. Next, it elaborates on my contributions to our Android application and proceeds to describe the methods by which we provide intervention and feedback to the runner, including a plan for future work. The paper finally offers some concluding thoughts on our project, and suggests possible improvements.

1.2 Speed Estimation Prototype

1.2.1 Motivation for Speed Estimation

For long distance runs like the marathon where the athletes are expected to cover large distances in a short time period, pacing oneself optimally throughout the course of the run is generally believed to be of significant importance [4]. Keeping this in mind, we wanted to keep track of the speed of the marathon runner throughout the course of the run. It is an important parameter to consider for implementing our performance model (refer to Eugene Song's technical contributions paper for more details) and will also be a useful reference

point for intervention.

While researching various accurate speed estimation algorithms we kept the following considerations in mind:

- The algorithm must only use data from sensors that are in-built on the smartphone, such as the accelerometer. We did not want the runner to don a number of wearable devices to estimate various parameters.
- The algorithm must allow the runner to carry the phone in a convenient location such as the pants or jacket pocket. We did not want to inconvenience the runners with awkwardly positioned belts that would not allow them easy access to their phone or would cause them discomfort while running.

While there are a number of smartphone apps which claim to report running speed [5], a large majority of them lack accuracy and/or reliability. A number of them (like MapMyRun, RunKeeper and Runtastic) use GPS to estimate the distance covered by the runner in a specific amount of time. GPS, however is largely dependent on the satellite signal strength at the runner's location and is also vulnerable to attacks and therefore quite unreliable [6]. Other applications calculate speed using cadence. They estimate the runner's stride length based on their height and calculate speed as product of cadence and stride length. This method is inaccurate because stride length does not remain constant during a long run and keeps changing depending on the slope of the road, and the energy levels of the runner. To avoid these errors generated by common speed estimation methods, we did an extensive literature review to help us decide on a reliable algorithm for speed estimation. We considered both kinematic models and statistical models. A more detailed description of our literature review can be found in Serven's paper. We settled on a 2012 paper titled "Online Pose Classification and Walking Speed Estimation using Handheld Devices" by Park et al as it seemed reliable and showed low error of less than 12 - 15%[7]. The algorithm uses only accelerometer data and allows the user to carry the accelerometer device (the smartphone in our case) in their pocket/hand/bag or held to their ear. Therefore, it satisfies our original

requirements.

1.2.2 Algorithm for Speed Estimation Prototype

The chosen algorithm uses a statistical model to estimate speed. For the purpose of our prototype, the accelerometer data are collected at the 'SENSOR_DELAY_GAME' rate supported by the Android sensor API which should typically correspond to 50Hz. Since these raw accelerometer data are not uniform in time and do not have a consistent sampling rate, the data are then re-sampled to get a sampling rate of 24Hz. We parse the data using a sliding window technique, where each window consists of 2.65 seconds of data (equivalent to 256 data points) with a 50% overlap between windows. It is assumed that the running speed remains constant during this window. Within these windows the accelerometer data are converted into the frequency domain (using Fast Fourier Transform) for feature extraction for the machine learning model. The frequency of human motion generally falls below 12Hz and so we assume that frequencies above that cutoff are mostly noise. To get rid of these higher frequencies we apply a low pass Butterworth filter of order 11 to the accelerometer signal.

For speed estimation, we use regularized least squares (RLS), a regression algorithm that benefits from regularization and the use of kernels. The features we extract from the accelerometer data are i) the frequency component of the magnitude of the acceleration signal (X_M) [Eq 1.2] where $|M(f_i)|$ is the absolute value of the frequency components at a sample point and ii) the spectral energy (x_E) [Eq 1.1].

$$x_E = \sum_{j=1}^{N} |M(f_j)|^2 \tag{1.1}$$

$$X_M = \left(\frac{|M(f_1)|}{\sqrt{x_E}}, ..., \frac{|M(f_N)|}{\sqrt{x_E}} \right) \tag{1.2}$$

These two features are combined using a custom kernel function [Eq 1.3] which is sum of radial basis function (RBF) kernels. These kernels are used to train and test the RLS model. A formulation of the RLS model can be seen in [Eq 1.4]

$$K_s(x, x') = exp\left(\frac{\|X_M - X'_M\|^2}{2\sigma_M^2}\right) + exp\left(-\frac{(x_E - x'_E)^2}{2\sigma_E^2}\right) \tag{1.3}$$

$$RLS : \min_{f \in \mathcal{H}} \frac{1}{2} \sum_{i=1}^{n} (f(X_i) - Y_i)^2 + \frac{\lambda}{2}\|f\|_k^2 \tag{1.4}$$

Here $\mathcal{H}$ is the set of all functions $f(\cdot) = \sum_{i=1}^{n} c_i k(X_i, \cdot)$ for some $c \in \mathbb{R}^n$ where k is the concerned kernel function.

λ is the regularization coefficient which is set to a value of 0.01 to prevent over-fitting of the model. σ_M and σ_E are half the median pairwise distances between values of X_M and x_E respectively. Y_i is the labeled speed value and X_i is the feature vector in a window such that $X_i = \begin{bmatrix} X_M^T \\ x_E \end{bmatrix}$. This paper will not delve into the details of RLS and will skip directly to the resultant formulae used in our algorithm. More on RLS can be found in [8] and [9].

By solving a system of linear equations we find that $c = (K + \lambda I)^{-1} Y$ where K is the kernel matrix for the entire training set and I is the identity matrix. The prediction at a new test point with feature vector X_* is given by $f(X_*) = \sum c_i k(X_i, X_*)$. The algorithm pipeline for the running speed prediction prototype can be seen in Figure 1.1

1.2.3 Prototype Implementation

As with all machine learning models, our speed estimator needed to be trained before it could predict speeds. As an initial working prototype we implemented the algorithm in MATLAB for training the model as well as testing its accuracy for speed prediction. This was meant to serve as an indicator of whether the chosen speed estimation method would satisfy our purpose in the domain of long-distance running. If the prototype showed promise, we would be able to integrate it into our smartphone application. The implementation in MATLAB therefore provided a relatively quick way to gauge whether the algorithm was

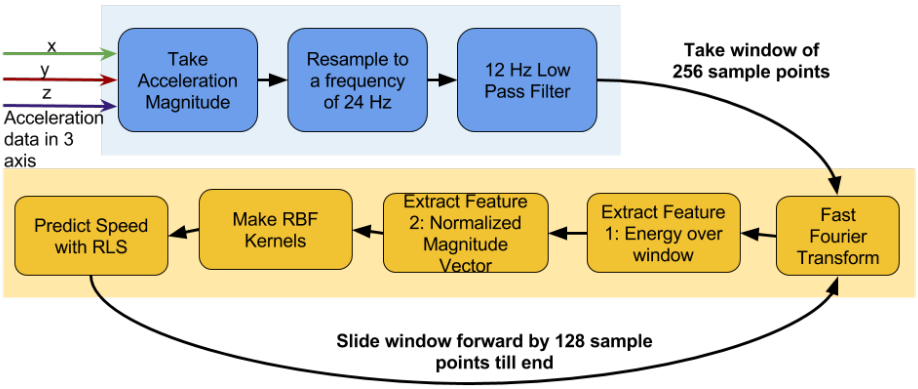

Figure 1.1: Speed prediction pipeline

worth pursuing.

We used our own collected data (more on this in section 1.3) to train the model and test it. For the purpose of the prototype we used only limited running data: 196 data points from 4 participants; which is an inadequate number when compared with the 2853 training samples used by Park et al. Our final Java implementation incorporated a larger amount of training data.

1.2.4 Results and Conclusions

With this limited amount of data however, we could see that the algorithm is promising. We tested our model using the leave-one-subject-out method (the model was tested on a subject who was not part of the training data set) and found that predictions at lower speeds were quite accurate. Our results are illustrated in Figure 1.2. We see that the prediction holds good in the range of values where we have maximum amount of training data, which is in the range of 1m/s to 3.5 m/s. Our training data did not have enough samples in the range of 4m/s and therefore those speeds give us inaccurate predictions. We therefore conclude

that training the model with more data at higher speeds will increase the accuracy of our prediction.

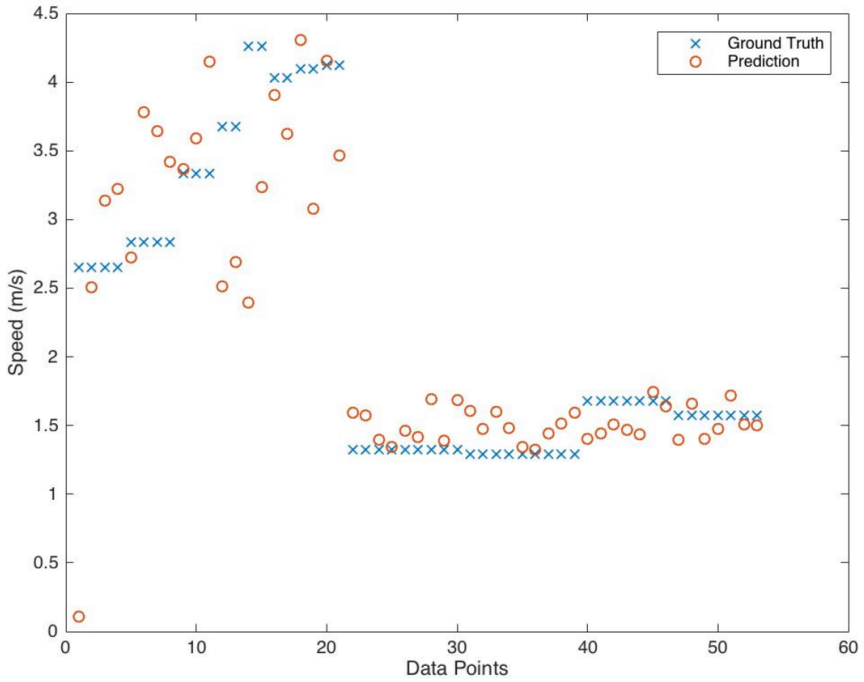

Figure 1.2: Speed prediction using leave-one-out method (RMS error = 0.6053m/s)

Lucas Serven's paper goes into the details of implementing the speed estimation algorithm in Java and integrating it into the telemonitoring framework. In order to put this speed predictor to test, we will be conducting our pilot study based on this developed model. A concern lies in the fact that the paper [7] uses this speed estimation model only for walking speeds while our application would require it to adapt to much higher running speeds. While we have reason to believe that this algorithm is more reliable than most existing running speed estimators on smartphone applications, the extent to which this is true is yet to be determined with more pilot studies and user feedback. Another concern lies in the fact that running the model on the smartphone would mean storing large amounts of data from the training set in the phone's limited memory. This may have the effect of slowing down the app or preventing it from running accurately in the case where sufficient memory is unavailable.

1.2.5 Results After Further Data Collection and Testing

After conducting the first round of pilot studies on ourselves, we found that the real-time speed prediction was consistently inaccurate for ground truth average speeds between 3m/s and 4m/s. Our predictor gave us much lower values in these ranges. Fortunately, through these pilot studies were able to amass a larger amount of data to train with. This should help us improve the accuracy of our predictor. We now have 1784 data points over a range of speeds [Figure 1.3] collected from 4 participants. Figure 1.4 shows us another test carried out (again with the leave-one-subject-out method) with the larger training data set. The final implementation of the prediction algorithm for our application is done in Java (refer to Serven's paper for details) where we use cross-validation to arrive at optimized values of λ, σ_m and σ_e.

Further work must be done to carry out cross validation to find the best value for the regularization coefficient with the new training data. Subsequently, further rounds of pilot studies must be carried out to evaluate the accuracy of the speed prediction algorithm in real-time.

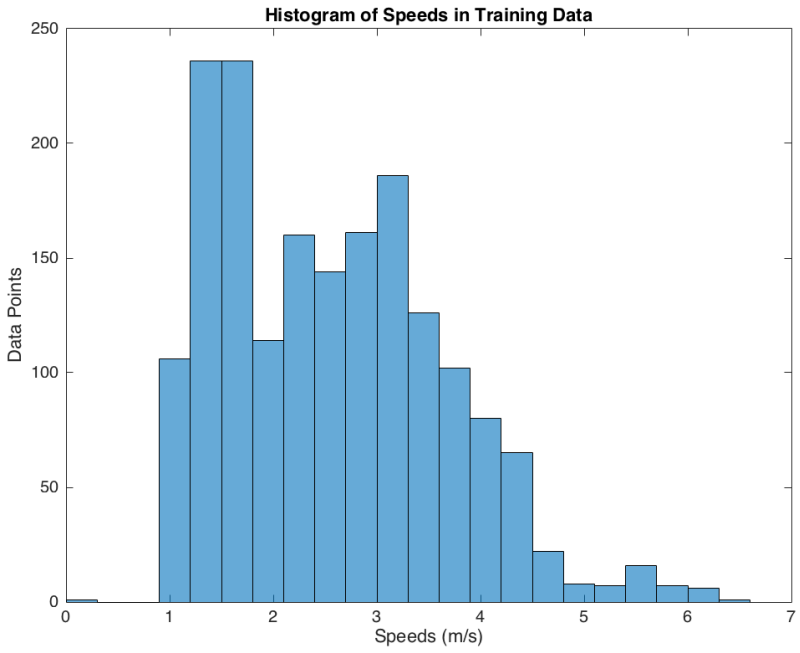

Figure 1.3: Number of sample points collected for different speed ranges

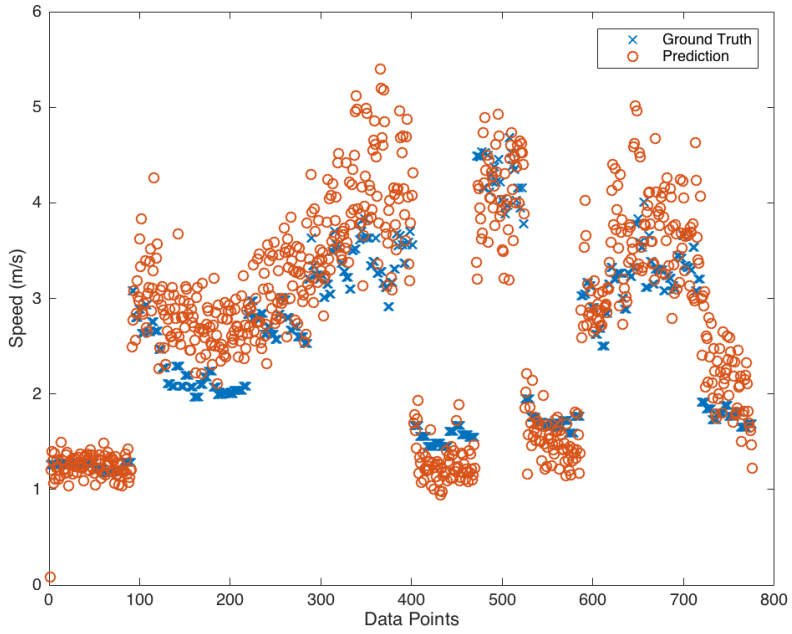

Figure 1.4: Speed prediction using leave-one-out method after further data collection(RMS error = 0.5455m/s)

1.3 Data Collection

1.3.1 Motivation for Data Collection

During our literature review, we found that the papers we referred to for estimation of speed and cadence did not give us a sufficiently detailed description of the methods used for data collection and settings under which they were carried out. We decided to fill this gap by following a more structured approach for data collection.

It is important to collect running data for speed and cadence estimation for the following reasons:

- Speed estimation uses a machine learning algorithm that needs to be trained. Training requires a large number of data points from a number of subjects.
- Verification and testing the accuracy of our algorithms for both speed and cadence require raw accelerometer data as well as the ground truth information.

To satisfy this requirement, I built two separate Android applications (only to serve as helpers). One to collect raw accelerometer data that would serve as inputs to the two algorithms and another to keep track of the ground truth. For cadence, the ground truth is the number of steps taken while the accelerometer data was being collected. For speed, the ground truth is the time in which a specified distance was covered.

1.3.2 Android Application for Data Collection

Raw Accelerometer Data Collection App

Android has several libraries [10] that facilitate data collection from on-board sensors such as the accelerometer. Using some of these libraries I wrote an Android application that records raw accelerometer data with a time stamp. The runner presses the start button when they start running and the stop button after the end of the run [Figure 1.5]. In this period the app will save a file onto the phone's memory which contains raw accelerometer values on the x, y and z axes along with the time at which these were recorded. During the run, the phone will be in the runner's pocket or attached to a belt near the hip. Our algorithms use only the magnitude of the acceleration in three axes and therefore the orientation of the phone within the pocket of the runner does not matter. This data would help us determine the reliability of our cadence and speed estimators. A sample set of the raw accelerometer data with the ground truth can be seen in Figure 1.6

Ground Truth App

To be sure that our estimators are accurate we must also keep track of the ground truth. With regard to cadence, this means keeping track of how many steps the runner took during the course of their run. Instead of having to keep count or have an observer keep count of this, this application will help serve this purpose. Every time the runner takes 5 steps, they touch a button on the screen [Figure 1.5] which updates the number of steps they took along with the time-stamp and saves it on a file in the phone's memory. We can compare this information to the results obtained from our cadence estimator to determine its correctness.

This means that during an initial test run, the runner will be carrying two phones, one in their pocket that estimates step count from raw accelerometer data and one in their hand that keeps track of ground truth. Our cadence estimator has given us satisfactory results, further details can be found in Asuncion's paper.

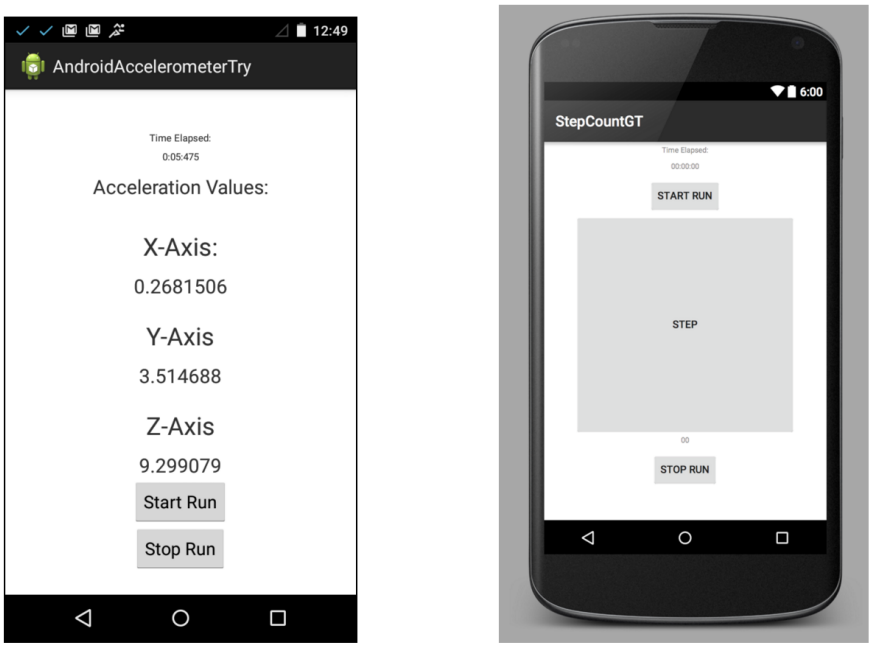

Figure 1.5: App for raw accelerometer data collection(left) and app for ground truth logging (right)

For speed the same app can be used to keep track of time. The runner presses the step button every time they cover a set distance (30ft in our case). This tells us the time it took the runner to cover a distance of 30ft which can give us their average speed for that distance. Again, while collecting data, the runner carries two phones: one in their pocket for gathering raw accelerometer values and another in their hand to keep track of distance covered.

1.3.3 Data Collection Methods

Data were collected by the members of our project team using the two apps described above. For evaluation of the cadence estimator, we collected several sets of walking and running data with the smartphone placed both in the pants pocket as well as in the jacket pocket in different orientations.

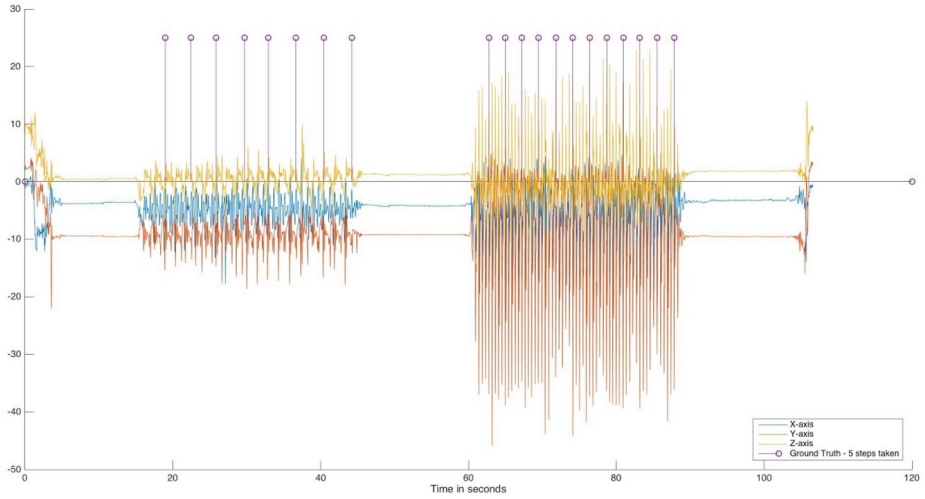

Figure 1.6: Raw accelerometer data together with ground truth collected in steps of 5

Collecting an exact ground truth for speed is not currently possible. When we run, all parts of our body do not follow a straight trajectory.The smartphone's accelerometer will experience, upward, downward and back and forth motions during the course of the run if its is placed in the runner's pocket. This prevents us from directly estimating speed as an integral of acceleration. There is no reliable way to be sure of the runner's speed at any instance of time. We attempted to take runs on different speed settings on a treadmill as this could give us a larger amount of well labelled data, however we observed that running on a treadmill does not produce the same pattern of accelerometer data as running on a static surface. Therefore, we opted to gather ground truth by calculating the average speed of the runner over short distances of 30ft. Four participants collected speed data while running, jogging and walking. After collection, raw accelerometer data on all three axes were labeled with a timestamp (in nanoseconds) and the ground truth speed (calculated from time intervals recorded using the ground truth app and the preset distance). Three participants' data were used to train the model and the last participant's data were used to validate its predictions. After performing validation, all the data were used for training. A greater amount of data increased the accuracy of speed predictions.

In order to be able to subsequently test our application on long distance runners for feedback, validation and testing, we submitted a proposal for conducting pilot studies to

our Institutional Review Board (IRB). This involved designing the proposal, planning out a procedure for the study, as well as all the related documentation. After a few rounds of comments and feedback, our proposal was approved.

1.4 Running Coach Android Application Development

Our Android application, called 'Running Coach' provides much of the functionality mentioned in the introduction. Of these, the aspects I worked on have been illustrated and explained below. Some modifications have been made to our original plan to satisfy the recommendations of the IRB.

1.4.1 Physical Parameter Settings

Among other things, our performance model uses the runners' height, weight, leg length, gender, age, target cadence and the date by which they would like to achieve that target cadence to formulate a training regimen relative to the start date. These inputs are provided directly by the user to the application on a Settings screen illustrated below. The subject ID is a unique code provided to each runner for the purpose of the pilot study to preserve their anonymity. Once set this cannot be changed. On installation, the app will direct the user directly to the settings screen so that the recruiting researcher may input their subject ID. Once this has been set the, the input is disabled so that user would not be able to make any changes to it. The user may also choose to not enter any of the values in the given list of physical and demographic parameters [Figure 1.7]. These parameters are:

- Gender
- Age
- Weight
- Height
- Leg length

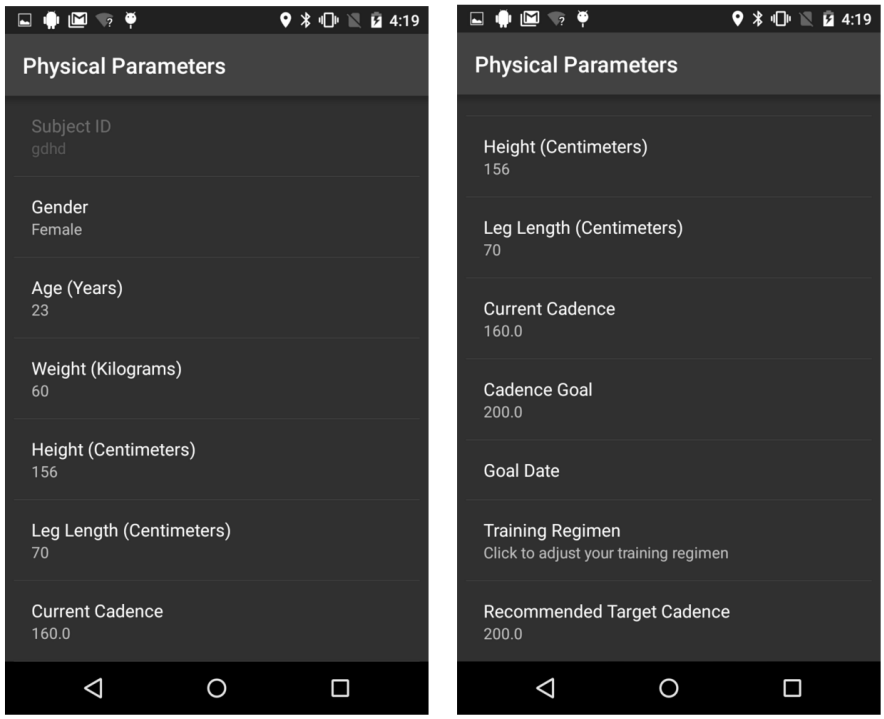

Figure 1.7: Physical and demographic parameter settings

1.4.2 Post-Run Survey

For the purpose of our pilot study, we want the user to give us feedback about the app after each run to see how closely the metrics of the actual run coincided with that provided by the application.To allow for this, we implemented a post run survey to see how tired the subject felt after that run (relative to how tired they typically felt during a run) and how accurate the provided information was. There are also a couple of open-ended questions which permit the subject to tell us how the app can improve. All these questions are presented in the form of a survey which is timestamped and sent to the server on completion. The survey screen can be seen in Figure 1.8. The questions (addressed to the subject) asked in the survey are the following:

1. How tired were you on a scale of 1-5 where 3 is your typical level of fatigue after long runs prior to using the app, 5 is very tired and 1 is least tired?

2. After viewing your run data, were any of the measurements inaccurate to the best of your assessment? (Choose all that apply from Speed, Cadence, Heart Rate, Stride

Length, Energy Expenditure, Distance)

3. If you selected any of the choices in the previous question, please explain.

4. Please provide any other comments regarding your experience using the app.

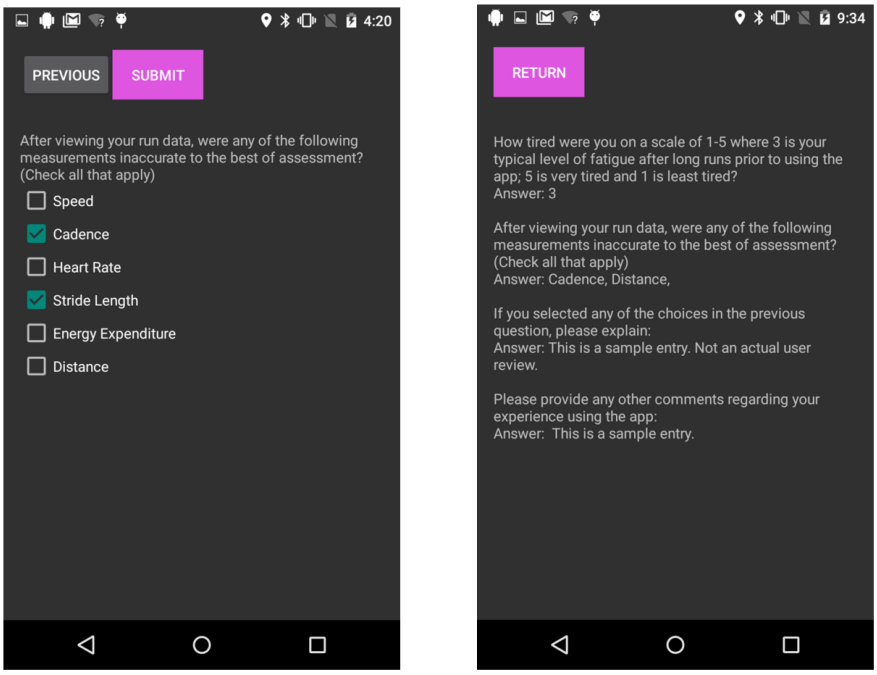

Figure 1.8: Post-run survey

1.4.3 Pre-Run and Post-Run Heart Rate Estimation

The Master of Engineering Capstone team from 2015 who were working on this project had developed two heart rate estimators for the telemonitoring framework. One estimates heart rate from the video of the user's face taken using the front camera of the phone [11]. The other estimates heart rate from the image of the user's finger, taken using the cell phone's back camera [12]. We wanted to measure the runner's heart rate value before and after a long distance run using these techniques so as to monitor their health and vital signs and for future studies. We also wanted to give the users the ability to choose whether or not they would like these measurements to be made. When the user wants to start their run, they press the 'Start' button on the application's home screen. The application then asks the

user whether they would like their heart rate measured from an image of their face (Figure 1.9). If the user says yes, a new screen will open, where the measurement occurs; if the user says no, the app asks them whether they would like the heart rate to be measured from the finger and acts accordingly (Figure 1.10). A similar procedure occurs when the user indicates that they have finished a run. The user may choose to take either measurement or both measurements. As soon as the measurement is taken, it is sent to the server with the subject ID and timestamp.

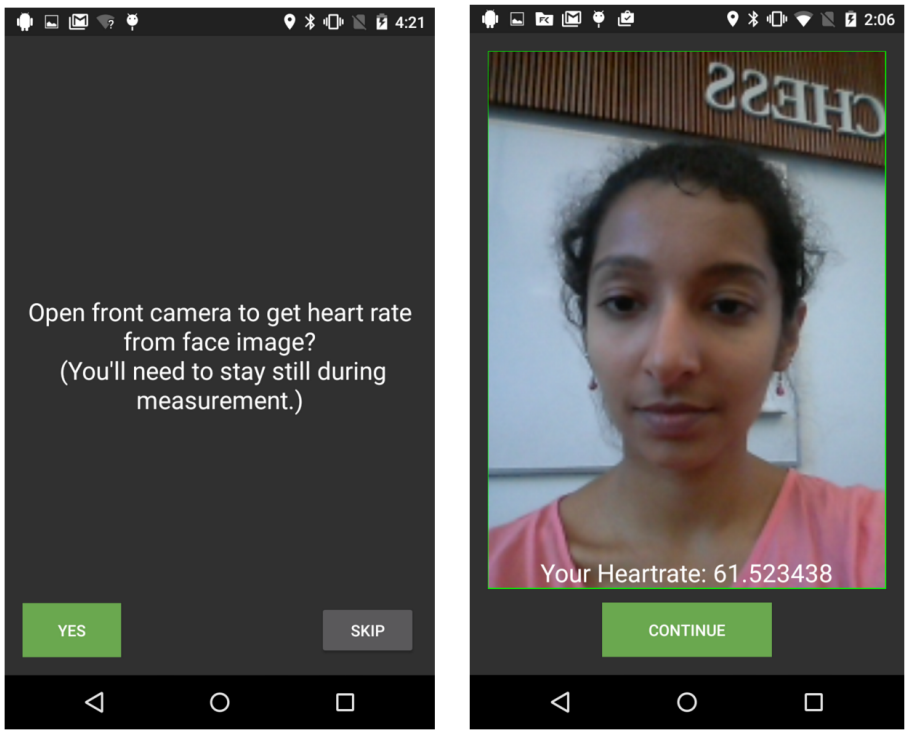

Figure 1.9: Option to estimate heart rate from face image

1.4.4 Concluding Remarks on Android Application Development

This section has covered only a few aspects of our Android application. The others can be found in my teammates' papers. These aspects include (but are not limited to) starting the various estimators and extractors we implemented, displaying the results of each run, and setting up a training regimen.

As a result of the IRB's recommendations, instead of providing intervention to the runners

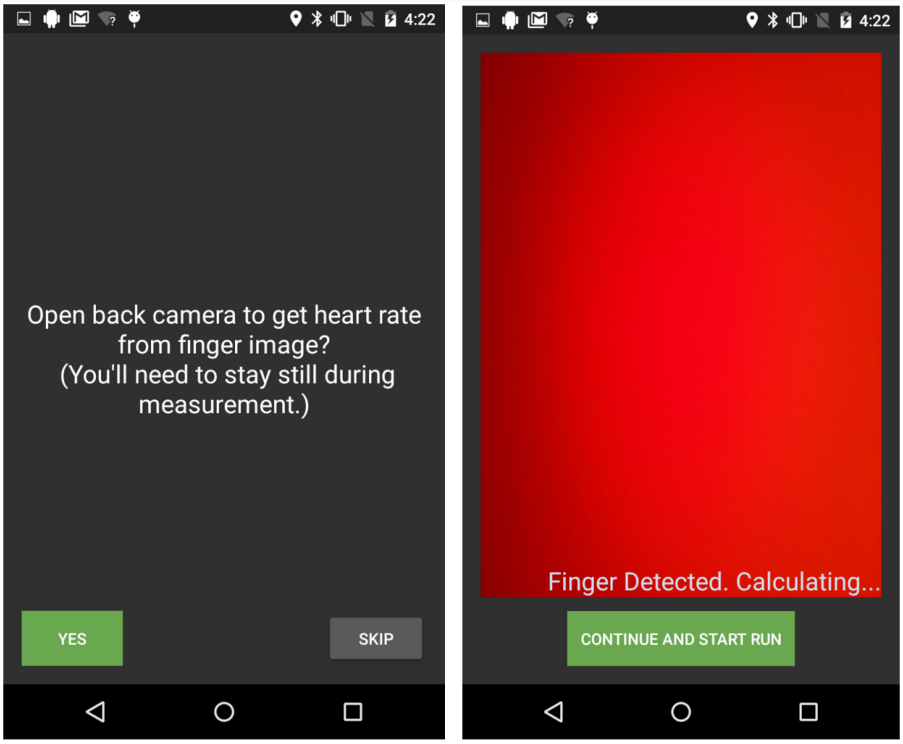

Figure 1.10: Option to estimate heart rate from finger image

depending on what we think their target cadence should be, we let the runners choose for themselves what they would like to set as their target cadence.

In future iterations of the app, we would improve both our user interface as well as the robustness of the application. We would also add more user friendly intervention techniques and also use GPS information to investigate and optimize the route followed by the user.

1.5 Intervention Delivery Methods

1.5.1 Motivation for Intervention Delivery

For the duration of the run, our long distance runner must try to maintain the target cadence indicated by their chosen training regimen. In order to let the runner know what their cadence or speed is at any point during the run, we must provide a form of feedback or intervention that will not hamper their run. For instance, the runner should not have to constantly look at their smartphone screen to know how far or close their are to their target

cadence. We therefore have to provide haptic or audio feedback to the user in the form of vibrations or tempo-controlled music to indicate to the runner that they need to run faster or slower to stay on track with their preset goals.

At present, our 'Running Coach' application will vibrate as well as generate a small tone when the runner is significantly far from their target cadence. This provision can be turned off by the user from the application settings.

In future iterations of the application we would like to incorporate a music player which plays songs that have a tempo (beats per minute) that matches the cadence (steps per minute) that the runner should be going at. This will allow for a more enjoyable form of reinforcement where the runner does not need to repeatedly look at their phone screen during the run.

1.5.2 Proposed Methods and Tempo Estimation

To begin the process of intervention described above, we follow the following steps:

1. Scan for and collect audio files from the users' smartphone
2. Extract the tempo of each song
3. Label each song with the corresponding tempo value
4. Get the target cadence for the run
5. Play music tracks with tempo that is as close as possible to the value of cadence (or half the value of the cadence, so that the runner can take two steps per beat of the song).

The first goal is to extract the tempo of the song. This problem is non-trivial considering that a most mainstream music tracks have several layers of notes and beats and a simple Fourier transform of the music will not give us an accurate tempo. Accurate tempo detection is a problem that is still being tackled by the scientific community. Some of the popular methods use the the spectral product and the autocorrelation function to get the periodicity of a onset notes extracted from the music [13]. Onsets are detected to extract the notable

features of a piece of music using a combination of spectral analysis and spectral energy flux estimation [14]. Another method breaks up the music signal into various frequency bands, does an envelope detection, splits them into different parts based on changes in sound, and applied a comb-filterbank to choose the highest energy component which would give the tempo of the song [15]. These algorithms are generally computationally complex or take a significant amount of time to arrive at a result. Our purpose is to make a quick, simple algorithm that can quickly compute the tempo for a potentially large number of songs and also choose songs that have clearly distinguishable and distinct beats that runners will be able to match their steps to.

To implement this we modified the first algorithm detailed in [16] which keeps track of beats by noting instances where the instantaneous energy (we consider a window of 0.02 seconds as being instantaneous) is greater than the average energy by a certain threshold. We use these energy spikes to find patterns where spikes repeat after more or less fixed intervals of time. The interval that occurs most often is considered as the periodicity of the beats from which tempo can be calculated.

1.5.3 Prototype Implementation

The algorithm was implemented in MATLAB. A diagrammatic representation can be found in Figure 1.11.

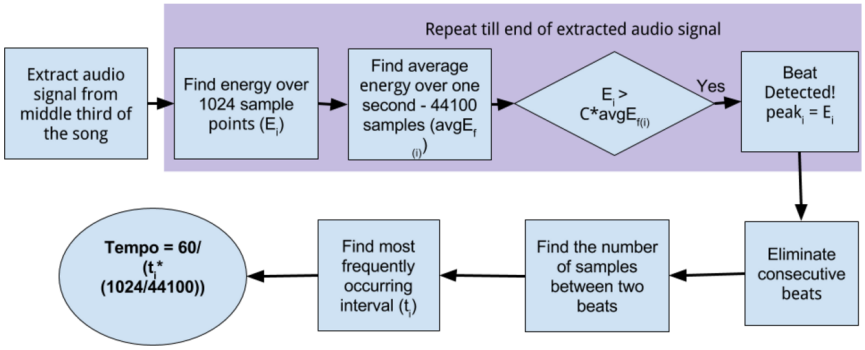

Figure 1.11: Tempo detection algorithm

Here we refer to a period of 0.02 seconds (corresponding to 1024 sample points for the

standard sampling frequency of 44.1KHz) as being instantaneous. Energy E_i is the sum of squared magnitudes of the audio signal for each of 1024 points in a sliding window. Average energy $avgE_{f(i)}$ is the average of E_i over 1 second which comes out to an average of 43 values of E_i. Each $avgE_{f(i)}$ is only compared with the E_i that lies at the center of the window. Subsequently E_i and $avgE_{f(i)}$ slide forward by 1024 points. Figure 1.12 illustrates instances with beats by contrasting the instantaneous energy and average energy over a set of sample points. The value of C (which is the coefficient of $avgE_{f(i)}$ when comparing with E_i) was originally taken to be 1.3 based on [16] but must be optimized using cross validation techniques to arrive at a more optimized value.

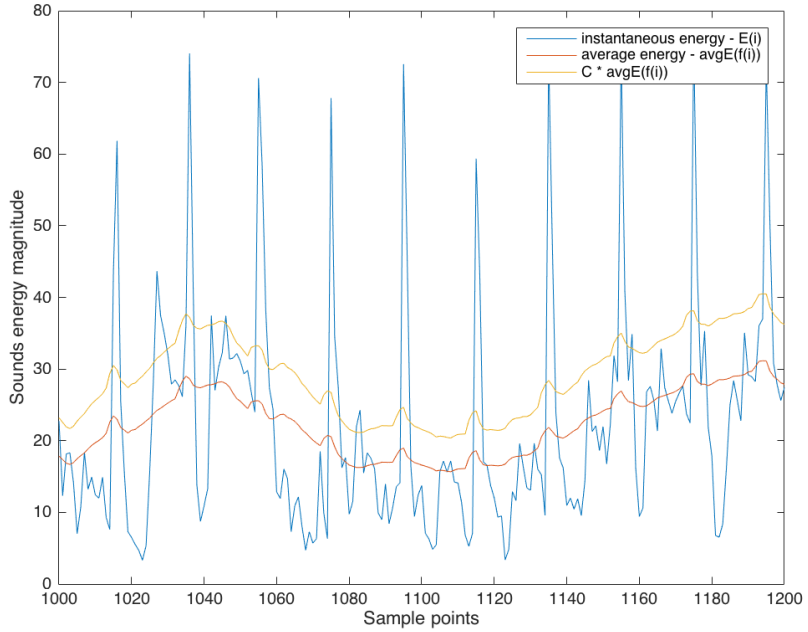

Figure 1.12: Illustration of beat detection

1.5.4 Results and Future Work

The above algorithm was tested using data collected by Martin McKinney and Dirk Moelants at Philips Research, the ground truth of which was established by having a number of humans tap to the music to keep track of its tempo [17]. This data set consists of 20 tracks of 30 seconds each with music from different genres. Of the 20 tracks, we get an accurate tempo

(a tempo is considered accurate if it falls within 10 beats per minute of the ground truth) for 11 (55% accuracy). However on listening to these tracks it can be seen that the songs on which the tempo detection fails consist mostly of violins, or wind instruments which do not intrinsically provide distinct beats. The algorithm performs much better (80% accuracy) on techno, rap, pop, R&B and rock songs which have drums or other percussion that dictate the beats. We may assume that this algorithm performs well enough to suit our purpose since we assume that while providing intervention we would want to only provide songs which would distinctly tell the runner what the required cadence is.

There is still work to be done in order to improve the performance of this algorithm. More specifically, the value of 'C' is yet to be optimised. In order to do this, the algorithm must be cross validated using a much larger number (around 100) of labelled songs to determine the value of of 'C' that would give us the best prediction. It is then to be tested on a much larger number of songs rather than our currently trivial set of 20.

As a first step, a basic music player application has been implemented on Android which collects the audio files stored on the runner's phone, displays them in a list and allows the user to choose which song they want to play (Figure 1.13). Work remains to be done in order to extract the tempo from these songs and play music that matches with the runner's target cadence. This music player has not been integrated into the first iteration of our Running Coach application.

We intend carry out subsequent pilot studies to verify our claim that providing intervention would improve the performance as well as satisfaction levels of the runner. A goal for the future is to gauge the difference in feedback between subjects who did receive intervention in different forms and subjects who did not. The results of this hypothesis test would help us modify our intervention techniques to fit the requirement and would also benefit anyone who wishes to develop an application in this area in the future.

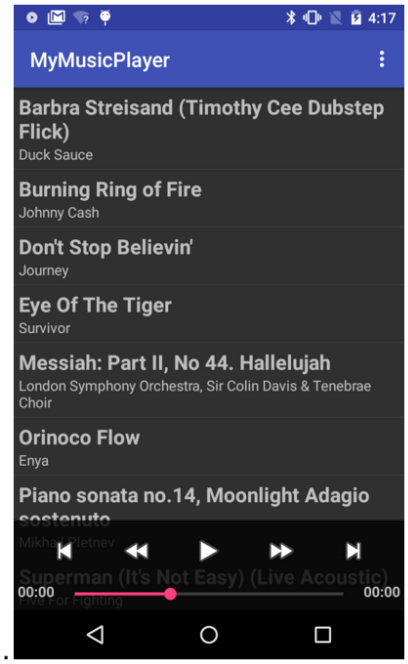

Figure 1.13: Basic music player on Android

1.6 Conclusion

We have sufficient reason to believe that telehealth frameworks like the one that our Android application is contributing to has much scope and potential in the fields of sports and medicine. The Whole System Demonstrator(WSD) launched by the UK in 2008 is the largest telehealth trial in the world. Results from the WSD showed that telehealth could reduce emergency admissions and patient mortality by 20% and 45% respectively [18].

All the current and past work associated with the Berkeley Telemonitoring framework is available on our website [19] with tutorials and and illustrations and much scope for further extension. The development of our Running Coach app as well as the data collected for these during our tests and pilot studies use the existing capabilities of the Berkeley Telemonitoring framework presented by Aranki et al in [2] and also add to it. We believe that this framework will benefit not just long distance runners and coaches but also future developers and computer scientists who seek to extend this framework further; and medical professionals and researchers who will be able to utilize this framework to make efficient health-care and tele-medicine more accessible and affordable.

Chapter 2

Engineering Leadership

2.1 Industry and Market Analysis Overview

While there exist several smartphone based platforms that can be used to create applications for various purposes, such as Canvas, Appery.io and Mobile Rodie [20] most of these offer limited functionality and access to sensors. Additionally, these products lack the ability to easily build predictive models for automated generation of personalized interventions. More generally, there are no such platforms that cater to the issue of telehealth. Our telemonitoring framework, targeted towards doctors and coaches, addresses this unmet need.

To guide the expansion of the framework, we will consider the design and implementation of new features in the context of a commercial application that would be used by marathon trainees. To this end, it is useful to perform a market and industry analysis on existing fitness tracking technology. By examining consumer behavior and industry offerings, we can better understand what functionality is missing and what features athletes desire.

2.2 Market Analysis

According to surveys [21] in 2014 there were more than 1200 marathon events held within the US, with a total of 550,637 finishers. These are both all-time high statistics, with the number of marathon finishers growing about 1.8% from 2013 to 2014. A survey of marathon

runners showed that 74% of them relied on wearable technology for training and 88% of them relied on said technology for motivation [22]. Between 2014 and 2015, the number of wearables purchased is said to have nearly tripled from 17.6 million to 51.2 million [23]. Of Internet users who exercise between the ages of 18-34, 38% of males and 21% of females use wearable fitness trackers [24]. Wearable technology has clearly entered into the mainstream, especially in the area of fitness training with fitness trackers. Marathon runners are no exception. With their ubiquity and proclivity for training technology, they represent an acceptable target market for our application.

2.3 Porter's Five Forces Analysis

We will now conduct a Porter's Five Forces analysis of our mobile application for marathon runners to contextualize it in the industry and develop a strategy for differentiating and promoting it [25].

2.3.1 Bargaining Power of Buyers

Buyers have strong bargaining power only when they are consolidated. Consumers of fitness tracking products are numerous, but diffuse in their buying patterns. Buyers are many and demand is great, weakening the bargaining power of buyers.

2.3.2 Bargaining Power of Suppliers

The power of suppliers refers to the power of businesses that provide materials to another business to demand increased prices for materials [25]. The application is developed for the Android platform, and cell phones have become a commodity, indicating a weak bargaining power.

2.3.3 Threat of New Entrants

New entrants have the potential to bring new ideas to a market. The market of activity monitors poses few barriers and connected fitness trackers are projected to grow from $2 billion to $5.4 billion from 2014 to 2019 [26]. With the burgeoning of the Internet of Things, it is expected that there will be new players in many applications of telemonitoring. Thus, the threat of new entrants is perceived to be strong.

2.3.4 Threat of Substitutes

A product from a different industry that offers the same benefits to consumers is referred to as a substitute product. Substitute products pose a threat because there is possibility of replacement for a company's product [25]. One substitute product for runners training for marathons is meeting one-on-one or in small groups with dedicated professional trainers and coaches. There is an approximately $1.5 billion industry existing in intensive personal athletic training in the United States [27]. This includes firms and independent individuals who provide services granting personalized attention to athletes training for sports seasons or upcoming events such as marathons. However, human trainers conducting in-person training generate problems not seen in the activity monitor/trainer application. For example, scheduling is a factor for this substitute, as the athlete would need to train according to the trainer's schedule and location. Having a human trainer is also significantly more expensive than using an activity monitor. The application does not come with these added cost and conditions. For these reasons we believe that the threat of substitutes is weak.

2.3.5 Rivalry Amongst Existing Competitors

Rivalry can pose a great threat when the rivals compete over price instead of features. The market for tracking and training of fitness, including endurance running, is a crowded one. In this market, our application will be competing with a variety of technologies, such as smartphone apps and specialized fitness tracking hardware. We will need to ensure that our feature offerings are differentiated in order to avoid significant threat from price-based

rivalry.

Wearable fitness tracking devices have seen widespread adoption among runners and other athletes. There are several subcategories of device functionality in this area, ranging in metrics measured, accuracy of these metrics, and price. These include step counters such as the Fitbit One or Nike+ FuelBand at the lower end of functionality and price, GPS-based speed and distance monitors like the Garmin ForeRunner 620 or TomTom Runner at the higher end, and multi-functional devices like smartwatches, such as the Apple Watch or Pebble Time that have some built-in fitness features [28].

Other competing fitness devices include specialized peripheral hardware, such as chest straps to monitor heart rate, shoe inserts to track impact and step duration, and devices that help athletes recover from training in terms of bloodflow and muscle relaxation, such as the Firefly Recovery System [29]. These products are more targeted at health monitoring and feedback for runners, which we can compete with by providing without specialized hardware outside of the mobile phone itself.

Additionally, given the demand for personal training, new products which provide personalized feedback, such as the Moov, have already begun to appear. Moov's successful crowdfunding campaign indicates a demand for fitness trackers that can provide this type of feedback [30]. Major players are pushing for greater personalization. For example, FitBit, a key player in wearable fitness tracking, acquired the startup FitStar in 2015 [31] which provides users with personalized instructional videos. Finally, our application will be competing with a host of other smartphone fitness applications. A huge market for personal fitness tracking exists in the app stores of the smartphones that so many Americans already carry with them daily. A study [32] estimated that in 2012 there were over 40 thousand health and fitness applications available for mobile phones, reaching over 500,000 active users, and that number has only increased in the past few years. A wide variety of fitness and run tracking, goal-setting and socially competitive, and motivational applications are available. Some of the most popular apps specifically targeted at runners are RunKeeper, MapMyRun, and Runtastic [28]. On the more creative side are apps like Zombies, Run

which provides audio motivation in a narrative form, taking a runner through customizable missions in a fictional environment.

Given the great number of players in this industry, the threat of existing rivals is strong. However, given the still largely unexplored area of personalized coaching within the crowded space of fitness tracking technology, we believe that this rivalry will primarily be features-based.

2.4 Technology Strategy

Considering our market research and Porter's Five Forces analysis, we have developed a strategy for our product in order to minimize the threats posed to our product. Our strategy revolves around marketing to customers based on the features offered by our product, particularly focusing on measurement and real-time feedback regarding performance metrics, such as speed and cadence. For instance, despite its importance, many fitness tracking solutions do not measure cadence. Furthermore, the products that do are typically not transparent about the estimation algorithms used and their accuracies. Even for those that do report accuracy, the algorithms used are still unpublished, and the accuracy of specific metrics, such as cadence, are conspicuously missing [33]. Our application uses algorithms backed by published scientific literature, and the accuracy of our implementation will be further measured and published. Furthermore, the framework on which the application is built includes a fault-tolerant client-server protocol for secure and convenient data syncing, and a wide library of well-tested data collection and analytics functionality to support our application's features and ensure they remain reliable and easy to use. Raising the standards of information transparency, estimation accuracy, and application reliability would not only allow our application to gain traction in the market if we were to actively promote it, but would also impose barriers to new entrants.

Bibliography

[1] A. T. Association. (2012). "what is telemedicine?" american telemedicine association, [Online]. Available: http://www.americantelemed.org/about-telemedicine/what-is-telemedicine.

[2] D. Aranki, G. Kurillo, A. Mani, P. Azar, J. van Gaalen, Q. Peng, P. Nigam, M. P. Reddy, S. Sankavaram, Q. Wu, and R. Bajcsy, "A telemonitoring framework for android devices," in *Proceedings of the 1st IEEE Conference on Connected Health: Applications, Systems and Engineering Technologies*, to appear, IEEE, Jun. 2016.

[3] D. Aranki, G. Kurillo, P. Yan, D. M. Liebovitz, and R. Bajcsy, "Real-time tele-monitoring of patients with chronic heart-failure using a smartphone: Lessons learned,"

[4] R. C. Coach Jeff. (2011). Why pacing is important for runners, [Online]. Available: http://runnersconnect.net/running-training-articles/improve-your-pacing-skills/.

[5] K. Munoz. (2016). 11 best mobile apps to run faster and smarter, [Online]. Available: http://greatist.com/fitness/best-running-apps-faster-smarter.

[6] J. S. Warner and R. G. Johnston, "Gps spoofing countermeasures," *Homeland Security Journal*, vol. 25, no. 2, pp. 19–27, 2003.

[7] J.-g. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie, "Online pose classification and walking speed estimation using handheld devices," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 2012, pp. 113–122.

[8] B. Schölkopf and A. J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[9] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.

[10] G. Inc. (2012). Sensors overview — android developers, [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[11] Q. Peng, P. Azar, A. Mani, and J. Van Gaalen, "Expanded tele-health platform for android," Master's thesis, EECS Department, University of California, Berkeley, May 2015. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-90.html.

[12] J. Van Gaalen, P. Azar, A. Mani, and Q. Peng, "Expanded tele-health platform for android," Master's thesis, EECS Department, University of California, Berkeley, May 2015. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-86.html.

[13] M. A. Alonso, G. Richard, and B. David, "Tempo and beat estimation of musical signals.," in *ISMIR*, 2004.

[14] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[15] C. et al. (2001). Beat this ¿ beat detection algorithm, [Online]. Available: https://www.clear.rice.edu/elec301/Projects01/beat_sync/beatalgo.html.

[16] Gamedev.Net. (). Beat detection algorithms, [Online]. Available: http://archive.gamedev.net/archive/reference/programming/features/beatdetection/index.html.

[17] Music-Ir.Org. (2006). 2014:audio tempo estimation - mirex wiki, [Online]. Available: http://www.music-ir.org/mirex/wiki/2014:Audio_Tempo_Estimation#Practice_Data.

[18] U. D. of Health. (2011). Whole system demonstrator programme: Headline findings, [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/215264/dh_131689.pdf.

[19] (2016). The berkeley telemonitoring project - privacy-aware health telemonitoring, [Online]. Available: https://telemonitoring.berkeley.edu/.

[20] G. Smith. (2013). 10 excellent platforms for building mobile apps, [Online]. Available: http://mashable.com/2013/12/03/build-mobile-apps/#SXF0URANsqqg.

[21] Running USA. (2015). 2014 running usa annual marathon report, [Online]. Available: http://www.runningusa.org/marathon-report-2015.

[22] Freescale. (2014). The next evolution in running, [Online]. Available: https://web.archive.org/web/20141223192158/http://blogs.freescale.com/iot/2014/12/wearables-next-evolution-in-running-marathon/.

[23] Gfk. (2015). Gfk forecasts 51 million wearables will be bought globally in 2015, [Online]. Available: gfk.com/news-and-events/press-room/press-releases/pages/gfk-forecasts-51-million-wearables-sold-globally-2015.aspx.

[24] Mintel, "Exercise trends - us - october 2014," Mintel, Market report, 2014.

[25] M. E. Porter, "The five competitive forces that shape strategy," *Harvard Business Review*, vol. 86, no. 1, pp. 78–93, 2008.

[26] Parks Associates. (2015). Global revenues from connected fitness trackers to exceed $5 billion by 2019, [Online]. Available: http://www.parksassociates.com/blog/article/pr-march2015-whcc.

[27] D. Witter, "Ibisworld industry report 61162: Sports coaching in the us," IBISworld, Industry report, 2015.

[28] J. Carter. (2013). 10 best running gadgets: The top tech for training, [Online]. Available: http://www.techradar.com/us/news/world-of-tech/roundup/10-best-running-gadgets-the-top-tech-for-training-1157180/1.

[29] K. Alger. (2014). Training for a marathon: Tech for half marathon, marathons and beyond, [Online]. Available: http://www.t3.com/features/training-for-a-marathon-half-marathon-ultra-and-beyond.

[30] J. Colao. (2014). Who needs kickstarter? exercise sensor moov raises $1 million in 15 days, [Online]. Available: http://www.forbes.com/sites/jjcolao/2014/03/14/exercise-sensor-moov-raises-1-million-in-15-days-without-kickstarter/#704c414614e3.

[31] R. Lawler. (2015). Fitbit confirms fitstar acquisition to bring training to its fitness portfolio, [Online]. Available: http://techcrunch.com/2015/03/05/fitbit-confirms-fitstar-acquisition-to-bring-training-to-its-fitness-portfolio/.

[32] Stanfy. (2013). Fitness in mobile: Case-study, [Online]. Available: http://www.slideshare.net/stanfymobile/fitness-cs-22962137.

[33] Garmin. (2016). How accurate are the running dynamics from the hrm-run and how was the accuracy tested? [Online]. Available: https://support.garmin.com/support/searchSupport/case.faces?caseId=%7B435cc8c0-e2e1-11e3-47b1-000000000000%7D.