Human Motion Segmentation by Data Point Classification

Jonathan Feng-Shun Lin, Vladimir Joukov and Dana Kulić

Abstract— **Contemporary physiotherapy and rehabilitation practice uses subjective measures for motion evaluation and requires time-consuming supervision. Algorithms that can accurately segment patient movement would provide valuable data for progress tracking and on-line patient feedback. In this paper, we propose a two-class classifier approach to label each data point in the patient movement data as either a segment point or a non-segment point. The proposed technique was applied to 20 healthy subjects performing lower body rehabilitation exercises, and achieves a segmentation accuracy of 82%.**

I. Introduction

To assist physiotherapy assessment and diagnosis, body worn sensors can be used to measure human movement [1]. When continuous data of patient movement is collected, the exercise data needs to be segmented to isolate when a patient begins and ends each exercise repetition. These segments can be used to extract useful metrics like mean velocity, range of motion, or the timed-up and go [2], allowing physiotherapists to assess their patient's progress quantitatively, and to provide patient feedback.

Motion segmentation, the process of locating the start and end locations of the movements of interest, is difficult due to: (1) large dimensionality of the time-series data [3], and (2) intra- and inter-personal spatial and temporal variability between patient movements due to stature, fitness and health conditions. For rehabilitation supervisory systems, the algorithm should also be on-line, in order to provide real-time feedback to the patient.

Numerous techniques have been proposed for motion segmentation. A common approach is to use zero-crossings and thresholds that characterize joint trajectory direction change as segment points, *e.g.* velocity crossings [4], joint acceleration [5] or linear acceleration [6]. However, these algorithms tend to over-segment.

Alternatively, the movement data can be modelled explicitly, *e.g.* via piecewise linear approximation [7]. Segments are declared when one linear segment ends and the next one begins. However, this method requires careful tuning of the regression error to prevent over-fitting.

Dynamic Time Warping (DTW) [8] has also been employed, where the temporal variations between the template and observation data are removed by warping the time scale of the observation to the template. Although dynamic time warping can accurately segment observation data with significant temporal variations from the template, it does not scale well to higher dimensions, and cannot be used on-line.

The Hidden Markov Model (HMM) [9] can also be used to model movement data. Researchers have used the HMM to model key poses as states in movement templates [10], and segment when the windowed observation data is sufficiently similar to the templates. Alternatively, each state of the HMM can be used to describe a movement primitive [11], and segments are declared on state transition.

Classification techniques such as the Support Vector Machine (SVM) [12] have also been considered. Several different SVMs can be trained to identify data as belonging to a particular class, and segment when the label changes [13]. Decision trees [14], *k*-Nearest Neighbour (*k*-NN) [15] and Artificial Neural Networks (ANN) [16] have all been used in a similar fashion. However, these approaches perform segmentation only when one motion type transitions to another, exercise supervision algorithms also require the ability to segment when the same action is performed multiple times.

This paper proposes an on-line segmentation algorithm that can segment between different movement primitives, as well as repetitions of the same primitive, by classifying individual data points as a *segment point* (p_1) or a *non-segment point* (p_0). The proposed method is appealing as this converts the difficult temporal segmentation task into an easier classification task. Once segment points are determined, additional classification can be done to identify the underlying motions.

II. Proposed Approach

This paper describes a method to utilize classifiers to discriminate between p_1 and p_0 points at each time point. However, using classifiers in this manner to perform segmentation is not trivial, and gives rise to several issues unique to classifier algorithms. Unlike the HMM or the DTW algorithms, classifiers do not inherently consider temporal information, which is an important aspect to movement data.

Appropriate generation of p_1 points is also important. Classifiers require training prior to use, so labelled data is needed *a priori*. Manually labelled segments typically specify a single time point to denote the start and the end of a given exemplar, which is not suitable for classifiers, as there is minimum difference between the data point at time t_n and at time $t_{n\pm1}$.

Lastly, unbalanced datasets are also a concern. For example, in a given set of exercise data, there are only a small number of p_1 points compared to p_0 points.

J. Lin, V. Joukov, and D. Kulić are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (email: jf2lin@uwaterloo.ca; vjoukov@uwaterloo.ca; dkulic@uwaterloo.ca)

978-1-4244-7929-0/14/$26.00 ©2014 IEEE

A. Model Training

A classifier requires sample data, named *motion exemplars*, in order to extract the characteristics of p_1 required for segmentation. The exemplars consist of annotated segment boundaries of each exercise repetition. The motion data can be measured by a sensor system, such as inertial measurement units (IMUs) [6], [10], and annotated by experts via video playback [11], [10].

Using these training exemplars, the classifier constructs a model or *template* to differentiate between p_1 and p_0. This section describes the components of the training process.

1) Normalization: The exemplar data are normalized to reduce the impact of the inter- and intra-participant variability, by subtracting the initial value from the data.

2) Manual Segment Point Expansion: In order to increase the number of segment training points (and decrease data imbalance) and include points very similar to the segment point in the segment exemplar set, an additional n_{exp} points before and after each manually labelled segment point are denoted as p_1. In addition, the data points between the end of one segment and the start of the next are all labelled as p_1. This assumes that points when the demonstrator is at rest between motions are also considered segment points, and allows for additional training data points to be added that share similar characteristics as the manual p_1 points.

3) Outlier Rejection: The quality of the training data is checked by examining its velocity, in order to remove velocity spikes. In a given exemplar, the peak velocity for each continuous segment and non-segment block is determined, and clustered by *k*-means, where *k* = 2. If the higher cluster in the *k*-means contains only one or two peaks, the segments containing these velocity spikes are removed from consideration for training. Data between t_1 and the start of the first segment is also rejected from the training data, to remove any noise due to data collection initialization. The joint angles examined are determined by calculating the variance of the exemplar, over the whole dataset, and selecting the joint angle that varies the most.

In addition, any segment and non-segment block that has velocities above some threshold is also removed from the training data, in case numerous peaks exists in a given block, and the *k*-means method did not reject the velocity spikes. This threshold would be tuned to the dataset.

4) Input Vector Stacking: Classifier techniques do not typically consider temporal factors. Although the input vector could include data with temporal information, such as joint velocity, additional temporal data may be required to adequately capture the temporal nature of the movement data. To consider short term temporal effects, the input vector is stacked, so that a given data point includes data from a few time steps before and after the current data point. That is, $t_{use} = [t_{n-n_{stack}} \cdots t_{n-1}, t_n, t_{n+1} \cdots t_{n+n_{stack}}]$. n_{stack} requires tuning to optimize for the data.

5) Downsampling: In a given dataset, there will be significantly more p_0 than p_1 points. To reduce the impact of unbalanced data, two layers of downsampling are employed. (1) The number of p_1 and p_0 for each exemplar are noted, and the smallest values are denoted as $p_{1_{min}}$ and $p_{0_{min}}$, respectively. Each exemplar is randomly downsampled to match $p_{1_{min}}$ and $p_{0_{min}}$, to eliminate the unbalanced dataset. (2) The full training data is resampled to make sure it does not exceed a limit of 7500 data points for either p_1 or p_0, as the installation of MATLAB used is unable to create arrays larger than 7500^2, which is required for certain classifiers. For p_1, it is randomly resampled. For p_0, Gaussian resampling is used to favour p_0 points close to existing p_1. See Figure 1 for an illustration.

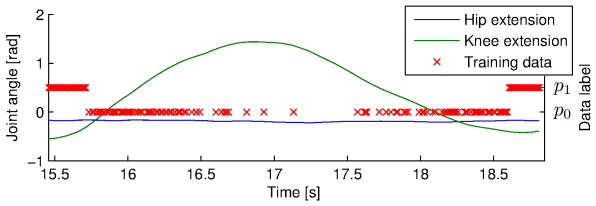

Fig. 1. Knee extension training data. The red points at $y = 0$ denote the training data selected for p_0, sampled by Gaussian resampling, favoring non-segment points closer to the boundary. The red points at $y = 0.5$ denote the training data selected for p_1, which has no downsampling.

6) Segmentation Classifier Training: After the above steps, the individual exemplar features are concatenated into a data matrix, and passed into a three-stage training process, consisting of a dimensionality reduction algorithm, a base classifier, and an aggregator. Common approaches to classification involve either utilizing dimensionality reduction or feature selection, then using a simple classifier, or forgoing any dimensional alternation and using a powerful classifier instead. The different combinations of these three components are compared:

1) Dimensionality reduction
   - No transformation applied
   - Principal Component Analysis [17] (PCA)
   - Fisher's Discriminate Analysis [17] (FDA)
2) Classifier
   - *k*-Nearest Neighbour [17] (*k*-NN)
   - Quadratic Discriminate Analysis [17] (QDA)
   - Radial Basis Function [18] (RDF)
   - Support Vector Machine [12] (SVM)
   - Artificial Neural Networks [19] (ANN)
3) Aggregation
   - No aggregation algorithm
   - Boosting [20]
   - Bagging [21]

B. Novel Data Classification

To classify, each set of observation data is processed in a similar fashion as the training data. The input vector is normalized by subtracting the initial value, then stacked with data points from before and after the time point itself. If a dimensionality reduction algorithm is used, it is then applied to the processed observation data before classification begins. If n_{stack} data stacking occurred, then time points t_1 to $t_{n_{stack}}$ and $t_{n-n_{stack}}$ to t_n cannot be classified since not

enough data is available to be stacked, and thus are excluded from the classification effort. Unlike the training data, no downsampling or noise rejection procedure is applied to the observation data. For verification, the ground truth p_1 points are expanded by n_{exp}.

To compare the proposed approach to temporal methods, each cluster of p_1 points is converted into a single temporal segment point by clustering the segment and non-segment points. Clusters shorter than n_{exp} in length are removed. Each cluster is then converted into an ending point for the n^{th} segment, and a starting point for the $n + 1^{th}$ segment. These segment points are declared n_{exp} from the two edges of the cluster, or one-third of the length of the cluster, if the cluster is shorter than n_{exp} in length.

III. Experiments

A. Motion Database

The algorithm was tested with a database of 20 healthy participants performing 20 repetitions of 5 rehabilitation motion types each. On average, each repetition took 3 seconds to complete. The exercises performed were: knee extension while seated, sit to stand, squats, knee/hip flexion while supine and hip extension while supine. The data was collected via 3 Shimmer IMU sensors [22], transmitting at 128 Hz. The IMU data was used to compute joint angles via an Extended Kalman Filter (EKF) [1]; the joint angles were used as the input features for classification. The input feature vector for the classifier consisted of the joint positions and velocities for each time step and the joint positions and velocities at the previous and subsequent 15 timesteps. For this dataset, manual segments of the exercises were generated and labelled by an expert watching the motion in video playback of motion capture data that was collected simultaneously.

All processing and algorithm implementation were done in MATLAB 8.0. All analysis was performed in MATLAB, along with the libsvm Toolbox [23], the Toolbox for Dimensionality Reduction [24], the Bayes Net Toolbox [25], and the ReBEL toolkit [26].

B. Algorithm Implementation Details

For a majority of the algorithms examined, standard configurations were used, and the tuning parameters are examined in Section III-D.

For the PCA and the FDA, the optimal number of principal components (PCs) was determined dynamically, via the elbow approach. The elbow is determined by ordering the fraction of total variance in the data represented by each principal component, then selecting the top *k* components that contain some amount of variance. This threshold was set to 80%. The *k* determined by the elbow method is reported. Alternatively, *k* can be manually tuned.

Soft-margin SVM was used to account for poor separability in the data. A feedforward ANN was examined for the purposes of this paper.

For boosting, the AdaBoost [20] is employed. However, the AdaBoost algorithm is formulated to work with classifiers that can accept weighted data points, which does not apply to all classifiers. Instead, a resampling scheme, based on the data weights, is employed [27].

C. Verification

To calculate the segmentation accuracy, each point in the observation is labelled p_1 or p_0. The number of correctly identified segment points, the true positives (TP), as well as all false positives (FP) and false negatives (FN), are aggregated together and reported as the $F_{1_{Seg}}$ score, which represents a measure that aggregates both precision and recall accuracy. The p_0 points are not considered for the $F_{1_{Seg}}$ as the larger volume of the p_0 points may obscure the true segmentation accuracy of the model. The $F_{1_{Seg}}$ score is calculated as follows:

$$F_{1_{Seg}} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

The overall classification accuracy, including both segment and non-segment points, can be assessed by incorporating the number of true negatives (TN) into the F_1 score. Here, the p_0 points are included in the calculations to assess the labelling accuracy within and between each of the primitive segment points. The $F_{1_{Class}}$ score is calculated as follows:

$$F_{1_{Class}} = \frac{2 \cdot (TP + TN)}{2 \cdot (TP + TN) + FN + FP}$$

For the temporal segment points, an algorithmic segment point is declared as TP if it is within some t_{err} of a manual segment point. If multiple segment points appear within the t_{err} of the same manual point, the extra points are declared as FP. If algorithmic segment points are declared outside of the t_{err} range of any manual segment point, it is also declared a FP. If no algorithmic segment points appear within the t_{err} of a manual segment point, it is declared a FN.

D. Test Parameters

Both the joint angle and the joint velocity are chosen to be part of the dataset in order to incorporate temporal information in the segmentation process. Preliminary testing showed that a high n_{stack} results in higher computation time, and is set to 15 to balance between runtime and accuracy. n_{exp} selection was more difficult, as it changes the number of manual p_1 points, thus altering the ground truth data. The higher n_{exp}, the more p_1 training points became available. For comparison to existing work, n_{exp} was set to 25, which corresponds to 0.2 seconds [10].

Input data summary:

- Input datatype: Joint angle q and joint velocity $\dot{q}$
- Manual segment point expansion: $n_{exp} = [25]$
- Input vector stacking: $n_{stack} = [15]$

Dimensionality reduction algorithm:

- No dimensionality reduction
- PCA and FDA: PCs set by elbow at 80% or set to 2

Classifier algorithm:

- *k*-NN: $k = [3, 9]$
- RBF: number of RBFs used, $n_{RBF} = [10, 20]$

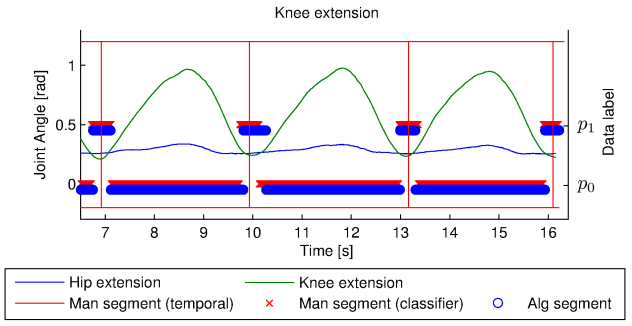

Fig. 2. Segmented knee extension motion. Points at around the $y = 0$ area are manual (red) and algorithmic (blue) points denoted as p_0, while points at around $y = 0.5$ are p_1 points.

- SVM: kernel function used, linear, polynomial, radial
- ANN: layers and neuron count, $n_{layers} = [10, 10], [10, 10, 10], [20, 20, 20]$

Aggregator algorithm:

- No aggregator
- Boosting: iteration count $n_{iter} = [3, 5]$
- Bagging: iteration count $n_{iter} = [3, 5]$

E. Experimental Configurations

Two different configurations were tested:

1) Segmentation Using Individualized Templates: Templates were generated from all 5 motions from a single subject, and tested on another set of the same motions from the same participant, to evaluate robustness against intra-subject variations. Data from all was used in this test.

2) Segmentation Using Generalized Templates: Templates were generated from all 5 motions from 5 different subjects, and tested on another set of 5 participants, to evaluate robustness against inter-subject variability. The training-testing sets were rotated through the 20 subjects, for a 4-fold cross-validation.

IV. Results

Table I shows the results of the individualized templates. Table II shows the results for the generalized templates. Figure 2 shows an example of the segmentation. The rank order in the tables was determined based on the $F_{1_{Seg}}$ score of the generalized templates, and was used for both tables.

Overwhelmingly, the top performing classifiers utilize PCA. Using the elbow method, PCA selects between 30 to 40 dimensions to represent 80% of the total variability, confirming that the data is redundant and correlated, so that a lower dimensionality feature vector can be used.

The classifiers that reported the highest $F_{1_{Seg}}$ were the SVM, the ANN and the *k*-NN. These classifiers provided high accuracy in both the individualized template and the generalized template tests, suggesting that these classifiers are suitable for both intra- and inter-subject segmentation. The high processing costs of *k*-NN makes it unsuitable for on-line applications, however.

The tables show that aggregated techniques do not improve segmentation accuracy, as the top 10 classifiers include both non-aggregated and aggregated variants. Note that some aggregation is already performed by the sampling technique, which preferably selects non-segment points closer to the segmentation boundary for training. These results suggest that the training data sampling scheme is sufficient, since the sampling scheme emphasizes the p_0 points close to the p_1, which are likely to have a higher chance of misclassification.

The misclassifications in the classifiers generally stem from the tendency of the classifiers to over-declare p_1 points, leading to a high FP score.

Individualized templates reports higher accuracy than generalized templates, in both classifier and temporal F_1, underscoring that intra-subject variability is easier to handle than inter-subject variability.

The temporal accuracy is lower than the classifier accuracy, and is due to the temporal localization component. The simple conversion approach used generates algorithmic segments that are wider or narrower than the manual segments, thus leading to a poorer result, even though the algorithmic segments approximately overlap the manual segments. The improved performance between $t_{err} = 0.2s$ and $0.3s$ shows that many algorithmic segments sit just outside the boundary of the manual segments, and are flagged as FN instead of TP. In several instances, the manual segments are delayed due to the reaction speed in the expert performing the labelling, and the segments suggested by the algorithm may be more suitable to denote the actual location of the segment.

The proposed approach outperforms prior work [10], particularly for individualized templates. At $t_{err} = 0.2s$, for individualized templates, the proposed algorithm reports a top temporal F_1 score of 92%, compared to a F_1 score of 85% in [10]. For generalized templates, the proposed algorithm reports a top temporal F_1 score of 80%, compared to a F_1 score of 84% from the prior work [10]. The two approaches achieve similar performance at $t_{err} = 0.3s$, 93% and 95%, respectively. The proposed algorithm requires less parameter tuning and does not require velocity crossings [10] to occur at all segment points, making it easier to deploy, and apply to a wider number of exercises.

V. Conclusion

Time-series segmentation can be reformulated as a two-class classification problem. Using PCA SVM, a dataset consisting of 20 healthy subjects performing 5 rehabilitation exercises was segmented with a F_1 score of 88% when individualized templates are used, and a F_1 score of 82% when generalized templates are used. ANN and *k*-NN also perform well, but *k*-NN suffers from long run-time.

For future work, movements from rehabilitation subjects will be examined, to verify the generalizability of the proposed algorithm to the target population.

VI. Acknowledgements

The authors wish to thank Dr. Mitchell Fergenbaum and Joanna Wong for their advice and feedback regarding rehabilitation and physical therapy, as well as the research participants from the University of Waterloo for their time.

TABLE I

SEGMENTATION RESULTS USING INDIVIDUALIZED TEMPLATES, WHERE THE INPUT VECTOR CONSISTS OF q AND $\dot{q}$, $n_{exp} = 25$ AND $n_{stack} = 15$. THE F_1 ACCURACY FOR TRAINING AND TESTING RESULTS ARE REPORTED. FOR BREVITY, ONLY THE TOP 10 RESULTS IS REPORTED. THE TRAINING TIME ACCOUNTS FOR CLASSIFIER TRAINING TIME, WHILE THE TESTING TIME ACCOUNTS FOR CLASSIFICATION TIME.

	Classifier Parameter			**Training**		**Testing**		**Temporal** F_1		**Time [s]**	
	Dim Reduct	Classifier	Aggregator	$F1_{Seg}$	$F1_{Class}$	$F1_{Seg}$	$F1_{Class}$	$t_{err} = 0.2s$	$t_{err} = 0.3s$	Training	Testing
1	PCA	SVM, radial	None	97%	99%	88%	97%	92%	95%	183	114
2	PCA	SVM, radial	Bagging, 5	96%	98%	88%	97%	92%	95%	217	141
3	PCA	SVM, radial	Bagging, 3	96%	98%	88%	97%	91%	95%	290	143
4	PCA	SVM, radial	Boosting, 5	96%	98%	88%	97%	92%	95%	225	107
5	PCA	ANN, 10-10-10	Bagging, 5	99%	100%	81%	95%	79%	85%	2031	87
6	PCA	ANN, 20-20-20	Bagging, 5	100%	100%	82%	95%	80%	86%	10257	94
7	PCA	SVM, radial	Boosting, 3	96%	98%	88%	97%	92%	95%	149	62
8	PCA	ANN, 10-10	Bagging, 5	99%	100%	81%	95%	78%	85%	10965	114
9	PCA	ANN, 10-10	Bagging, 3	99%	100%	80%	95%	76%	83%	6675	91
10	PCA	*k*-NN, 9	Bagging, 5	98%	99%	88%	97%	91%	96%	8536	8741

TABLE II

SEGMENTATION RESULTS USING GENERALIZED TEMPLATES, WHERE THE INPUT VECTOR CONSISTS OF q AND $\dot{q}$, $n_{exp} = 25$ AND $n_{stack} = 15$. THE F_1 ACCURACY FOR TRAINING AND TESTING RESULTS ARE REPORTED. FOR BREVITY, ONLY THE TOP 10 RESULTS IS REPORTED. THE TRAINING TIME ACCOUNTS FOR CLASSIFIER TRAINING TIME, WHILE THE TESTING TIME ACCOUNTS FOR CLASSIFICATION TIME.

	Classifier Parameter			**Training**		**Testing**		**Temporal** F_1		**Time [s]**	
	Dim Reduct	Classifier	Aggregator	$F1_{Seg}$	$F1_{Class}$	$F1_{Seg}$	$F1_{Class}$	$t_{err} = 0.2s$	$t_{err} = 0.3s$	Training	Testing
1	PCA	SVM, radial	None	94%	97%	82%	95%	80%	87%	98	406
2	PCA	SVM, radial	Bagging, 5	94%	97%	82%	95%	79%	87%	445	1559
3	PCA	SVM, radial	Bagging, 3	94%	97%	81%	95%	79%	87%	274	972
4	PCA	SVM, radial	Boosting, 5	94%	97%	81%	95%	78%	86%	421	1521
5	PCA	ANN, 10-10-10	Bagging, 5	95%	97%	81%	95%	80%	88%	1564	185
6	PCA	ANN, 20-20-20	Bagging, 5	96%	98%	80%	94%	77%	86%	5452	202
7	PCA	SVM, radial	Boosting, 3	94%	97%	82%	95%	78%	87%	764	2259
8	PCA	ANN, 10-10	Bagging, 5	94%	97%	82%	95%	80%	88%	1653	173
9	PCA	ANN, 10-10	Bagging, 3	93%	96%	81%	95%	79%	87%	958	149
10	PCA	*k*-NN, 9	Bagging, 5	94%	97%	81%	94%	78%	85%	3175	23974

REFERENCES

[1] J. F. S. Lin and D. Kulić, "Human pose recovery using wireless inertial measurement units," *Physiol Meas*, vol. 33, pp. 2099–2115, 2012.

[2] S. S. Ng and C. W. Hui-Chan, "The timed up and go test: Its reliability and association with lower-limb impairments and locomotor capacities in people with chronic stroke," *Arch Phys Med Rehabil*, vol. 86, pp. 1641–1647, 2005.

[3] M. G. Pandy, "Computer modeling and simulation of human movement," *Annu Rev Biomed Eng*, vol. 3, pp. 245–273, 2001.

[4] T. Zhang *et al.*, "Imu based single stride identification of humans," in *IEEE Symp Robot Human Interactive Commun*, 2013, pp. 220–225.

[5] G. Guerra-Filho and Y. Aloimonos, "A language for human action," *Computer*, vol. 40, pp. 42–51, 2007.

[6] S. Kozina *et al.*, "Dynamic signal segmentation for activity recognition," in *Proc Int Joint Conf Artificial Intell*, 2011, pp. 15–22.

[7] E. Keogh *et al.*, *Data Mining in Time Series Databases*. World Scientific Publishing, 2004, vol. 57, ch. 1, pp. 1–22.

[8] W. Ilg *et al.*, "On the representation, learning and transfer of spatio-temporal movement characteristics," *Int J Hum Robot*, vol. 1, pp. 613–636, 2004.

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc IEEE*, vol. 77, pp. 257–286, 1989.

[10] J. F. S. Lin and D. Kulić, "On-line segmentation of human motion for automated rehabilitation exercise analysis," *IEEE Trans Neural Syst Rehabil Eng*, vol. 22, pp. 168–180, 2013.

[11] D. Kulić *et al.*, "Online segmentation and clustering from continuous observation of whole body motions," *IEEE Trans Robot*, vol. 25, pp. 1158–1166, 2009.

[12] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min Knowl Discov*, vol. 2, pp. 121–167, 1998.

[13] S. Ekvall *et al.*, "Online task recognition and real-time adaptive assistance for computer-aided machine control," *IEEE Trans Robot*, vol. 22, pp. 1029–1033, 2006.

[14] A. Ataya *et al.*, "Improving activity recognition using temporal coherence," in *Proc Int Conf IEEE Eng Med Biol Soc*, 2013.

[15] L. Lu *et al.*, "Content analysis for audio classification and segmentation," *IEEE Speech Audio Process*, vol. 10, pp. 504–516, 2002.

[16] Z. Liu *et al.*, "Audio feature extraction and analysis for scene segmentation and classification," *J VLSI Signal Process Syst Signal Image Video Technol*, vol. 20, pp. 61–79, 1998.

[17] A. K. Jain *et al.*, "Statistical pattern recognition: a review," *IEEE Trans Pattern Anal Mach Intell*, vol. 22, pp. 4–37, 2000.

[18] M. J. L. Orr, "Introduction to radial basis function networks," Centre for Cognitive Science, University of Edinburgh, Tech. Rep., 1996.

[19] A. K. Jain *et al.*, "Artificial neural networks: A tutorial," *Computer*, vol. 29, pp. 31–44, 1996.

[20] Y. Freund and R. E. Schapire, "A short introduction to boosting," *J Japanese Soc Artificial Intell*, vol. 14, pp. 771–780, 1999.

[21] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[22] A. Burns *et al.*, "Shimmer: A wireless sensor platform for noninvasive biomedical research," *IEEE Sensors J*, vol. 10, pp. 1527–1534, 2010.

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans Intell Syst Technol*, vol. 2, pp. 27:1–27, 2011.

[24] L. Van der Maaten *et al.*, "Dimensionality reduction: A comparative review," *J Mach Learn Res*, vol. 10, pp. 1–41, 2009.

[25] K. Murphy, "Bayes net toolbox for matlab," code.google.com/p/bnt/, 1998.

[26] R. van der Merwe and E. A. Wan, "Recursive bayesian estimation library (ReBEL) and toolkit for MATLAB," choosh.csee.ogi.edu/rebel/, 2006.

[27] C. Seiffert *et al.*, "Resampling or reweighting: A comparison of boosting implementations," in *IEEE Int Conf Tools Artificial Intell*, vol. 1, 2008, pp. 445–451.