

Tracking Free-Weight Exercises

Keng-hao Chang¹, Mike Y. Chen², and John Canny¹

¹Berkeley Institute of Design, Computer Science Division
University of California, Berkeley, CA 94720 USA
{kenghao, jfc}@cs.berkeley.edu

²Ludic Labs, USA
mike@ludic-labs.com

Abstract. Weight training, in addition to aerobic exercises, is an important component of a balanced exercise program. However, mechanisms for tracking free weight exercises have not yet been explored. In this paper, we study methods that automatically recognize what type of exercise you are doing and how many repetitions you have done so far. We incorporated a three-axis accelerometer into a workout glove to track hand movements and put another accelerometer on a user's waist to track body posture. To recognize types of exercises, we tried two methods: a Naïve Bayes Classifier and Hidden Markov Models. To count repetitions developed and tested two algorithms: a peak counting algorithm and a method using the Viterbi algorithm with a Hidden Markov Model. Our experimental results showed overall recognition accuracy of around 90% over nine different exercises, and overall miscount rate of around 5%. We believe that the promising results will potentially contribute to the vision of a digital personal trainer, create a new experience for exercising, and enable physical and psychological well-being.

1 Introduction

Exercise is an important contributor to physical and psychological well-being. Regular exercise reduces many chronic diseases, such as heart/cardiovascular diseases, diabetes, hypertension, obesity, etc [1][2][3]. A recent Surgeon General report indicated that approximately 300,000 U.S. deaths are associated with obesity and overweight each year. Proper exercises and related interventions are effective in ameliorating symptoms and improving health [4].

To help people exercise effectively, several recent works focus on tracking and user feedback via exercise management systems. For example, in the category of aerobic exercises such as bicycling, swimming, and running, there are accelerometer and GPS-based pedometers to track running pace and distance, ECG monitors to track exertion [5], and electronic exercise machines such as treadmills, elliptical trainers, stair climbers and stationary bikes. Weight training, in addition to aerobic exercises, is an important component of a balanced exercise program [6]. However, mechanisms for tracking free weight exercises have not yet been explored. Weight training involves combinations of different types of exercises, varying weight amount to lift, number of repetitions and sets to be done, and so on. Managing a diverse training

sequence should be well supported on site. During the process of working out, people may forget their progress, skip steps, or miscount a sequence. Even though people may try to organize by keeping notes on their progress, this is tedious and easily turns the workout into a chore.

We are exploring several applications in weight exercise management. The first is an *exercise tracker*: a system which automatically keeps track of your progress of free weight exercises. You only have to focus on doing the exercises without worrying about remembering your progress. It can be *accessed anytime, anywhere*: this system is embedded in the mobile device that you normally bring with you when you walk into a gym, illustrated in Fig. 1. You can check it before, during, or after your training process. The second application is an *exercise planner*: you can review your exercise history from the mobile device and this system can help you design a proper exercise plan. Eventually we would like to build a *digital personal trainer*: it warns you if you exercise too hard or in incorrect form.



Fig. 1. Scenario of an exercise tracker

This paper focuses on exercise tracking: it explores methods that automatically recognize what type of exercise you are doing and how many repetitions you have done so far. In fact, recognizing exercise types belongs to the area of activity recognition, which has been extensively explored in the past few years. In this work, we applied some well-studied methods and found that these methods can also achieve good results in our application. We first incorporated a three-axis accelerometer into a workout glove to track hand movements, and put another accelerometer on a user's waist to track body posture. We investigated the effectiveness of two methods to recognize exercise type: Naïve Bayes Classifiers and Hidden Markov Models. In addition, since the number of repetitions is another important factor of weight training, merely recognizing types of exercises is insufficient for an exercise tracker. So, we developed and tested two methods to count repetitions: a peak counting algorithm and a method using the state sequence predicted by Hidden Markov Models.

Experimental results proved that the methods can be applied to a variety of exercises, users, and conditions. For the *exercise recognition goal*, it achieved 95% accuracy based on single user data, and the accuracy was around 85% when we cross-validated training results with new user data. For the *counting goal*, both proposed methods achieved around 5% miscount rate in general, which means if a user performs 100 repetitions, the system may miscount by less than 5 repetitions. The experimental results were based on exercise data we collected by asking ten subjects

to perform nine different exercises, with different weight settings. The total length of the data was 9740 seconds (162.5 minutes), with a total of 4925 repetitions.

The remainder of this paper is organized as follows. Section 1 describes the related work. Section 2 describes a taxonomy of exercises and the rationale for using accelerometers to approach free weight exercises. Section 3 presents the development and evaluation of the algorithms. Finally, Section 4 states our future work and conclusions.

2 Related Work

The related work is categorized into the following two categories: exercise-specific work and studies of activity recognition. In the first category, FitLinxx [7] used sensors to track the usage of weight machines and showed training progress on a built-in display. Nonetheless, their sensors and tracking methods cannot be directly applied to free weight exercises, since FitLinxx only tracks predefined exercise routines on specific machines, and cannot perform the initial identification of type. In addition, iPod + Nike [8] tracked jogging and used music feedbacks to promote a new exercise experience. In fact, both products promote the idea of a digital personal trainer to coach exercise. As for research projects, House_n [9] is building a system to recognize gym- and exercise-related activities with accelerometers. They also intend to better estimate calorie expenditure in real time, which is different from our goal of tracking and eventually creating a digital personal trainer for free weight exercises.

In the category of activity recognition, there have been a large number of studies using different sensors, including accelerometers, gyroscopes, microphones, barometers, RFID readers, and GPS units, to recognize a variety of activities, and those studies use a variety of different machine learning techniques. Rather than developing brand new techniques, we applied and extended existing work to test the applicability within this new domain. For example, Mithil [10] and House_n [11] have shown great success using accelerometers to recognize activities such as walking, running, bicycling, etc, so we adopted their methods in which a Naïve Bayes Classifier was used to classify features extracted from sliding windows. We further leverage the characteristics of free weight exercise. For example, we only used accelerometers, rather than using both accelerometers and gyroscopes as in Minnen's work [12]. The design choice was based on the observation that sensors attached and constrained to gloves should be rotated with less degree of freedom in compare to being rotated freely in the air with no constraint (i.e. fixed on hands) and using only accelerometers is sufficient. The gravity effect on accelerometers was therefore emphasized and extracted as an extra feature. In fact, both their and our approaches achieved comparable 80~90% recognition accuracy. Hidden Markov Models have been widely applied. For example, Ward [13] and Georgia Tech Gesture Toolkit [14] applied HMMs to acceleration data. Moreover, Benbasat [15] identified a forward/backward (or a combination of both) hand movement as an atom, and they regarded a gesture as a composition of gesture atoms. Similarly, we identified "atoms" in acceleration data to count repetitions with two different approaches.

More techniques were proposed to contribute to the area of activity recognition. Minnen [12] made an unsupervised learning technique to avoid labeling effort. They

first used information theory to identify best motifs, and then, they applied HMM to learn motif sets, which is exactly what we applied to our work. Wynatt [16] and Hamid [17] also proposed similar methods. In addition, Lester [18] applied the AdaBoost technique to let algorithms selecting best set of features, which is different from traditional approaches and what we did in this work to tune features manually. As for the settings in evaluation, we applied methods to recognize sets in isolation, rather than to recognize the entire exercise session. In future work, we would address this harder setup, as those by Lester[16], Ward[13], and Subramanya [19].

3 Approaching Free Weight Exercises with Accelerometers

This paper addresses two goals. The first *recognition goal* is to detect what type of exercise a user is doing. The second *counting goal* is to count how many repetitions she has been performing so far. To approach them, we provide a taxonomy of free weight exercises, discuss the accelerometer-based approach, and provide rationale and deployment details.

3.1 The Taxonomy of Free Weight Exercises

To fulfill the goals, we have identified the most common, representative free weight exercises in the gym environment in Table 1. Each exercise listed here is common and representative in a sense that people frequently use those exercises to train each individual muscle group in the human body [20]. Exercises are listed based on the muscle groups they are designed to train. For example, to train the arms, people often perform *bicep curls* for biceps and *tricep curls* for triceps. To train the upper body, users perform *bench press* and *flye* to work on their chest muscles. They also perform *bent-over row* to strengthen their upper backs, and use *lateral raise* to train shoulder muscles. Finally, in the lower body category, people use *deadlift* to train quadriceps and *standing calf raise* to train calves. Table 1 also lists the posture required to perform each exercise. The details of how to perform each free weight exercise can be found in reference [21], and some of the exercises will be explained in section 2.2 and section 2.3. We regard these exercises as the targets of our paper and we want to

Table 1. Representaive and commonly performed exercises for each muscle group

	Exercise	Muscle groups	Body part	Posture
1	Biceps curl	Biceps	Arms	Standing/Sitting
2	Tricep curl	Triceps		Standing/Sitting
3	Bench press	Chest	Upper Body	Lying
4	Flye			Lying
5	Bent-over row	Upper back		Standing
6	Lateral raise	Shoulders		Standing
7	Overhead dumbbell press			Standing/Sitting
8	Deadlift	Quadriceps		Lower Body
9	Standing calf raises	Calves	Standing	

figure out whether it is possible to track them well. Throughout this paper, these commonly performed exercises are used as examples to explain our work and as baselines to test our system, in order to prove that our system is applicable to real world gym environments.

3.2 The Accelerometer Glove and the Posture Clip

To track the various free weight exercises listed in Table 1, we use accelerometers and incorporated acceleration data with machine learning techniques. Since in free weight exercises people hold and move weights, shown in Fig. 3, we instrumented a three-axis accelerometer onto a workout glove on the right hand, shown in Fig. 2-(a). We say this setting is applicable because people usually wear workout gloves when they perform free weight exercises. We call it the *accelerometer glove* throughout this paper. The three-axis accelerometer used is an off-the-shelf product, called WiTilt v2.5 by Spark Fun Electronics [22]. WiTilt v2.5 employs a FreeScale MMA7260Q triple-axis accelerometer. The accelerometer samples acceleration at a frequency of 80 Hz and in the range of $\pm 1.5g$. In addition, WiTilt v2.5 has Bluetooth wireless connectivity.

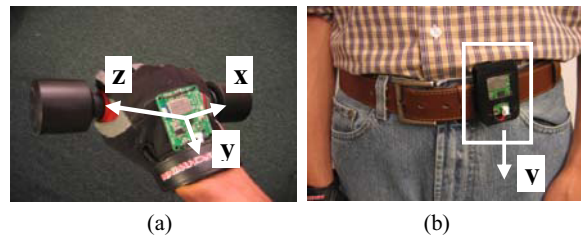


Fig. 2. (a) The *accelerometer glove* and the directions of three axes on the accelerometer. The *z*-axis is vertical to the palm. (b) The *posture clip*.

In addition to hand movements, it is also important to differentiate whether people are standing or lying on a bench. For example in Fig.3, the hand movement of the *overhead dumbbell press* is quite similar with the hand movements of *bench press*, in that people push free weights straight up to the air in both cases. But these two exercises are essentially different because the posture of lying down makes chest muscles to be trained in *bench press*, while standing/sitting helps training shoulder muscles in the *overhead dumbbell press*. Therefore, we use a complementary posture clip to detect postures during exercises, shown in Fig. 2-(b). The posture clip is also made of a WiTilt v2.5 three-axis accelerometer.

Considering the trade-off between accuracy, cost, and the nature of exercises themselves, we claim that it is sufficient to use only accelerometers to track free weight exercises. Although accelerometers are much less expensive and much smaller than Inertia Measurement Systems [23], three-axis accelerometers cannot track the six degrees of freedom that Inertia Measurement Systems can. However, because human motions are relatively restricted, the inherently lower variation of acceleration patterns sensed by accelerometers fixed on gloves should be sufficient for tracking

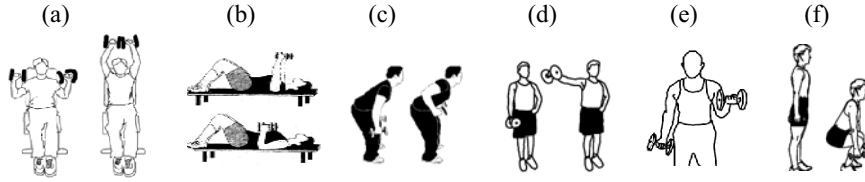


Fig. 3. Illustrations of (a) overhead dumbbell press, (b) bench press, (c) bent-over row, (d) lateral raise, (e) bicep curls, and (f) deadlift

free weight exercises well. In fact, results of this paper do justify that using accelerometers are sufficient to track free weight exercises. In addition, we have once considered putting accelerometers on free weights and use free weights themselves to track exercises. However, as we have just mentioned, the incapability of sensing six degrees of freedom keeps us from going in this direction.

The other important focus of tracking free weight exercise is to know how much weight a user is lifting. This can be enabled by simply attaching RFIDs on free weights, instrumenting a glove with an RFID reader, and keeping RFID-weight mappings. As the RFID-related work [24] is pretty mature, we are not focusing on this in our paper and consider it as future work when we are building a real system.

3.3 Acceleration Responses for Free Weight Exercises

In this subsection we discuss how different exercises result in different acceleration patterns and how repetitions of exercises respond in acceleration data, which serve as a basis to design algorithms described in Section 3.

Recognition Goal (how exercises differ from each other in acceleration data)

With the *accelerometer glove* and the *posture clip*, let's take a look at the acceleration data resulting from the nine exercises in Fig. 4. We only plot the data from the *accelerometer glove* to maintain legibility. Some of them are quite distinct from each other, which is due to both the characteristics of exercises and the characteristics of accelerometers themselves. For exercise characteristics, movements from different exercises impose acceleration in different axes, which we called the *major axis*. As for accelerometer characteristics, since different exercise postures make different axes vertical to the ground, gravity affects acceleration readings in the corresponding axes, which we called the *gravity effect*. For example, although both the *overhead dumbbell press* and the *bent-over row* cause acceleration in the same *y-axis*¹ (see references in Fig. 2 and Fig. 3), the average acceleration magnitudes are different. It's the gravity effect making the difference: in the *overhead dumbbell press*, hands are raised up such that the positive direction of *y-axis* faces down to the ground, so gravity makes the acceleration value larger. On the contrary, gravity² pulls the acceleration value of

¹ Since there are three axes in both the *acceleration glove* and the *posture clip*, for cases where we don't explicitly state where the axis is from, by default we mean the axis from the glove.

² When an axis directs to the ground and the accelerometer is stable, the accelerometer returns value around 800 (+1g). When the axis faces to the air, it returns value around 200 (-1g).

bent-over row and makes it smaller. The similar idea can also be applied to the comparison of *deadlift*, *standing calf raise*, and *bent-over row*. They are similar because they all impose acceleration in y -axis, and they all have the same gravity effect because all y -axes face up. However, *deadlift* and *standing calf raise* are different from *bent-over row* because the body movements in *deadlift* and *standing calf raise* make the *posture clip* move up and down in the y -axis on the clip whereas bodies stand still in *bent-over row*.

Take another example, the *bicep curl* shown in Fig. 3. It's an exercise in which users bend their arms to curl the weights toward their shoulders, and lower their arms to the starting position. The arm-bending movement not only induces acceleration in the z -axis, but gravity also affects the acceleration readings of the y and z axes alternately: while the arms are bent at the closest position to the shoulders, the y -axis is affected the most by the gravity. In contrast, the z -axis has the largest gravity effect while arms are in the horizontal position during the bending movement. Other exercises such as *tricep curls*, *lateral raise*, and *flye* all lie in the same family. They are similar because they all create acceleration change in more than one axis, but they are different in the combination of axes and in the directions facing ground, i.e. different gravity effect.

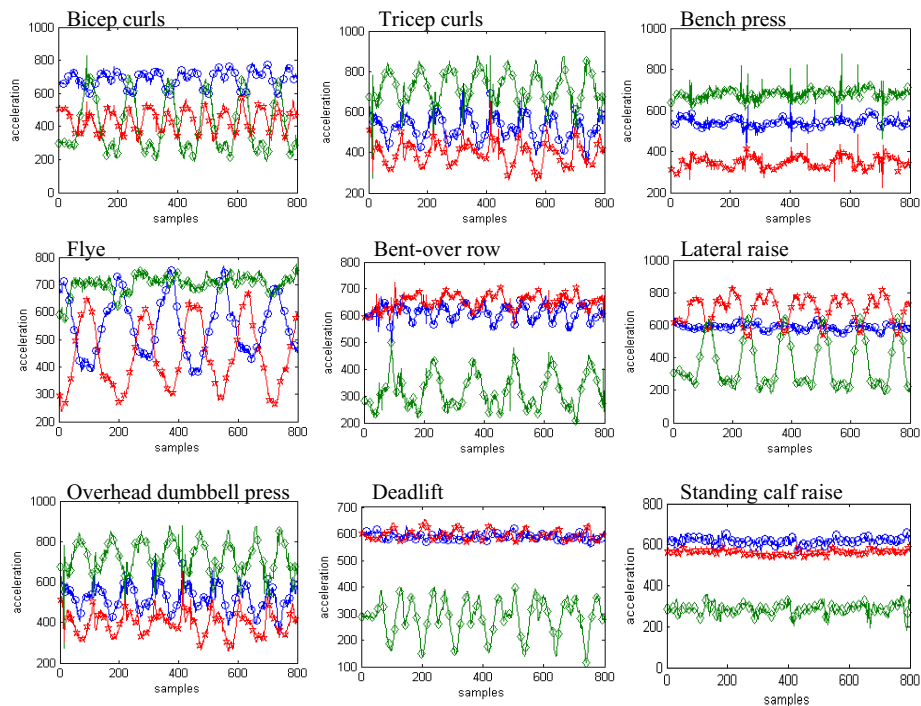


Fig. 4. Acceleration data from the *acceleration glove* in each of the nine exercises. X-axis is shown with blue circles. Y-axis is shown with green diamonds. Z-axis is shown with red stars.

Counting Goal (how repetitions of exercises show up in acceleration data)

Here we take the *overhead dumbbell press* as an example to explain how repetitions of free weight exercise show up in acceleration data. In general, people start this exercise by keeping the dumbbells to the sides of the shoulders (*starting position*). Then, they smoothly lift the dumbbells overhead until the arms are straight (*forward movement*). They slowly lower the dumbbells back to the starting position (it's called a *backward movement*), and repeat. In general, repetition pattern of all free weight exercises includes a starting position, a forward movement, and a backward movement. Fig. 5 shows the pattern of repetitions for the *overhead dumbbell press*. As we can see from Fig. 5, there exists a repetition pattern in acceleration and we give the reason for such pattern in the following. During the *overhead dumbbell press*, since the user's hand goes in the "negative" direction of y-axis in the glove during the forward movement (Fig. 2), muscles should induce "negative" acceleration to turn the velocity of the hand to "negative" and make it move. Around the end of the movement, she has to slow down her hand and stop her hand in the air, so "positive" acceleration occurs to turn the negative velocity of her hand to zero. As a result, the acceleration starting as negative and turning to be positive makes the pattern look like a "right \mathcal{X} " (\wedge): because the shape of the forward movement is similar to the second stroke of the letter \mathcal{X} . On the other hand, the backward movement makes the pattern look like a "left \mathcal{X} " (∇).

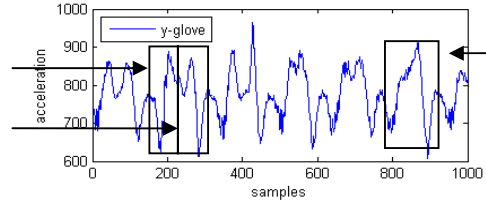


Fig. 5. The acceleration data in the y-axis of the glove while doing *overhead dumbbell press*. The rectangles mark repetition patterns and patterns of "left/right \mathcal{X} ".

Nonetheless, the repetition pattern may vary from the "left / right \mathcal{X} " form. In some cases, the right end of "left \mathcal{X} " overlaps with the left end of "right \mathcal{X} " and merges into a "V" shape (∇) (or into the shape of \wedge if the right end of the "right \mathcal{X} " overlaps with the left end of the "left \mathcal{X} "), also shown in Fig. 5. Such a case happens if users do not pause and start the backward movements right after forward movements. We found that occurrences of the merging situations are exercise-dependent. For example, in the *lateral raise* exercise, users tend to pause for a short while when their hands are lifted to the horizon, which causes the "left \mathcal{X} " plus "right \mathcal{X} " pattern. The case happens in *deadlift* as well. On the contrary, in the *bicep curl* exercises, users usually put their hands down right after they curl their hands up to the highest position. As a result, we must design an algorithm able to count repetitions whether a waveform has "left \mathcal{X} " plus "right \mathcal{X} " shapes, or the overlapped version "V". In other words, we must be able to consider whether users pause between forward and backward movements.

4 Algorithms and Experimental Results

This section presents several algorithms we have used to achieve the *recognition goal* and *counting goal*, along with experimental results. In summary, these algorithms follow the diagram shown in Fig. 6. Acceleration data is first streamed from an *accelerometer glove* and a *posture clip*. Data is then fed into the *type recognizer* which serves the *recognition goal*. After the recognizer recognizes what type of exercise a given segment of a data stream belongs to, it labels the acceleration data streams with a type and feeds them into the *repetition counter*. Then, because different exercises cause acceleration in different axes, the *repetition counter*, serving the *counting goal*, counts the number of repetitions from the acceleration data in the corresponding axis based on the *type* labeled. Finally, the *repetition counter* reports the type of exercise and how many repetitions a user performed. Such reports can be used by many possible weight exercise management applications in the future.

For the *recognition goal*, we present a set of features we selected and we show how we applied them to Naïve Bayes Classifier (NBC) and Hidden Markov Model (HMM), which are chosen based on past success [10][11][13]. We do not mean to constrain to these two methods. These methods are used to verify the selected features and hardware settings. For the *counting goal*, we developed algorithms that detect peaks in acceleration data and detect repetitions in the state sequence predicted from Hidden Markov Model. In addition, this paper compares the pros and cons of the algorithms in the experimental results subsections.

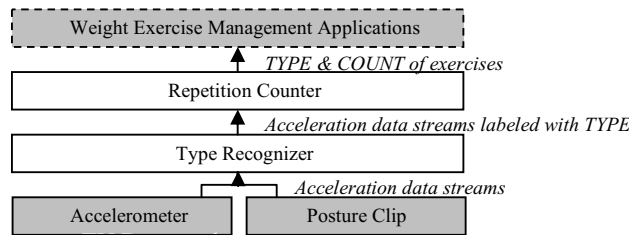


Fig. 6. Flow diagram

4.1 Algorithms of Recognition Goal

Naïve Bayes Classifier and Hidden Markov Model

Here we briefly introduce how to use Naïve Bayes Classifier and Hidden Markov Model. For the NBC, first features are extracted from sliding windows on acceleration data streams, each sliding window overlaps 50% with its adjacent windows. Then, the feature vector of each sliding window is fed into a NBC, which has been trained with other feature vectors into a Gaussian mixture. The NBC classifies each sliding window as a certain type of exercise based on the highest likelihood. Finally, the algorithm collects and reports a continuous sequence of windows with the same classified type. For HMM, we have to train a separate HMM for each individual exercise. Given any sequence of interest, the algorithm first extracts and transforms data as a stream of features. Then, it calculates likelihoods from each trained HMM.

Finally, it classifies the sequence to a specific type based on the HMM with the largest likelihood.

The Feature Space

Based on the discussion in section 2.3, we selected a set of features to identify the major axes, the gravity effect, and the correlation feature. The major axes are the axis in which users impose acceleration (for the particular exercise). The correlation feature is used to capture whether an exercise imposes acceleration in multiple axes at the same time. In particular, we customized the feature calculation while applying them to NBC and HMM: in NBC we calculated a feature vector for each sliding window while in HMM we transformed the stream of acceleration to a stream of features.

To identify the major axes, we found that it didn't achieve good accuracy if we directly applied raw acceleration data. Therefore, we integrate it to some extent to achieve good results. In NBC, an energy feature was chosen; we first applied Fast Fourier Transform to a sliding window and calculated the summation of the results. In other words, it summed up the squared magnitudes of waves. It is also divided by the window size to make the result independent of window size. In HMM, the velocity feature was chosen. It not only presents lower order than acceleration data, it also allows HMM to model temporal relationships, since velocity of a time point depends on the velocity of the previous time point. Velocity is the integral of acceleration and is approximated by cumulatively summing over acceleration and adding an adjustment factor that keeps the velocity over time from diverging. The adjustment factor works as the constant component in the integration equation. Doing this can make sure the calculated obeys the nature that after a long period of time, the velocity of human hands should still change between positive, zero and negative values, rather than diverging.

To capture the gravity effect, the peak magnitude feature is captured. It's calculated as the average magnitude values of positive and negative peaks in a sliding window. In particular in HMM, the peak magnitude of a sample in time t are calculated as the average positive and negative peak magnitudes in the sliding window centered in t (e.g. the sliding window of $[t-win/2, t+win/2]$). In fact, before we finalized the feature set for HMM, we didn't choose peak magnitudes as features. Instead, we used only raw acceleration data and tried to model peaks with more states. For the case between *flye* and *bench press*, it turned out raw acceleration feature didn't work well. There reason is the following. Since the acceleration in y-axis of *bench press* vibrates in the "sub-range" where the acceleration of *flye* vibrates (see Fig. 4), the acceleration of *bench press* falls exactly in the Gaussian mixture in the Hidden Markov Model of *flye*, which was trained to model acceleration of *flye* falling in the sub-range. As a result, it produced an ambiguous likelihood. The feature of peak magnitudes, instead, can avoid this problem by only providing information that acceleration of *flye* vibrates "larger" than *bench press*.

Finally, the correlation feature is calculated in the same way as first major-axis feature (i.e. the energy for NBC and the velocity for HMM), but it's calculated from the acceleration difference between each pair of axes. There are a total of four pairs of axes we consider, including every pair of axes from the *accelerometer glove*, (e.g. x - y , y - z and x - z in the glove), and the pair of the y -axis from the *acceleration glove* and from y -axis in the *posture clip* (e.g. y_{glove} - $y_{posture}$). The reason we choose the

pair of y_{glove} and $y_{posture}$ is to capture that the posture clip is moved up and down in y -axis in *deadlift* and *standing calf raise*, described in section 2.3. In addition, both the first and second features are calculated for each of the three axes from the *accelerometer glove* and the three axes from the *posture clip*. As a result, there are 22 features in a feature vector. Features are globally standardized so as to avoid numerical complications with the model learning algorithm.

4.2 Evaluation of Recognition Goal

To figure out whether the algorithms can be applied to a variety of users, say users with different gender, height, weight, level of experience with free weight exercises, etc., we collected data by asking *ten subjects* to perform the nine exercises listed in Table 1. There were eight male and two female subjects. To figure out whether the algorithms work well in different situations, we designed a data collection process in which we asked subjects to perform exercises with dumbbells of *different weights*. Weight difference is assumed to affect users' exercise performance. For example, light weights make users perform faster. Heavier weights cause users to move slower and may affect users' form.

Before collecting data, we showed subjects (or helped them review) how to perform each of the nine exercises. Then we asked subjects to do each exercise in three sets. Subjects performed 15 repetitions of each set and we manually started and stopped the recording process around each set. In addition, each set was designed with a different weight setting. We asked subjects to do the first set with normal weights, the second set with heavy weights, and finally the third set with light weights. Since the term of heaviness varies from subject to subject, depending on experience of free weight exercises, gender, height, etc., we asked subjects to decide the suitable heavy/light weights. Subjects stopped exercising if they felt tired. On the other hands, we let subjects do more than 15 repetitions if they could. In summary, we have collected data for 9740 seconds (162.5 minutes), with a total of 4925 repetitions. We applied methods to sets in isolation, rather than to the entire exercise session. In future work, we will address this harder setup.

Here we give details of the implementation of the algorithms. The implementation is based on the Bayes Net Toolbox in Matlab [25] and HMM Toolbox in Matlab [26]. The acceleration data streams are filtered with a low-pass filter before feature calculation. For NBC, each sliding window 256 samples long, approximated as a duration of 3 seconds at the sampling rate of 80Hz. The duration is selected to be large enough to capture dynamics in exercises. For HMM, each Hidden Markov Model was trained with parameters of one mixture and optimal number of states, which was around 3.

Two Cross-validation Protocols and Results

We tested both algorithms with two cross-validation protocols. The first *user-specific* protocol checks the algorithmic robustness for each single user, under different weight situations. Classifiers were trained on two out of three weight settings for each subject and tested on the remaining weight setting. This user-specific protocol was repeated for all ten subjects. The second *leave-one-subject-out* protocol aims to understand the robustness under user variety. Classifiers were trained on all the data from all subjects except one. The classifiers were then tested on the data from the

only subject left out of the training set. This leave-one-subject-out validation process was repeated for all ten subjects.

Table 2 lists the confusion matrices and recognition accuracy of the NBC using two cross-validation protocols. The user-specific protocol results in an overall 95% accuracy listed in Table 2-(a). The leave-one-subject-out protocol shows an overall accuracy of 85% in Table 2-(b). Table 3 lists the confusion matrices and recognition accuracy of Hidden Markov Models using the leave-one-subject-out protocol. As Hidden Markov Models can recognize the exercise type of a long sequence of data, Table 3-(b) lists the accuracy of each set. Nonetheless, to compare with NBC, we

Table 2. Confusion matrices and recognition accuracy of Naïve Bayes Classifier by (a) the user-specific protocol and (b) the leave-one-subject-out protocol: number in (i, j) of the matrix means number of sliding windows in exercise i recognized as exercise j

(a) User-specific protocol											(b) Leave-one-subject-out protocol										
	Bicep	Tricep	Bench	Flye	Bent.	Later.	Overh.	Dead.	Stand.		Bicep	Tricep	Bench	Flye	Bent.	Later.	Overh.	Dead.	Stand.		
Bicep	649	0	0	0	0	1	1	0	0	0.99	Bicep	651	0	0	0	0	0	0	0	1.00	
Tricep	0	663	0	0	0	0	0	0	0	1.00	Tricep	0	663	0	0	0	0	0	0	1.00	
Bench.	0	0	791	1	0	0	0	0	0	0.99	Bench	0	0	675	117	0	0	0	0	0.85	
Flye	0	0	14	775	0	0	0	0	0	0.98	Flye	0	0	20	769	0	0	0	0	0.97	
Bent.	0	0	0	0	520	0	0	2	0	0.99	Bent.	0	0	0	0	439	20	0	55	1	0.84
Later.	0	0	0	0	1	599	0	1	2	0.99	Later.	0	0	0	0	4	595	0	0	4	0.98
Over.	0	0	0	0	0	0	522	0	0	1.00	Over.	9	3	0	0	0	0	510	0	0	0.97
Deadl.	0	0	0	0	6	3	0	612	43	0.92	Deadl.	0	0	0	0	14	0	0	393	257	0.59
Stand.	0	0	0	0	0	0	0	15	551	0.97	Stand.	0	0	0	0	0	0	0	58	508	0.89

Table 3. Confusion matrices and recognition accuracy of Hidden Markov Model by the leave-one-subject-out protocol. (a) The number in (i, j) of the matrix means number of sliding windows in exercise i recognized as exercise j . (b) The number in (i, j) of the matrix means number of sets in exercise i recognized as exercise j .

(a) Window-based											(b) Sequence-based										
	Bicep	Tricep	Bench	Flye	Bent.	Later.	Overh.	Dead.	Stand.		Bicep	Tricep	Bench	Flye	Bent.	Later.	Overh.	Dead.	Stand.		
Bicep	640	1	0	0	0	9	0	0	0	0.98	Bicep	36	0	0	0	0	0	0	0	1.00	
Tricep	0	651	0	0	0	10	0	0	0	0.99	Tricep	0	34	0	0	0	0	0	0	1.00	
Bench	0	0	692	100	0	0	0	0	0	0.87	Bench	0	0	32	5	0	0	0	0	0.86	
Flye	0	0	8	767	0	0	0	0	0	0.93	Flye	0	0	2	31	0	0	0	0	0.94	
Bent.	14	0	0	0	497	10	0	1	0	0.93	Bent.	1	0	0	0	29	2	0	0	0.91	
Later.	0	0	0	0	15	586	0	0	0	0.97	Later.	0	0	0	0	1	31	0	0	0.97	
Over.	52	21	0	0	0	0	447	0	0	0.86	Over.	3	0	0	0	0	0	29	0	0.90	
Deadl.	0	0	0	0	0	0	0	569	94	0.86	Deadl.	0	0	0	0	0	0	0	29	4	0.87
Stand.	0	0	0	0	6	3	0	184	372	0.67	Stand.	0	0	0	0	0	0	0	11	20	0.64

explicitly segmented data into sliding windows and used HMM to recognize data in each window. This worked by feeding the stream of features of each window to HMM. Table 3-(a) lists the results. As we can see, both methods achieved similar results.

Nonetheless, when we tried to test HMM using the user-specific protocol, it turned out the results were not acceptable. We do not list the results here. We believed that the data was not sufficient to train HMM well in the user-specific protocol, since we could only use data from two out of three sets to train the model, which was too few for HMM. In addition, because of the fact that there are multiple state settings and therefore multiple Gaussian mixtures to be trained in HMM, in comparison with only one Gaussian Mixture in the NBC, the same amount of data that is sufficient is for NBC may be too little for the HMM. Nonetheless, since we used the same feature set in both HMM and NBC, we believe if we collect enough data from each individual user in the future, HMMs may achieve comparable results.

In general, the NBC performs well in the user-specific protocol, so we conclude that the algorithm is robust for every single user, even with different weight settings. The reasons are the following. Sometimes heavier weights made subjects perform exercises slower, but the NBC is still able to capture the acceleration dynamics since we chose big sliding windows. Sometimes their hands shook, but resulting signal noise was filtered out by a low-pass filter. More importantly, there is a factor favoring such results: we found individuals tended to perform exercises in consistent patterns.

However, consistent patterns by individuals don't imply different users would perform exercises with the same pattern. As a result, it shows worse performance in the leave-one-subject-out protocol. For example, it is ambiguous between *deadlift* and *standing calf raise*. The reason is that some subjects tended to do *deadlift* slowly and caused smaller acceleration in *y*-axis, whereas some others performed *standing calf raise* more abruptly and caused larger acceleration in *y*-axis. The two cases make *deadlift* (with decreased acceleration) and *standing calf raise* (with increased acceleration) alike; making it harder to differentiate them. Others including *bent-over row* versus *deadlift*, *bench press* versus *flye*, and *bent-over row* versus *lateral* also present a similar ambiguity. However, there was an important implication. Although it showed that it's less robust under a variety of users, the robustness of tracking individuals leads us to an opportunity: calibration. Calibration done for each individual user can boost the recognition performance, and such recognizers would become persistent throughout the use of that user.

Discussion

In fact, the ambiguity problem comes from free weight exercises themselves: some exercises would impose similar acceleration responses, on the hardware setting we currently have. For *deadlift* and *standing calf raise*, situations would happen that just can't tell from the acceleration by the glove and clip. The acceleration only shows people up and down in *y*-axis. Nonetheless, if we add extra accelerometers on thighs, it's easier to differentiate them, since people would have to bend their knees in *deadlift*. Similarly, it could also help in differentiating *bench press* and *flye* if we add another accelerometer on the upper arm. In other words, it's a tradeoff between

minimalist design and more clues for better accuracy. Actually, we have already made the promise by adding an accelerometer clip, since posture is an important factor to be captured.

Another solution is to try dynamic time warping (DTW) [27]. In particular, we found that *standing calf raise* usually responses like a sharp peak and *deadlift* usually performs like a rather flatter one. DTW may identify the shape difference better and could give us more hints. As this could only be applied to *deadlift* and *standing calf raise* at this point, I would say it's only a partial solution.

In addition, based on the evaluation results, it turned out that the feature of simple peak magnitudes was better than raw acceleration. This also led to another conclusion that the dynamics of gravity effect should be well-captured to track free weight exercises. Peak magnitudes achieved good results. However, dropping raw acceleration from the feature space is actually not the best idea for the long term goal of detecting whether users do exercise in a proper form, since raw acceleration data contains the most details.

4.3 Algorithms of Counting Goal

Peak Detection

To count how many repetitions a user has done so far, we have to look at the acceleration data, find the pattern that repeats itself, and count how many repetitions there are. As we just mentioned, each exercise causes acceleration in certain axis (or a combination of axes), which we have called the *major axis*. Experiments show that we can avoid the noise that exists in the other axes and get better results if we find patterns only in the major axis can. The major axis is chosen based on the exercise type labeled on the acceleration data by the type recognizer. In Table 4, we list the major axis that we chose for each of the nine exercises. For those exercises associated with acceleration changes in two axes, such as *bicep curl*, we picked one of them as the major axis.

In section 2.3, we mentioned that a pair of forward/backward movements causes acceleration with a shape similar to the shape "left/right \mathcal{X} " or shape " \mathcal{V} ". From those observations, we developed an algorithm that applies a strong low-pass filter with high order to the acceleration data in the major axis. As an example shown in Fig. 7, this can filter a pair of "left \mathcal{X} " and "right \mathcal{X} " waveforms into an overlapped shape " \mathcal{V} ", making the acceleration data follow " \mathcal{V} " patterns. Then, the algorithm simply counts the number of peaks in the resulting data. A peak is identified by checking whether in data there is a decrease followed by an increase, and vice versa. We have evaluated the algorithm with different parameters and will discuss the results next.

Table 4. The major axes chosen for free weight exercises

Major axis	Free Weight Exercises
x-axis	Tricep curl
y-axis	Biceps curl, bench press, flye, bent-over row, shoulder press, deadlift, standing calf raises
z-axis	Lateral raise

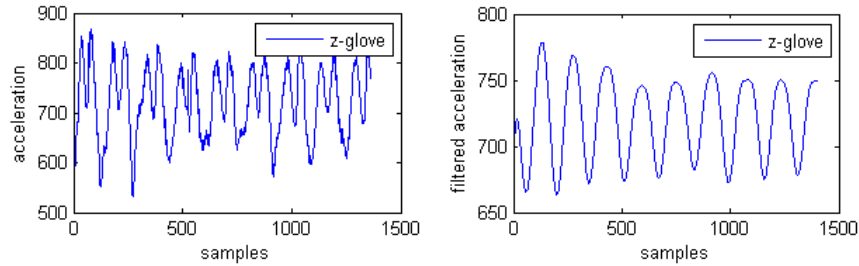


Fig. 7. (a) The raw acceleration data in the major axis (z -axis) of *lateral raise* and (b) the acceleration data being applied with a strong low-pass filter

State Sequence in Hidden Markov Models

The peak detection algorithm is expected to depend strongly on the selection of a good filter: it is necessary to find a low-pass filter with exact strength to correctly filter data into "U" patterns. So, we decided to leverage the modeling power of Hidden Markov Models for this problem. The Hidden Markov Model should be able to model and tolerate the coexistence of the pattern of "left/right X" and "U", and predict corresponding state sequences. We reduced the problem of counting repetitions in acceleration data (with continuous values) into a problem of counting repetitions in the predicted state sequence (with finite states). The algorithm worked by first calling the Viterbi algorithm [28] to predict the state sequence from acceleration data. Then, it counted the repeating patterns in the state sequence.

We trained a *counting version* of Hidden Markov Models for each exercise, using the features extracted from the acceleration data in the major axis, listed in Table 5. The features included acceleration and velocity, in which the calculation of velocity has been described in section 3.1. In addition, features were normalized to make sure it predicts a good state sequence: feature values were shifted to let its waveform vibrate around the zero point and are scaled to be bounded by a specific range ($\pm d$). The reason is to model the pattern of "left/right X" and "U", because it's more important to model the change of the waveforms rather than absolute magnitudes.

4.4 Evaluation of Counting Goal

Peak Detection

To obtain the best filter, we tried a number of low-pass filters with different strengths on acceleration data, and fed the filtered data into the peak detection algorithm. The counting results are listed in Table 5, with filters of order $n = 64, 96,$ and 128 samples. The filter coefficients were generated by the *fir1*³ function by MATLAB, with order n and with low-pass frequency of $1/n$ of the Nyquist frequency. As the sampling frequency of the accelerometers is 80Hz, the filters are on the order of 0.8, 1.2, and 1.6 seconds. As we can see, this simple algorithm achieves acceptable counting results, which is less than five percent error rate. In other words, if a user performs

³ The *fir1* function implements the classical windowed linear-phase FIR digital filter design [29].

100 repetitions of an exercise, the algorithm miscounts five reps. Nonetheless, the *bench press* had a 16% of miscount rate. We found that since subjects lay on a bench to do a *bench press*, some tended to have tiny preparation motion (by lowering arms a little), before they pressed the dumbbells up, and caused more peaks to be counted.

The algorithm is simple, easy to implement, and achieves good performance, but there are problems. Since the key is to successfully filter “left/right \mathcal{X} ” patterns into an overlapped version of “ \mathcal{U} ”, the choice of filter parameters thus becomes tricky. If we choose a filter not strong enough, i.e. with smaller order, it may not be able to merge into “ \mathcal{U} ”. However, if we choose a filter too strong, it may instead merge two adjacent “ \mathcal{U} ” together and increase the miscount rate. For example, the order of 64 gives *bicep curl* the best count accuracy, whereas *deadlift* must be applied with a filter of order higher than 94 to achieve acceptable accuracy. Therefore, we can find in Table 5 that there is not a filter setting that achieves the global best accuracy.

In addition, we can expect that the filter strength depends on how long a user pauses between forward/backward movements. If a user pauses longer, for example the user is using heavier weights, the order of a filter should be large enough to merge the “left \mathcal{X} ” and “right \mathcal{X} ” together, which makes a fixed filter setting more infeasible.

Table 5. Counting accuracy of the peak detection algorithm. The table lists results using low pass filters of different strengths, each with order 64, order 96, and order 128. The miss count, miss count rate, and the base actual count for each exercise are listed.⁴

Exercise		Bicep curl	Tricep curl	Bench press	Flye	Bent. row	Later. raise	Over. press	Deadl. calf r.	Stand. calf r.
Actual Count		596	566	640	561	535	507	523	570	541
Order 64	Miss Count	3	22	137	28	5	143	31	130	80
	Error Rate	.0050	.0389	.2141	.0499	.0093	.2821	.0593	.2281	.1479
Order 96	Miss Count	8	18	105	19	17	41	61	25	36
	Error Rate	.0134	.0318	.1641	.0339	.0318	.0809	.1166	.0439	.0665
Order 128	Miss Count	25	36	127	43	58	85	161	49	108
	Error Rate	.0419	.0636	.1984	.0766	.1084	.1677	.3078	.0860	.1996

State Sequence Prediction with Hidden Markov Models

Table 6 shows that this method can achieve overall good accuracy. Here was the process of the experiment. We first applied a low-pass filter of order $n=32$ to the acceleration data, and extracted features to train each Hidden Markov Model. It turned out that trained Hidden Markov Models can map hidden states to different parts of a repetition: the starting position, the forward/backward movements, the ending position of forward movements, and the pause between forward/backward movements. Then, we selected one of the states as an anchor state, say the state of forward movement, to serve as the evidence that the pattern repeats. In other words,

⁴ Actual count is the number of repetitions subjects actually performed, which was manually counted during data collection. Miss count is the absolute value of the difference between the actual count and the count from the repetition counter. Error rate is calculated as miss count over actual count.

we count how many segments are predicted to be the anchor state. The error, as expected, came from the incorrect prediction of the state sequence, which might results from noise or different user patterns. The errors can be categorized as insertion, deletion or substitution errors, which could be solved using sequence reconstruction [30] or autocorrelation. We leave this as our future work. Similar to the peak counting algorithm, *bench press* cannot be counted well because of the preparation motions some subjects did.

Table 6. Counting accuracy of state sequence in Hidden Markvo Model. The miss count, miss count rate, and the base actual count for each exercise are listed.

Exercise	Bicep curl	Tricep curl	Bench press	Flye	Bent. row	Later. raise	Over. press	Deadl.	Stand. calf r.
Actual Count	596	566	640	561	535	507	523	570	541
Miss Count	8	19	90	44	18	4	47	26	41
Error Rate	.0134	.0336	.1719	.0784	.0336	.0073	.0899	.0456	.0758

Discussion

For counting algorithms, although the filtering and peak counting approach can count pretty well, without really looking into data patterns, it is less adaptive. One way to improve it could be using FFT to approximate the length of repetition first and then applying an adaptive filter to the data. The approach of the state sequence and Hidden Markov Model is of course more adaptive, but it requires more work to train. Another outcome of the peak detection algorithm is to help detect improper form. It's important to maintain constant and reasonable speed doing free weight exercise, and measuring the duration between peaks can help achieve the goal.

5 Conclusion and Future Work

The paper proposes a new application domain with free weight exercises: an exercise tracker, an exercises planner, and a digital personal trainer. It describes a feature space and the evaluation of methods to track free weight exercises. We incorporated a three-axis accelerometer into a workout glove to track hand movements and put another accelerometer on a user's waist to track body posture. With the accelerometer settings, both approaches of Naïve Bayes Classifier and Hidden Markov Models resulted in around 90% accuracy to recognize the types of exercise. We also developed a peak counting algorithm and used state sequence with Hidden Markov Model to count how many repetitions you have done so far, which achieved around a 5% miscount rate. We also discuss the implications in the evaluation sections.

One of our future works would be developing a real-time system in mobile devices. To achieve that, we have to first train a garbage model for movements that do not belong to any of the target exercises to avoid false positives. In addition, we are planning to model exercises in detail in order to detect improper form, which would be able to serve as the basis of a *digital personal trainer*.

References

1. Ades, P.A.: Cardiac rehabilitation and secondary prevention of coronary heart disease. *The New England journal of medicine* 111, 369–376 (2001)
2. Shephard, R.J., Balady, G.J.: Exercise as Cardiovascular Therapy. *Circulation*, New York 99, 963–972 (1999)
3. Sothorn, M.S., Loftin, M., Suskind, R.M., Udall, J.N., Blecker, U.: The health benefits of physical activity in children and adolescents: implications for chronic disease prevention. *European Journal of Pediatrics* 158(4), 271–274 (1999)
4. Dishman, E.: Inventing wellness systems for aging in place. *IEEE Computer* 37, 34–41 (2004)
5. Healey, J., Logan, B.: Wearable Wellness Monitoring Using ECG and Accelerometer Data. HP Laboratory Tech Report, HPL-2005-134 (July 2005)
6. The President’s Council on Physical Fitness and Sports. Fitness fundamentals guidelines for personal exercise programs. online council publications (2003), <http://www.fitness.gov/fitness.htm>
7. FitLinxx, <http://www.fitlinxx.com/brand.htm>
8. Apple – Nike + iPod, <http://www.apple.com/ipod/nike/>
9. Intille, S.S.: Designing a home of the future. *IEEE Pervasive Computing* 1(2), 76–82 (2002)
10. DeVaul, R., Sung, M., Gips, J., Pentland, A.S.: MITHril 2003: Applications and Architecture. In: *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, October 2005, pp. 4–11. IEEE Computer Society Press, Los Alamitos (2005)
11. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) *PERVASIVE 2004*. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
12. Minnen, D., Starner, T., Essa, I., Isbel, C.: Discovering Characteristics Actions from On-Body Sensor Data. In: *Int. Symp. On Wearable Computing (ISWC)*, Montreux, CH, October 2006, pp. 11–18 (2006)
13. Ward, J.A., Lukowicz, P., Tröster, G.: Gesture spotting using wrist worn microphone and 3-axis accelerometer. In: *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, vol. 121, pp. 99–104 (2005)
14. Westeyn, T., Brashear, H., Atrash, A., Starner, T.: Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. In: *Proceedings of International Conference on Perceptive and Multimodal User Interfaces 2003*, pp. 85–92 (2003)
15. Benbasat, A.Y., Paradiso, J.A.: An Inertial Measurement Framework for Gesture Recognition and Applications. In: Wachsmuth, I., Sowa, T. (eds.) *GW 2001*. LNCS (LNAD), vol. 2298, pp. 9–20. Springer, Heidelberg (2002)
16. Wyatt, D., Philipose, M., Chouhury, T.: Unsupervised Activity Recognition using Automatically Mined Common Sense. In: *proceeding of the Twentieth National Conference on Artificial Intelligence, AAAI2005*, Pittsburg, PA, July 2005, pp. 21–27 (2005)
17. Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., Isbell, C.: Discovery and Characterization of Activities from Event-Streams. In: *Proceedings of UAI 2005*, Edinburgh, Scotland, pp. 71–78 (2006)
18. Lester, J., Choudhury, T., Kern, N., Borriello, G., Hannaford, B.: A Hybrid Discriminative –Generative Approach for Modeling Human Activities. In: *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI 2005*, July 2005, pp. 766–772 (2005)

19. Subramanya, A., Raj, A., Blimes, J., Fox, D.: Recognizing Activities and Spatial Context Using Wearable Sensors. In: Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (2006)
20. Delavier, F.: Strength Training Anatomy, 2nd edn. Human Kinetics Publishers (2005)
21. Cane, J., Glickman, J., Johnson-Cane, D.: Complete Idiot's Guide to Weight Training. Alpha Books (1999)
22. Spark Fun Electronics, <http://www.sparkfun.com/>
23. Benbasat, A.Y.: An Inertial Measurement Unit for User Interfaces, Master Thesis, MIT Media Lab (September 2000)
24. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hähnel, D.: Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing* 3(4), 50–57 (2004)
25. Murphy, K.P.: The Bayes net toolbox for MATLAB. Technical report 94720-1776, Dept. Comp. Sci., University of California at Berkeley, Berkeley, CA (2001)
26. Murphy, K.P.: Hidden Markov Model (HMM) Toolbox for Matlab, <http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>
27. Hartmann, B., Abdulla, L., Mittal, M., Klemmer, S.R.: Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In: Proceedings of ACM CHI 2007 Conference on Human Factors in Computing Systems, CHI2007, pp. 145–154. ACM Press, New York (2007)
28. Forney, G.D.: The Viterbi algorithm. *Proceedings of IEEE* 61(3), 268–278 (1973)
29. Programs for Digital Signal Processing, Algorithm 5.2. IEEE Press, New York (1979)
30. Lenenshtein, V.I.: Efficient reconstruction of sequences from their subsequences or super sequences. *J. Comb. Theory Ser. A* 93(2), 310–332 (2001)