Depth-Based Human Fall Detection via Shape Features and Improved Extreme Learning Machine

Xin Ma, *Member, IEEE*, Haibo Wang, Bingxia Xue, Mingang Zhou, Bing Ji, and Yibin Li, *Member, IEEE*

***Abstract*—Falls are one of the major causes leading to injury of elderly people. Using wearable devices for fall detection has a high cost and may cause inconvenience to the daily lives of the elderly. In this paper, we present an automated fall detection approach that requires only a low-cost depth camera. Our approach combines two computer vision techniques—shape-based fall characterization and a learning-based classifier to distinguish falls from other daily actions. Given a fall video clip, we extract curvature scale space (CSS) features of human silhouettes at each frame and represent the action by a bag of CSS words (BoCSS). Then, we utilize the extreme learning machine (ELM) classifier to identify the BoCSS representation of a fall from those of other actions. In order to eliminate the sensitivity of ELM to its hyperparameters, we present a variable-length particle swarm optimization algorithm to optimize the number of hidden neurons, corresponding input weights, and biases of ELM. Using a low-cost Kinect depth camera, we build an action dataset that consists of six types of actions (falling, bending, sitting, squatting, walking, and lying) from ten subjects. Experimenting with the dataset shows that our approach can achieve up to 91.15% sensitivity, 77.14% specificity, and 86.83% accuracy. On a public dataset, our approach performs comparably to state-of-the-art fall detection methods that need multiple cameras.**

***Index Terms*—Curvature scale space (CSS), extreme learning machine (ELM), fall detection, particle swarm optimization, shape contour.**

I. Introduction

AGING has become a worldwide issue that significantly raises healthcare expenditure. As the World Health Organization has reported [1], about 28–35% of people who are 65 and older fall each year, and the percentage goes up to 32–42% for people 70+. Since more and more elderly people are living alone, automated fall detection becomes a useful technique for prompt fall alarm.

There have existed many device-based solutions to automatic fall detection [2], [3], such as those using wearable accelerometers or gyroscopes embedded in garments [4], [5]. Although a wearable device is insensitive to environment, wearing it for a long time will cause inconvenience to one's daily activities. The *smart home* solution [6], [7] attempted to unobtrusively detect falls without using wearable sensors. They install ambient devices, such as vibration, sound sensors, infrared motion detectors, and pressure sensors, at multiple positions of a room to record daily activities. By fusing the information of these sensors, fall can be detected and alarmed with a high accuracy [8]–[11]. However, using multiple sensors significantly raises the cost of the solution. Moreover, putting many sensors in a room will bring side effect on the health of the elderly people.

Recent advances in computer vision have made it possible to automatically detect human fall with a commodity camera [12]. Such a camera has a low price, and when installed remotely, would not disturb the normal life of the detected person. As compared to other sensing modalities, camera can provide richer semantic information about the person as well as his/her surrounding environment. Therefore, it is possible to simultaneously extract multiple visual cues, including human location, gait pattern, walking speed, and posture, from a single camera. Vision-based fall detection requires a distinctive feature representation to characterize fall action and a classification model to distinguish fall from other daily activities. Although there have been many attempts, vision-based fall detection remains an open problem.

Human shape is the first clue one would explore for fall detection. Human fall can be identified by analyzing human shape changes over a short period. Existing shape-based fall detectors approximate human silhouettes by a regular bounding box or an ellipse, and extract geometric attributes such as aspect ratio, orientation [13], [14], or edge points [15] as representative fall attributes. However, the accuracy of these methods is limited by the proximity of the shape attributes. Fall action lasts relatively short as compared to other actions [16]. Therefore, motion analysis has also been used for fall detection, such as those encoding motion contour in the motion energy image [17] or in the integrated spatiotemporal energy map [18]. However, fall of elderly people might last longer than that of young people. Therefore, different motion models are often required to study the fall behaviors of young and elderly people.

3-D shape information is more robust to viewpoint and partial occlusion than 2-D silhouette. Using a multicamera system, 3-D volume distributions are extracted for fall detection in [19]. If a majority of the distribution goes close to the floor within a short time, fall alarm will be triggered. Then in [20], centroid and orientation of person voxel are computed for fall detection. However, reconstructing 3-D human shape with a multicamera

Manuscript received July 13, 2013; revised November 22, 2013 and January 25, 2014; accepted January 29, 2014. Date of publication February 3, 2014; date of current version November 3, 2014. This work was supported in part by National Natural Science Foundation of China under Grants 61240052, 61203279, and 61233014, in part by the Natural Science Foundation of Shandong Province, China (No. ZR2012FM036), and in part by the Independent Innovation Foundation of Shandong University, China (No. 2012JC005). X. Ma and H. Wang contributed equally to this work. X. Ma and Y. Li are corresponding authors of this paper.

The authors are with the School of Control Science and Engineering, Shandong University, Jinan, Shandong, China (e-mail: maxin@sdu.edu.cn; hbwang1427@gmail.com; xuebingxia0811@163.com; succzhou@163.com; b.ji@sdu.edu.cn; liyb@sdu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JBHI.2014.2304357

system is computationally demanding, which offsets the gain of using 3-D shape. The recently emerged low-cost depth camera—Microsoft Kinect—makes it readily feasible to use 3-D depth information with no computational cost. The depth image acquired from Kinect encodes the distance from a visible subject to the Kinect sensor. By means of infrared light sensing, Kinect is invariant to visible lights and works day and night. Moreover, the depth image well masks identity information of the person being detected. With the usage of depth maps from Kinect, 3-D skeleton [21], [22], 3-D head detection [23], and 3-D gait information [24] can all be readily extracted for action recognition. Unfortunately, to our best knowledge, there are still no attempts that particularly use Kinect to detect human falls.

State-of-the-art classification models have been widely explored for fall detection. A fuzzy neural network classifier was used in [25] for recognizing *standing, bending, sitting*, and *lying*. Fall action is reported if body posture changes from *standing* to *lying* in a short period. However, the performance of neural network is limited by the lack of large-scale human fall dataset. In [26], the *k*-nearest neighbor classifier is used for fall detection, with simple silhouette features. Fuzzy logic has been also used to recognize static postures (lying, squatting, sitting, and standing) [27], and a 74.29% recognition accuracy is reported in their experiments. The computationally efficient support vector machine (SVM) classifier has also been used. Using binary SVM to classify fall from normal activities can yield a 4.8% error rate [16]. By fusing shape and context information, a directed acyclic graph SVM [28] can achieve an upto 97.08% fall detection accuracy and a 0.8% false detection rate in a simulated home environment. Since their datasets and used features are different, it is difficult to accurately rank the performances of these classifiers.

The recently developed extreme learning machine (ELM) [29] has a comparable performance to SVM yet it: 1) is extremely fast to compute and 2) has the capacity to deal with highly noisy data. When using ELM for fall classification, variations of pose, viewpoint, and background can be potentially well handled. Moreover, ELM has a closed-form solution, not suffering from the common issues of gradient-based optimization such as low learning rate and local minima. The drawback of ELM lies in its sensitivity to input parameters, including the number of hidden neurons, input weights, and hidden layer bias values [30].

In this paper, we present a new vision-based approach for automated fall detection in an indoor environment. The approach relies on a Kinect depth camera to extract human silhouette information. We intentionally do not use color images in order to avoid uncovering the privacy of detected person. Then we extract curvature scale space (CSS) features [31], [32] of the extracted silhouette at each frame. A Bag of words model (BoW) [33] built upon the CSS features (BoCSS) is finally used to characterize fall action. The CSS feature is invariant to translation, rotation, scaling, and local shape deformation. On the other hand, BoCSS is invariant to the length of fall action. Therefore, our approach is able to detect both rapid and slow falls. After that, we use the ELM model to classify falls from other daily activities. In order to eliminate the sensitivity of ELM to the number of hidden neurons, input weights, and bias values, we propose a variable-length particle swarm optimization (VPSO) algorithm to simultaneously optimize the three parameters. For clarity, we term the parameter-optimized ELM as VPSO–ELM. To validate the efficiency of VPSO–ELM, we build an action dataset using a Kinect camera. Since it is difficult to capture real falls of elderly people, ten young men and women subjects intentionally fall down. We also capture five other types of actions (walking, sitting, squatting, bending, and lying) for the evaluation purpose. In short, main contributions of the paper include:

1) A novel shape-based fall representation approach, which is invariant to human translation, rotation, scaling, and action length.
2) A novel classifier—VPSO–ELM, which uses particle swarm to optimize the hyperparameters of ELM classifier so as to improve its generalization ability.
3) A dataset including six activity categories of ten subjects (including falling, bending, sitting, squatting, walking, and lying) for public fall detection benchmarking.

The rest of the paper is organized as follows. In Section II, we present details of the proposed fall detection approach. In Section III, we describe dataset and experimental settings. Following that, Section IV provides experimental results and discussions. Finally, we draw conclusions in Section V.

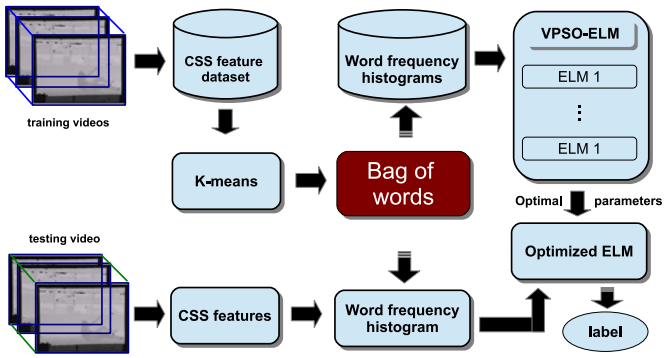


Fig. 1. Workflow of the proposed fall detection approach. CSS feature is extracted at each video frame. BoW is then applied to represent video action. At training time, the ELM classifier is learned with the optimal parameters found by VPSO.

II. Proposed Method

Fig. 1 shows workflow of the proposed fall detection approach. Briefly speaking, the approach involves three main modules. First, CSS feature is extracted at each frame. Second, BoW model is built to represent video action. Third, VPSO optimizes the hyperparameters of ELM. In the following, we describe details of the three modules.

A. Silhouette Extraction From Depth Image

The first step is to extract human silhouette from depth image. We first applied the adaptive Gaussian mixture model (GMM) [34] to segment human from background. Since Kinect is robust against visible lights, the GMM model is only smoothly updated

over time. Following the segmentation, we detect silhouette using the Canny edge detector [35].

B. Extracting Curvature Scale Space Features

The second step is to find a distinctive and compact feature to represent the detected human silhouette. We use the CSS-based representation [31], [32], due to its robustness to translation, rotation, scaling, and local deformation. To build the CSS feature, we convolve the extracted silhouette with a set of Gaussian functions. Given a planar curve $\Gamma(x, y)$ with (x, y) at Cartesian coordinates, first we reparameterize it by its arc length u: $\Gamma(u) = (x(u), y(u))$. Then, we convolve $\Gamma(u)$ with a set of Gaussian functions $\{g(u, \sigma); \sigma = 1, 2, \ldots, n\}$: $X(u, \sigma) = x(u) * g(u, \sigma)$, $Y(u, \sigma) = y(u) * g(u, \sigma)$, to obtain a set of convolved curves $\{\Gamma_\sigma = (X(u, \sigma), Y(u, \sigma))\}$. Here, $*$ indicates the convolution operator. As σ increases, the ordered sequence of the convolved curves $\{\Gamma_\sigma\}$, which is also referred to as the evolution of Γ, will be generated. The curvature κ of each Γ_σ is defined as

$$\kappa(u, \sigma) = \frac{X_u(u, \sigma)Y_{uu}(u, \sigma) - X_{uu}(u, \sigma)Y_u(u, \sigma)}{(X_u^2(u, \sigma) + Y_u^2(u, \sigma))^{3/2}} \quad (1)$$

where

$$X_u(u, \sigma) = x(u) * g_u(u, \sigma)$$
$$Y_u(u, \sigma) = y(u) * g_u(u, \sigma)$$
$$X_{uu}(u, \sigma) = x(u) * g_{uu}(u, \sigma)$$
$$Y_{uu}(u, \sigma) = y(u) * g_{uu}(u, \sigma)$$

g_u and g_{uu} are the first and second derivatives of $g(u, \sigma)$ with respect to u, respectively.

The CSS image of Γ is defined at $\kappa(u, \sigma) = 0$, the zero-crossing (ZP) point of all Γ. There are two types of ZP: ZP_+ is the start point of a concavity arc where $\kappa(u, \sigma)$ changes from negative to positive, and ZP_- is the start point of a convexity arc where $\kappa(u, \sigma)$ changes from positive to negative. On a closed curve, ZP_+ and ZP_- always appear in a pair. The arc between a pair of ZP_+ and ZP_- is either concave (ZP_+, ZP_-) or convex (ZP_-, ZP_+). Since it is extracted from the curvatures at multiple scales, ZP is invariant to rotation, translation, and uniform scaling. To make it further robust against noise and local deformation, we resample the CSS features by curve interpolation. Fig. 2 demonstrates how to create CSS features. During the curve evolution, the σ value keeps increasing until Γ_σ shrinks to a circle-like convex contour where all ZPs disappear. On a CSS image, the (u, σ) coordinates of all ZPs form a set of continuous curves. The maxima point of each curve is most distinctive. Therefore, the (u, σ) coordinates of all maxima points are used as our CSS feature vector. Note that u is length-normalized.

C. Action Representation via BoW

By extracting CSS features at each frame, a video clip is transformed to be a collection of framewise CSS features. To represent an action with their CSS features, we use the BoW model [33]. In the first stage of BoW, *K*-means clustering is applied over all feature vectors to generate a codebook. Each cluster center is a codeword, as a representative of similar feature vectors. Then, by mapping the collected vectors of a video clip to the codebook, we have a histogram of occurrence counts of the words, which is the BoW representation of video action. Since each CSS feature is a 2-D vector, generating the BoW model is real-time. The K value is critical for the *K*-means clustering. As shown in Fig. 4(c), we empirically find that $K = 40$ produces the best fall detection performance.

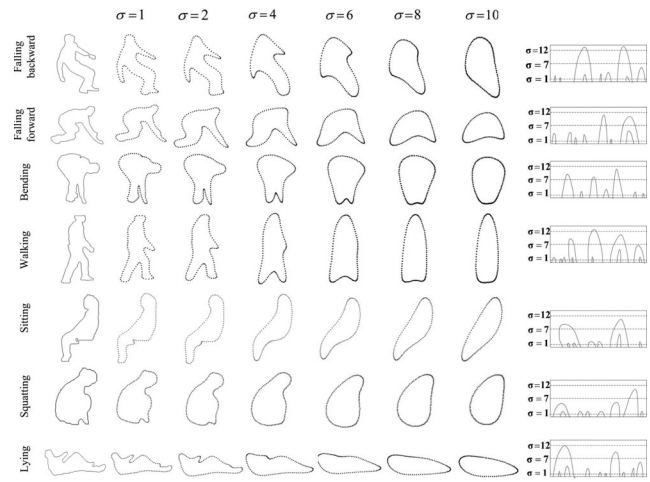


Fig. 2. Illustration of extracting CSS features. The left most column shows the extracted silhouettes of various actions. As σ increases from 0 to 10, the silhouettes shrink and eventually evolve to circle-like shapes. Zero-crossing points on the evolved curves constitute a CSS image (right most column).

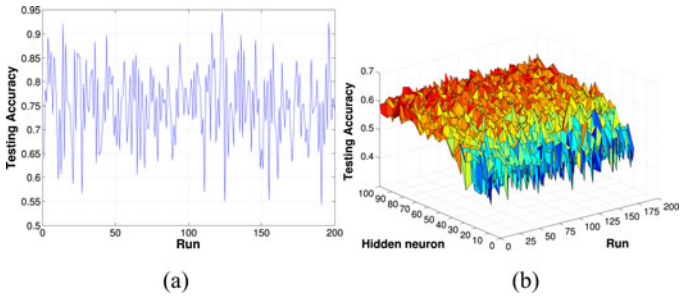


Fig. 3. (a) Testing ELM with 20 hidden neurons, random input weights and biases over 200 runs. (b) Testing ELM as varying hidden neurons from 20 to 100 over 200 runs. Details about the experiments can be found in Section III.

D. VPSO–ELM for Fall Detection

ELM [29] is a single-hidden-layer feedforward neural network. Given a set of samples $\{\mathbf{x}_j\}$ and their labels $\{\mathbf{t}_j\}$, ELM having L hidden neurons is modeled by

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{\omega}_i \cdot \mathbf{x}_j + \mathbf{b}_i) = \mathbf{t}_j, j = 1, \ldots N \quad (2)$$

where $g(x)$ is the activation function of ELM. $\boldsymbol{\omega}_i$, $\mathbf{b}_i$, and $\boldsymbol{\beta}_i$ represent input weights, biases, and output weights of the ith hidden neuron, respectively. Rewriting (2) in a matrix form yields $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, where $\mathbf{T} = [\mathbf{t}_1^T, \ldots, \mathbf{t}_N^T]^T$, and $\mathbf{H}$ is the hidden

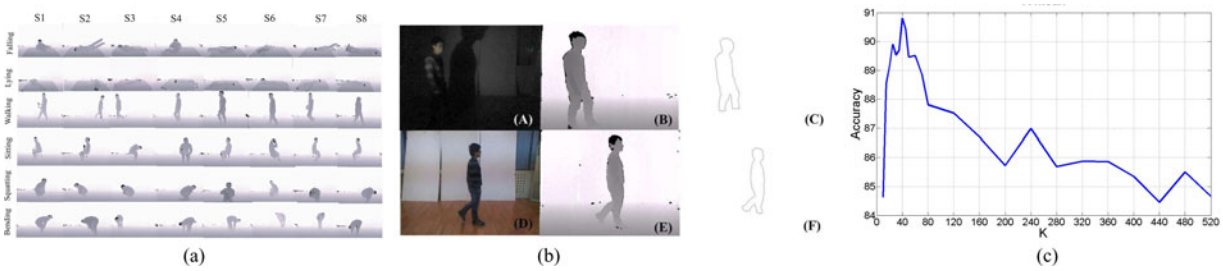


Fig. 4. (a) SDUFall Dataset. Shown here are six action categories (rows) from eight subjects (columns): falling, lying, walking, sitting, squatting, and bending. (b) The insensitivity of silhouette extraction to illumination. Whether light is turned off (a) or on (d), human silhouette [(b) and (e)] is clearly visible since Kinect is insensitive to visible lights. Therefore, as (c) and (f) shows, silhouettes can be successfully extracted in both situations. (c) The choice of K for K-means clustering. The highest fall detection accuracy is obtained when K equals to 40.

layer output matrix,

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{\omega}_1 \mathbf{x}_1 + \mathbf{b}_1) & \cdots & g(\boldsymbol{\omega}_L \mathbf{x}_1 + \mathbf{b}_L) \\ \vdots & \cdots & \vdots \\ g(\boldsymbol{\omega}_1 \mathbf{x}_N + \mathbf{b}_1) & \cdots & g(\boldsymbol{\omega}_L \mathbf{x}_N + \mathbf{b}_L) \end{bmatrix}_{N \times L}. \quad (3)$$

The ith column of $\mathbf{H}$ is the output of the ith hidden neuron with respect to the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$. By assigning random values to $\{\boldsymbol{\omega}_i\}$ and $\{\mathbf{b}_i\}$, ELM optimizes $\boldsymbol{\beta}$ via least squares.

As shown in Fig. 3(a), the input weights and biases dramatically affect the performance of ELM (Details about the experiments can be found in Section III). Moreover, as Fig. 3(b) shows, the accuracy of ELM classifier reaches maximum with 60–70 hidden neurons. Less neurons are inadequate to deal with sample variations while more leads to overfitting. These experiments motivate us to use particle swarm optimization to optimize the ELM parameters.

1) *PSO–ELM*: Before introducing VPSO–ELM, we first introduce PSO–ELM, which uses standard particle swarm optimization (PSO) to optimize the parameters of ELM. PSO was introduced to simulate the social behaviors of bird flocking on stochastic optimization [36]. A PSO model consists of a swarm of particles, each of which is encoded by a variable vector $\mathbf{p}_i = (p_{i1}, \ldots, p_{id})$, and its velocity $\mathbf{v}_i = (v_{i1}, \ldots, v_{id})$. PSO randomly initializes each particle and iteratively updates their values in terms of optimizing a fitness function f. For ELM optimization, f is referred to as the validation accuracy of ELM. At each iteration, each particle knows its own best position $\mathbf{p}_i^*$ as well as the position of the optimal particle $\mathbf{p}_g^*$ in the swarm. The update strategy to $\mathbf{p}_i$ and $\mathbf{v}_i$ is

$$\begin{aligned} \mathbf{v}_i(t+1) &= \omega \mathbf{v}_i(t) + c_1 r_1 (\mathbf{p}_i^*(t) - \mathbf{p}_i(t)) \\ &\quad + c_2 r_2 (\mathbf{p}_g^*(t) - \mathbf{p}_i(t)) \\ \mathbf{p}_i(t+1) &= \mathbf{p}_i(t) + \mathbf{v}_i(t+1) \end{aligned} \quad (4)$$

where ω is an inertia factor, $r_1 \in [0,1]$ and $r_2 \in [0,1]$ are used to maintain the diversity of the population. $c_1 > 0$ is the coefficient of the self-recognition component, while $c_2 > 0$ is the coefficient of the social component. In (4), a particle can decide where to move, based on its own best position as well as the experience of the optimal particle in the swarm. The inertia factor ω is determined by

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{\text{iter}_{\max}} \times t$$

where $\omega_{\max}$ and $\omega_{\min}$ is the upper and lower bounds of ω, respectively. $\text{iter}_{\max}$ is the maximally allowed iteration number and t is the current iteration count. Once $\mathbf{p}_i$ is updated, its own best position $\mathbf{p}_i^*$ is updated according to

$$\mathbf{p}_i^*(t+1) = \begin{cases} \mathbf{p}_i^*(t) & \text{if } f(\mathbf{p}_i(t+1)) \le f(\mathbf{p}_i^*(t)) \\ \mathbf{p}_i(t+1) & \text{if } f(\mathbf{p}_i(t+1)) \ge f(\mathbf{p}_i^*(t)). \end{cases}$$

After then, the globally optimal particle $\mathbf{p}_g^*$ is found by

$$\mathbf{p}_g^* = \arg\max_{\mathbf{p}_i^*} (f(\mathbf{p}_1^*), f(\mathbf{p}_2^*), \ldots, f(\mathbf{p}_m^*)). \quad (5)$$

By treating an ELM as a particle, with its input weights and biases as the unknowns, the particle variable $\mathbf{p}$ can be expressed as a matrix

$$\mathbf{p} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} & b_1 \\ w_{21} & w_{22} & \cdots & w_{2n} & b_2 \\ & & \cdots & & \\ w_{L1} & w_{L2} & \cdots & w_{Ln} & b_L \end{bmatrix}_{L \times (n+1)} \quad (6)$$

where L is the number of hidden neurons, $(w_{i1} \;\; w_{i2} \cdots w_{in} \;\; b_i)$ denote the input weights and biases of ith hidden neuron, $i = 1, \ldots, L$. n is the feature dimension. The optimal values of input weights and biases correspond to the position of the globally optimal particle.

Note that in (4) and (6), each particle (an ELM classifier) is required to have the same size, i.e., the number of hidden neurons for each ELM classifier must be the same. It follows that the optimal number of hidden neurons in PSO–ELM must be determined before optimizing the input weights and biases of ELM. In the following, we introduce a variable-length PSO–ELM that allows the particles in PSO–ELM having different lengths, from which we then find the optimal particle length.

2) *VPSO–ELM:* To determine the optimal number of hidden neurons in PSO, we propose variable-length PSO (VPSO), which allows particles to have different lengths. For the purpose, as opposed to (4), we define new update equations that can handle particles with different sizes. In VPSO, before its update, each particle in the swarm is first compared

against the globally optimal particle. If the size of the particle is different to that of the optimal particle, the particle will be updated according to a new update strategy; otherwise, the algorithm moves to next particle. Here are the details:

a) If the row number of the ith particle $\mathbf{p}_i(t)$, nr_i, is larger than that of the best particle $\mathbf{p}_g^*(t)$, nr_g, we randomly select nr_g rows from the nr_i rows of $\mathbf{p}_i(t)$, $\mathbf{v}_i(t)$, and $\mathbf{p}_i^*(t)$, respectively, and set

$$\mathbf{p}_i'(t) = \mathbf{p}_i(t)\left(\frac{nr_g}{nr_i}\right), \mathbf{v}_i'(t) = \mathbf{v}_i(t)\left(\frac{nr_g}{nr_i}\right)$$
$$\mathbf{p}_i'^*(t) = \mathbf{p}_i^*(t)\left(\frac{nr_g}{nr_i}\right). \quad (7)$$

Afterwards, $\mathbf{p}_i'(t)$, $\mathbf{v}_i'(t)$, and $\mathbf{p}_i'^*(t)$ are updated according to (4). For the unselected entries, $\overline{\mathbf{p}_i'(t)}$, $\overline{\mathbf{v}_i'(t)}$, and $\overline{\mathbf{p}_i'^*(t)}$, we update them according to

$$\overline{\mathbf{v}_i'(t+1)} = \omega\overline{\mathbf{v}_i'(t)} + c_1 r_1(\overline{\mathbf{p}_i'^*(t)} - \overline{\mathbf{p}_i'(t)}),$$
$$\overline{\mathbf{p}_i'(t+1)} = \overline{\mathbf{p}_i'(t)} + \overline{\mathbf{v}_i'(t+1)}. \quad (8)$$

Finally, we restack $\mathbf{p}_i'(t+1)$ and $\overline{\mathbf{p}_i'(t+1)}$, and $\mathbf{v}_i'(t+1)$ and $\overline{\mathbf{v}_i'(t+1)}$. Such an update scheme guarantees that particle size remains unchanged during the swarm evolution.

b) If nr_i, is smaller than nr_g, we first randomly select nr_i rows from the nr_g rows of $\mathbf{p}_g^*(t)$, set

$$\mathbf{p}_{gi}^*(t) = \mathbf{p}_g^*(t)\left(\frac{nr_i}{nr_g}\right) \quad (9)$$

and then update $\mathbf{p}_i(t)$ and $\mathbf{v}_i(t)$ according to (4).

As a summary, VPSO–ELM proceeds as follows:

1) Create an initial population with each particle as an ELM; the number of hidden neurons for each particle is randomly selected from [20, 100];
2) Calculate the fitness function for each particle;
3) Generate a new population using the update strategy described in (7)–(9);
4) If termination criterion is satisfied, stop and output the best weights, biases, and number of hidden neurons; otherwise, return to step 2.

III. Experiments

A. Dataset

SDUFall Dataset: We built a depth action dataset with a low-cost Kinect camera. The dataset has been released at our project site[1] for public fall detection benchmarking. Ten young men and women subjects participate in the study. Each subject does the following six actions 30 times: falling down, bending, squatting, sitting, lying, and walking. Since it is difficult to capture real falls, the subjects intentionally fall down. The 30 times for each action differ in that each time we randomly change one or more of the following conditions: carrying or not carrying large object, turning the light on or off, changing room layout, changing direction and position relative to the camera. A Kinect camera was installed 1.5 m high to record the actions. A total of $6 \times 10 \times 30 = 1800$ video clips are collected. Video frame is at size of 320×240, recorded at 30 fps in AVI format. The average video length is about 8 s, which is baseline for unit action segmentation. Six action (falling, lying, walking, sitting, squatting, and bending) examples from different subjects are shown in Fig. 4(a).

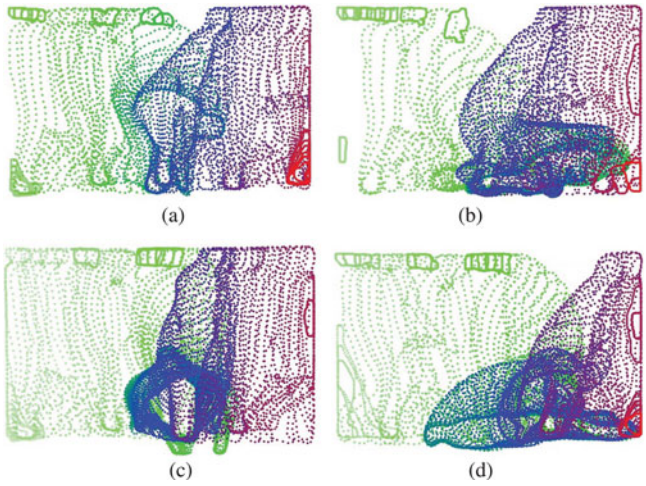


Fig. 5. Motion evolutions of bending, falling forward, sitting, and lying. Regardless of time order, lying is the most confounding activity to fall, which is consistent with the confusion matrices in Fig. 6. (a) Bending. (b) Falling forward. (c) Sitting. (d) Lying.

MultiCam Dataset: The MultiCam dataset was originally used in [20]. In the dataset, a healthy subject performed 24 realistic scenarios. The first 22 scenarios contain a fall and confounding events, the last two ones contain only confounding events (11 crouching position, nine sitting position, and four lying on a sofa position) under an eight-camera configuration. The study area has a size of 7 m $\times$ 4 m, with a table, a chair, and a sofa to simulate a real living room. Video streams have a size of 720×480 and were recorded at 30 fps.

B. Experimental Settings

Feature extraction was implemented in C++ while VPSO–ELM was implemented in MATLAB. All the experiments were conducted with MATLAB 7.10 (R2010a) on a PC with Intel (R) Core (TM) i3-2120 CPU and 2.00 GB RAM.

On SDUFall dataset, we perform six-class classification and then estimate fall-versus-nonfall classification metrics. Using the same BoCSS features, we compare different classifiers: one-versus-all multiclass SVM [37], ELM with random initializations, ELM with PSO optimized weights and biases (PSO–ELM), ELM with VPSO optimized weights, biases, and number of hidden neurons (VPSO–ELM). For SVM, we use a linear kernel with the parameter $C = 1$. The experiment runs 100 times with leave-one-subject-out cross validations. PSO–ELM and VPSO–ELM optimize hyperparameters by further splitting the training dataset into training and validation set with leave-one-subject-out cross validations. On MultiCam dataset, we perform

[1]http://www.sucro.org/homepage/wanghaibo/SDUFall.html

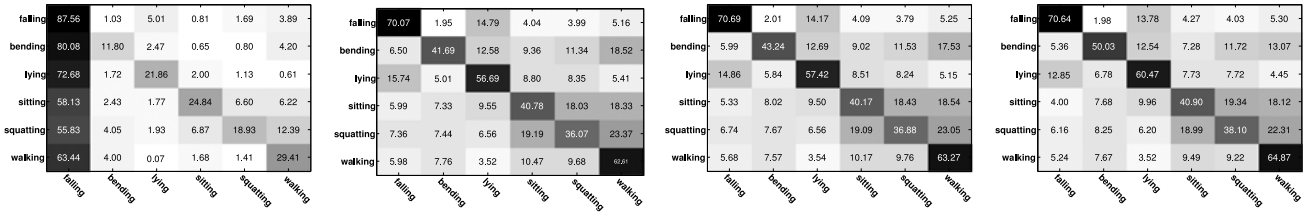


Fig. 6. Confusion matrices of action classification with One-versus-All SVM, ELM, PSO–ELM, and VPSO–ELM, respectively. VPSO–ELM obtains the highest classification accuracies. While One-versus-All SVM is better than VPSO–ELM in fall detection accuracy, it has significantly higher false negatives. Lying is most confounding activity to fall, which we can also see in Fig. 5. Please see the corresponding texts for more analysis. (a) One-versus-All SVM. (b) ELM. (c) PSO–ELM. (d) VPSO–ELM.

fall-versus-nonfall binary classification 100 times with leave-one-out cross validations. On the dataset, we compare the proposed VPSO–ELM approach against the reported results of [19], which needs multiple cameras.

IV. Results and Discussion

A. Influence of Illumination on Silhouette Extraction

Since Kinect uses infrared light for depth sensing, it is insensitive to visible light. This is also illustrated in Fig. 4(b). In the scenario, human silhouettes are successfully extracted even when the room goes fully dark. This is a clear advantage over the methods using color images [19].

B. Assessing Hyperparameters

The number of *K*-means clusters is critical to building the BoW model. In Fig. 4(c), we evaluate the influence of K on fall detection. This was done on the SDUFall dataset. $K = 40$ yields the highest classification accuracy, which indicates that 40 clusters well represent the overall training CSS features. Therefore, $K = 40$ is used throughout our experiments.

The input weights and biases for ELM were randomly initialized from $[-1, +1]$. For each ELM particle, the number of hidden neurons were randomly initialized from $[20, 100]$. VPSO finds the optimal number that is then used for training a final ELM. In our experiments, the parameters of VPSO have no big influence on the results. Therefore, we empirically fix them as the maximum inertia weight $\omega_{\max} = 1.2$, the minimum of inertia weight $\omega_{\min} = 0.73$, the maximum iteration termination criterion $\text{iter}_{\max} = 200$, the coefficient of self-recognition component $c_1 = 1.5$, the coefficient of social component $c_2 = 1.5$, and the population size is 50.

C. Performance on SDUFall Dataset

Fig. 6 shows the classification results on SDUFall. VPSO–ELM has smaller misclassification errors as compared to other methods. The improvement of VPSO–ELM over PSO–ELM reflects the influence of the number of hidden neurons. The One-versus-All SVM is the one more likely to misclassify other activities as fall. This is mainly because multiclass SVM is not as natural a multiclass classifier as ELM. The most confounding activity to fall is lying since about 14% of fall is misclassified as lying. This reflects the similarity of the two activities as also illustrated in Fig. 5. Regardless of time order, the silhouettes of falling appear similar to that of lying. There are also 5.3% of falls miscategorized as walking. This happens because some subject falls down and gets up too quickly.

TABLE I
Fall-Versus-Nonfall Classification Results ($\pm std$) on SDUFall

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)
SVM	63.12±5.57	57.40±5.54	75.78±15.79
ELM	84.36±6.02	89.16±5.68	76.51±9.74
PSO-ELM	85.34±5.41	89.96±4.78	77.07±9.77
VPSO-ELM	**86.83±4.11**	**91.15±3.82**	**77.14±9.25**

TABLE II
Fall-Versus-Nonfall Classification Results ($\pm std$) on MultiCam

Approach	Accuracy (%)	Sensitivity (%)	Specificity (%)
BoCSS+VPSO-ELM	97.20±1.72	99.93±1.02	91.97±4.25
[19] with 3 cameras	N/A	80.6±2.1	100.0±0.0
[19] with 4 cameras	N/A	99.7±0.1	99.8±0.0

TABLE III
Time Cost of Classification on SDUFall

Classifier/Time	SVM	ELM	PSO-ELM	VPSO-ELM
Training time (s)	2.6832	0.0218	62.1925	413.2950
Testing time (s)	2.3868	0.0019	0.0018	0.0024

By treating the five daily activities (sitting, walking, squatting, lying, and bending) as a nonfall category, we are able to measure the fall-versus-nonfall errors. Table I shows such results based on the confusion matrices in Fig. 6. VPSO–ELM is about 2% more accurate than ELM and 1% than PSO–ELM. Similar improvements exist in the sensitivity and specificity measurements. Note that VPSO–ELM goes better but since STDs of accuracy estimation are higher than the differences in accuracy, this result has to be tested in the future with more data/different datasets.

D. Performance on MultiCam Dataset

Table II shows the classification results on MultiCam dataset. As compared to [19], our approach is better than the three-camera solution but worse than the four-camera one. Since we need only a single camera, the cost of our approach is lower than that of [19].

E. Time Cost

On SDUFall dataset, we compared the time costs of SVM, ELM, PSO–ELM, and VPSO–ELM. As shown in Table III, VPSO–ELM needs the longest time to optimize the parameters

TABLE IV
TIME COST ON MULTICAM

Approach	time cost
Our system	30 fps
[19] with 3 cameras, implemented in GPU	16 fps
[19] with 8 cameras, implemented in GPU	10 fps

of ELM. However, since training is only needed one time, this is affordable in practice. Once learned, the running time of ELM, PSO–ELM, and VPSO-ELM have almost no difference. While SVM is the fastest to train, its running time is also the longest. The BoCSS feature building runs at about 30 fps. For a unit action that typically lasts 8 s, our approach needs only about 8 s to recognize it. Table IV compares the time cost of our approach with that of [19] on MultiCam. Our approach is about two times faster than the GPU implementation of [19].

V. CONCLUSION

In this paper, we presented a new vision-based fall detection approach that uses only one low-cost Kinect depth camera. There are two critical components in our approach. First, we represent video action by a bag-of-words model built upon distinctive CSS features of human silhouette. Second, for fall classification, we propose VPSO–ELM that utilizes particle swarm optimization to optimize the model parameters of ELM. By design, VPSO–ELM can choose optimal values for the input weights, biases, and number of hidden neurons to which ELM is sensitive. Extensive experiments on a self-captured dataset shows that fusing CSS and VPSO–ELM can achieve an upto 86.83% fall detection accuracy with a single depth camera.

SDUFall dataset contains five daily activities and intentional falls from ten subjects. In the future, we will acquire more daily activities and increase the number of subjects. Moreover, the study is limited due to the lack of real falls. Since it is difficult to acquire real falls, we will seek to collaborate with nursing homes to capture real falls of elderly people.

REFERENCES

[1] W. H. O. Ageing and L. C. Unit, *WHO global report on falls prevention in older age*. World Health Org., Geneva, Switzerland, 2008.

[2] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013, pp. 1.

[3] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller, "A smart home application to eldercare: Current status and lessons learned," *Technol. Health Care*, vol. 17, no. 3, pp. 183–201, 2009.

[4] T. Shany, S. Redmond, M. Narayanan, and N. Lovell, "Sensors-based wearable systems for monitoring of human movement and falls," *IEEE Sensors J.*, vol. 12, no. 3, pp. 658–670, Mar. 2012.

[5] G. Zhao, Z. Mei, D. Liang, K. Ivanov, Y. Guo, Y. Wang, and L. Wang, "Exploration and implementation of a pre-impact fall recognition method based on an inertial body sensor network," *Sensors*, vol. 12, no. 11, pp. 15338–15355, 2012.

[6] G. Demiris and B. Hensel., "Technologies for an aging society: A systematic review of smart home applications," *Yearb. Med. Inform.*, vol. 32, pp. 33–40, 2008.

[7] N. Suryadevara, A. Gaddam, R. Rayudu, and S. Mukhopadhyay, "Wireless sensors network based safe home to care elderly people: Behaviour detection," *Sensors Actuators A: Phys.*, vol. 186, pp. 277–283, 2012.

[8] A. Ariani, S. Redmond, D. Chang, and N. Lovell, "Simulated unobtrusive falls detection with multiple persons," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 11, pp. 3185–3196, Nov. 2012.

[9] C. Doukas and I. Maglogiannis, "Emergency fall incidents detection in assisted living environments utilizing motion, sound, and visual perceptual components," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 2, pp. 277–289, Mar. 2011.

[10] Y. Li, K. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 5, pp. 1291–1301, May 2012.

[11] Y. Zigel, D. Litvak, and I. Gannot, "A method for automatic fall detection of elderly people using floor vibrations and sound-proof of concept on human mimicking doll falls," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 12, pp. 2858–2867, Dec. 2009.

[12] O. Popoola and K. Wang, "Video-based abnormal human behavior recognition—A review," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev.*, vol. 42, no. 6, pp. 865–878, Nov. 2012.

[13] V. Vishwakarma, C. Mandal, and S. Sural, "Automatic detection of human fall in video," in *Proc. Int. Conf. Pattern Recogn. Mach. Intell.*, 2007, pp. 616–623.

[14] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Fall detection from human shape and motion history using video surveillance," in *Proc. 21st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2007, vol. 2, pp. 875–880.

[15] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 611–622, May 2011.

[16] B. Mirmahboub, S. Samavi, N. Karimi, and S. Shirani, "Automatic monocular system for human fall detection based on variations in silhouette area," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 2, pp. 427–436, Feb. 2013.

[17] H. Qian, Y. Mao, W. Xiang, and Z. Wang, "Recognition of human activities using SVM multi-class classifier," *Pattern Recogn. Lett.*, vol. 31, no. 2, pp. 100–111, 2010.

[18] Y. T. Liao, C.-L. Huang, and S.-C. Hsu, "Slip and fall event detection using Bayesian belief network," *Pattern Recog.*, vol. 45, no. 1, pp. 24–32, 2012.

[19] E. Auvinet, F. Multon, A. Saint-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-D silhouette vertical distribution," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 2, pp. 290–300, Mar. 2011.

[20] M. Yu, S. Naqvi, A. Rhuma, and J. Chambers, "One class boundary method classifiers for application in a video-based fall detection system," *IET Comput. Vision*, vol. 6, no. 2, pp. 90–100, 2012.

[21] R. Planinc and M. Kampel, "Introducing the use of depth data for fall detection," *Personal Ubiquitous Comput.*, vol. 17, no. 6, pp. 1063–1072, 2011.

[22] G. Mastorakis and D. Makris, "Fall detection system using kinect's infrared sensor," *J. Real-Time Image Process.*, pp. 1–12, Mar. 2012.

[23] A. T. Nghiem, E. Auvinet, and J. Meunier, "Head detection using kinect camera and its application to fall detection," in *Proc. 11th Int. Conf. Inf. Sci., Signal Process. Appl.*, 2012, pp. 164–169.

[24] G. Parra-Dominguez, B. Taati, and A. Mihailidis, "3D human motion analysis to detect abnormal events on stairs," in *Proc. Second Int. Conf. 3D Imaging, Modeling, Process., Visualization Transmiss.*, 2012, pp. 97–103.

[25] C.-F. Juang and C.-M. Chang, "Human body posture classification by a neural fuzzy network and home care system application," *IEEE Trans. Syst., Man Cybern., Part A: Syst. Humans*, vol. 37, no. 6, pp. 984–994, Nov. 2007.

[26] C.-L. Liu, C.-H. Lee, and P.-M. Lin, "A fall detection system using k-nearest neighbor classifier," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 7174–7181, 2010.

[27] D. Brulin, Y. Benezeth, and E. Courtial, "Posture recognition based on fuzzy logic for home monitoring of the elderly," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 5, pp. 974–982, Sep. 2012.

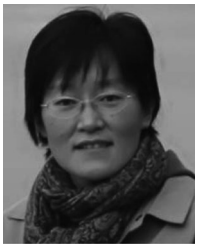
[28] M. Yu, A. Rhuma, S. Naqvi, L. Wang, and J. Chambers, "A posture recognition-based fall detection system for monitoring an elderly person in a smart home environment," *IEEE Trans. Inf. Technol. Biomed.*, vol. 16, no. 6, pp. 1274–1286, Nov. 2012.

[29] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2004, vol. 2, pp. 985–990.

[30] Q. Zhu, A. Qin, P. Suganthan, and G. Huang, "Evolutionary extreme learning machine," *Pattern Recogn.*, vol. 38, no. 10, pp. 1759–1763, 2005.

[31] F. Mokhtarian and A. K. Mackworth, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 8, pp. 789–805, Aug. 1992.

[32] F. Mokhtarian, "Silhouette-based isolated object recognition through curvature scale space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 5, pp. 539–544, May 1995.
[33] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn.*, 2005, vol. 2, pp. 524–531.
[34] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," Proc. IEEE Comput. Soc. Conf. Comput. Vis., Fort Collins, CO, USA, 1999, vol. 2.
[35] L. Ding and A. Goshtasby, "On the canny edge detector," *Pattern Recogn.*, vol. 34, no. 3, pp. 721–725, 2001.
[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, vol. 4, pp. 1942–1948.
[37] R. Rifkin and A. Klautau, "In defense of one-versus-all classification," *J. Mach. Learning Res.*, vol. 5, pp. 101–141, 2004.



Xin Ma (M'10) received the B.S. degree in industrial automation and the M.S. degree in automation from Shandong Polytech University (now Shandong University), Shandong, China, in 1991 and 1994, respectively. She received the Ph.D. degree in aircraft control, guidance, and simulation from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1998. She is currently a Professor at Shandong University. Her current research interests include artificial intelligence, machine vision, human–robot interaction, and mobile robots.

Haibo Wang received the B.S. degree in logistics engineering from the School of Control Science and Engineering, Shandong University, Shandong, China. He had been a cotutelle M.S.–Ph.D. student in the LIAMA and Laboratoire d'Informatique Fondamentale de Lille (LIFL) labs from 2005 to 2010. He received the Ph.D. degree in computer science from LIFL/INRIA Lille-Nord-Europe, Université des Sciences et Technologies de Lille, Lille, France, and the Ph.D. degree in pattern recognition and intelligent system from LIAMA/National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2010.

He is currently a Lecturer at Shandong University. His research interests include computer vision, pattern recognition, and their applications to virtual reality.

Bingxia Xue received the B.S. degree in automation from Shandong University, Shandong, China, in 2012. She is currently working toward the Master's degree in control science and engineering in Shandong University.

Her research interests include artificial intelligence and pattern recognition.

Mingang Zhou received the B.S. degree in automation and the Master's degree in control science and engineering from Shandong University, Shandong, China, in 2010 and 2013, respectively.

His research interests included machine vision and pattern recognition.

Bing Ji received the B.S. degree in electronic engineering and the Master's degree in signal and information processing from Xidian University, Xian, China, in 2007 and 2009, respectively, and the Ph.D. degree in medical engineering from University of Hull, Hull, U.K., in 2012.

He is currently a Lecturer at Shandong University, Shandong, China. His research interests include cloud robotics and computational simulation of biological system.

Yibin Li (M'10) received the B.S. degree in automation from Tianjin University, Tianjin, China, in 1982, the Master's degree in electrical automation from Shandong University of Science and Technology, Shandong, China, in 1990, and the Ph.D degree in automation from Tianjin University, China, in 2008.

From 1982 to 2003, he worked with Shandong University of Science and Technology, China. Since 2003, he has been the Director of Center for Robotics, Shandong University. His research interests include robotics, intelligent control theories, and computer control system.