GymBuddy: Personalised Mobile Weight Training Instructor to Improve Workout Experience

Lorraine Choi, Daniil Tarakanov, Guang Yang and Aditya Sakhuja
Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ
Email: {fyc12, dt811, gy912, as10512}@imperial.ac.uk
Supervisor: Dr. Yiannis Demiris

***Abstract*—Awareness in health and fitness have been trending among the younger population, due to their desire in staying healthy and achieving an appealing body shape. One effective way to tone up the body is to perform weight training, while many beginners were unfamiliar with the correct exercising techniques or put off by the potential injury by using dumbbells. This lack of knowledge is often compensated by hiring personal trainers, but many consumers find personal trainers unaffordable or cause inflexibility in scheduling sessions with them. Therefore, *GymBuddy*, a mobile personalised smart fitness trainer, is developed with the aim to improve the exercise effectiveness and lower the chance of getting injured. *GymBuddy* uses a Myo armband that measures electromyography (EMG), arm position and acceleration, together with a mobile application, to provide real-time advice, exercise suggestions and the users' past exercise data.**

I. Introduction

Weight training is a kind of resistance training, which can increase muscle strength and muscle endurance, resulting in a gain in muscle size (muscle hypertrophy) [1]. When fitness experts are designing exercise routines, one common principle to use is F.I.T.T., which stands for frequency, intensity, time and type. All four factors would have to be tailored to individuals according to a person's age, sex, current fitness level and the available resources for training in order to maximise exercise effectiveness and avoid injuries [2] [3].

GymBuddy is a mobile application that acts as a personal gym trainer for weight training in arms, and is used in aid with a Myo armband, which contains 8 surface electromyographic (sEMG) sensors, a gyroscope, accelerometer and magnetometer. It is important for *GymBuddy* to be able to assess an individual's fitness ability to ensure the designed activities is at an optimal level, which means it is not too high that the person will injure, but not too low where the fitness goal will be under-achieved.

At the moment, *GymBuddy* focuses on training forearm and bicep muscles since Myo could only extract reliable sEMG data on the forearm. The app would provide personalised fitness advice to users, as well as real-time feedback on their performance and warnings if muscle fatigues too quickly or incorrect postures are detected.

Since *GymBuddy* is a system that has to be used in aid of Myo and a smartphone, the target users would be tech-savvy students or young professionals that are interested in weight training, and are looking for flexibility, personalisation and are willing to invest in and try out new gadgets.

II. Hypothesis

The aim of this project is to investigate the effect of having an auto-generated training programme through muscle fatigue detection and posture tracking on the user experience during fitness activities. The following hypotheses are proposed:

1) Novice subjects are more motivated to do weight training with a personalised smart fitness trainer.
2) Novice subjects using the *GymBuddy* wearable system will be more satisfied with their workout than those not using the system.

III. Related Work

Skulpt, founded by neurology professor at Harvard Medical School and a PhD in Electrical Engineering from MIT in 2009, is a system that provides fitness advice based on the user's physiological measurements. The system consists of a handheld device that measures body fat percentage and muscle quality by using a lab-based technology Electrical Impedance Myography (EIM) [4], and an app that provides personalised exercise and nutrition guidance. The device could measures up to 24 muscle groups, and allows user to see progress in specific areas. However, Skulpt does not provide any real-time tracking such as counting repetitions or give safety advices, which are very important to novice users.

The personal wellness coach [5] is a wearable device which can collect and analyse health data, then provide real-time feedback. It helps users to achieve customised training goals and monitor their heart rate to prevent harm. This virtual coach is able to differentiate between struggling and non-struggling states, and count repetitions in order to provide real-time encouragement. However, this system requires a computer for data processing, thus the mobility is rather limited. Additionally, the sensors used are bulky and obtrusive, limiting the users' uptake during the usability study.

Buttussi and Chittaro developed MOPET [6], which is a wearable system for fitness training. MOPET is implemented using a PocketPC, together with a heart rate monitor and 3D accelerometer. This context-aware and user-adaptive system provides the user with GPS navigation for jogging routes, visualises information about their speed and heart rate, provides motivational and safety advices, and suggests appropriate exercises. The user model consists of personal information, physiological information and the users' experience on an exercise. MOPET's user interface features a 3D embodied agent that speaks and provides real-time suggestions. This wearable system is mainly geared towards outdoor exercises.

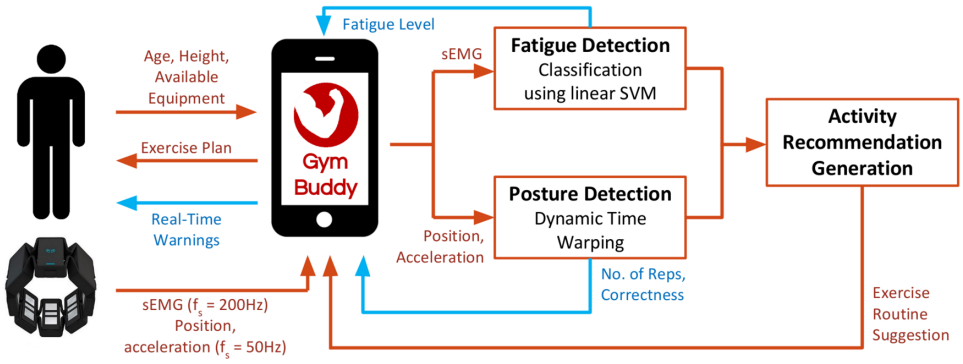

Fig. 1: High Level System Overview

IV. System Design

The high-level overview of the *GymBuddy* system is illustrated in Fig 1. The system consists of a Myo armband, which provides real-time EMG signal to track muscle status and position information for posture tracking, and a fitness app that provides real-time feedback and fitness advice to the user.

The app could be broken down into three major modules, which are (1) local muscle fatigue detection to avoid under- or over-training, (2) posture tracking to monitor the user's posture correctness and performance, and (3) recommendation generation which designs a personalised exercise routine according to the user's ability. When the *GymBuddy* application is first launched, a $ViewController$ is presented, which prompts the user to connect the app to Myo. The connection is performed via Bluetooth by manually selecting Myo from the list, although on every subsequent launch the Myo will be connected automatically, as the app will store the armband's device ID in CoreData. Once Myo is connected to the device, the user will be asked to calibrate Myo by performing a pre-defined pose. Once the Myo is calibrated, the $ProfileViewController$ is presented, which prompts the user to enter their height, age and gender. This data is stored in CoreData, and is automatically populated on subsequent launches of the app. Once the user has entered or confirmed their personal information, a suggestion with type of exercise, dumbbell weight, number of reps will be displayed to the user, together with a graphical tutorial of the exercise. Once the user begins an exercise, $StatusViewController$ is presented, which indicates the number of repetitions left, time passed, current muscular fatigue status, speed and posture correctness. On the $StatusViewController$'s initialisation, $FatigueController$ and $CorrectnessController$ are initialised to update the performance indicators. Once the user ends a set, $StatusViewController$'s state changes to $resting$, the indicators are hidden and suggestions for the number of repetitions and weights to be used during the next set. Additionally, the timer changes to a suggested resting time, and begins counting down. At the end of the last set, the user is again presented with an $InstructionViewController$, which provides suggestions for the next exercise.

A. Muscle Fatigue Detection

The objective of the *Muscle Fatigue Detection and Prediction* module is to track the health of muscles during the weight training, and also records how quickly a user's muscle would fatigue in order to provide the appropriate exercise prescription.

1) Background: Researchers have been evaluating ways to determine local muscle fatigue by analysing sEMG. Using sEMG for muscle fatigue analysis is particularly useful as it is non-invasive, and also allows real-time tracking. The 8 medical grade EMG sensors on the Myo armband will allow *GymBuddy* to extract the raw EMG signal from the user's forearm by using the SDK released by Thalmic Labs [7], and therefore skeletal muscle fatigue analysis would be a good choice for this system. Many researches have conducted time-domain and frequency-domain signal processing techniques to analyse skeletal muscle fatigue.

a) Time Domain Analysis: Sörnmo and Laguna have stated that "The amplitude of the surface EMG is a fundamental quantity which increases monotonically with the force developed in the muscle" [8]. In order to evaluate the sEMG amplitude, the mean absolute value (MAV) and root mean square (RMS) could be calculated to observe the changes in signal amplitude and signal power respectively. An increase in MAV and RMS should be observed as muscles fatigue. The two metrics are defined by the following equations:

$$MAV = \frac{\sum_n |A_n|}{N} \qquad RMS = \sqrt{\frac{\sum_n {A_n}^2}{N}}$$

where A_n is the n^{th} EMG amplitude measurement of N observations.

Cifrek et al. have suggested that the sEMG amplitude is seldom being used to determine muscle fatigue alone [9]. Usually the amplitude is used with other spectral analysis to indicate local muscle fatigue.

b) Frequency Domain Analysis: Although much information could be extracted from the amplitude characterisation of the sEMG in terms of MAV or RMS, another common approach is to conduct power spectral analysis [8]. During the muscle fatiguing process, the power spectrum of the sEMG compresses, which leads to a shift of the spectrum towards the lower frequencies [9] [10] [8]. Therefore, it would be difficult to observe and quantify such slowing behaviour of the signal using time domain analysis [8]. There are many spectral parameters that could be used to analyse the spectral shape, and among all variables, the most popular metrics being used are the mean frequency (MNF) and median frequency (MDF) of the power spectrum [10] [11] [8]. As the spectral distribution

shifts to the lower frequencies when muscle fatigues, the MNF and MDF of the spectrum will fall. The equations for MNF and MDF calculation are stated as below:

$$MNF = \frac{\int_0^{\pi} \omega S_x(e^{j\omega}) d\omega}{\int_0^{\pi} S_x(e^{j\omega}) d\omega} \qquad MDF = \frac{1}{2} \int_0^{\frac{\omega_s}{2}} S_x d\omega$$

where ω is frequency $(rads^{-1})$, ω_s is the sampling frequency, S_x is the power spectral density of the EMG signal.

Therefore, the rate of change of MAV, RMS, MNF and MDF could act as features to classify the level of fatigue.

2) Implementation: The input of the module would be the raw EMG signals from the 8 sensor pods in the Myo armband, and the fatigue level would be passed to the *Real-time Feedback* module for warnings if it exceeds a certain threshold, and the rate of muscle fatiguing would be passed to the *Workout Routine Generation* module for workout routine generation.

At a high-level view, the module would extract raw EMG of the user's forearm from Myo, then extract features from the signal. The features would be sent to a classification model to determine the muscle fatigue level using a trained model. Lastly, the fatigue level and rate of fatiguing will be passed to the *Real-Time Feedback* and *Workout Routine Generation* modules respectively.

Based on the reviewed literature, 11 features are being selected, which are: mean absolute value (MAV), root mean square (RMS), mean frequency (MNF), median frequency (MDF), Age, Sex, Height, Dumbbell Weight, Dominant or Non-Dominant Arm, Fitness Activity and User ID. Borg Scale of Perceived Exertion, which is a measure of perceived exertion during physical activity on a scale of 6-20, is being used as the response class (fatigue level) [11]. The levels are (1) Very light $(score = [6, 9])$, (2) Light $(score = [10, 12])$, (3) Somewhat hard $(score = [13, 14])$, (4) Hard $(score = [15, 16])$, (5) Very hard $(score = [17, 20])$.

EMG data was collected from 3 test subjects aged between 21 to 22 on both arms using the Myo armband to choose a classification algorithm and train the classifying model. The test subjects were first asked to keep his arm relaxed and rest on a surface for 30 seconds. Next, the test subjects were instructed to perform forearm curls with 1kg and 3kg dumbbells in 1 minute intervals respectively. sEMG data was recorded throughout each session, and test subject is asked to rate how hard they feel they have been working using the Borg Scale.

To find the most suitable classification algorithm, the classification learner in MATLAB is being utilised. Firstly, MAV, RMS, MNF and MDF are being calculated based on the collected data. The MAV and RMS of signal amplitude, and MNF and MDF of power spectrum (computed by the in-built Fast Fourier transform $fft()$ in MATLAB) were extracted every 0.5 second. Every 30 seconds, a regression line is being fitted into each computed set of data points to observe the change of metrics over time. The extracted features are fed into the classification learner, and a 95% accuracy is reached by using Linear Support Vector Machines (SVM) with a 5-fold cross-validation. Therefore, the linear SVM algorithm is chosen for the classification of muscle fatigue level.

In order to implement the classification in an iOS environment, two sub-modules were built in the fatigue controller: feature extraction and classification. In feature extraction, four functions are written to calculate the MAV, RMS, MNF and MDF. As frequency analysis is required, it is essential to perform fourier transform in Swift. The $vDSP_fft_zipD$ function in the $Accelerate$ library is utilised to perform fft on the extracted sEMG signals for every 128 samples (around 0.6 seconds). The reason of using 128 samples instead of 0.5 seconds is because the $vDSP_fft_zipD$ function would only perform accurate transformation when the length of input vector is of power of 2. This could potentially be caused by how they perform zero-padding.

For the implementation of linear SVM in Swift, an open source library *swift-libsvm* that is built based on the public domain LIBSVM repository is imported to the *GymBuddy* app [12]. Ideally, the model for classification should be trained based on data from similar users initially, then based on the user's own data since it is more personalised. But due to the time constraints and the scale of this project, a training data set collected from 3 test subjects are being hard-coded in the system, and will train a model during the initialisation of the app. The classifying function will be called every 30 seconds. Since the fatigue level of the 8 electrodes are classified separately, the average value of the 8 fatigue levels will be passed to the *Activity Recommendation* module.

B. Posture Tracking

1) Background: The correctness tracking module is used to count exercise repetitions, and indicate speed and posture correctness. These three metrics are updated in real-time and displayed to the user, in order to provide feedback on how well they are performing, and how many repetitions of an exercise they have left to do. These allows novice users to correct their technique, akin to the traditional personal trainers.

Exercise repetition counting and posture correctness tracking have been a subject of research by M. Kranz et al. during their development of the GymSkill Android application. [13] GymSkill provides automated assessment for balance board training, by calibrating the device, recording orientation data and comparing it against the model data. Calibration was performed by placing the smartphone on the balance board, and loading the board's calibration settings via NFC. These calibration settings include values of pitch and roll corresponding to events where board touches the ground. The orientation data was recorded in a form of Euler Angles, and was then normalised to a common value range with zero-mean. This data was mapped to $[-1, +1]$, corresponding to the maximum displacement angles, as per calibration settings. The resultant orientation data was analysed locally to describe interesting portions of user movement, and globally to describe the overall movement quality. Local analysis was performed using Principal Component Breakdown Analysis (PCBA) algorithm, which reduces the data dimensionality using eigenvectors and measures the Principal Component Analysis (PCA) model variance, thereby highlighting time periods with discontinuous movement. Global analysis was performed by estimating the dominant axes in order to derive the Empirical Distribution Function (EDF). The actual EDF is compared to the model EDF using standard Kullback-Leibler divergence (KLd), which

is used to calculate the performance score. Repetition counting was performed by counting the number of zero crossings in orientation data, where two zero crossings corresponded to one repetition.

2) Implementation: The correctness tracking module consists of three Swift controllers: $CalibrationViewController$, $StatusViewController$ and $CorrectnessController$. $CalibrationViewController$ samples the Myo's orientation values at a rate of 50 Hz. These orientation values are stored in a quaternion representation, which is a four-dimensional rotation vector that can be represented as $H = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, where x, y and z represent the axis of rotation, while w represents the angle rotated through. [14] Quaternions become useful when representing an object's physical rotation is space, as they allow for automatic interpolation between positions, regardless of how the key frames were originally defined, as opposed to the Euler angles, which suffer from the Gimbal lock, where one degree of rotational freedom is lost. When the "Calibrate" button is pressed, the $CalibrationViewController$ stores the conjugate of latest orientation quaternion as a global frame of reference. A quaternion conjugate is defined as $\bar{H} = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}$.

$StatusViewController$, which is used for displaying real-time updates while the user performs an exercise, samples orientation values at 50 Hz and multiplies the current orientation quaternion by the global frame of reference in order to obtain an orientation in the frame of reference defined during the calibration. [15] Quaternion multiplication is non-commutative, thus the order of multiplication is important. The scalar value w in of the resultant quaternion is then stored in the global orientation array. Such use of quaternions together with calibration is important in order to obtain consistent values between exercises, which allows us to compare the actual orientations with the model values. w can take floating-point values in $[-1, 1]$, where 1 indicates that the arm is in calibration position.

$CorrectnessController$ is active whenever the user performs an exercise, and is responsible for repetition counting, and calculation of speed and posture correctness scores, using the orientation values stored in the global orientation array by the $StatusViewController$. The three resultant metrics are then stored as corresponding parameters of the $Status$ data model.

a) Ground Truth: The ground truth orientation values were recorded using the *GymBuddy* application by one of team members, following illustrations found on various fitness websites. An ideal repetition for each exercise has been extracted using the rep splitting algorithm, every orientation value within the resultant array was multiplied by 100 and converted into an integer, in order to facilitate the posture correctness calculation. The ideal exercise speed in repetitions per minute (rpm) was measured using a stopwatch. The model data was stored within the ideal data model.

Repetition counting is performed every 0.5 seconds by counting the number of crossings of a pre-determined orientation scalar threshold in a negative direction, which represents the movement of an arm away from the initial position. This threshold is currently defined as 0.8 in order capture repetitions of both the bicep curl and forearm extension exercises, accounting for the events where the user fails to lower their arm completely. The resultant rep count is then appended to the corresponding Status data model parameter, as well as to the local $reps$ array, whose length is capped at 10 elements, in order to facilitate a sliding window speed correctness calculation.

b) Reps counting: Repetition counting is performed every 0.5 seconds by counting the number of crossings of a pre-determined orientation scalar threshold in a negative direction, which represents the movement of an arm away from the initial position. This threshold is currently defined as 0.8 in order capture repetitions of both the bicep curl and forearm extension exercises, accounting for the events where the user fails to lower their arm completely. The resultant rep count is then appended to the corresponding Status data model parameter, as well as to the local $reps$ array, whose length is capped at 10 elements, in order to facilitate a sliding window speed correctness calculation.

c) Speed correctness: Every 0.5 seconds all the values within the local $reps$ array are summed, which results in the number of repetitions over the last 5 seconds. This sum is then multiplied by 12 to give the repetition speed over the last 5 seconds in rpm. Finally, the ideal speed, which is specified in the $Ideal$ data model, is subtracted from the product in order to obtain the speed deviation in rpm. Thus, if the user performs exercise at an ideal speed, then the deviation will be 0 rpm. This deviation is then stored in the corresponding $Status$ data model parameter.

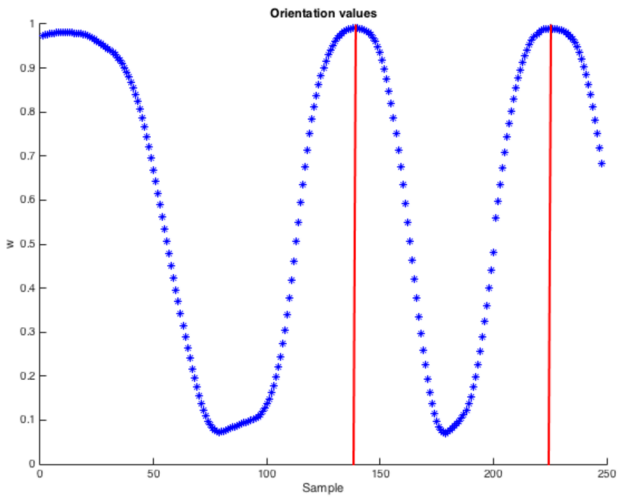

Fig. 2: Repetition splitting

d) Posture correctness: Posture correctness is calculated every 5 seconds by extracting full repetitions from the previous 5 seconds of orientation data, applying Dynamic Time Warping (DTW) to calculate the path cost between the ideal repetition and the last actual repetition, and finally using the path cost to compute the posture correctness score as an integer in $[0, 100]$. This score is then stored in the corresponding parameter of the $Status$ data model. Currently, the score is calculated every 5 seconds in order to ensure that there exists at least one full repetition in that time period, and to allow time for the computation-intensive DTW algorithm to complete.

The repetition splitting algorithm works by finding turning

points in an array of orientation values, indicating their topology (peak or trough), and then slicing the orientation array according to the $[peak, trough, peak]$ pattern, as illustrated in Fig 2. The peaks are found by iterating through the array, checking if each element is higher than the element 10 positions prior and 10 positions ahead by more than 0.01, and finding position of the maximum value in the resultant 21-element slice. The troughs are found in the similar fashion. Such sliding frame technique is necessary, as the orientation values are sampled at 50 Hz, thus false turning points may exist if the user's arm is slightly shaking.

Similarity between the model orientation time series, and the actual time series is measured using the DTW algorithm. Prior to the computation, the actual time series is multiplied by 100, and converted into an integer to facilitate DTW. DTW is an elastic similarity measure that is able to handle time warping, which is necessary, as users may exercise at a different pace from the model exercise. [16] We have implemented DTW in Swift, similar to Batra's Python implementation. [17] This implementation attempts to find the optimal warp path between the two temporal signals, such that the accumulated path cost is minimised. An example of such warping is illustrated in Fig 3, where the path cost was calculated to be 10982, which corresponds to the posture correctness score of 78. As we are most interested in finding the similarity score, the path cost is computed using Euclidean Distances between the warped pairs of points, where cost is a non-negative integer.

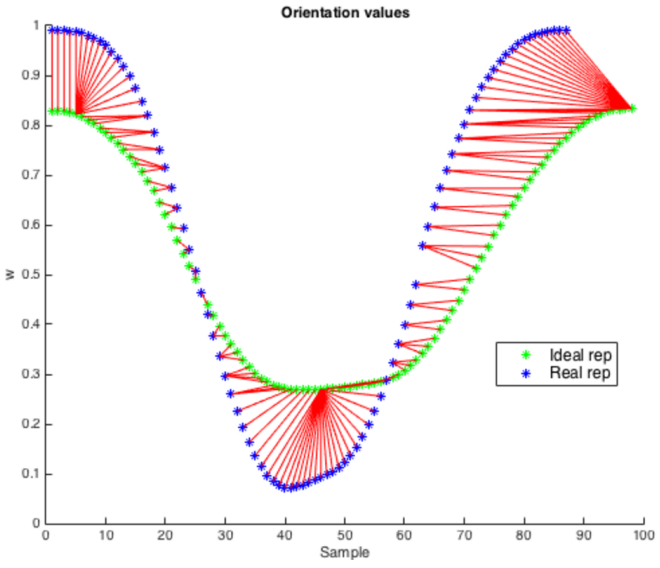

Fig. 3: Optimal path using DTW

The posture correctness score is computed as follows: $Score = max(100 - round(cost/500), 0)$.

C. Training Programme Recommendation

1) Background: A decision tree is implemented to give routine suggestions based on the fatigue level and history data. Decision trees are commonly used in machine learning to represent classifiers. [18]. A tree generally consists of a root, edges, nodes and leaves. The root does not have incoming edges and leaves have no outgoing edges. The internal nodes will split all instance space into two groups according to a defined threshold. Each leaf stores the probability of distribution of class labels. The complexity of a tree can be represented in several ways, 1) the total number of nodes, 2) total number of leaves, 3) depth of the tree. The complexity of a tree is closely related to its accuracy. The advantage of decision trees is that it is easy to read and debug. It can be converted into conditional loops and can also deal with different types of inputs.

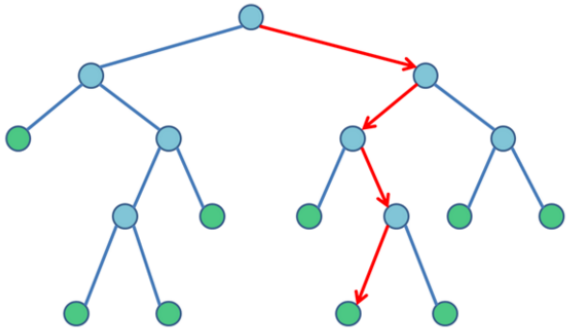

Fig. 4: Decision tree

2) Implementation: In our system, the suggestion contains three factors, 1) N_rep number of repetitions of one action, 2) m weight of dumbbells, 3) t_rest how long should user rest. These factors are tweaked according to the fatigue level which is updated when user has finished one session of exercise. Fatigue level is categorised into 5 levels. t_rest will increase gradually if the fatigue level is 4 or 5, and will be reduced when fatigue level is 1 or 2. Maximal and minimal rest duration are defined as t_max and t_min respectively. N_rep goes down when $t_rest >= t_max$, and up if $t_rest <= t_min$. The value m will slightly increase once the user has done enough exercises with the current weight. N_rep, m, t_rest are all updated every day based on the history data. Some threshold values in the decision tree, for example, the definition of long/short rest time and heavy/light weight, will be adjusted correspondingly if the performance and status of a user are improved or diminished. All the parameters are highly personalised because they are continuously learning from users and the past data.

D. User Interface

The user interface is an important factor for the success of an app. It identifies the purpose as well as the intention of the app and allows the user to derive the most out of it. 3 main goals were set for the user interface when designing *GymBuddy*: intuitive, engaging and useful. Keeping these in mind, every element of the app, from the ease of Myo armband synchronisation with the iPhone to the finely detailed view of key metrics during workout, reflects how the three above were achieved.

The design process began with the emphasis on determining what information will be presented to the user. To achieve this, all the information was segmented into meaningful chunks and separated ViewControllers to create separated information outlets. For instance, one ViewController was setup to guide the user through the process of connecting a synchronising the Myo Armband controller by displaying required instructions.

Next, the information presented should be intuitive for the users. This means that the user interface was re-assessed to

remove all distractions and create a simple user journey. Carrying forward the example of the synchronised ViewController, the instructions on how to perform the curls were visualised using animated GIFs to guide the user through the process.

Finally, the task of making the app engaging for the users requires an evaluation in how users interact with the app and how much interaction would be ideal. For this, the interaction needs of the app is altered based on the current state of the user. For instance, while the user is exercising, the app requires virtually no interaction and only acts as a continuous outflow of tracking metrics from the workout. However, before the workout begins, the user will be asked to calibrate the system as the placement of Myo armband could affect the posture tracking a lot.

Overall, the aim of *GymBuddy* is to create an encouraging experience to motivate users throughout their workout. While the internal capabilities of the app to cater to the user are crucial in creating that experience, it is the user interface that allows for virtually anyone to download the app and starting working out easily.

V. Evaluation

A. Methodology

In order to evaluate the functionality of *GymBuddy* and to test the proposed hypotheses in Section II, two methodologies were adopted: survey with potential users of *GymBuddy*, and evaluation with professional fitness trainers.

1) User Survey for Hypothesis Testing: Two questionnaires were devised to test the proposed hypotheses, which are: (1) Novice subjects are more motivated to do weight training with a personalised smart fitness trainer, and (2) Novice subjects using the *GymBuddy* wearable system will be more satisfied with their workout.

a) Hypothesis 1: Improved Motivation: This survey was designed to evaluate whether *GymBuddy* would motivate more users to undergo weight training. The target responders would be young people with little or no knowledge in weight training, or beginners in weight training in general. The questions in survey are:

1) Are you undergoing weight training regularly?
   Ans: (0) Yes, (1) No
2) If not, what is the main reason that hinders you from weight training?
   Ans: (0) Not interested, (1) No knowledge on it
3) Will you try to do weight training if there was a smartphone app that functions as a personal trainer and designs exercise routine for your needs?
   Ans: (0) Yes, (1) No

b) Hypothesis 2: Improved Satisfaction During Workout: The second questionnaire aimed at evaluating the second hypothesis on whether users will be more satisfied with their workout when they have used *GymBuddy* during exercising. Participants had to have no or minimal experience in weight training, and was asked to workout in two short sessions: first without the use of *GymBuddy*, then followed by a session with the aid of *GymBuddy*. Participants will be asked rate how satisfied they are with exercising experience on a scale of 1 to 5 after each session.

2) Evaluation with Professionals: Ethos is a state-of-the-art sports centre in Imperial College London based in South Kensington. A meeting with two professional fitness instructors from Ethos was arranged in order to allow the trainers to try using the *GymBuddy* system and comment on the fitness advice provided. Also, trainers are asked to perform bicep curls with dumbbells correctly and demonstrate how beginners perform bicep curls wrongly. The correctness scores of both correct and incorrect curls are recorded to evaluate the posture tracking functionality. Furthermore, the instructors were asked to rate how hard they think they are exercising using the Borg's scale, and the collected EMG data is sent to MATLAB and the iOS app to classify the fatigue level. The predicted results are will be compared to the actual fatigue level perceived by the instructors.

B. Results in User Survey for Hypothesis Testing

1) Hypothesis 1: Improved Motivation: 10 responses were collected for this survey, where all responders are students in Department of Electrical Engineering in Imperial College London and within the age of 20-23. The survey responses are details as below:

1) Are you undergoing weight training regularly?

	Yes	**No**
No. of Responses	9	1

2) If not, what is the main reason that hinders you from weight training?

	Not interested	**No knowledge**
No. of Responses	2	7

3) Will you try to do weight training if there was a smartphone app that functions as a personal trainer and designs exercise routine for your needs?

	Yes	**No**
No. of Responses	8	2

In order to evaluate whether *GymBuddy* motivates more novice users to undergo weight training, the responses in question 1 and 3 in survey is being re-grouped into Table I.

Undergo weight training?	**Yes**	**No**
Without GymBuddy	1	9
With GymBuddy	8	2

TABLE I: Willingness to undergo weight training with and without GymBuddy

The Pearson's Chi-squared test gives a p-value of 0.001654 with the collected responses. Using a 0.05 significance level, the null hypothesis is rejected as p-value 0.001654 is smaller than the 0.05 significance level. However, a chi-square test could potentially give an inaccurate inference with such a small sample size. Therefore, the Fisher's exact test, which is typically applied to small sample sizes, is employed as well and resulted in a p-value of 0.005477. Therefore, the null hypothesis is rejected as well with a significance level of 0.05 using the Fisher's exact test.

2) Hypothesis 2: Improved Satisfaction During Workout: 6 test subjects aged between 21-23 with no or little weight

training experience (<1 month) participated in the second test, which evaluates whether *GymBuddy* helps to improve satisfaction during workout. The results are listed in Table II.

	Average Score
Without GymBuddy	2.67
With GymBuddy	3.92

TABLE II: Satisfaction score with and without GymBuddy

Using a two-tail paired t-test, a p-value of 0.0476 is computed. Therefore, the null hypothesis is rejected with a significance level of 0.05.

C. Results for Evaluation with Professionals

1) Overall Qualitative Evaluation: During the conversation with the 2 professional fitness instructors from Ethos Sports Centre (one male and one female), the *GymBuddy* team has asked the instructors to try the system and comment on the fitness advice provided by *GymBuddy* and their user experience.

Both instructors agree that the suggested routines are appropriate for beginners, and mentioned the incremental resting time as the workout circuit goes on complements with the recommended routine. They have also agreed that the timer and speedometer could also help users to keep track of their performance.

The instructors have confirmed that it is good to monitor muscle fatigue while a person is working out as beginners sometimes would over-estimate their ability or think getting muscle soreness quickly in first few trainings are normal. The fatigue detection functionality should be able to reduce injury caused by the misjudgement of muscle use. They have also complemented the tracking of posture correctness as beginners would often be unsure about how a movement is being performed, which could greatly increase the chance of getting injured or reduce the exercise effectiveness.

The female instructor said she feels comfortable when she was exercising with the Myo armband on. The male trainer mentioned it feels alright to wear the Myo armband on his forearm, but it was slightly too tight for him as he has a relatively thick forearm. Therefore, it is likely that he would pull down the Myo armband during the resting time between the routines, which could cause problem to our calibration.

The gym trainers have also pointed out that it is important to look at the body fat percentage of the user instead of just height and weight. This is because both a fat person or a muscular person could weigh heavy. Therefore, knowing the body fat percentage could help the system to better estimate the fitness ability of the user. Also, they have suggested the system to include the user's fitness goal, such as to build up muscle size or to tone up. The prior would involve training in muscle strength, which is to perform less repetitions with heavier weight; and the latter requires doing more repetition with lighter weights. Therefore, knowing the fitness goal would affect how personal trainers design exercise routines as well.

2) Functionality Evaluation: The fitness instructors were asked to perform two sets of bicep curls: first set correctly and second incorrectly. The result is displayed as a box plot in Fig 5, which shows clearly that when the trainer is doing bicep curls correctly, the score is always above 90. While when incorrect curls are performed, the score would be a lot lower, depending on how incorrect the movement is.

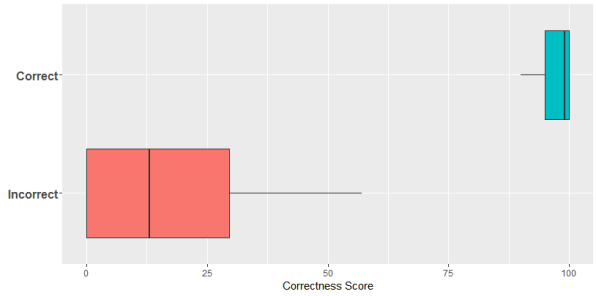

Fig. 5: Comparison of Correctness Score with Correct and Incorrect Bicep Curls

For the exercise speed correctness, the fitness instructors were asked to perform exercises at a pace that they would recommend, while allowed the novice users to exercise at their own pace. During the fitness instructors' exercise, the speed correctness indicator has always remained within the corresponding indicator's green area. During the novice users' exercise, the speed correctness indicator has varied over a wide range. These results are in line with our expectations, hence the speed correctness is deemed to be functioning correctly.

For fatigue level classification, the instructors were asked to rate how hard they think they have worked based on the Borg's scale, which acts as the actual fatigue level. The EMG data are processed in both MATLAB and the *GymBuddy* app, and the predicted fatigue levels are plotted against the actual fatigue level, as shown in Fig 6. The green line is the targeted predicted values, which means the prediction matches with the actual value. From the Fig 6, linear SVM in MATLAB gives a relatively high precision, while the swift-libsvm library in iOS tends to over-estimate the fatigue level. Although the same set of data was used to train the classification model in both MATLAB and swift-libsvm, since this is a multi-class classification, potentially the two models have assigned weightings to the binary classifiers differently and lead to different predicted values.

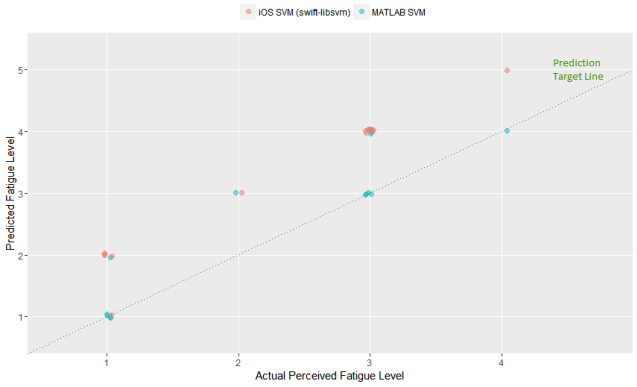

Fig. 6: Comparison of Predicted and Actual Fatigue Level

VI. Future Work

For the future development of *GymBuddy*, it would be beneficial to add an ability for the user to specify their weight and body fat percentage in order for *GymBuddy* to better

estimate the user's fitness ability and provide more suitable routine recommendations. Allowing the user to specify their fitness goal will allow *GymBuddy* to provide more personalised routine recommendations, as different goals require different reps/weights combination. For example, strength training may necessitate fewer reps, more sets and higher weights, while focusing on muscular gains may necessitate more reps, less sets and lower weights. [19]

At the moment, the app will only work with one Myo armband, and therefore the posture tracking and fatigue detection will only be available on one arm. As the non-dominant arm is usually weaker and therefore more likely to be fatigued or injured, therefore users are recommended to wear Myo armband on the non-dominant arm. As *GymBuddy* develops, it will be highly beneficial to allow the app to track data from two Myo at the same time, which would allow tracking on both arms.

In order to provide more accurate indicators throughout the exercise, it would be necessary to update the posture correctness indicator every 0.5 seconds, as opposed to the current refresh rate of 5 seconds. Currently the user may exercise for 5 seconds falsely thinking that their posture is correct or incorrect, whereas a higher rate of updates will provide them with real-time information on the correctness of their exercise.

In the current version of *GymBuddy*, some routine suggestions may be widely inaccurate, which is caused by the user stopping mid-set due to distraction. *GymBuddy* will perceive this as the user was struggling, and would end up suggesting an abnormally low number of reps in the next set. In order to avoid such cases, *GymBuddy* would have to discard the outlier set to provide reliable routine recommendations in the future.

Currently, *GymBuddy* can only provide indicators and suggestions for two upper-body weight exercises: bicep curl and forearm extension. The list of exercises could be extended to include more upper-body exercises, as well as for the other muscle groups. This will allow the user to enjoy a more balanced workout routine, and achieve their goals faster.

In the course of future development, the fatigue level classification needs to be further optimised in order to improve the detection accuracy. The current classification tends to overestimate the fatigue level, which could lower the exercise effectiveness since the user's capacity is under-estimated. Therefore, improving the fatigue detection accuracy will improve the quality of routine suggestions, as it heavily relies on the past fatigue variations.

Finally, there is a potential to further develop the *GymBuddy* brand in order to improve the novice users' workout satisfaction by employing other sensors to measure other important indicators, such as ECG, which will allow the user to understand their body better, and for *GymBuddy* to provide more accurate routine recommendations.

VII. Conclusion

In conclusion, a mobile system that provides advice and monitor performance for novice users in weight training is designed and developed. The *GymBuddy* system consists of a wearable Myo armband, which measures EMG signals, accelerometer and magnetometer data of the user's forearm, and an iOS application that interacts with the user. Based on the survey responses, *GymBuddy* have motivated more people with little workout experience to undergo weight training and enhances their workout experience. Fitness professionals have also evaluated the fitness advices provided by *GymBuddy* and verified the functionality of the system, and were satisfied with the level of precision in monitoring and suggestions provided by the system. The future development of *GymBuddy* should focus on improving the user's workout experience by providing more accurate indicators, and extending the variety of available exercises. This will allow *GymBuddy* to become a viable substitute to hiring a personal trainer for novice users.

References

[1] V. H. Heyward and A. Gibson, *Advanced fitness assessment and exercise prescription 7th edition*. Human kinetics, 2014.

[2] A. Barisic and S. T. Leatherdale, "Importance of frequency, intensity, time and type (fitt) in physical activity assessment for epidemiological research," *Canadian Journal of Public Health*, vol. 102, no. 3, p. 174, 2011.

[3] A. A. of Pediatrics, "The fitt plan for physical activity," 21 November 2015 2015.

[4] Skulpt, "Skulpt - measure body fat percentage and muscle quality." [Online]. Available: http://www.skulpt.me/optimizely/v2/

[5] R. Asselin, G. Ortiz, J. Pui, A. Smailagic, and C. Kissling, "Implementation and evaluation of the personal wellness coach," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. IEEE, 2005, pp. 529–535.

[6] F. Buttussi and L. Chittaro, "Mopet: A context-aware and user-adaptive wearable system for fitness training," *Artificial Intelligence in Medicine*, vol. 42, no. 2, pp. 153–163, 2008.

[7] P. Bernhardt, "myocraft: Logging imu and raw emg data."

[8] L. Srnmo and P. Laguna, *Bioelectrical signal processing in cardiac and neurological applications*. Academic Press, 2005.

[9] M. Cifrek, V. Medved, S. Tonkovi, and S. Ostoji, "Surface emg based muscle fatigue evaluation in biomechanics," *Clinical Biomechanics*, vol. 24, no. 4, pp. 327–340, 2009.

[10] S. R. Alty and A. Georgakis, "Mean frequency estimation of surface emg signals using filterbank methods," in *Signal Processing Conference, 2011 19th European*. IEEE, 2011, pp. 1387–1390.

[11] Y. Lee and Y. Chee, "Evaluation of the effectiveness of muscle assistive device using muscle fatigue analysis," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, July 2013, pp. 2112–2115.

[12] K. Coble, "Kevincoble/swift-libsvm," Jan 2016. [Online]. Available: https://github.com/kevincoble/swift-libsvm

[13] M. Kranz, A. Möller, N. Hammerla, S. Diewald, T. Plötz, P. Olivier, and L. Roalter, "The mobile fitness coach: Towards individualized skill assessment using personalized mobile devices," *Pervasive and Mobile Computing*, vol. 9, no. 2, pp. 203–215, 2013.

[14] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH computer graphics*, vol. 19, no. 3. ACM, 1985, pp. 245–254.

[15] P. Bernhardt. (2015) How i learned to stop worrying and love quaternions. [Online]. Available: http://developerblog.myo.com/quaternions/

[16] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[17] N. Batra. (2014) Programatically understanding dynamic time warping. [Online]. Available: http://nipunbatra.github.io/2014/07/dtw/

[18] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[19] M. Robertson. (2015) High reps, low reps? which rep scheme is best? [Online]. Available: http://www.bodybuilding.com/fun/high-reps-low-reps-which-rep-scheme-is-best.html