2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Human Activity Recognition using Smart Phone Embedded Sensors: A Linear Dynamical Systems Method

Wen Wang, Huaping Liu, Lianzhi Yu, Fuchun Sun

***Abstract*—This paper presents a novel framework of human activity recognition with time series collected from inertial sensors. We model each action sequence with a collection of Linear Dynamic Systems (LDSs), each LDS describing a small patch of the sequence. A codebook is formed by using the K-medoids clustering algorithm and a Bag-of-Systems (BoS) is developed to represent the time series. A great advantage of this method is that the complicated feature design procedure is avoided and the LDSs can well capture the dynamics of the time series. Our experiment validation on public dataset shows the promising results.**

I. INTRODUCTION

In the last decade, human activity recognition has became an important emerging field of research within context-aware systems[1],[2]. Ref. [3] presented a wearable activity sensor system and a systematic activity classification scheme for the classification of human daily physical activities. The wearable activity sensor system, consisting of two activity sensor modules worn on users' dominant hand wrists and ankles, is used for collecting activity acceleration signals. Other similar studies focused on how one can use a variety of accelerometers to identify a range of user activities.

Mobile phones or smart phones are rapidly becoming the central computer and communication device in people's lives. Importantly, today's smart phones are programmable and come with a growing set of cheap powerful embedded sensors, such as accelerometer, digital compass, gyroscope, GPS, microphone, and camera, which are enabling the emergence of personal, group, and community scale sensing applications[4]. For the sake of economy and easy to carry, a lot of works fixed attention on the applications that can be built on smart phones. These works include identifying a user's activity level and predicting their energy consumption, detecting a fall and the movements after the fall, and monitoring user activities levels in order to promote health and fitness. Here are some typical examples.

Obesity prevention requires individuals to have healthy eating and physical activity awareness in their daily lives. Ref. [5] presented a novel health-aware smart phone system (Health Aware) which utilizes the embedded accelerometer to monitor daily physical activities. The physical activities were categorized into walking steps and running steps during the day. Ref. [6] proposed a system for fall detecting using an embedded tri-axial accelerometer sensor smart phone. If the user falls down, hurts hardly and cannot move himself, the system alerts pre-specified guardian with a message via SMS. Therefore, fallen man can be cared immediately.

Ref. [7] investigated the current directions of activity recognition using inertial sensors, with potential application in the healthcare, wellbeing and sports. He summarized five main steps involved in the activity recognition: preprocessing, segmentation, feature extraction, dimensionality reduction and classification.

The most widely used features to train the classifier can be divided into 3 categories: time-domain features, frequency-domain features and time-frequency domain features. Time-domain features such as mean, variance, root mean square and correlation coefficient , and so on , are directly derived from a raw data segment .The features are basic waveform characteristics and signal statistics. Ref. [8] extracted the mean, standard deviation, energy and correlation; Ref. [9] selected average absolute difference, average resultant acceleration, time between peaks and binned distribution value as the representation of the preprocessed data. Frequency-domain features include discrete FFT coefficient, spectral energy [10], spectral entropy [11]. Time-Frequency domain features are used to investigate both time and frequency characteristics of complex signals and they generally employ wavelet techniques [12].

On the other hand, Linear Dynamical System (LDS) becomes an important tool for modeling time series in engineering, controls and economics as well as the physical and social sciences. In [13], the LDS model has been successfully used for dynamic texture description. In [14], this model was developed for teaching control courses. Very recently, Ref. [15] proposed a Bag-of-Systems (BoS) model to describe complicated dynamic texture. This method was motivated by the popular Bag-of-Words (BoW) model and used un-ordered multiple local LDSs to represent a whole video sequence. Inspired by such works, we regard human activities signal as the output of an intrinsic dynamic system. To give a more complete representation for the human activity time series, we leverage a method which is similar to BoS. We represent time series with a set of LDSs parameters, which are called feature descriptors. These feature descriptors are used to form a codebook and the distribution of all descriptors in this time series over the codebook is used to

All of authors are with Department of Computer Science and Technology, Tsinghua University, State Key Laboratory of Intelligent Technology and Systems, TNLIST, Beijing, China. Wen Wang and Lianzhi Yu are also with School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China. hpliu@tsinghua.edu.cn.

This work is jointly supported by the National Key Project for Basic Research of China (2013CB329403), the National Natural Science Foundation of China (Grants No: 91120011 and 61210013), Tsinghua Self-innovation Project (Grant No: 20111081111) and Tsinghua University Initiative Scientific Research Program (Grant No: 20131089295).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

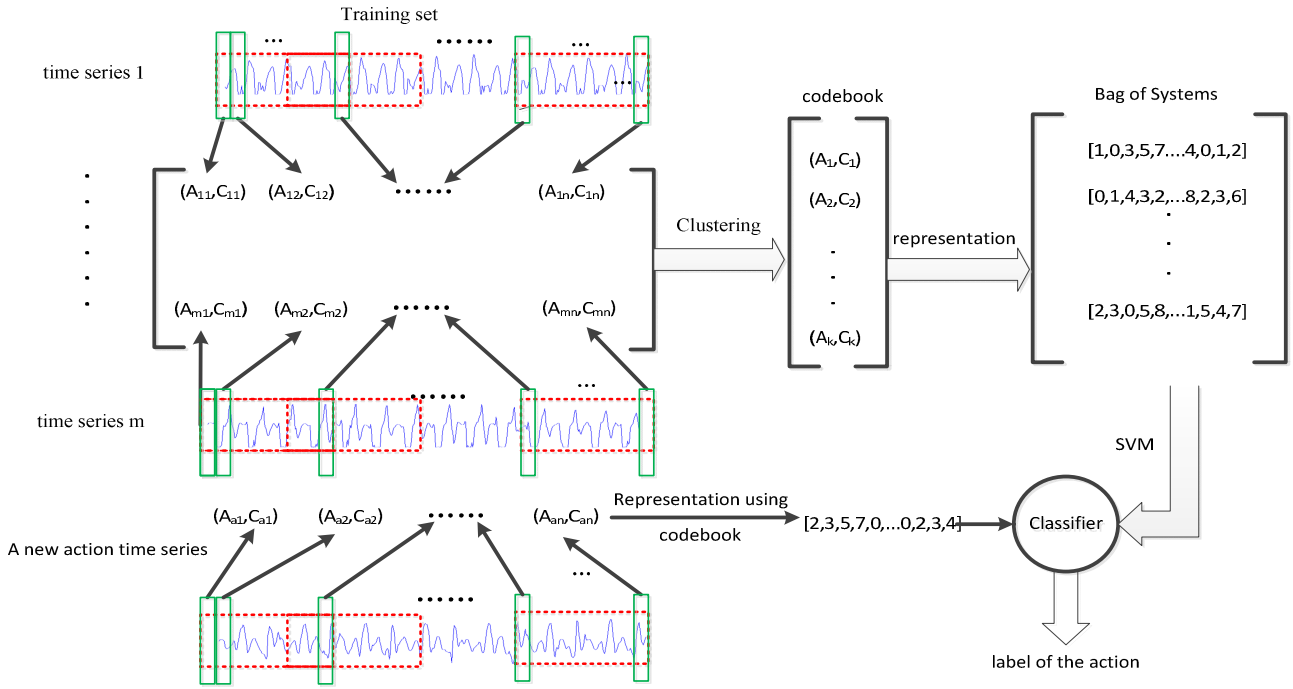


Fig.1. Architecture of our framework

represent the whole time series. New time series are categorized by comparing their distribution to those time series in the training set using traditional classifier.

The main contribution of this paper is that BoS is developed to depict the characteristics of the time-series collected with the smart phone sensor. To the best knowledge of the authors, this is the first time for such a method to be used for human activity recognition. Please note that in [16] the human activity recognition was addressed by k-nearest neighbor method. However, such a method depends on very complicated features, which should be designed by human. In this paper, such a tedious procedure is avoided and we just use the LDSs to represent the whole dynamics of the time-series. In this regard, the proposed method is more principle and requires less feature design. The experiment results show that the proposed method can obtain comparable results with [16].

The rest of this paper is organized as follows. In Section II the overall architecture is illustrated. Section III reviews LDS and the metric for LDSs. In Section IV we categorize time series using the approach of BoS. Section V provides some experimental results. Finally, the conclusion is given in section VI.

II. Architecture

The framework of human activity recognition is inspired by two aspects. The first is the BoF approach to classify time series [17] and the second is the bag of dynamical systems [15] in categorizing dynamic textures. The steps in our framework are as follows:

1. Extract features and corresponding LDS descriptors from the training set.
2. Form codebook using K-medoids clustering algorithm.
3. Represent time series using the formed codebook.
4. Train a classifier using the representation vector and corresponding labels.
5. Given a new time series, infer which class it belongs to using the trained classifier.

Our framework can be illustrated in Fig.1 Note that, in order to illustrate clearly, the time series drawn in Fig.1 are single-dimensional. In fact, they are 9-dimensional in our work. In the training set, subsequences (*red rectangle*) are sampled from each time series and features are extracted from the intervals (*green rectangles*). Features are then grouped into *K* groups and the centers of all groups are selected to form codebook *(middle*). Once the codebook is formed, all of the actions can be represented by the codebook (*right*). A SVM classifier is then trained by the representation vectors and corresponding labels. After the classifier is trained, the recognition can be performed. Given a new action time series, the features are extracted by the same method in the training produce, and represent the time series with the formed codebook. Finally, we can infer which action the time series belongs to using the trained classifier (*bottom*).

III. Brief Review About LDS

A. LDS Representation for Time Series

Assume that a time series $\{y(t)\}_{t=1\ldots\tau}, y(t)\in\Re^m$ is a realization of a second-order stationary stochastic process [13]. This means that the joint statistics between two time instances is shift-invariant.

In our paper, we assume that there exists symmetric positive-definite matrices $Q\in\Re^{n\times n}$ and $R\in\Re^{m\times m}$ such that

$$\begin{cases} x(t+1) = Ax(t) + v(t) \quad v(t) \sim N(0,Q); x(0) = x_0 \\ y(t) = Cx(t) + \omega(t) \quad \omega(t) \sim N(0,R) \end{cases} \tag{1}$$

where $x(t) \in \Re^m$ is the hidden state at time t with initial condition $x(0) = x_0$, $A \in \Re^{n \times n}$ models the dynamics of the hidden state, $C \in \Re^{m \times n}$ maps the hidden state to the output of the system , $v(t)$ and $\omega(t)$ are driven by Gaussian white noise.

It is well known that the choices of matrices of A , C , Q is not unique, but we can find a canonical model realization to represent each equivalence class[18].

Subspace methods calculate LDS parameters by first decomposing a matrix of observations to generate an estimate of the underlying state sequences. The most straightforward technique is singular value decomposition (SVD). $Y = \{y(t)\}_{t=1...\tau}$ denotes the matrix of observations which is the input to SVD. SVD yields $Y \approx U \Sigma V^T$ where $U \in \Re^{m \times n}$ and $V \in \Re^{\tau \times n}$ have orthonormal columns $\{u_i\}$ and $\{v_i\}$, and $\Sigma = diag\{\sigma_1, ..., \sigma_n\}$ contains the singular values. So we get the estimates of C and X where

$$\hat{C} = U \qquad \hat{X} = \Sigma V^T \tag{2}$$

The least squares estimate of A is:

$$\hat{A} = X_{1:\tau} X^{\dagger}_{0:\tau-1} \tag{3}$$

where，† denotes the Moore-Penrose inverse.

Stability is a desirable characteristic for linear dynamical systems, but in the above estimation procedure, the algorithm does not enforce stability. In [19], the author proposed a novel method for learning stable dynamical systems, the authors formulate an approximation of the problem as a convex program, start with a solution to a relaxed version of the program, and incrementally add constraints to improve stability. Rather than continuing to generate constraints until reach a feasible solution, the authors test stability at each step, because the convex program is only an approximation of desired problem, this early stopping rule can yield a higher-quality solution. Readers can refer [19] for more details.

Finally, the noise convariance Q can be estimated from

$$\hat{Q}(\tau) = \frac{1}{\tau - 1} \sum_{t=1}^{t-1} \hat{v}(t) \hat{v}^T(t) \tag{4}$$

where $\hat{v}(t) = \hat{x}(t+1) - \hat{A}\hat{x}(t)$. So far, we have got the LDSs parameters.

B. Martin Distance

To compare two different LDSs, an appropriate distance should be defined. One family of distances between two LDSs is based on the subspace angles between the two systems. The subspace angles are defined as the principal angles between the observability subspaces associated with the two model parameters. In the following we give a brief review about the Martin distance, of which detail can be found in [20] and [21].

Given two LDSs $M_1 = (A_1, C_1)$ and $M_2 = (A_2, C_2)$, we define

$$A = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \in \Re^{2n \times 2n} \tag{5}$$

and

$$C = [C_1, C_2] \in \Re^{2n \times 2n} \tag{6}$$

The observability subspace is the range-space of the extended observability matrix of the LDS defined by

$$O_{\infty}(M) = [C^T, (CA)^T, (CA^2)^T, ...]^T \tag{7}$$

The calculation of the subspace angles between the two models is performed by first solving for Q from the Lyapunov equation $A^T Q A - Q = -C^T C$, where

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \in \Re^{2n \times 2n} \tag{8}$$

The cosine of the subspace angles $\{\theta_i\}_{i=1}^n$ is calculated as

$$\cos^2 \theta_i = eigenvalue(Q_{11}^{-1} Q_{12} Q_{22}^{-1} Q_{21}) \tag{9}$$

Using the subspace angles, the Martin distance between M_1 and M_2 is defined as

$$d(M_1, M_2)^2 = -\log \prod_{i=1}^{n} \cos^2 \theta_i \tag{10}$$

A more detailed explanation about how to calculate the squared cosines of the angles and a more complete mathematical derivation can be found in [20] ,[21].

IV. Categorizing Timer Series Using a Bag of Dynamical Systems

The LDS introduced in the above section can be used to characterize the dynamics of the time series. However, its' representative capability is weak since it is a simple linear model, while practical time series always contain complicated dynamics. To tackle this problem, we extract multiple subsequences from the time series and use the BoS method to contract features for classification. In this section we first introduce the feature extraction, then the codebook design and the time series representation. Finally we give the classification method.

A. Features Extraction

Features are extracted from the raw time series using windows with size of *L*. Feature extraction on windows with *50%* overlap has demonstrated success in previous work [6].The size of *L* is decided by the length of time series. In our experiment, there are 500 sample points, our results show that $L = 100$ works well (described in section V). For each subsequence, we divide it into *d* intervals. So for the training set, totally, *N* features are extracted. As described above, the features are a set of LDSs parameters.

B. Codebook Formation

In the traditional BoF framework, once the features and the corresponding descriptors are extracted, the descriptors are clustered into k groups using clustering algorithm such as K-Means or K-medoids. The centers of the groups are selected to form the codebook. Unfortunately, in our paper, the descriptors are the LDSs parameters, which lie in the non-Euclidean manifold. In order to solve this problem, we used a metric between two LDSs parameters, that is, Martin distance. In section A, we extract N features $F = \{A_i, C_i\}_{i=1}^{N}$ in the training set. Consequently, the distance matrix $D \in \Re^{N \times N}$ can be calculated by Martin's approach. Similar to the K-Means algorithm, the K-medoids algorithm starts by selecting a subset of data points as cluster centers. The remaining points are then grouped based on their distances to these chosen cluster centers. Given the clustering, the cluster centers are updated as the medoid of the data points within each group. A medoid of a set of data points is the data point that minimizes the sum of squared distances to all other data points. Therefore, the medoid is similar to the centroid of the data points, except that it is restricted to be one of the data points. Consequently, after running the K-medoids algorithm on the distance matrix D, we can directly obtain a codebook $W = \{W_1, W_2, ...W_K\} \subset F$

C. Timer Series Representation

Each time series can be represented by the codebook. The simplest representation is called Term Frequency (TF) [15] and is defined as:

$$h_i = \frac{c_i}{\sum_{i=1}^{k} c_i} \quad i = 1, 2, ...k \tag{11}$$

where c_i is the times that codeword W_i "*occurs*" in a time series. Note that, we select the nearest codeword away from the feature descriptor as the "*occurs*" codeword. Obviously, $\sum c_i$ equals the number of features extracted from the series. After the representation procedure, for each time series we can get a probability histogram and a label. Fig.2 shows the process of BoS representation of a time series.

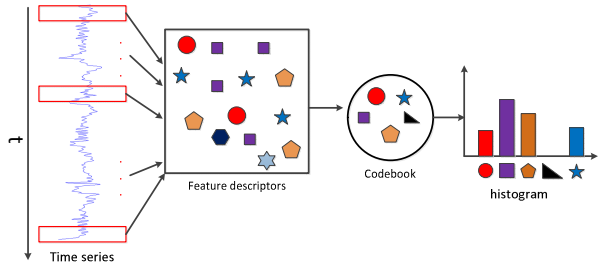


Fig.2 Time Series Representation

D. Classification

The final stage is to design a classifier for recognition. Here the well-known SVM is used to predict the label of time series [22], [23]. Of course some other multi-class methods can also be adopted such as KNN and decision tree. In the case of multiple classes, the training data are labeled $\{h_i, l_i\}_{i=1}^{N}$, where h_i is the feature vector, l_i is the label and N is the number of the training samples.

The linear SVM aims to learn L linear functions $\left\{\omega_c^T h \mid c \in \psi\right\}$, where ω_c is the weight vector. We use a one-versus-all strategy to train L binary linear SVMs, each solving the following unconstrained convex optimization problem

$$\min_{\omega_c} \left\{ J(\omega_c) = \|\omega_c\|_2^2 + C \sum_{i=1}^{n} \ell(\omega_c; l_i^c, h_i) \right\} \tag{12}$$

where C is a parameter which is determined by the cross-validation. A larger C corresponds to the assignment of higher penalties to errors. The SVM constructs a hyperplane to classify the data and the weight vector ω_c serves as the normal vector to the hyperplane.

For multi-class classification, we set $l_i^c = 1$ if $l_i = c$, otherwise, $l_i^c = -1$, while $\ell(\omega_c; l_i^c, h_i)$ is a hinge loss function which is defined as:

$$\ell(\omega_c; l_i^c, h_i) = \left[\max(0, \omega_c^T h \cdot l_i^c - 1) \right]^2 \tag{13}$$

For a test sample feature vector h_{new}, its class label is predicted by

$$l_{new} = \max_{c \in \psi} \omega_c^T h_{new} \tag{14}$$

However, many problems are not linearly separable. For these problems, kernel-based SVMs are often used. In this work the Radial-Basis-Function (RBF) kernel defined as $\kappa(H_1, H_2) = e^{\gamma \times d(H_1, H_2)}$ is adopted, where γ is a free parameter and $d(H_1, H_2)$ is the distance on the space of histogram. The parameters in the RBF SVM classifier are determined by cross-validation.

V. Experiments and Results

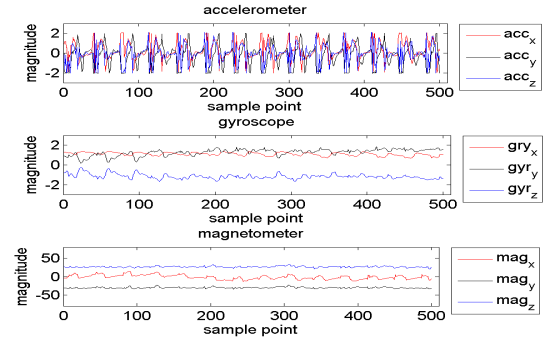
A. Dataset

We utilize a dataset called UCF-iPhone Data set which is provided by the University of Central Florida [16]. There are 9 subjects who were called to bind the phone to their belts on the right hand side. Nine actions (*Bike, Climbing, Descending, Gymbike, Jumping, Running, Standing, Treadmill, and Walking*) are performed and each action is recorded for 5 times using the single *60Hz* IMU built in the phone. Each action sequence was trimmed to 8.33 seconds (*500 sample points*).

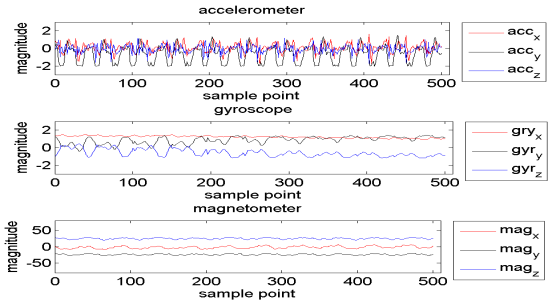
Figs.3 shows several typical time series in the data set. Three inertial sensors are used to collect data: accelerometer, gyroscope and magnetometer. In Figs.3 (a) and (b), the sensor signals change acutely and periodically. However, the signal of *Standing* is unchanged except for some noise.

Intuitively, *Jumping* and *Running* are easy to be confused; *Standing* is easy to be distinguished from *Jumping* and *Running*.

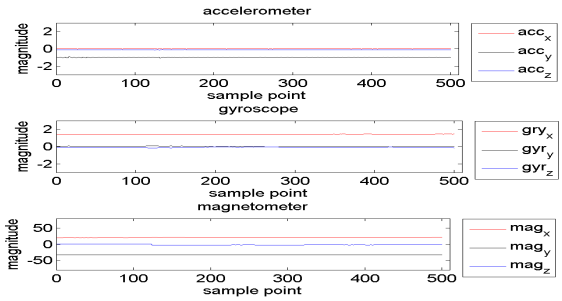
B. Experimental Results



(a) *Jumping*



(b) *Running*



(c) *Standing*

Figs.3. Several typical time series of different actions

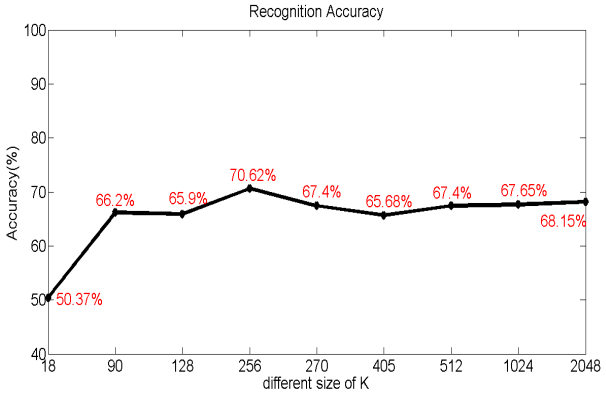


Fig.4. The impact of the number of K on the classification performance.

As described above, the smart phone is bound on the belt of the volunteer. Intuitively, the smart phone should have same movements with the body. In fact, the data collected from the smart phone can not reflect the action completely, especially the strenuous activities such as *Running* and *Jumping*. These uncertainties will increase difficulties for activity recognition.

In our experiment, all of the descriptors in the database are used to form the codebook. In section IV, we have discussed that the size of L is important, so we have to select it carefully. For our data, $L = 100$ works well. We also use the leave-one-subject-out validation test to evaluate the classifiers' ability to recognize unacquainted actions. Classifiers are trained on representation vectors and corresponding labels expect the subject left out of the training data set to test the classifier. The recognition result is given in Fig.4.

Fig.4 shows the performance when K= *18, 90, 128, 256, 270, 405, 512, 1024* and *2048*, respectively. The experimental results indicate that an appropriate K leads to good performance. As K is more than *256*, the performance decreases. The reason is that a large K results too many codebook elements and two similar LDS descriptors may be separated into different clusters, although they are both very close to the boundary. When $K = 256$, our method performs the best. The highest accuracy using the proposed method is about *71%*, which is a littler higher than the results (*68%*) in [16]. Although the performance improvement is not very significant, an important advantage lies in the fact that in the proposed method, no extra feature selection procedure is needed and the LDS naturally characterizes the intrinsic dynamics of the time series, on the contrary in [16], 13 ad-hoc features are designed by the human. In this sense, the proposed method gives a more principle solution to tackle this problem.

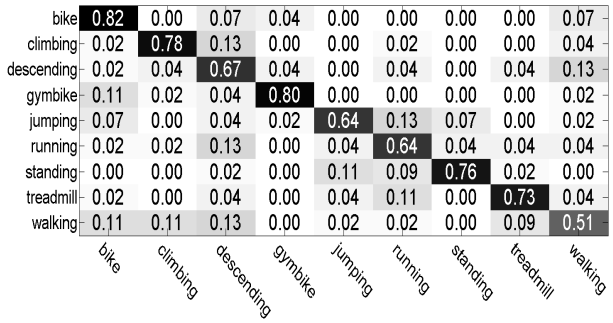


Fig.5. Confusion matrix for best recognition result

In order to find which activities are harder to be recognized relatively, we analyzed the confusion matrix. Fig.5 shows the aggregate confusion matrix for the best overall recognition result (*70.62%*). The confusion matrix gives information about the actual and predicted classifications done by the SVM (RBF kernel) classifier. The labels of abscissa and ordinate are the names of nine actions described in section *A*. The elements on primary diagonal represent the recognition accuracies to different actions, and other elements in *row x* and *column y* show the percentage of action *x* recognized as action *y*. We can see that different actions obtain different recognition accuracies. For example, *Biking* and *Gym bike* obtains higher accuracy (*82%* and *80%*) respectively, but *Walking* obtains the lowest accuracy (*51%*). It is often recognized as *Bike*, *Climbing* and *Descending*. *Jumping* and *Running* are usually confused with each other. This result is reasonable, because the raw signals for *Jumping* and *Running* are indeed similar in Figs.3. In [16], *Climbing* and *Descending* are always confused even used the hierarchical

classifier, but using our method these two actions can be recognized accurately.

To show the advantage of BoS which adopt multiple LDSs to depict a whole time series, we design a baseline approach which adopts one single LDS model to depict the time series. The modeling procedure is same as the description in section III, except that the considered data comes from the whole time series. In this case, SVM classifier cannot be adopted and we use the naïve k-nearest neighbor classifier which utilizes the Martin distance. Table Ⅰ shows the recognition results for k=1, 3, 5, 7, 9 respectively. It is obvious that all of the results are inferior to the BoS method. The main reason is that BoS adopt multiple LDSs and therefore the dynamics of the time series can be well characterized.

TABLE I
CLASSIFICATION ACCURACY WITH DIFFERENT NUMBER OF k

k=1	*k=3*	*k=5*	*k=7*	*k=9*
57.78%	59.26%	61.23%	61.23%	60.74%

VI. CONCLUSION AND OUTLOOK

This paper proposes a novel framework of human activity recognition. The main difference between our method and previous work is that LDSs parameters are selected as features to represent time series. To describe time series more accurately, a codebook is formed by a set of LDSs parameters. Finally, we get a set of distribution vectors. In our paper, we used the instances and labels to train SVM classifier. The average recognition results for 9 different human activities in the UCF-iPhone Data set using the proposed framework can be achieved to about 71%. A great advantage of this method is that the complicated feature design procedure is avoided and the LDSs can well capture the dynamics of the time series. To achieve better recognition results, in future studies, the approach to extract features can be changed to grab more essential features of time series and other faster classifier algorithm such as ELM can be used[24].

REFERENCES

[1] H. Cheng, Z. Liu, Y. Zhao, G. Ye, X. Sun, "Real world activity summary for senior home monitoring," *Multimedia Tools and Applications*, July 2011, pp.1–4.

[2] H. Cheng, Z. Liu, L. Hou, J. Yang, "Sparsity induced similarity measure and its applications," *IEEE Trans. Circuits and Systems for Video Technology*, in press.

[3] F. C. Chuang, Y. T. Yang, and T. P. Kao, "A wearable activity sensor system and its physical activity classification scheme," *in Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Jun. 2012. pp. 1–6.

[4] N. D. Lane, E. Miluzzo, L. Hong, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Trans. Communications Magazine*, vol. 48, pp. 140–150, Sept. 2010.

[5] C. Gao, F. Kong, and J. Tan, "Health aware: Tackling obesity with health aware smart phone systems," in *Proc. IEEE Int. Conf. Robotics and Biomimetics (ROBIO)*, Dec. 2009, pp.1549 –1554.

[6] Z. T. Zhao, Y.Q. Chen, and J.F. Liu "Fall detecting and alarming based on mobile phone," in *Proc. 7th Int. Conf. Ubiquitous Intelligence & Computing and 7th Int. Conf. Autonomic & Trusted Computing(UIC/ATC)*, Oct. 2010, pp.494–497.

[7] A. Akin, B. Stephan, M. Mihai, M. Raluca, and H.Paul, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," in *Proc. 23rd Int. Conf. Architecture of Computing Systems (ARCS)*, Feb. 2010, pp. 1–10.

[8] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Proc. 17th National Conf. Artificial Intelligence*, vol. 20, 2005, pp. 1541–1546.

[9] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," in *Proc. 4th International Workshop on Knowledge Discovery from Sensor Data*, vol. 12, no. 2, Dec. 2010, pp. 74–82.

[10] T.Huynh, and B.Schiele, "Analyzing features for activity recognition," in *Proc. Joint Conf. Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, 2005, pp. 159–163.

[11] S. Wang, J. Yang, N. Chen, X. Chen, and Q, Zhang, "Human activity recognition with user-free accelerometers in the sensor networks," in *Proc. IEEE Int. Conf. Neural Networks and Brain*, vol.2, 2005, pp. 1212–1217.

[12] J. Mantyjarvi, J. Himberg, and T. Seppanen, "Recognizing human motion with multiple acceleration sensors," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 2, Oct. 2001, pp. 747–752.

[13] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, Feb. 2003.

[14] H. Liu, W. Xiao, H. Zhao, F. Sun, "Learning and understanding system stability using illustrative dynamic texture examples," *IEEE Trans. Education*, vol.57, no.1, pp.4–11, Feb. 2014.

[15] R. Vidal, R. Chaudhry, and R. Vidal, "Categorizing dynamic textures using a bag of dynamical systems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 342–353, Feb. 2013.

[16] C. McCall, K. Reddy and M. Shah," Macro-class selection for hierarchical K-NN classification of inertial sensor data," in *Proc. 2nd Int. Conf. Pervasive and Embedded Computing and Communication Systems*, PECCS 2012, Feb. 2012, pp. 106–114.

[17] M. G. Baydogan, G. Runger, E. Tuv, "A bag-of-features framework to classify time series," *IEEE Trans. Pattern Analysis and Machine* Intelligence, vol. 35, no. 11, pp. 2796–2802, Nov. 2013.

[18] P. Saisan, G. Doretto, Y. Wu, S. Soatto, "Dynamic texture recognition," in *Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol.2, 2001, pp. 58–63.

[19] S. M. Siddiqi, B. Boots, and G. J. Gordon, "A constraint generation approach to learning stable linear dynamical systems," in *Proc. Int. Conf. Neural Information Processing Systems*, Dec. 2007, pp. 1329–1336.

[20] K. D. Cock, and B. D. Moor, "Subspace angles between ARMA models," *Systems & Control Letters*, vol. 46, no. 4, pp. 265–270, Jul. 2002.

[21] R. J. Martin. "A metric for ARMA processes," *IEEE Trans. Signal Processing*, vol. 48, no. 4, pp. 1164–1170, Apr. 2000.

[22] J. Yang, K. Yu, Y. Gong, T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 1794–1801.

[23] H. Liu, Y. Liu, F. Sun, "Traffic sign recognition using group sparse coding," *Information Science*, vol.266, pp.75–89, 2014.

[24] G.Huang, Q. Zhu, CK. Siew. "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.