RunningCoach – Cadence Training System for Long-Distance Runners

Daniel Aranki[*]
daranki@cs.berkeley.edu

Uma Balakrishnan[†]
bkrishnan.uma@gmail.com

Hannah Sarver[†]
hbsarver@gmail.com

Lucas Serven[†]
lserven@gmail.com

Carlos Asuncion[†]
c.asun93@gmail.com

Kaidi Du[*]
kaidi_du@berkeley.edu

Caitlin Gruis[*]
cgruis@berkeley.edu

Gao Xian Peh[*]
pehgaoxian@berkeley.edu

Yu Xiao[*]
xiaoyu9376@berkeley.edu

Ruzena Bajcsy[*]
bajcsy@eecs.berkeley.edu

ABSTRACT

Long-distance running is a category of sports that is injury-prone. Half of the injuries sustained in long-distance running are at the knee and are attributed to the inability of the lower extremity joints to sufficiently handle the load applied during initial stance. Furthermore, cadence (steps per minute) has been identified as a factor that is strongly associated with running-related injuries. Increasing cadence results in reduced energy absorption at the hip and the knee, thus reducing the risk of some common running injuries. Therefore, it is vital for runners to run at an appropriate running cadence in order to minimize risk of injury. In this paper, we present an mHealth system that remotely monitors running cadence levels of runners in a continuous fashion, among other variables, and provides immediate feedback to runners in an effort to help them optimize their running cadence. We also present some initial findings based on a feasibility study we are currently conducting using this system.

INTRODUCTION

The modern ubiquity of sensing hardware and networking capabilities creates the potential for monitoring, coaching and intervention, even remotely. The Berkeley Telemonitoring Project[1] seeks to tap into this potential by providing a platform on which to quickly and easily build real-time telemonitoring and intervention apps that can be deployed on hardware running the Android operating system [2]. These apps can include those involving health monitoring or sports coaching. While fitness tracking technology has become commonplace in recent years, commercial devices often obfuscate their proprietary algorithms. The public is not privy to how these devices detect steps, measure speed or measure cadence. This furthermore limits the ability of the scientific community to easily validate these algorithms [1]. An additional advantage of the Berkeley Telemonitoring Project is its open source status. Any algorithms, such as those used for our RunningCoach app, can be verified by the community.

To demonstrate the capability of the Berkeley Telemonitoring Project, we designed a coaching app for long-distance runners–RunningCoach–and implemented it using the Berkeley Telemonitoring Framework. The app aims to optimize the runner's cadence, defined as *steps taken per minute*, in an effort to reduce injury and improve running performance, since up to 79% of long-distance runners are expected to sustain a running-related injury in the lower extremities [19]. Proper cadence has been connected with reducing impact forces on joints [9], reducing muscle soreness and fatigue [15] and increasing efficiency of oxygen use [8]. The multitude of advantages to cadence control is apparent. Consqeuently, we are interested in developing interventional models to adjust runners' cadences for greater safety and efficiency in running. For that purpose, we built RunningCoach to measure the cadences of runners, among other relevant variables.

Conforming to the design principles of the Berkeley Telemonitoring Framework, RunningCoach consists of two nodes: the client and the server. The client is the Android-based device, such as a smartphone, and the corresponding application that

PervasiveHealth '17, May 23–26, 2017, Barcelona, Spain

ACM ISBN 978-1-4503-6363- 1/17/05. . . $15.00
https://doi.org/10.1145/3154862.3154935

[1]The Berkeley Telemonitoring Project: https://telemonitoring.berkeley.edu

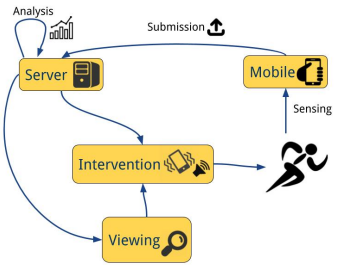


Figure 1: The telemonitoring flow of RunningCoach.

interacts directly with the environment being monitored. The client receives input from the environment via sensors, while indicating interventions as an output. The client node is designed to submit the sensed and estimated fitness parameters of the runner to the server.

On the other end, the server analyzes the data it receives from the client and generates intervention if necessary. The RunningCoach server also contains a user interface that allows the viewing and visualization of the runners' data. The workflow of RunningCoach is depicted in Figure 1. In addition, we are conducting a user study with long-distance runners to test the efficacy of this system. Currently, we have 6 subjects enrolled in the ongoing feasibility phase of the study; and in this paper, we report on its early findings.

The rest of the paper is organized as follows. In *Related Work*, we survey the literature for similar works. In *Client Design*, we describe the mobile app design and its implementation. We then describe the server design and its implementation in *Server Design*. We present the early findings of the feasibility study in *Feasibility Study* and summarize in *Discussion and Future Work*.

RELATED WORK

Running-related injuries have been extensively studied in the literature. Moreover, there exists a plethora of commercial products that assist runners in their running exercises. These commercial products include apps, wearables and insoles that measure fitness markers such as energy expenditure, speed, distance, heart rate and cadence. For the sake of brevity, we focus in this section on scientific literature that study the link of running cadence to running-related injuries. Many have studied the relationship between music and setting the running cadence through auditory-motor synchronization [4, 17]. These studies provide evidence of the ability of music to affect the running cadence and ultimately improve performance [4] and reduce injuries [9, 17]. Among the most common causes of injury are lack of stretching, excess of training, and wearing the wrong shoes [16]. Other studies quantified the incidence and factors of lower extremity running injuries in long-distance running. We refer the reader to the systematic review by van Gent et al. [19]. It is worth noting the study by Heiderscheit et al. [9] studying the joint mechanics as a function of cadence and the cadence effect on running injuries.

Other studies focused on the environmental factors in relation to injuries and performance in long-distance running. Marr and Ely studied the effect of air pollution on the performance of runners in marathons [10]. More generally, El Helou et al. studied the relationship between more generic environmental factors (e.g., atmospheric moisture and pressure, humidity and temperature) and the performance of runners in marathons [6]. Péronnet et al. focused on devising an index of endurance (high utilization of maximal oxygen uptake) for marathon running [13]. Our purpose, which we aim to achieve after a series of studies, is to develop a recommendation algorithm that outputs an optimal cadence level for a runner based on her physical parameters (e.g., age, gender, height and weight).

CLIENT DESIGN

RunningCoach goals

There are two purposes for the client node in RunningCoach, which is an Android app designed for smartphones running Android version 4.3 (Jelly Bean MR2) or newer. The first purpose is to measure, estimate and sense the runner's environment and running-related features. The second purpose is to provide feedback and intervention to the runner when necessary in an effort to optimize their cadence and other running-related parameters.

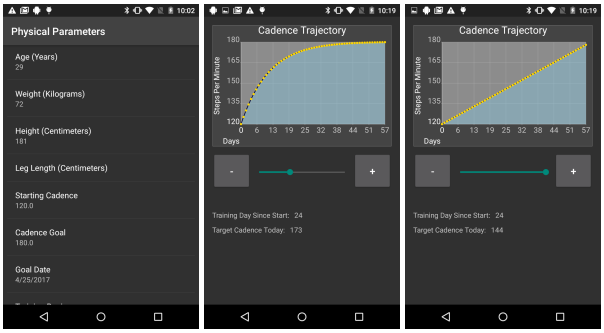
Training regimen

For the app to achieve the coaching goals stated above, it is necessary to establish a training regimen for the runners. Since different runners have different levels of experience and body types, a single rigid coaching model is not necessarily generalizable for all runners. To handle the heterogeneity of users, the smartphone app collects two types of inputs before the first run, namely the runner's i) physical parameters and ii) desired cadence improvement trajectory curve. In the current iteration of the study, the baseline and target cadence of the runner are both set manually by the runner. The collected physical parameters include age, gender, height, weight and leg length (Figure 2a).

Once the physical parameters are provided, the runner can configure the training regimen by selecting the target date by which she or he wishes to achieve the target cadence (default is 90 days) and selecting the steepness of regimen. The generalized cadence training regimen follows the following exponential curve

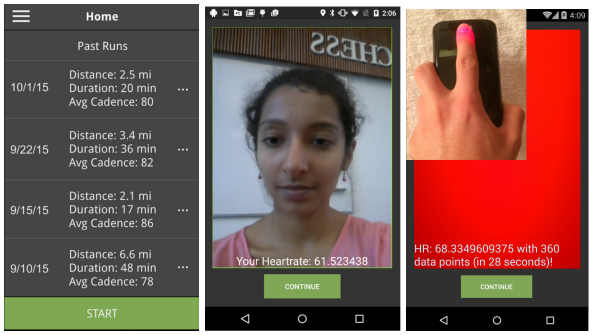
$$C(d) = \frac{C_N \cdot e^{\alpha N} - C_0}{e^{\alpha N} - 1} - \frac{C_0 - C_N}{e^{-\alpha N} - 1} \cdot e^{\alpha d} \qquad (1)$$

where $C(d)$ is the cadence at day d, C_0 is the initial cadence, C_N is the target goal cadence, α is the steepness of the training regimen and N is the length of the training regimen by days. By providing C_0, N, C_N and α, we establish a regimen that sets the target cadence over time as the runner trains. The curve described in Equation (1) is essentially a function of the form $C(d) = A + B \cdot e^{-\alpha d}$ that passes through the points $(0, C_0)$ and (N, C_N). Note that the training regimen defined in Equation (1) can be used to define a linear-improvement training regimen as α approaches 0. That is, $\lim_{\alpha \to 0} C(d) = C_0 + \frac{C_N - C_0}{N} \cdot d$. The proof of this claim can be found in [3]. The rationale



(a) (b) (c)

Figure 2: Screenshots of the physical parameters and training regimen portion of the RunningCoach app. (a) Runner profile; (b) cadence training regimen with exponential improvements; and (c) cadence training regimen with linear improvements.



(a) (b) (c)

Figure 3: (a) The start screen with the list of previous runs; (b) an example of the face-video-based heart rate measurement screen; and (c) an example of the finger-video-based heart rate measurement screen.

behind following a curve to gradually reach the target cadence is minimizing the risk of injury due to sudden changes in the running routine.

Figure 2b depicts an exponential-improvement training regimen for the cadence ($\alpha > 0$), while Figure 2c depicts a linear-improvement training regimen for the cadence ($\alpha \to 0$).

In the current version of the system, the cadence settings are manual. Ultimately, we wish to devise a recommendation algorithm that would suggest, for each runner, i) an ideal cadence level; and ii) a recommended steepness for the training regimen, based on her or his provided physical parameters in Figure 2a. We will use the collected data to train a model that can automatically recommend an improvement trajectory based on a runner's physical parameters and current running level. For more information about the training regimen, we refer the reader to [3].

Running experience

The home screen of the app is depicted in Figure 3a. Once a runner clicks on the **START** button, the app prompts her or him to take two measurements of heart rate. The first heart rate measurement is based on an estimate from a video of the runner's face. The implementation, which is described in [18], is a real-time version of the algorithm described in [14]. Figure 3b displays a screenshot of the screen collecting this measurement.

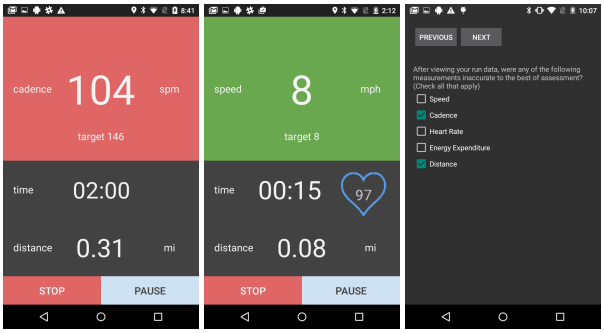
The second heart rate measurement is based on an estimate from a video of the runner's index finger. The implementation, which is described in [18], is based on the algorithm described in [5]. Figure 3c displays a screenshot of the screen collecting this measurement, with a photo of a runner's hand while taking the measurement. Both of these heart rate measurements, which can be skipped, are collected to establish the resting heart rate of the runner and to test the usability and accuracy of these methods.

After taking both measurements the app starts the run. During the run, the phone is expected to be on the runner's body, ideally on the waistline. During the run, the app monitors the runner's cadence, speed, distance covered, energy expenditure, heart rate (using a chest strap) and location through global positioning system (GPS), providing feedback to the runner. Depending on the preferences set by the runner, the feedback alerts the runner whenever she or he deviates more than a preset percentage from the target cadence of the day (following Equation (1)) or from a preset target speed. The default deviation percentage is 10%. If a significant deviation from the target is detected for more than 20 seconds, the app provides haptic and auditory feedback (vibration and audio beeps) to inform the runner whether to increase or decrease her or his cadence or speed to come closer to the target. Additionally, if the runner happens to be looking at the phone, visual feedback is provided in the form of a color change when there is a deviation from the target. Figure 4a depicts a screenshot of the running screen showing cadence that is out of the 10% of the target cadence of the day, whereas Figure 4b depicts a screenshot of the running screen showing speed that is within 10% of the preset target speed, in addition to a heart rate reading.

Post-run survey

When the runner finishes a run by clicking the **STOP** button, the app prompts her or him to take two measurements of heart rate, similar to the ones before the run. Similarly, these two measurements can be skipped. Furthermore, we wanted to enable the user to give us feedback about the app after each run to see how closely the metrics of the actual run coincided with those estimated by the app. 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. We also supplied two 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 questions asked in the post-run survey are i) "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



(a) (b) (c)

Figure 4: (a) A screen during a run showing cadence that deviated from the target cadence of the day; (b) a screen during a run showing running speed and a heart rate measurement; and (c) an example of a question in the post-run survey.

and 1 is least tired?" ii) "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, Energy Expenditure, Distance);" iii) "If you selected any of the choices in the previous question, please explain;" and iv) "Please provide any other comments regarding your experience using the app." Figure 4c depicts an example of a question from the post-run survey as displayed in the app.

Software architecture

The software objects on the client that are pertinent to sensing and measurement are extractors and estimators. Extractors are responsible for reading data directly from phone sensors (e.g., global positioning system (GPS)), while estimators derive secondary measurements from the sensor readings provided by the extractors (e.g., estimating cadence from the accelerometer sensor). Data that are to be stored persistently, transmitted to the server, or both, are placed into encapsulators, which are software objects that facilitate data storage and transfer in a fault-tolerant manner [2].

A background monitoring service runs continuously and asynchronously as the app is collecting data, even when the app is not in the foreground. This service's roles are to initialize the data collection and to maintain its continuity throughout the killing of processes that occurs as part of the Android operating system's standard resource management routines. Additionally, the monitoring service mediates the association of encapsulators and server handlers, ensuring that data is streamed to the server, while, again, being tolerant of potentially disruptive events from the operating system.

Next, we will briefly describe the design of the estimators for cadence and speed, which are now incorporated in the Berkeley Telemonitoring Framework. The other estimators used in RunningCoach were implemented in the Berkeley Telemonitoring Framework prior to this work and are described in [2].

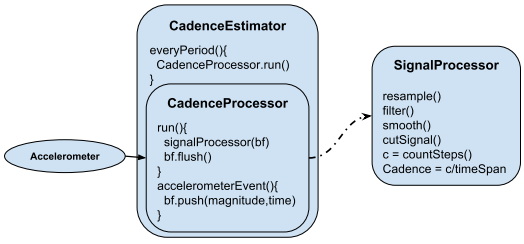


Figure 5: A class diagram for the CadenceEstimator, CadenceProcessor and SignalProcessor describing the architecture and the pipeline of the cadence estimation algorithm.

Cadence estimator design

Motivation

There is a plethora of commercial solutions for measuring the cadence of a runner. Some of them measure cadence through an external device, while others do so using built-in sensors of a smartphone and an app. Access to the cadence data through some of these solutions may be controlled with a restricted application program interface (API) that may be web-based. Such an API would require Internet access, which may be unavailable during the run. Moreover, other solutions completely restrict data through the use of propriety apps. Overall, these options can limit the functionality and portability of the RunningCoach app.

Furthermore, techniques used in existing commercial solutions for cadence estimation are not public information and therefore verification for accuracy would be complex. For these reasons, we decided to develop a cadence estimator, which employs verifiable algorithms presented in academic work. The method of cadence estimation utilizes the sensors built into the smartphone, staying true to the theme of the framework. For this implementation of cadence estimation, we will assume the use of built-in accelerometers and sensor interface provided by the Android API.

Algorithm

The step counting algorithm presented by Mladenov and Mock was the most promising approach of those surveyed [11]. The main reasons for choosing this algorithm are i) simplicity; and ii) accuracy (as reported in [11], compared to the other algorithms surveyed). This algorithm is also independent of the phone's orientation (by design) and can tolerate small deviations in the phone's placement on the body [11]. The algorithm is a simple peak detection process, which considers peaks above a dynamic threshold to be steps. Also, although accelerometers on phones produce 3-D acceleration information, this algorithm utilizes the magnitude of the acceleration vector, rendering it independent of orientation. This further makes the algorithm weakly dependent on the placement of the phone on the person. This algorithm does assume, however, that the accelerometer data are sampled at a fixed $50Hz$ rate. Since the accelerometer data provided by Android are

not sampled at a fixed sampling rate, we first resample it at $50Hz$.[2]

Mladenov and Mock's algorithm for step detection consists of a pre-processing step to filter noise out from the data using a 20th order low-pass Butterworth filter. A dynamic peak mean is calculated by accumulating and averaging the magnitudes of points with negative backward slope and positive forward slope. Finally, peaks are classified as steps if they lie above the calculated dynamic mean. Figure 5 outlines the pipeline of the algorithm in the form of a class diagram. More information about the algorithm can be found in [11] and more details about its implementation can be found in [3].

Speed estimator design

Motivation

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 believed to be of significant importance [7]. Therefore, speed estimation should be implemented to be used as part of the performance model and a point of intervention. As with cadence estimation, the ideal speed estimation algorithm would meet the following requirements: i) the algorithm must only use data from sensors that are in-built on the smartphone, such as the accelerometer, as it is not ideal for the runner to don a number of wearable devices to estimate various parameters; and ii) the algorithm must allow the runner to carry the phone in a convenient location such as the pants or jacket pocket as it would be an inconvenience to the runner to run with awkwardly positioned belts that would not allow him easy access to his phone or would cause him discomfort while running.

While there are a number of smartphone apps that claim to report running speed, a large majority of them lack either openness, accuracy or reliability. A number of them 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 can therefore be quite unreliable [20]. For usability and testing purposes, however, we chose to include a GPS-based speed estimate in our design. Other apps 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 can also be 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 reviewed multiple alternative algorithms in the literature (in addition to the GPS-based speed estimation). We considered both kinematic models and statistical models. Our choice was to implement the algorithm described by Park et al. in [12] as it seemed reliable and reported low error rates of less than 12 - 15%. 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 stated desiderata.

[2]See http://developer.android.com/reference/android/hardware/SensorManager.html

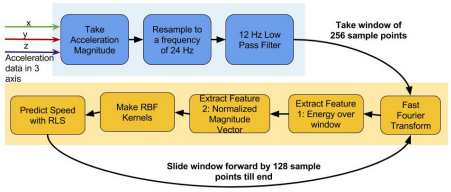


Figure 6: The speed estimation pipeline.

Algorithm

The chosen algorithm uses a statistical model to estimate speed. Since these raw accelerometer data are not provided at a fixed sampling rate, the data are re-sampled to get a sampling rate of $24Hz$. The data are then parsed using a sliding window technique, where each window consists of 256 data points with a 50% overlap between windows. The algorithm assumes that the running speed remains constant during each 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 [12]. 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 kernel functions. The features extracted from the accelerometer data are i) the spectral energy $X_E = \sum_{i=1}^{N} \mid M(f_i) \mid^2$, where $\mid M(f_i) \mid$ is the absolute value of the frequency components at frequency f_i; and ii) the frequency component of the magnitudes of the acceleration signal $X_M = [\mid M(f_1) \mid, \cdots, \mid M(f_N) \mid]$.

$$K_s(X^{(i)}, X^{(j)}) = exp\left(-\frac{\mid\mid X_M^{(i)} - X_M^{(j)} \mid\mid}{2\sigma_M^2}\right) + exp\left(-\frac{\mid\mid X_E^{(i)} - X_E^{(j)} \mid\mid}{2\sigma_E^2}\right) \quad (2)$$

$$RLS : \min_{f \in \mathscr{H}} \sum_{i=1}^{n} \left(f(X^{(i)}) - Y^{(i)}\right)^2 + \lambda \mid\mid f \mid\mid_k^2 \quad (3)$$

These $N+1$ features are combined using the custom kernel function described in Equation (2), which is sum of two radial basis function (RBF) kernels. σ_M and σ_E are half the median pairwise distances between values of X_M and X_E, respectively. This kernel is used to train and test the RLS model. The formulation of the RLS model for our problem is described in Equation (3), where $\mathscr{H}$ is the set of all functions $f(\cdot) = \sum_{i=1}^{n} c_i k_s(X_i, \cdot)$ for some $c \in R^n$.

The constant λ is a regularization coefficient, which is set to a value of 0.01 in our implementation to prevent over-fitting of the model. $Y^{(i)}$ is the labeled speed value and $X^{(i)}$ is the feature vector of the ith datapoint in a window such that $X^{(i)} =$

$\left[X_M^{(i)}, X_E^{(i)}\right]^T$. By solving a system of linear equations we find that the solution to Equation (3) is $c = (K + \lambda I)^{-1} Y$, where K is the kernel matrix $K_{i,j} = \left[k(X^{(i)}, X^{(j)})\right]$ 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 $f(X^*) = \sum_{i=1}^{n} c_i k(X^{(i)}, X^*)$ (for $X^{(i)}$ datapoints in the training set). The algorithm pipeline for the running speed prediction prototype is depicted in Figure 6. More details about the algorithm and its implementation can be found in [12, 3].

Validation

Although GPS-based speed estimation may be inaccurate, it is the only other estimate of speed available during the study. Until we fully validate the accelerometer-based speed estimator, the GPS-based speed estimator will be used for the runner feedback; but both estimates of speed will be collected. In this setting, we will be able to further assess the accelerometer-based speed estimation algorithm by comparing its output to the estimates made by the GPS-based speed estimator.

A note on data collection

In order to train and test the described estimation algorithms, labeled datasets are needed (where the data are accelerometer data and the labels are the corresponding cadence or speed). During the literature review, we found little details about the methods used for the collection of labeled accelerometer data, particularly for speed estimation purposes. There was also little information about the settings under which such data collection was carried out.

The collection of ground truth labels for speed is not a trivial endeavor. One could think about multiple settings to achieve this. A few examples include running on a treadmill (where the speed is preset), calculating the time needed to cover a set distance (assuming the speed is constant) or utilizing computer vision techniques to estimate speed from (possibly depth) video. The treadmill setting is promising except for the fact that the data generated by it would most likely only be valid for estimation of speed while running on a treadmill. That is, the data are likely not useful if the purpose of the algorithm is to estimate running speeds outdoors like our setting. The computer vision setting is complex and requires an additional validation step to make sure that the computer vision algorithm estimating speed is credible, which requires knowing the ground truth speeds, thus creating a circular situation. We believe it is important for the literature to report on the methodology of the data collection used in non-trivial settings like this, which we believe was lacking in this particular case.

In our setting, we built and utilized two separate helper Android apps. Namely, the first was designed to collect raw accelerometer data that served as inputs in the development of the two estimation algorithms (cadence and speed). The second app was designed as a clicker app (a screen with a button) to log the timestamps of every click. During the collection of training data, the person collecting the data carried two smartphones. The first phone was mounted around the hip, running the first app to collect raw accelerometer data. The second phone was carried by the collector in her or his hand, running the clicker app.

For the cadence estimator, the person collecting the data triggered a click every set number of steps (usually 1 for walking and 5 for running). Counting the clicks served as a ground truth of the number of steps taken (in addition to when the steps were taken, through the logged timestamps)

As for the speed estimator, the data collection procedure was conducted on a running track as follows. We set visual markers on the floor at fixed and known distances and asked the person collecting the data to click every time she or he passes a marker while running. The clicker was then used to calculate the average speed of the person between markers. Note that this method assumes that the speed is constant between markers, which is a good assumption whenever the markers are close to each other. On the other hand, the closer we bring the markers to each other, the higher the inaccuracy due to the delays in clicks compared to the event of passing the markers (because of the natural reaction time of the person collecting the data). To balance this tradeoff, we decided to use 30 feet as the fixed distance between markers based on the ranges of speeds in our application. Note that we collected data at different walking and running speeds and rates, from 5 subjects, for the cadence and speed estimation algorithms development.

SERVER DESIGN

The server is the other major node of the RunningCoach system and is responsible for receiving data from the Android smartphone client. It consists of two subcomponents: i) a Java-based *backend* responsible for communicating with the client node; and ii) a web-based *dashboard* for visualization of the collected data. Both components are running on a Red Hat Enterprise Linux version 6.8 server.

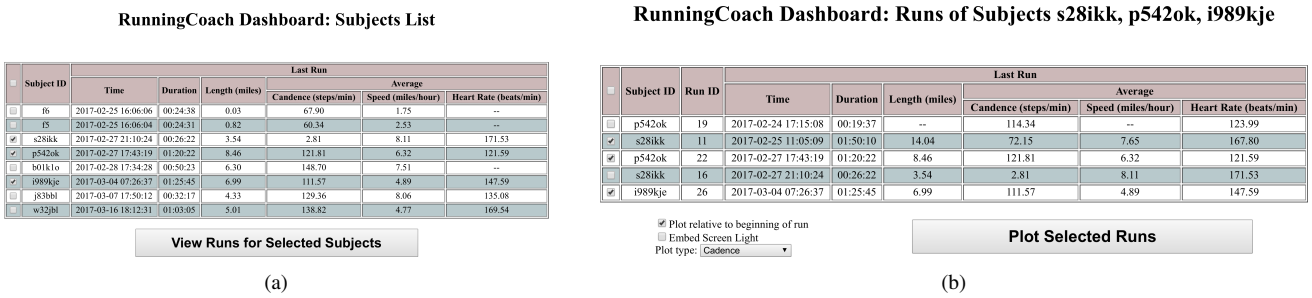
Backend

The RunningCoach server backend is written in Java and uses the Berkeley Telemonitoring Framework. To communicate with the client nodes, the backend uses the Tele-Interfacing (TI) protocol with Transport Layer Security (TLS), as described in [2]. The backend receives the data from the client nodes in the form of data jobs, unpacks the jobs back into encapsulators using job handlers (conforming to the Berkeley Telemonitoring Framework) and stores the data on a MySQL database. Each data job is attached to a subject identifier that is uniquely set to each runner in the app, which is used to identify the source of the data. The subject identifier is stored along with the data in the MySQL database.

The dashboard

In addition to the backend, there is a need for an easy-to-use solution to sieve out conveyable insights from our data (for the researchers). As such, we designed a web-based interface to visualize the subject data that we have collected from the RunningCoach app. The dashboard is designed as follows. The first screen, depicted in Figure 7a, lists out the subjects and summary statistics for their last respective recorded runs. From that screen, one can choose a subset of the subjects, by selecting their corresponding checkboxes, to further investigate their data in more detail.

Subsequently, clicking the **View Runs for Selected Subjects** button from the first screen loads the second screen of the



(a) (b)

Figure 7: (a) The first dashboard screen, displaying summary statistics of each runner's last recorded run; and (b) the subsequent screen listing all the runs of the previously selected subjects from (a).

dashboard. This screen, depicted in Figure 7b, lists all of the recorded runs of the selected subset of subjects from the first screen. The list includes, for each run, i) the subject ID; ii) time of the run; and iii) various summarizing parameters of the run (such as average hear-beat rate, cadence and energy expenditure). The screen allows the researcher to select a subset of these runs to investigate in further detail. There are six different run parameters that can be visualized, namely i) GPS; ii) heart rate; iii) energy expenditure, iv) cadence; v) speed; and vi) distance. The dashboard allows plotting these parameters (except GPS) on either an absolute time-scale or a relative time-scale (relative to the beginning of the run).

Once the **Plot Selected Runs** button has been clicked, the dashboard loads a third screen that visualizes the selected parameter for the selected runs. The visualization is done by using the Javascript-based data visualization libraries D3.js[3], dc.js[4] and Leaflet.js.[5] In Figure 8a, we display an example of a cadence plot for three runs. The dashboard includes an overview plot that enables further interaction with the data even closer by zooming in to a portion of the run, as depicted in Figure 8b. Other examples of plots for the same runs are displayed in Figures 8c to 8f for distance covered, energy expenditure, heart rate and (GPS-based) speed, respectively.

For the GPS visualization, the dashboard will create a blue marker per selected run on the map. Once a marker is selected, the dashboard visualizes the run on the map with some statistics about it, as depicted in Figure 9a. The dashboard can display a static or animated running path to the viewer depending on the viewer's choice.

FEASIBILITY STUDY

Methods

In order to validate the efficacy of the designed feedback and intervention, we are conducting a series of studies that will gradually guide the design of the next component in the system. The ultimate goal is to devise a recommendation algorithm that suggests a target cadence level for each runner based on her or his physical parameters and suggests a personalized training regimen based on the same parameters (recommends an α for Equation (1)). The first of these studies is a feasibility study designed to test the acceptability and usability of the app by runners. The study was approved by the UC Berkeley Institutional Review Board.

The screening process for the prospect subjects includes two criteria. First, the subject has to own an Android smartphone running Android version 4.3 (Jelly Bean MR2) or newer. Second, the subject has to already be a long-distance runner, in order to minimize further risks by participation in the study. To qualify as a long-distance runner, we use the following definition, which is consistent with the International Association of Athletics Federations (IAAF). A long-distance runner is someone who every week i) runs for at least 5 kilometers (or 3 miles) in distance; or ii) runs at least one session that is 1 hour or longer in duration.

During the main part of the study, the subjects are asked to carry the phone on their person in a comfortable spot (e.g., on the arm or on the hip) during their routine runs with the RunningCoach app running. In addition, the subjects are asked to wear a provided heart rate chest strap. The app collects information about their estimated energy expenditure, cadence, speed (GPS-based and accelerometer-based), heart rate and total distance covered, as described earlier. In addition to these estimates, the app collects the following two variables: i) whether or not the screen light is on; and ii) the battery level. Finally, at the conclusion of each subject's participation in the study, we administer a post-study privacy and acceptability survey. Since this is an ongoing study, we will not report on the findings of the post-study survey in this paper since it has not been administered yet.

The study has three main objectives. The first objective is to collect feedback from runners about the usability, usefulness and perceived accuracy of the system. The second objective is to validate the two heart rate estimation algorithms (finger-video-based and face-video-based) and the accelerometer-based speed estimation algorithm. The last objective is to collect data in order to train a cadence recommendation algorithm for runners.

We have currently recruited 6 subjects in the study: 3 women and 3 men. At the time of writing this paper, the runners

[3]D3.js: https://d3js.org/
[4]dc.js: http://dc-js.github.io/dc.js/
[5]Leaflet.js: http://leafletjs.com/

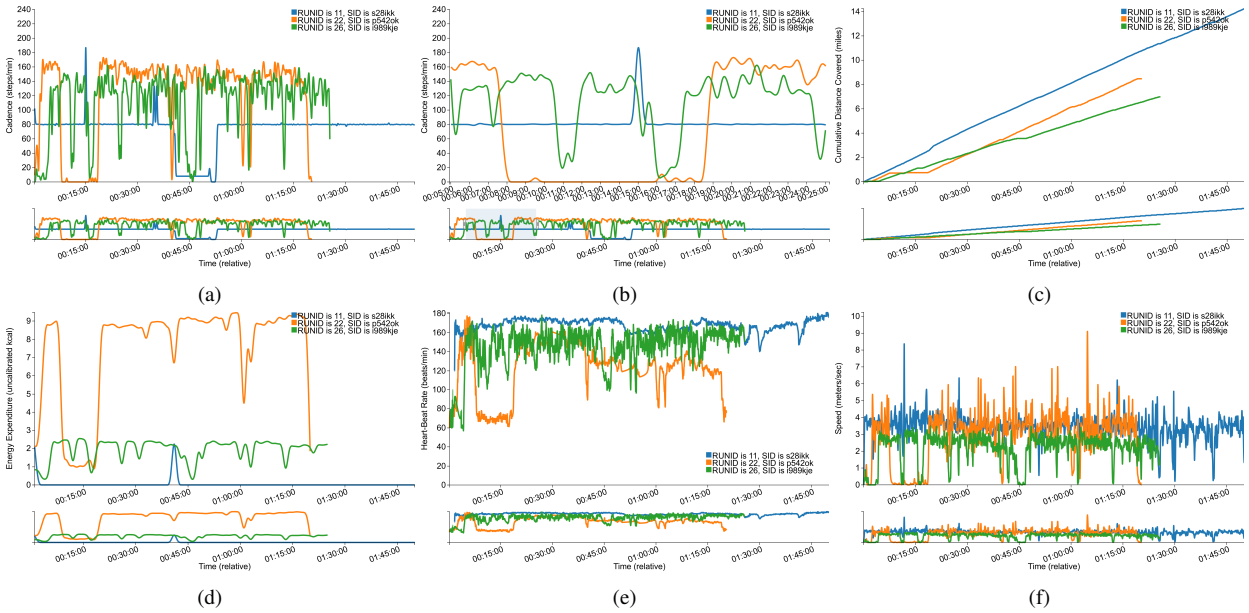


Figure 8: Examples of plots from the dashboard. (a) Cadence plot; (b) cadence plot with zoom in; (c) distance plot; (d) energy expenditure plot; (e) heart rate plot; and (f) GPS-based speed plot.

combined have used the system a total of 8 hours and 45 minutes in a period of 3 weeks.

Preliminary results – case studies

Since the study is ongoing, in this paper we report the primary results in a qualitative manner, based on case studies. After the conclusion of the study, quantitative methods and standardized protocols will be employed to assess accuracy and usability, and these results will be reported in future publications.

Regarding usability

Subject `b01k1o` reported that the heart rate chest strap caused her or him some skin irritation. The subject was instructed to stop using the strap immediately in order to limit the risk of harm. The strap itself was later tested and no fault in it was found.

During her or his first run, subject `p542ok` turned off GPS data collection in the app, disabling the collection of distance, speed and the run path. After that run, the subject turned the GPS data collection back on. No reason was provided by the subject; we suspect that the reason for disabling GPS data collection during that run is i) the phone wasn't charged in the beginning of the run (battery level was 35%): the runner did not want to risk her or his phone dying; ii) privacy: the runner wasn't comfortable sharing her or his location during the run; or iii) the phone's GPS radio was left turned off from before the run.

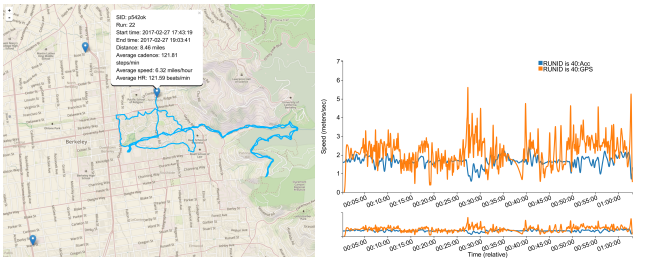
Using the data about the screen light, we found that subject `i989kje` seems to run with the RunningCoach app in foreground and the screen turned on throughout the majority of the run. We believe that the subject chose to do so in order to regularly check the app's interface showing her or his running parameters such as cadence, speed, distance, time and heart rate. All the other runners ran with the screen turned off, only using the haptic vibration as the sole means of feedback. It is worth noting that subjects `b01k1o` and `s28ikk` would sporadically turn the screen on during the run, presumably to check their running parameters.

Finally, subject `i989kje` provided a usability feedback in the app regarding the speed and cadence estimates, stating "I stopped a few times during the run and the app did not take it into account". The subject was referring to run 26 which is depicted in Figure 8. The data corroborates the feedback by the subject, which seems to have stopped on multiple occasions as can be seen from the figures, particularly Figure 8f. It is worth noting that the app provides an option to pause the current run, which the subject did not use in this instance.

In informal discussions with some subjects, they provided us with the feedback that they prefer to see their speed presented as in units of minutes/mile rather than miles/hour. Other than the cases stated above, the study is running smoothly in terms of engagement and data collection. In particular, there were no issues pertaining loss of data, app crashes or fault-intolerance, validating the fault-tolerance claims made in [2]. More feedback will be collected as the study moves forward, in particular during the post-study acceptability and usability survey that will be conducted at the conclusion of the study.

Regarding estimators' accuracy

Subject `s28ikk` found the cadence estimates to be inaccurate during run 11, which is depicted in Figures 8a and 8b. When asked to elaborate on why she or he thought the estimates were inaccurate (in the post-run survey), subject `s28ikk`'s answer was "Cadence too low; possibly because I held phone in hand". Other than this incident, which seems to have been caused by



(a) (b)

Figure 9: (a) an example GPS plot showing the path of run 22; and (b) a figure showing both GPS-based and Accelerometer-based estimates of speed for run 40.

the runner carrying the phone in her or his hand, the subjects voiced no complaints regarding the perceived accuracy of the cadence estimator.

On multiple occasions, subject `p542ok` found the heart rate data to be inaccurate, stating on one occasion (run 22, shown in Figure 8e): "I'm not at 70bpm immediately after a run." From the run data, we can tell that the video-based heart rate estimates did not yield any credible datapoints after that run, but the chest strap dropped and gave readings in the 70bpm range towards the end of the run. On another occasion, the subject used the post-run survey to report: "face and finger are way off as usual." The estimates in this case were seemingly off (lower than 55bpm), corroborating the feedback from the subject.

On one occasion, subject `p542ok` took video-based heart rate estimates without actually running. The subject did not report any perceived inaccuracy in the estimates, which were 61bpm and 62bpm for the finger-video-based and face-video-based estimates, respectively.

Early exploration of the collected data shows that the accelerometer-based speed estimator is not accurate enough for our purposes. Figure 9b depicts a plot comparing the GPS-based speed estimate to the Accelerometer-based speed estimate for a single run. As can be seen in the figure, although the Accelerometer-based speed estimator qualitatively matches the GPS-based speed estimator at times (e.g., minutes 13 to 17), it seems to be deviating from the GPS-based speed estimator (e.g., minutes 50 to 59). Further validation needs to happen including calculating the distribution of the estimation error (compared to the GPS-based estimates). However, we suspect that the main reasons for the lack of accuracy in the accelerometer-based speed estimator are i) more training data need to be collected with a finer resolution of speeds; and ii) the placement of the smartphone during the run in the study is less controlled than during the training data collection.

In Figures 8a and 8b, the cadence of the subject `p542ok` during run 22 seems to have dropped to almost 0 between minutes 8 and 18. At first, it may seem like an inaccurate series of estimates by the cadence estimator. Upon further investigation, however, we can see that the runner actually stopped running in that period, as corroborated by the flat distance covered plot during that period (Figure 8c), the drop in heart rate (Figure 8e) and the almost zero speed during the same period (Figure 8f). Moreover, except the reported perceived inaccuracy regarding heart rate towards the end of the run discussed above, the subject did not register any other complaints during that run, particularly to any estimates during the period in question.

DISCUSSION AND FUTURE WORK

Cadence is an important variable to optimize for long-distance runners because it improves overall performance and reduces risk of injury. In this work, we present an mHealth solution to long-distance running cadence-based coaching, called RunningCoach. We are conducting an ongoing usability and acceptability study with currently 6 participating subjects. Future versions of the system will include a music player that selects music with beats that are on the desired cadence for the day. The feedback from the subjects in this study will also be incorporated in the next version of the system (including the feedback in the post-study acceptability and usability survey).

The study, while ongoing, shows early signs of satisfaction by the subjects in terms of usability and perceived accuracy, except for the video-based heart rate estimates. There is however a need to develop systematic tools to automatically assess the accuracy of the estimates from some metadata. For instance, when a runner holds the phone in her or his hand, this needs to be detected and the produced cadence estimates, for example, need to be tagged to reflect that. More generally, we believe that each estimation algorithm must be engaged to an audit algorithm that verifies or quantifies the accuracy of each estimate produced by it.

We believe that it is important to describe the settings of data collection in scientific publications, particularly when the setting is non-trivial. Such description is vital for the readers to reproduce the research results as well as understanding the assumptions made during the data collection process. In addition, understanding the origin of the data collection helps the scientific community assess the validity of the presented results.

The data collected in this study will be used to train a recommendation algorithm that would suggest a cadence level for each runner based on her or his physical parameters. This will be done by leveraging each subject's self-reported fatigue level (as reported in the post-run survey) while running in different cadence levels. In addition to the fatigue level, we will investigate parametric models that use weight, height, age and gender to devise a cadence level that would closely mimic the choices made by the current subjects. In the next study, we will validate the efficacy of the recommendation algorithm with a larger cohort of subjects.

ACKNOWLEDGEMENTS

We would like to thank David M. Liebovitz, MD for suggesting long-distance running as an application to the Berkeley Telemonitoring Project and advising on the design of the app. Many thanks to Eugene Song and everyone involved in the Berkeley Telemonitoring Project at UC Berkeley for their hard work that made this work possible. This work was supported in part by the Center for Long-Term Cybersecurity (CLTC) at

UC Berkeley. The views expressed in this paper are those of the authors and do not necessarily represent the official views of the CLTC.

REFERENCES

1. Daniel Aranki, Gregorij Kurillo, Ruzena Bajcsy, Samee U. Khan, Ed., Albert Y. Zomaya, Ed., and Assad Abbas, Ed. 2017. *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Springer, Chapter Smartphone Based Real-Time Health Monitoring and Intervention. in print.
2. Daniel Aranki, Gregorij Kurillo, Adarsh Mani, Phillip Azar, Jochem van Gaalen, Quan Peng, Priyanka Nigam, Maya P. Reddy, Sneha Sankavaram, Qiyin Wu, and Ruzena Bajcsy. 2016. A Telemonitoring Framework for Android Devices. In *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. 282–291.
3. Carlos Asuncion, Uma Balakrishnan, Hannah Sarver, Lucas Serven, and Eugene Song. 2016. *A Telemonitoring Solution to Long-Distance Running Coaching*. Master's thesis. EECS Department, University of California, Berkeley. EECS-2016-107, EECS-2016-94, EECS-2016-85, EECS-2016-100 and EECS-2016-99.
4. Robert Jan Bood, Marijn Nijssen, John Van Der Kamp, and Melvyn Roerdink. 2013. The power of auditory-motor synchronization in sports: enhancing running performance by coupling cadence with the right beats. *PloS one* 8, 8 (2013), e70758.
5. Vikram Chandrasekaran, Ram Dantu, Srikanth Jonnada, Shanti Thiyagaraja, and Kalyan Pathapati Subbu. 2013. Cuffless differential blood pressure estimation using smart phones. *IEEE Transactions on Biomedical Engineering* 60, 4 (2013), 1080–1089.
6. Nour El Helou, Muriel Tafflet, Geoffroy Berthelot, Julien Tolaini, Andy Marc, Marion Guillaume, Christophe Hausswirth, and Jean-François Toussaint. 2012. Impact of environmental parameters on marathon running performance. *PloS one* 7, 5 (2012), e37407.
7. Shelly-lynn Florence and Joseph P Weir. 1997. Relationship of critical velocity to marathon running performance. *European journal of applied physiology and occupational physiology* 75, 3 (1997), 274–278.
8. Joseph Hamill, Timothy R Derrick, and Kenneth G Holt. 1995. Shock attenuation and stride frequency during running. *Human movement science* 14, 1 (1995), 45–60.
9. Bryan C Heiderscheit, Elizabeth S Chumanov, Max P Michalski, Christa M Wille, and Michael B Ryan. 2011. Effects of step rate manipulation on joint mechanics during running. *Medicine and science in sports and exercise* 43, 2 (2011), 296.
10. Linsey C Marr and Matthew R Ely. 2010. Effect of air pollution on marathon running performance. *Medicine and science in sports and exercise* 42, 3 (2010), 585–591.
11. Martin Mladenov and Michael Mock. 2009. A step counter service for Java-enabled devices using a built-in accelerometer. In *Proceedings of the 1st international workshop on context-aware middleware and services: affiliated with the 4th international conference on communication system software and middleware (COMSWARE 2009)*. ACM, 1–5.
12. Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. 2012. Online pose classification and walking speed estimation using handheld devices. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 113–122.
13. François Péronnet, Guy Thibault, Edward C Rhodes, and Donald C McKenzie. 1987. Correlation between ventilatory threshold and endurance capability in marathon runners. *Medicine and science in sports and exercise* 19, 6 (1987), 610–615.
14. Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. 2011. Advancements in noncontact, multiparameter physiological measurements using a webcam. *IEEE transactions on biomedical engineering* 58, 1 (2011), 7–11.
15. Ann V Rowlands, Roger G Eston, and Caroline Tilzey. 2001. Effect of stride length manipulation on symptoms of exercise-induced muscle damage and the repeated bout effect. *Journal of sports sciences* 19, 5 (2001), 333–340.
16. Bruno Tirotti Saragiotto, Tiê Parma Yamato, and Alexandre Dias Lopes. 2014. What do recreational runners think about risk factors for running injuries? A descriptive study of their beliefs and opinions. *journal of orthopaedic & sports physical therapy* 44, 10 (2014), 733–738.
17. Edith Van Dyck, Bart Moens, Jeska Buhmann, Michiel Demey, Esther Coorevits, Simone Dalla Bella, and Marc Leman. 2015. Spontaneous entrainment of running cadence to music tempo. *Sports medicine-open* 1, 1 (2015), 15.
18. Jochem van Gaalen and Quan Peng. 2015. *Expanded Tele-Health Platform for Android*. Master's thesis. EECS Department, University of California, Berkeley. EECS-2015-86 and EECS-2015-90.
19. Bobbie RN van Gent, Danny D Siem, Marienke van Middelkoop, Ton AG van Os, Sita SMA Bierma-Zeinstra, and Bart BW Koes. 2007. Incidence and determinants of lower extremity running injuries in long distance runners: a systematic review. *British journal of sports medicine* (2007).
20. Michael G Wing. 2011. Consumer-grade GPS receiver measurement accuracy in varying forest conditions. *Res J For* 5, 2 (2011), 78–88.