# Study of One Class Boundary Method Classifiers for Application in a Video-Based Fall Detection System

Miao Yu, Syed Mohsen Naqvi, Adel Rhuma, and Jonathon Chambers

Advanced Signal Processing Group, School of Electronic and Electrical Engineering

Loughborough University, Leicestershire, LE11 3TU, UK

{m.yu, s.m.r.naqvi, a.rhuma, j.a.chambers}@lboro.ac.uk

**Abstract**

In this paper, we introduce a video-based robust fall detection system for monitoring an elderly person in a smart room environment. Video features, namely the centroid and orientation of a voxel person, are extracted. The boundary method, which is an example one class classification technique, is then used to determine whether the incoming features lie in the 'fall region' of the feature space, and thereby effectively distinguishing a fall from other activities, such as walking, sitting, standing, crouching or lying. Four different types of boundary methods, k-center, k-th nearest neighbor, one class support vector machine and single class minimax probability machine are assessed on representative test datasets. The comparison is made on the following three aspects: 1). True positive rate, false positive rate and geometric means in detection 2). Robustness to noise in the training dataset 3). The computational time for the test phase. From the comparison results, we show that the single class minimax probability machine achieves the best overall performance. By applying one class classification techniques with 3-d features, we can obtain a more efficient fall detection system with acceptable performance, as shown in the experimental part; besides, it can avoid the drawbacks of other traditional fall detection methods.

**Index Terms**

voxel person, one class classification, boundary method, fall detection

## I. Introduction

There has been increasing public attention on fall detection in recent years. According to [1], falls are the leading cause of death due to injury among the elderly population and 87% of all fractures in this group are caused by falls. Although many falls do not result in injuries, 47% of non-injured fallers can not get up without assistance and this period of time spent immobile also affects their health. So, it is important to highlight that detecting a fall event at home is an indispensable part of elderly people's care.

Nowadays there are more and more concerns with respect to fall detection using computer vision-based methods, due to the fact that it can avoid the drawbacks, such as being inconvenient to wear and easily affected by noises, of the traditional wearable sensor-based and sound or vibration sensor-based methods as proposed in [2], [3] and [4]. There are various ways to detect a fall event using computer vision and signal processing techniques. In [5] and [6], C. Rougier et al. compare the values of certain extracted features with simple thresholds to make decisions. The head's 3-d velocity and the 2-d human shape information are extracted as features respectively. But when the head's velocity is high (such as fast nodding), a misclassification will occur and the 2-d human shape information is not always an effective measure to distinguish falls due to variations with respect to the distance of a person to the camera and the 3-d plane on which the person performs activities. C. Juang and C. Chang [7] on the other hand use an elegant self-constructing neural fuzzy inference network for posture recognition. In their method when a 'lying' posture is detected, certain rules are added to determine whether a fall occurs, such as counting the frame number from the first frame representing 'lying' to the previous closest standing posture frame; a fall is judged to have occurred when this number is less than a threshold. This procedure will become very complex however if various postures are considered. Moreover, the synthesis of a self-constructing neural fuzzy inference network is not robust to noise in the training set. As an effective tool for the classification problem, the support vector machine (SVM) technique is applied in [8], the extracted features are finally fed to a multi-class SVM for precise classification of motions and determination of a fall event. But for a multi-class SVM system, many two class SVMs which distinguish two activities need to be trained and the combination of the two class classifiers will lead to a very complex structure. In [9], a layered hidden Markov model (LHMM)-based approach is proposed to determine the state of the person (walking or falling) from a multiview pose classification strategy. Although the paper proposes an elegant way to use 'image rectification', as proposed in [9], to derive relationships between the 3-d angle corresponding to the individual's major orientation and the principle axis of an ellipse fitted to the human in a 2-d image, the construction of an LHMM is complex and for a particular activity, the corresponding LHMM needs to be constructed. So, as for the self-constructing neural fuzzy inference network, it seems complex and not effective to distinguish falls from other activities, for the LHMM models only lying and walking are constructed.

Our philosophy is based on the observations that although there are many non-fall activities (the most common non-fall activities are walking, sitting, standing, bending and lying but these can be extended to other activities when people do exercises, such as stretching, even rotating, so that the number is large), the fall activity has distinct characteristics and can be ascribed to one class. For the one class classification problem, there are two main methods: density method–which constructs the corresponding density model for one class and the boundary method–which defines the 'region' to which the one class belongs. However, the density method is limited since we normally do not know the particular density model that the training set obeys; besides, in order to obtain the exact density model, more samples are needed in the training dataset compared with the boundary method, as proposed in [10]. So, considering these disadvantages, the boundary method is preferred, and it is used to obtain a closed boundary to enclose the target data set (fall) while excluding the remaining data set (non-fall). In our work, a voxel person is constructed and the corresponding video features (centroid and orientation information) are extracted. For the feature extraction, we need to use at least two synchronized cameras and more cameras are needed to cope with the occlusion in the view of a certain camera. The boundary method-based one class classifiers, k-center, k-th nearest neighbor (k-th NN), one-class support vector machine (OCSVM) and single class minimax probability machine (SCMPM) are then constructed and their performances are compared.

Compared with the previous work, the main advantages of our method include:

1). Instead of using 2-d features (as in [5], [6], [8] and [7]), we use 3-d features derived from various view angles of multiple cameras in order to reliably detect falls in different directions.

2). In order to avoid the need to construct a model for each type of activity (such as walking, sitting, standing and crouching), as in [7], [8] and [9], we exploit a one-class classifier which captures the fall activity in a single class different from the non-fall classes. Our approach thereby has the advantage that training time and testing time for the proposed fall detection system will be reduced.

The structure of this paper is as follows: Section II shows the extraction of the video features. The boundary methods, k-center, k-th NN, OCSVM and SCMPM are introduced in Section III. Experimental simulations are presented in Section IV, together with comparisons between the different classifiers. Conclusions and future work are given in Section V.

## II. Video feature extraction

For the video features needed for the construction of the classifier, we use 3-d features obtained from a voxel person which is constructed by the method proposed in [11]. Traditional 2-d features for posture and motion velocity have the drawback that they are highly dependent on the viewpoint of the camera, e.g. consider a fall along the optical axis of the camera versus a fall perpendicular to it, as proposed in [12]. So, 2-d features are not efficient to be used for detection of falls that happen in different directions and that is the reason why we use 3-d features instead of 2-d features.

### A. *Voxel person construction*

In order to obtain the 3-d features, a voxel person is constructed from two cameras (although additional cameras can achieve a more accurate voxel person which better represents the real person, for the purpose of feature extraction for fall detection, due to the fact that only the person's 3-d centroid position and orientation angle are needed, two cameras are found to be enough). For the frame obtained by each camera, we use background subtraction to extract the foreground human body region, in particular, the codebook subtraction technique [13]. The advantages of the codebook background subtraction algorithm is that it is effective in compressed video frames and can deal with the problems of illumination change, and the change of background after training; besides, compared with other methods, such as the mixture-of-Gaussian (MoG) [14] and the kernel-based (KB) method [15], its computational time is less.

The voxel person is constructed from the background subtraction results of the two cameras' frames. And we have to find the correspondences between the pixels in each camera's recorded frame and the voxels in the 3-d space. This can be achieved by the camera calibration technique introduced in [16].

In order to reduce the computational complexity when the 3-d space is very large, we use a more time-efficient octree method as compared with that used in [11] for spatial partitioning. Initially, the space is divided into eight large cubes, for a pixel, a ray can be constructed and we can identify voxels that this ray intersects. Every intersected voxel is then further divided into eight and these small voxels are then checked to determine whether they are intersected by the ray or not. This procedure is repeated until the intersected voxels are reduced to a small size. And the resulting intersected voxels are the ones to which that pixel corresponds and the pixel-voxel relationship for one pixel is obtained.

Figure 1 shows the octree spatial partition method, two voxels (represented in blue) which are intersected by a ray connecting the lens center (black dot) and one pixel on the image plane (small rectangle) are further divided into eight smaller voxels respectively.

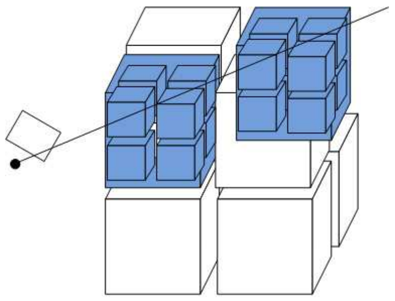

Fig. 1. The octree spatial partition for two voxels which are intersected by a ray

After we obtain the pixel-voxel relationship for all the pixels in the image plane of a camera, the union of voxels corresponding to foreground pixels for that camera forms a set of $P_i$ voxels, it is denoted as : $\mathbf{V}_t^i = \{\mathbf{V}_{t,1}^i, ......, \mathbf{V}_{t,P_i}^i\}$, where $t$ is the captured time, $i$ is the index of the camera, $\mathbf{V}_{t,i}^i$ represents a voxel and $P_i$ is the number of voxels for the i-th voxel set. The voxel person can finally be obtained from the intersection operation: $\mathbf{V}_t' = \bigcap_{i=1}^{C} \mathbf{V}_t^i$ for C cameras, where $C$ is the number of cameras,two in our case because for our case, there is no occlusion in either camera's view and two cameras are enough to recover the rough 3-d person.

The procedure of constructing the voxel person is summarized in Figure 2. Two cameras are used to record the video frames, the codebook background subtraction technique is used to extract the silhouettes of the person for the frames obtained from the two cameras. Each silhouette corresponds to a voxel block set in 3-d space and the intersection of the two sets yields a final voxel person.

## *B. Video feature extraction*

After we obtain the voxel person $\mathbf{V}_t'$, we can calculate the position of the centroid and a value called $gpsim_t$, which reflects the similarity of the voxel person's primary orientation, its definition could be found in equation (3).

The centroid of the voxel person at time t, $\mathbf{u}_t = [x_t, y_t, z_t]$ can be obtained by:

$$\mathbf{u}_t = (\frac{1}{P}) \sum_{j=1}^{P} \mathbf{V}_{t,j}' \tag{1}$$

where $\mathbf{V}_{t,j}'$ is the j-th voxel of a voxel person $\mathbf{V}_t'$ at time t.

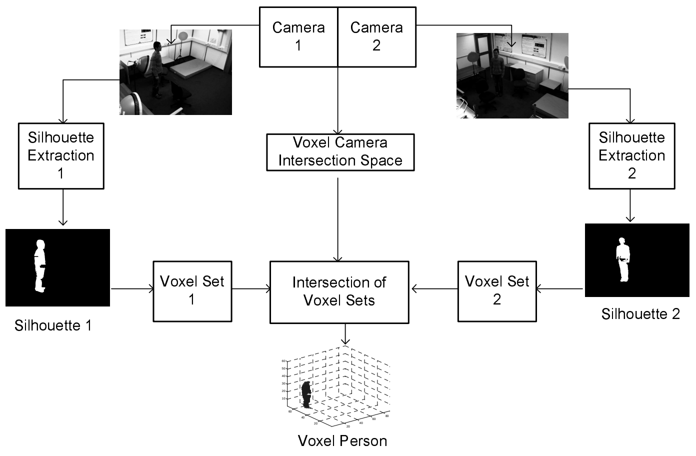

Fig. 2. The procedure of constructing the voxel person from two video camera measurements

We can choose the centroid's height information and differences of the centroid's horizontal and vertical positions in a time interval $\Delta_t$ as part of a feature for fall recognition. The horizontal variation of the centroid can be calculated as: $\sqrt{(x_{t+\Delta_t} - x_t)^2 + (y_{t+\Delta_t} - y_t)^2}$ and the vertical variation is: $|z_{t+\Delta_t} - z_t|$.

The covariance matrix used to define the eigen information is:

$$(\frac{1}{P})\sum_{j=1}^{P}(\mathbf{V}'_{t,j} - \mathbf{u}_t)(\mathbf{V}'_{t,j} - \mathbf{u}_t)^T \tag{2}$$

where $(\mathbf{V}'_{t,j} - \mathbf{u}_t)$ is the difference between the 3-d positions of the j-th voxel for a voxel person and the voxel person's centroid.

The eigenvalues and eigenvectors of the covariance matrix are calculated according to [17] and the eigenvector corresponding to the largest eigenvalue at time $t$ is denoted as $eigenvec_t$ and a value denoted by $gpsim_t$ is calculated by:

$$gpsim_t = max(eigenvec_t \cdot \langle 0,0,1\rangle^T, -eigenvec_t \cdot \langle 0,0,1\rangle^T) \tag{3}$$

If the person is upright, the value is near unity; if he or she is on the ground, the value is near zero.

This value, and its difference during one second $|gpsim_{t+\Delta_t} - gpsim_t|$ are chosen as the remaining elements of the feature vector. Finally, we obtain a 5-dimensional feature vector, it consists of the following elements:

1). The centroid's horizontal position change over $\Delta_t$

2). The centroid's vertical position change over $\Delta_t$

3). The centroid's horizontal position at the particular time

4). The $gpsim$ value change over $\Delta_t$

5). The $gpsim$ value at the particular time

Now that the video feature vector is defined we next consider the construction of the classifier.

## III. One class classification

Although we can not list all the boundary methods for one class classification, we discuss and compare four representative ones which have been widely applied in many practical areas: k-center, k-th NN, OCSVM and SCMPM. Our aim is to compare their performances so that we can recommend the most suitable classifier in the context of our fall detection system.

### A. k-center

The k-center method [10] is a one class classification method which covers the dataset with $k$ small balls with equal radius, the value of $k$ should be chosen properly, if $k$ is small, such as 1, we will get a very loose data description boundary; otherwise, if $k$ is big, many small balls will lead to a tight data description which could cause overfitting. It tries to minimize the following error:

$$\xi_{k-center} = max_i(min_k \parallel \mathbf{x}_i - \mathbf{u}_k \parallel^2) \tag{4}$$

where $\mathbf{x}_i$ is a training feature vector sample and $\parallel \cdot \parallel$ denotes Euclidean norm.

The ball centers $\mathbf{u}_k$ are placed on training objects and obtained by minimizing $\xi_{k-center}$, which is the maximum of all minimum distances between training objects and centers. And the resulting $\xi_{k-center}$, is the corresponding radius square of every ball.

For the testing phase, the distance from a test object $\mathbf{z}$ to the target set is calculated and defined as:

$$d_{k-center}(\mathbf{z}) = min_k \parallel \mathbf{z} - \mathbf{u}_k \parallel^2 \tag{5}$$

If $d_{k-center} \leq \xi_{k-center}$, then we assume that $\mathbf{z}$ belongs to the class.

### B. k-th NN

The k-th NN [10] is one of the boundary methods which avoids the explicit density estimation and only uses distances to the k-th nearest neighbor. It is derived from a local density estimation, where an hypersphere in $d$ dimension is centered around the test object $\mathbf{z}$. The volume of this cell is grown until it captures k objects from the training set. The local density is then estimated by [10]:

$$p_{NN}(\mathbf{z}) = \frac{k/N}{V_k(\parallel \mathbf{z} - NN_k^{tr}(\mathbf{z}) \parallel)} \tag{6}$$

where N is the number of training samples, $NN_k^{tr}(\mathbf{z})$ is the k-th nearest neighbor of $\mathbf{z}$ in the training set and $V_k$ is the volume of the cell.

In the k-th NN classifier, a test object $\mathbf{z}$ is accepted when its local density is larger or equal to the local density of its k-th nearest neighbor in the training set $NN_k^{tr}(\mathbf{z})$. So, the decision rule, as proposed in [10] is:

$$f_{NN_k^{tr}}(\mathbf{z}) = I(\frac{\frac{k}{N}}{V(\parallel \mathbf{z} - NN_k^{tr}(\mathbf{z}) \parallel)} \geq \frac{\frac{k}{N}}{V(\parallel NN_k^{tr}(\mathbf{z}) - NN_k^{tr}(NN_k^{tr}(\mathbf{z})) \parallel)}) \tag{7}$$

where $I(\cdot)$ is the indicator function and the value is unity if the term in the bracket holds, and is otherwise zero.

For a d-dimensional cell, its volume is: $V(r) = V_d r^d$ , where $V_d$ is the volume of a unit hypersphere in $d$ dimensions. This can be substituted into equation (7) to obtain:

$$f_{NN_k^{tr}}(\mathbf{z}) = I\left(\frac{\| \mathbf{z} - NN_k^{tr}(\mathbf{z}) \|}{\| NN_k^{tr}(\mathbf{z}) - NN_k^{tr}(NN_k^{tr}(\mathbf{z})) \|} \leq 1\right) \tag{8}$$

It means that if the distance from object **z** to its k-th nearest neighbor in the training set $NN_k^{tr}(\mathbf{z})$ is smaller than that from $NN_k^{tr}(\mathbf{z})$ to its k-th nearest neighbor in the training set, the object **z** is accepted as a sample from the distribution of the training class; otherwise not. In practice, the value of k must be determined empirically.

## *C. OCSVM*

The OCSVM is proposed in [18]. The basic idea behind OCSVM is that given a data set drawn from an underlying probability distribution $P$ for the minority class, OCSVM estimates a function $f$ that is positive in a region $S$ and negative in the complement, where $S$ is the 'most-likely region'– a subset of the input space such that a test point drawn from $P$ lies outside of $S$ equals some a priori specified value between 0 and 1.

The strategy of OCSVM is to map the training data into the feature space $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$ to separate them from the origin with maximum margin.

To design the classifier, we try to solve the following quadratic problem:

$$\min_{\mathbf{w},\xi,\rho} \qquad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu\ell}\sum_i \xi_i - \rho \quad \text{subject to} \quad (w \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \qquad \xi_i \geq 0 \tag{9}$$

Here, $\nu \in (0, 1]$ and $\ell$ is the number of training data samples, in [18], it is proven that $\nu$ is an upper bound on the fraction of outliers (samples outside the decision region) and it is also a lower bound on the fraction of SVs (samples on the decision boundary and outside the decision region). The nonzero slack variables $\xi_i$ are introduced to allow for the possibility of outliers (the data points which are not drawn from the distribution $P$), the range of $i$ is from 1 to $N$, where $N$ is the size of the training dataset .

For a new test point **x**, the decision function is:

$$f(\mathbf{x}) = sgn((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho) \tag{10}$$

where $sgn(\cdot)$ is a symbol function, if the term in the bracket is no less than zero, the result of this function is $+1$; otherwise, $-1$.

Using multipliers $\alpha_i, \beta_i \geq 0$, we introduce a Lagrangian [19]:

$$\begin{aligned} L(\mathbf{w}, \xi, \rho, \alpha, \beta) = \frac{1}{2} \| \mathbf{w} \|^2 + \frac{1}{\nu\ell}\sum_i \xi_i - \rho \\ - \sum_i \alpha_i((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho + \xi_i) - \sum_i \beta_i \xi_i \end{aligned} \tag{11}$$

We set the derivatives with respect to the primal variables $w, \xi, \rho$ equal to zero respectively and obtain:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i) \tag{12}$$

$$\alpha_i = \frac{1}{\nu\ell} - \beta_i \leq \frac{1}{\nu\ell} \tag{13}$$

$$\sum_i \alpha_i = 1 \tag{14}$$

According to the $KarushKuhnTucker$ conditions (KKT), the following constraints hold [19]:

$$\alpha_i((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho + \xi_i) = 0 \tag{15}$$

$$\beta_i \xi_i = 0 \tag{16}$$

By some deductions from the above results, we can obtain a dual problem as:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{ij} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu\ell}, \quad \sum_i \alpha_i = 1 \end{aligned} \tag{17}$$

Here $k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$ and it is called the 'kernel function'. According to the KKT conditions, the decision function follows as:

$$f(\mathbf{x}) = sgn(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho) \tag{18}$$

For the OCSVM, an appropriate kernel must be chosen, in this paper, we choose the Gaussian kernel with the form:

$$k(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \tag{19}$$

where $\sigma$ is the width of the Gaussian kernel as proposed in [20]) and needs to be tuned.

## D. SCMPM

Finally, we introduce a method called SCMPM proposed in [20]. This method seeks a half-space $Q(\mathbf{a}, b) = \{\mathbf{z} | \mathbf{a}^T\mathbf{z} \geq b\}$, such that the probability is at least $\alpha$, that the data lies in $Q$, while the Mahalanobis distance to the origin is maximized, for a certain class whose mean and covariance matrix $(\bar{\mathbf{x}}, \sum_{\mathbf{x}})$ are in a range of $(\bar{x}, \sum_x) \in \chi$, where $\chi = \{(\bar{\mathbf{x}}, \sum_x) : (\bar{\mathbf{x}} - \bar{\mathbf{x}^0})^T \sum_{\mathbf{x}}^{-1} (\bar{\mathbf{x}} - \bar{\mathbf{x}^0}) \leq \nu^2, \ \| \sum_{\mathbf{x}} - \sum_{\mathbf{x}}^0 \|_F \leq \rho\}$, and $\nu$ and $rho$ represent the uncertainties for the mean and covariance matrix respectively. That is, to solve the following problem:

$$\begin{aligned} & max_{\mathbf{a}\neq 0,b} \quad min_{(\bar{\mathbf{x}},\sum_{\mathbf{x}})\in\chi} \quad \frac{b}{\mathbf{a}^T\sum_{\mathbf{x}}\mathbf{a}} \\ \text{subject to} \quad & inf_{\mathbf{x}\sim(\bar{\mathbf{x}},\sum_{\mathbf{x}})} \quad Pr\{\mathbf{a}^T\mathbf{x}\geq b\}\geq\alpha\forall(\bar{\mathbf{x}},\sum_{\mathbf{x}})\in\chi \end{aligned} \tag{20}$$

where $inf_{\mathbf{x}\in X}f(\mathbf{x})$ represents the maximum value of $A$ for which the equation $f(\mathbf{x})\geq A$ in the domain $X$ holds.

Without loss of generality, we set b=1 and following the derivations from [20]. an equivalent problem can be obtained as:

$$\begin{aligned} & min_{\mathbf{a}} \quad \sqrt{\mathbf{a}^T(\sum_{\mathbf{x}}+\rho I_n)\mathbf{a}} \\ \text{subject to} \quad & \mathbf{a}^T\bar{\mathbf{x}}-1\geq(\kappa(\alpha)+\nu)\sqrt{\mathbf{a}^T(\sum_{\mathbf{x}}+\rho I_n)\mathbf{a}} \end{aligned} \tag{21}$$

where $\kappa(\alpha)=\sqrt{\alpha/1-\alpha}$. However, there exists a problem that when $\bar{\mathbf{x}}=\mathbf{0}$ or there is not a choice of $\alpha$ which makes problem (21) feasible, there is no solution. In order to address the problem, one can map the data with $R^f:\mathbf{x}\rightarrow\varphi(\mathbf{x})\sim(\bar{\varphi(\mathbf{x})},\sum_{\varphi(\mathbf{x})})$, to express and solve problem (21) in the feature space. And the problem can be converted to the following form by using an appropriate kernel function $K(\mathbf{z}_1,\mathbf{z}_2)=\varphi(\mathbf{z}_1)^T\varphi(\mathbf{z}_2)$ (here we consider the Gaussian kernel defined in equation (19):

$$\begin{aligned} & min_{\mathbf{r}} \quad \sqrt{\mathbf{r}^T(L^TL+\rho K)\mathbf{r}} \\ \text{subject to} \quad & \mathbf{r}^T\mathbf{k}-1\geq(\kappa(\alpha)+\nu)\sqrt{\mathbf{r}^T(L^TL+\rho K)\mathbf{r}} \end{aligned} \tag{22}$$

where $\mathbf{r},\mathbf{k}\in R^N$ and $[\mathbf{k}]_i=\frac{1}{N}\sum_{j=1}^{N}K(\mathbf{x}_j,\mathbf{x}_i)$. $L$ is defined as $L=(K-\mathbf{1}_N\mathbf{k}^T)/\sqrt{N}$, where $\mathbf{1}_N$ is a column vector with ones of dimension $N$ and $\mathbf{K}$ is the Gramian matrix and defined as $\mathbf{K}_{ij}=\varphi(\mathbf{z}_i)^T\varphi(\mathbf{z}_j)$ (Gaussian form in our case).

The solution of equation (22) is:

$$\gamma_*=\frac{(L^TL+\rho K)^{-1}\mathbf{k}}{\zeta^2-(\kappa(\alpha)+\nu)\zeta} \qquad \text{with} \qquad \zeta=\mathbf{k}^T(L^TL+\rho K)^{-1}\mathbf{k} \tag{23}$$

for a proper chosen $\alpha\leq\frac{(\zeta-\nu)^2)}{1+(\zeta-\nu)^2}$.

After finding the optimal $\mathbf{r}_*$, we can determine whether an incoming test point $\mathbf{z}$ belongs to the class by checking whether $\mathbf{a}_*^T\varphi(\mathbf{z})=\sum_{i=1}^{N}[\mathbf{r}_*]_iK(\mathbf{x}_i,\mathbf{z})\geq b$.

We next evaluate the four one class classifiers on real video datasets.

## IV. Experiments and evaluations

The experiments were performed in the Smart Room (the dimension of the Smart Room is $4.5m\times3.5m\times3m$), Loughborough University. There are two cameras which are located at the corners and they both cover the common activity area of the room. In our experiment, the camera we use is BASLER A312fc, the pixels (H × V) have dimension 780 × 580 with the size of $8.3um\times8.3um$ for one pixel. The maximum frame rate we use in the experiment is 15 fps. And the final output video format is AVI. The cameras are connected to the PCs, with one being server and others being clients. The StreamPix software (version 3.0) is installed in the PCs to perform video recordings and the format of the obtained video is avi. Video recordings are converted to consecutive $320 \times 240$ frames for further processing by MatLab. A synchronizer is used to ensure the synchronization of the cameras.

Figures 3 and 4 show the background subtraction results of the two cameras respectively for example frames in which the human is stretching, crouching, walking and lying. The codebook background subtraction technique is applied and the post-processing technique is also performed to fill in the small holes and remove noise.

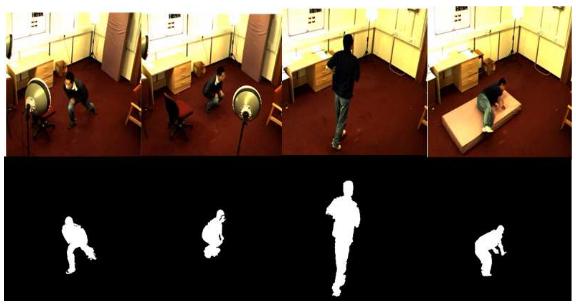

Fig. 3. Some background subtraction results for camera 1, from left to right–stretching, crouching, walking and lying

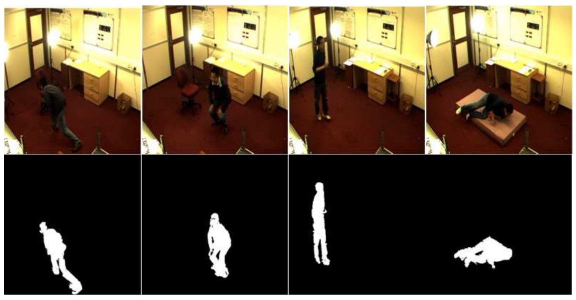

Fig. 4. Some background subtraction results for camera 2, from left to right–stretching, crouching, walking and lying

However, we have to remark that although the codebook subtraction algorithm is robust to gradual light change, as discussed in [13], when the light level suddenly changes, the background subtraction results can be poor and many background pixels can be mistaken as foreground ones. To overcome this problem, we adopt a strategy that when a large change of the percentage of the foreground pixels is detected (which indicates sudden light change), the background model for the codebook background subtraction algorithm is retrained. This is possible provided the foreground subject is not static for a long time during the training phase, which does not occur with a moving human.

Figure 5 shows the results obtained in different lighting conditions: (a) is one frame used for constructing the initial background model; (b) and (c) are the images with foreground objects before and after the sudden light change; (d) is the background subtraction result under the initial background model before the light changes. When the light changes suddenly, the original background model will fail and many background pixels will be mistaken as foreground ones, as shown in (e). We then retrain the background model and the new model is applied for background subtraction in the incoming frames: (f)

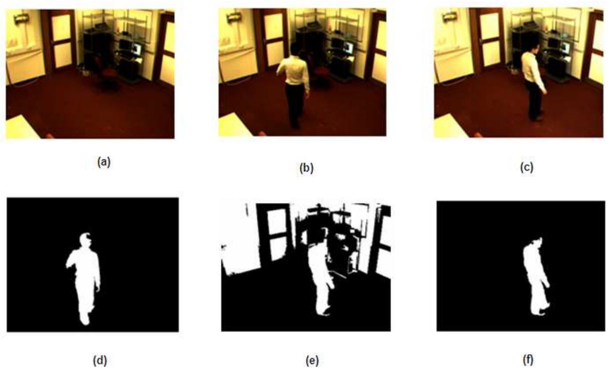

Fig. 5. Background subtraction results under sudden change of the light condition. (a) Original background image. (b) Image with foreground subject before the illumination change. (c) Image with foreground subject after the illumination change. (d) Background subtraction result under initial background model. (e) Background subtraction result under sudden illumination change. (f) Background subtraction result under retrained background model.

is the background subtraction result under the retrained background model after the light condition changes. In this way, we can deal with the problem of sudden light change, such as when an old person moves to turn on a light in a room.

Figure 6 shows the corresponding 3-d reconstruction voxel person results. The voxel person is reconstructed by the procedure proposed in II.A, the smallest voxels have size of $5cm * 5cm * 5cm$, so the normalized smart room's 3-d dimension with the size of smallest voxels is $90 * 70 * 60$ as it is shown in the figure. The finally obtained smallest voxels are marked blue and they form the voxel person which is an approximation to the real 3-d person.

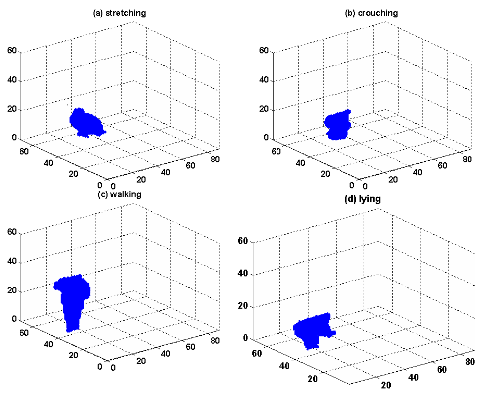

Fig. 6. The reconstructed 3-d voxel persons for stretching, crouching, walking and lying

From the 3-d voxel person, the video features are extracted. For the features extraction, we have to determine an appropriate value for $\Delta_t$ mentioned in Section II.B. It is a fact that if $\Delta_t$ is small, the changes of the orientation and centroid position for fall and some other activities (especially lying) will be similar so the corresponding features will be alike and overlap (as shown in the left subfigure of Figure 17), which shows the projection of the 40 fall features and 40 non-fall features onto 2-d space by principle component analysis (PCA) with $\Delta_t$=0.1s (the blue crosses represent the fall features and the red circles represent the non-fall features). And this value also should not be too large in order to reflect the changes of orientation and centroid position for a single activity happening during a short time interval. As highlighted in [21], a fall is a process lasting a short interval, normally 1-3 secs. Basing on these observations and considering that changes of orientation and centroid position during 1s for fall activity are different from those of non-fall activities, we set the $\Delta_t$ empirically to be 1s. The appropriateness of this selection is confirmed in the right subfigure of Figure 7, From the plot, we can see that the fall features and non-fall features are well separated by choosing $\Delta_t$ to be 1s.

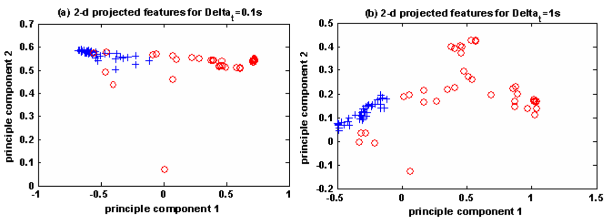

Fig. 7. The 2-D projection of the fall and non-fall feature vectors by using PCA for different time intervals

The fall evaluation of the classifiers is performed on three datasets. The first dataset is a training dataset, a stuntman simulates different types of fall activities, 40 video clips are recorded and the video features for these fall activities are extracted and used to build up the corresponding one of the four one class classifiers– k-center, k-th NN, OCSVM and SCMPM, for a particular one class classifier, its decision region can be defined as: $R = \{\mathbf{x}|F(\mathbf{x}) < 0\}$. However, the training dataset only consists of 3-d video features obtained from one person and thereby the resulting classifier is thereby not a universal solution for different people. But the video features for other people falling are only likely to be in a region which is a slight extension of the obtained region $R$, this slightly extended region could be represented as: $R' = \{\mathbf{x}|F(\mathbf{x}) < th\}$, where $th$ is an appropriate threshold.

In order to obtain the optimal $th$ which achieves high true positive rate (TPR) and low false positive rate (FPR), a validation dataset is needed. Our validation dataset consists of 40 fall activities and 40 non-fall activities simulated by different people. The receiver operating characteristic (ROC) curve of each classifier is plotted according to the variation of $th$ and we choose the $th$ of the ROC point which maximizes the geometric means–$\sqrt{TPR * (1 - FPR)}$. Besides, the classifier's own parameter will also be tuned according to the validation dataset. A grid search method is used here to obtain the optimal parameter set, and different classifiers' parameters which needed to be tuned are shown in Table I.

TABLE I

THE PARAMETERS NEEDED TO BE TUNED FOR DIFFERENT CLASSIFIERS

| Classifier | parameters needed to be tuned |
|---|---|
| k-center | k |
| k-th NN | k |
| OCSVM | $\nu$, $\sigma$ |
| SCMPM | $\alpha$, $\sigma$, $\nu$, $\rho$ |

For a particular parameter set, the corresponding classifier is constructed and the ROC curve is plotted. The AUC value, which is the area under the ROC curve, is calculated. The larger this value, the better the performance of the classifier is under this parameter set.

A grid search method is applied and the parameter set which maximizes the AUC value for the validation dataset is chosen. In our experiment, we found the optimal number of $ks$ for the k-center method and k-th NN are 7 and 1 respectively. And the optimal parameters for OCSVM are: $\nu = 0.1$ and $\sigma = 0.1$. For SCMPM, the obtained parameter set is: $\alpha = 1$, $\sigma = 0.17$, $\nu = 0.1$ and $\rho = 0$.

Under the optimal parameter setting, the ROC curves of the four classifiers for the validation dataset are shown in Figure 8. We can see under the optimal parameter setting, the AUC values are large (for SCMPM it is 1 and for the other classifiers the values are near to 1) for the four types of classifiers, which means good classification performance.

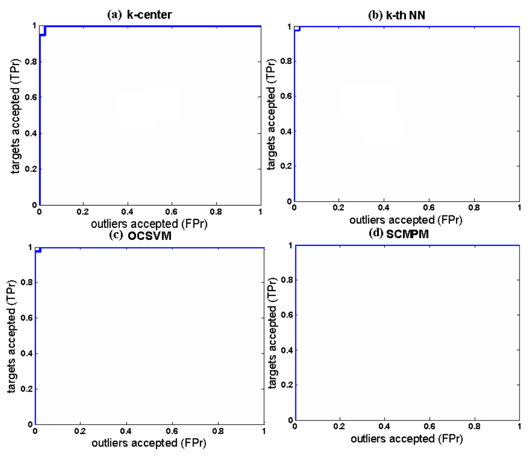

Fig. 8. The ROC curves for k-center, k-th NN, OCSVM and SCMPM respectively

According to the ROC curves, the optimal thresholds which maximize the geometric means (a large value means a high true positive rate and low false positive rate) are chosen and the corresponding classifiers are tested on a test dataset with 37 fall activities and 40 non-fall activities, the results are shown in Table II.

TABLE II

FALL DETECTION PERFORMANCE COMPARISON FOR FOUR TYPES OF ONE CLASS CLASSIFIERS

| | k-center | k-th NN | OCSVM | SCMPM |
|---|---|---|---|---|
| TPR | 92% | 100% | 100% | 100% |
| FPR | 0% | 2% | 2% | 0% |
| Geometric means | 0.96 | 0.99 | 0.99 | 1 |

We can see that among the four classifiers, SCMPM achieves the best performance with the Geometric means being 1, followed by OCSVM and k-th NN with 2% FPR. Considering in the real application of fall detection, the aim is to detect the falls with 100% TPR and an acceptable FPR which is not too high, one could conclude these three methods are all acceptable to be adopted for implementing a fall detection system. The k-center method can only achieve a TPR of 92%, for some falls it fails, which is inadequate for the real application.

And Figure 9 shows the visualized results on the 2-dimensional projected test dataset. The black line is the obtained decision boundary and the blue crosses and red stars represent the projected non-fall (outlier) and fall (target) features respectively. From this figure, we can see that the decision boundary for the k-center method is too tight while for k-th NN and OCSVM method the boundaries are relaxed with some outliers enclosed in the decision boundary, an intermediate one is obtained for SCMPM.

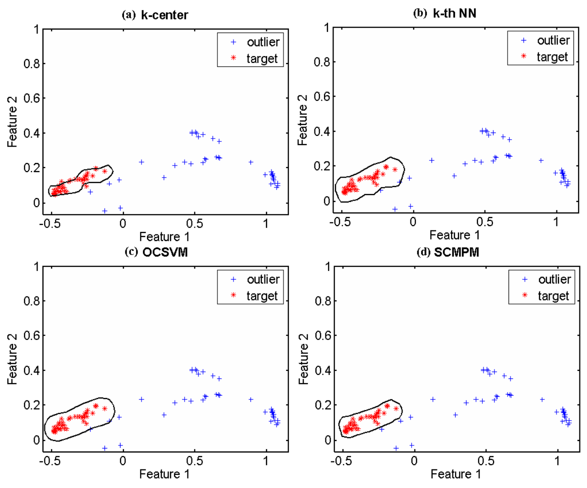

Fig. 9. The 2-D visualized plots of the test features and decision boundaries for the k-center, k-th NN, OCSVM and SCMPM respectively

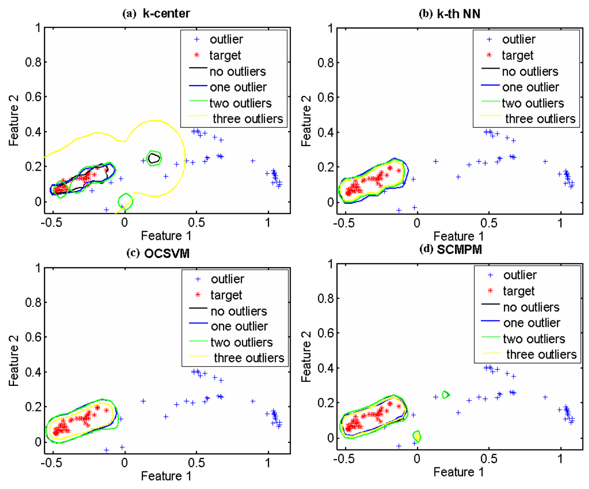

Fig. 10. The 2-D visualized plots of the test features and decision boundaries for the k-center, k-th NN, OCSVM and SCMPM respectively with different number of outliers in the training dataset

However, sometimes the training dataset will be imperfect and contain outliers. Figure 10 shows the shift of the decision boundary under the training dataset with different number of outliers (in this paper, we simulate the outliers by adding the original samples with a Gaussian noise $n\ N(\mathbf{0}, 0.1 * \mathbf{I})$, where $\mathbf{0}$ is a 5-d zero vector and $\mathbf{I}$ is a $5 \times 5$ identity matrix). The decision boundaries obtained with 0, 1, 2 and 3 outliers are marked black, blue, green and yellow respectively. From this figure, we can see that the k-center method is the least robust to outliers with large decision boundary changes when noises are added. For the other three methods, they are robust to noises to an extent by observing that when noise is added in the training dataset, the decision boundaries of k-th NN, OCSVM and SCMPM do not change too much, the TPRs remain the same with only a little change in FPRs. The quantitative result is summarized in Table III:

TABLE III
CLASSIFIERS' PERFORMANCE WITH DIFFERENT NUMBER OF NOISES IN TRAINING DATA SETS.

| | | k-center | k-thNN | OCSVM | SCMPM |
|---|---|---|---|---|---|
| No outlier | TPR | 92% | 100% | 100% | 100% |
| | FPR | 0% | 2% | 2% | 0% |
| One outlier | TPR | 92% | 100% | 100% | 100% |
| | FPR | 0% | 2% | 0% | 0% |
| Two outliers | TPR | 90% | 100% | 100% | 100% |
| | FPR | 2% | 0% | 4% | 2% |
| Three outliers | TPR | 100% | 100% | 100% | 100% |
| | FPR | 17% | 0% | 0% | 0% |
| Mean values | TPR | 93% | 100% | 100% | 100% |
| | FPR | 5% | 1% | 2% | 0.5% |

From Table III, we can see that the SCMPM is most robust to the noise in this case.

Another important aspect in the real application of a particular classification system is its computational time, theoretically, we can see that the computational time for k-center, k-th NN and SCMPM is $O(N)$, where $N$ is the number of samples in the training dataset. And the computational time for OCSVM is $O(SVs)$, where $SVs$ is the number of support vectors. Table IV shows the comparison of the computational time for the four different methods. Both the theoretical computational time and the actual one on the test dataset (77 samples) are provided. In the experiment, we used a PC with Intel (R) Core (TM)2 Duo CPU and 2 GB memory.

TABLE IV

COMPUTATIONAL TIME COMPARISON FOR FOUR TYPES OF ONE CLASS CLASSIFIERS

| | k-center | k-th NN | OCSVM | SCMPM |
|---|---|---|---|---|
| Theoretical results | $O(N)$ | $O(N)$ | $O(SVs)$ | $O(N)$ |
| Actual results($\times 10^{-2}s$) | 0.609 | 0.343 | 0.328 | 0.578 |

From Table IV, we can see that the computational time for OCSVM is less than other three methods. However, we notice that for all these methods, the computational time is at the level of $10^{-5}s$ for one sample. So, there is not much difference with respect to the computational time for these four methods and all of them can be implemented in real time.

In conclusion, we can see that for these four methods, SCMPM achieves the best performance on the test dataset. Unlike k-center, k-th NN, OCSVM and SCMPM are robust to noise–the obtained decision boundary does not change too much even though some noise is present in the training dataset. For the computational time, the differences between these four methods are very small and all of them are applicable in real time. At the end, we concluded that SCMPM is the best choice.

## V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new robust fall detection system based on a one class boundary method classifier. 3-d video features are extracted from the construction of a voxel person, and the corresponding boundary method-based one class classifiers are constructed and their performance is compared. From the experimental results, we can see that the SCMPM method achieves the best performance on the test dataset; besides, it is robust to noise in the training dataset and can be implemented in real time. So, SCMPM is the optimal one among these four boundary methods to choose as a classification method to detect falls.

A more robust fall detection system can potentially be achieved by the combination of audio and video information, which is well known as multimodal processing. The combination can be achieved at the feature level–to group the audio and video features into one vector or decision level–to train the classifiers by using audio and video features respectively and combining them in an elegant way. The combination of audio and video features for fall detection deserves further research.

## REFERENCES

[1] "http://www.medicalnewstoday.com/articles/142487.php, accessed april 2011," *accessed April 2011.*

[2] A. Bourke, C. Scanaill, K. Culhane, J. O'Brien, and G. Lyons, "An optimum accelerometer configuration and simple algorithm for accurately detecting falls," *In Proceedings of the 24th IASTED International Conference on Biomedical Engineering, Anaheim, CA, USA*, pp. 156–160, 2006.

[3] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jamsa, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait and Posture*, vol. 28, no. 2, pp. 285–291, 2008.

[4] M. Alwan, P. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," *The 2nd IEEE International Conference on Information and Communication Technologies: from Theory to Applications, Damascus, Syria*, April 2006.

[5] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Monocular 3d head tracking to detect falls of elderly people," *In Proceedings of the 28th International Conference of the IEEE Engineering in Medicine and Biology Society, New York, USA*, pp. 6384–6387, September 2006.

[6] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Fall detection from human shape and motion history using video surveillance," *In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, Niagara Falls, Canada*, pp. 875–880, May 2007.

[7] C. Juang and C. Chang, "Human body posture classification by a neural fuzzy network and home care system application," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 37, no. 6, pp. 984–994, 2007.

[8] H. Foroughi, A. Rezvanian, and A. Paziraee, "Robust fall detection using human shape and multi-class support vector machine," *In Proceedings of the Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 413–420, December 2008.

[9] N. Thome, S. Miguet, and S. Ambellouis, "A real-time, multiview fall detection system: A lhmm-based approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 10, pp. 1522–1532, 2008.

[10] D. Tax, "One-class classification," *PhD Thesis, TU Delft, NetherLand*, 2001.

[11] D. Anderson, R. Luke, J. Keller, M. Skubic, M. Rantz, and M. Aud, "Modeling human activity from voxel person using fuzzy logic," *IEEE Transaction on Fuzzy Systems*, vol. 17, no. 1, pp. 39–49, 2009.

[12] S. Zambanini, J. Machajdik, and M. Kampel, "Early versus late fusion in a multiple camera network for fall detection," *In Proceedings of the 34th Annual Workshop of the Austrian Association for Pattern Recognition, Zwettl, Austria*, pp. 15–22, May, 2010.

[13] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using code-book model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.

[14] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE International Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, June 1999.

[15] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," *In Proceedings of European Conference on Computer Vision 2000, Dublin, Ireland*, pp. 751–767, July 2000.

[16] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[17] G. Strang, "Introduction to Linear Algebra," W*ellesley*-C*ambridge* P*ress and* SIAM*, 4th* E*dition*, 2009.

[18] B. Scholkopf, J. Platt, J. Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[19] S. Boyd and L. Vandenberghe, "Convex optimization," *Cambridge University Press, 1st* E*dition*, 2004.

[20] G. Lanckriet, L. Ghaoui, C. Bhattacharyya, and M. Jordan, "Robust novelty detection with single-class mpm," *In Proceedings of Neural Information Processing Systems, Whistler, British Columbia, Canada*, pp. 905–912, 2002.

[21] X. Yu, "Approaches and principles of fall detection for elderly and patient," *In Proceedings of the 10th IEEE Int. Conf. on e-health Networking, Applications and Services, Singapore*, pp. 42–47, July 2008.

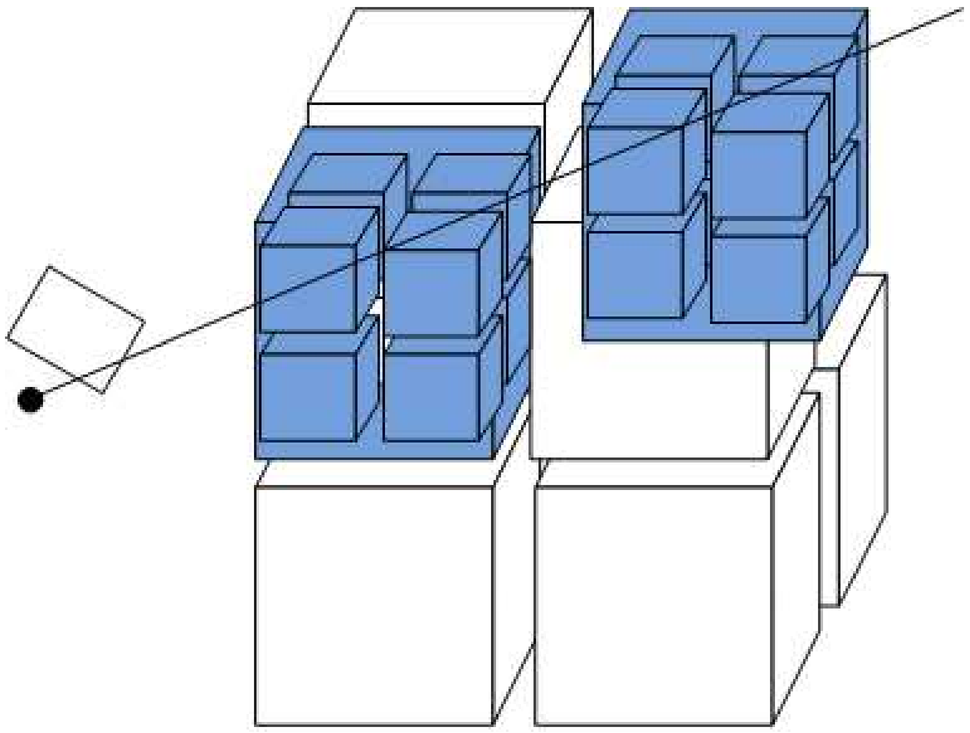
Fig. 11. The octree spatial partition for two voxels which are intersected by a ray

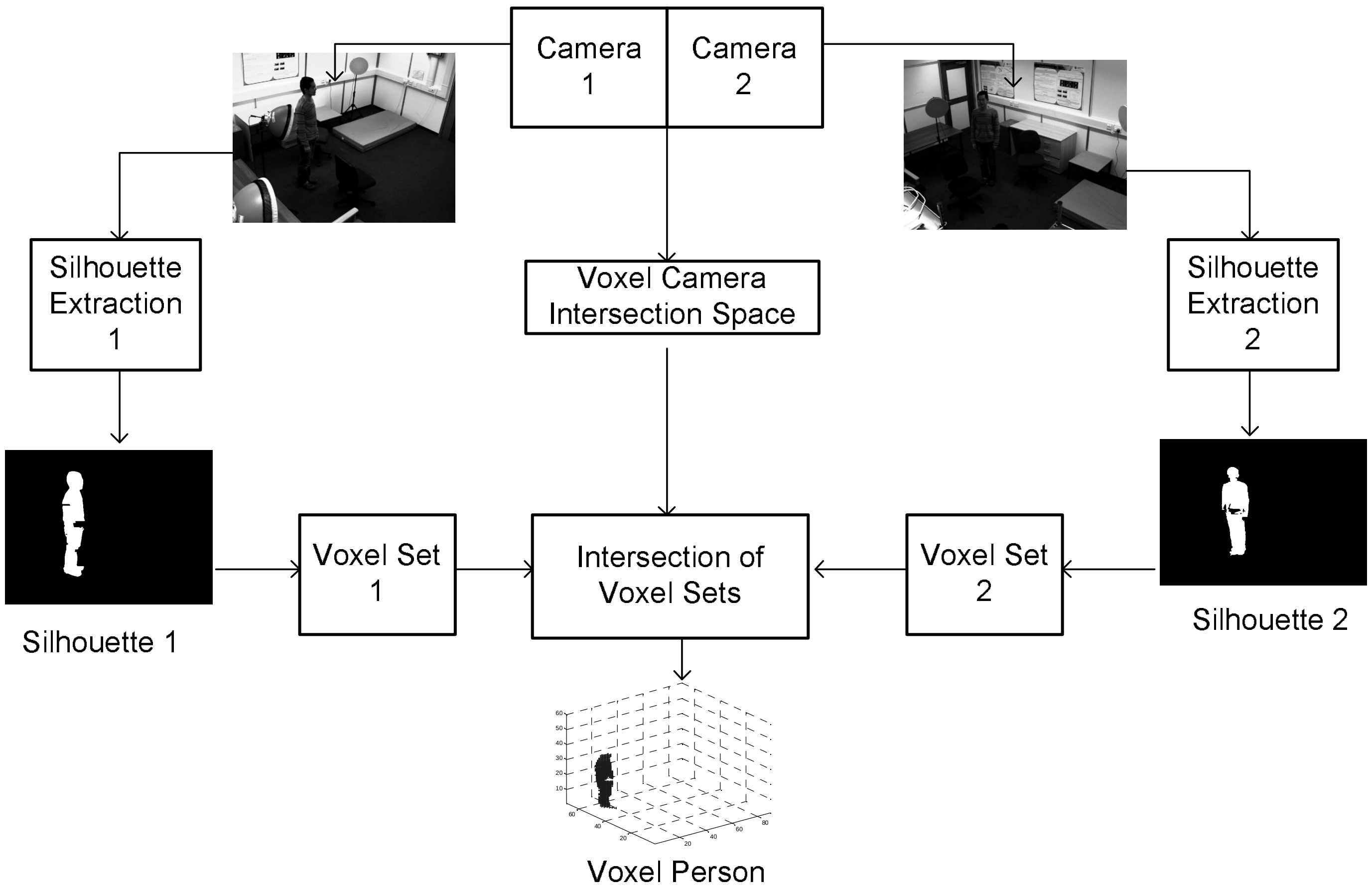

Fig. 12. The procedure of constructing the voxel person from two video camera measurements

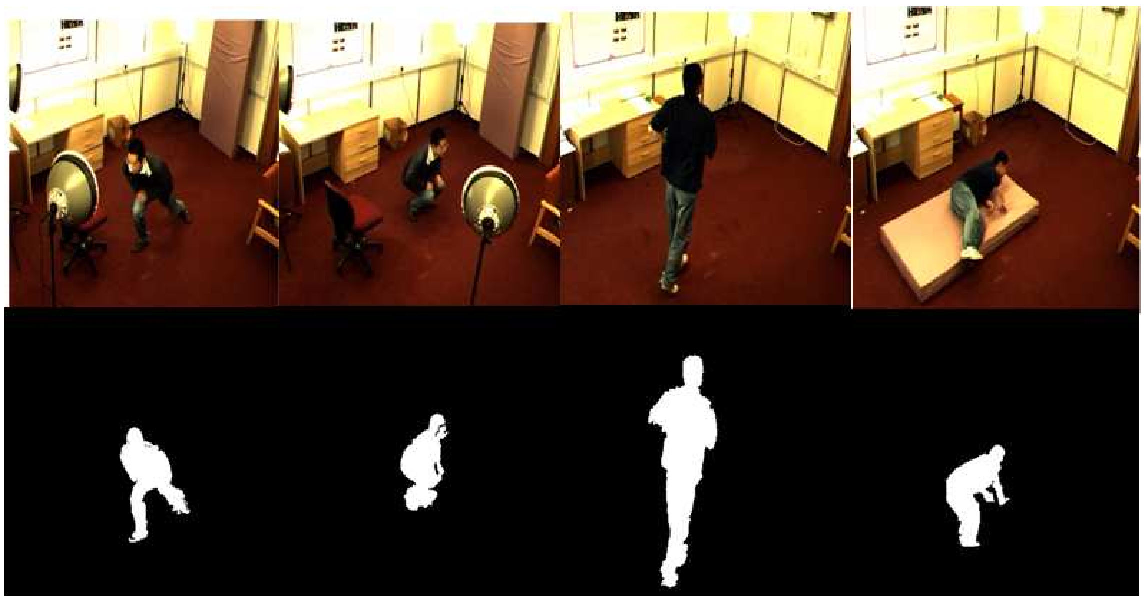

Fig. 13. Some background subtraction results for camera 1, from left to right–stretching, crouching, walking and lying

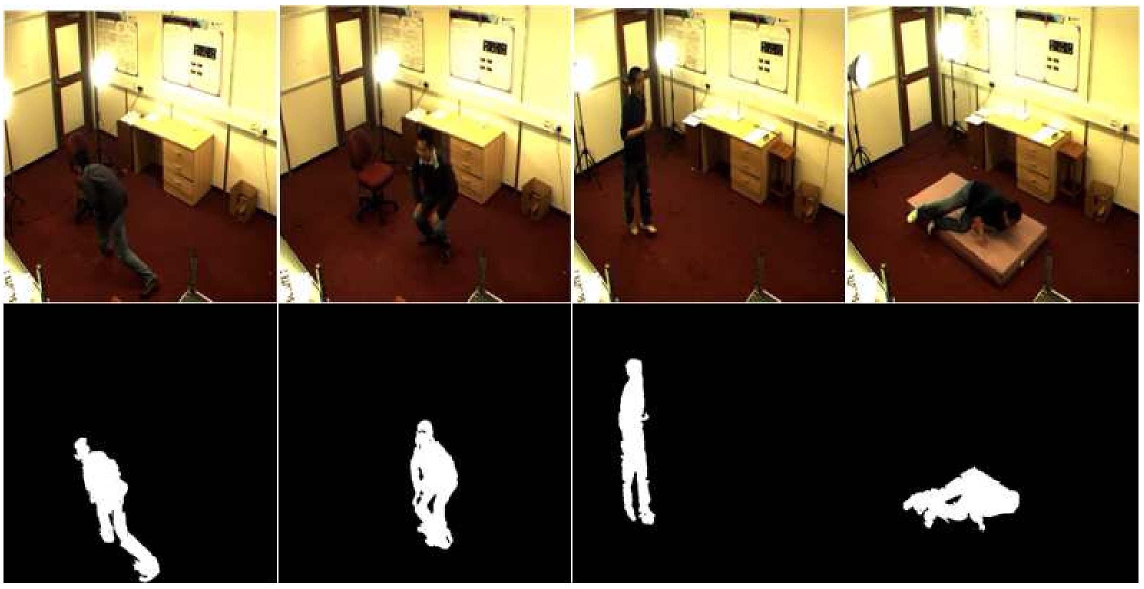

Fig. 14. Some background subtraction results for camera 2, from left to right–stretching, crouching, walking and lying

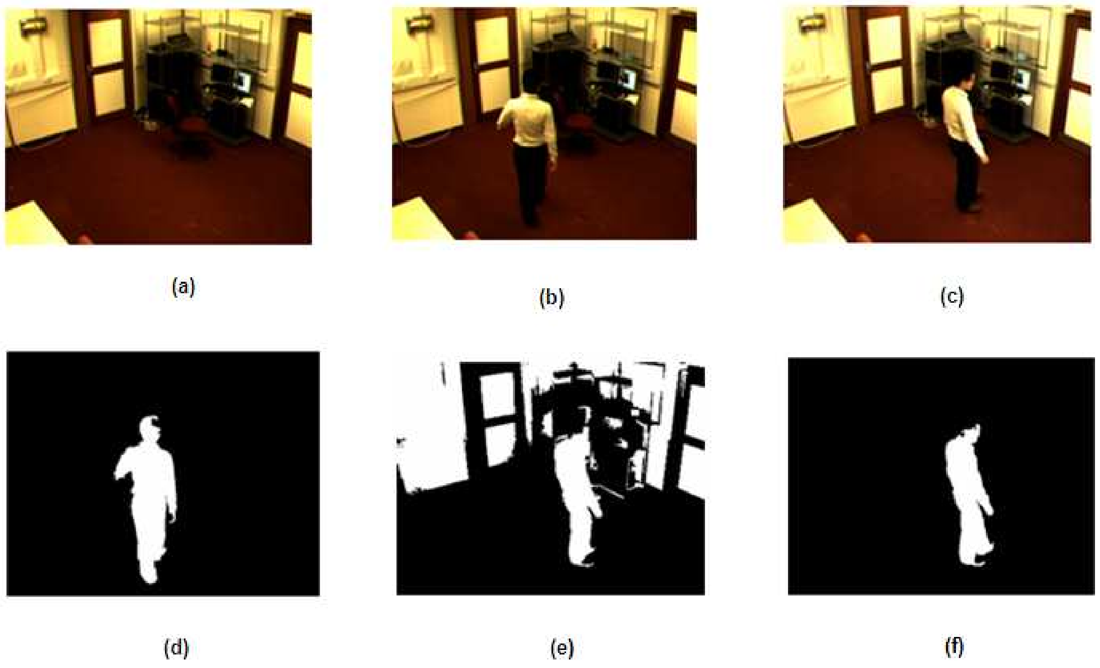

Fig. 15. Background subtraction results under sudden change of the light condition

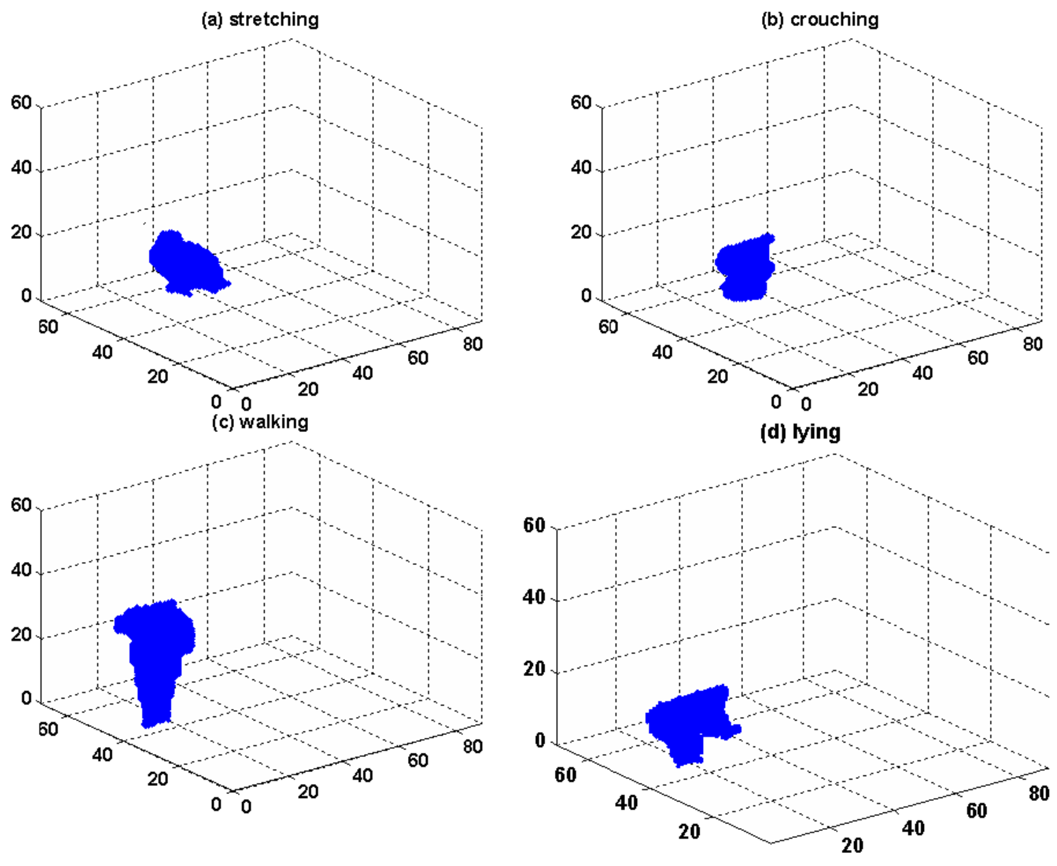

Fig. 16. The reconstructed 3-d voxel persons for stretching, crouching, walking and lying

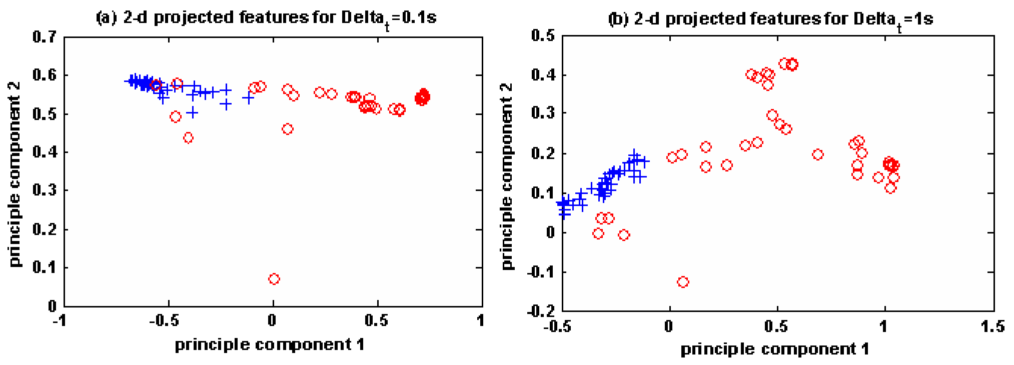

Fig. 17. The 2-D projection of the fall and non-fall feature vectors by using PCA for different time intervals

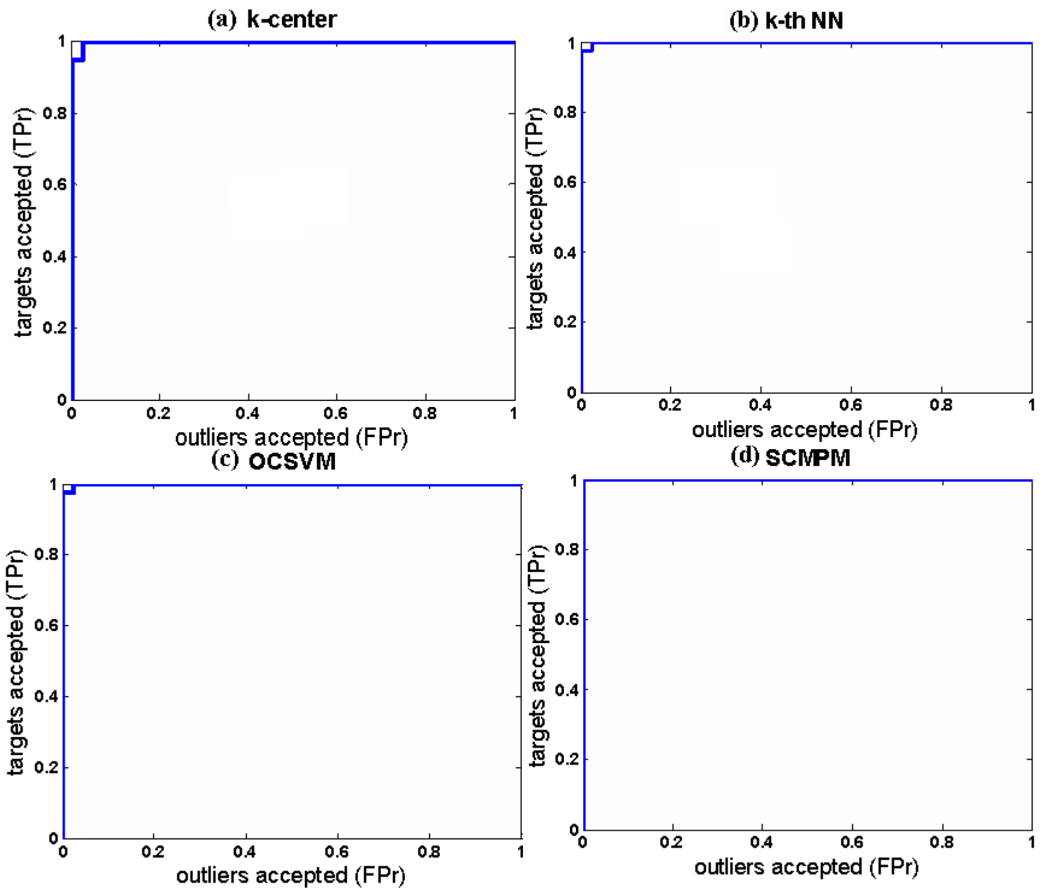

Fig. 18. The ROC curves for k-center, k-th NN, OCSVM and SCMPM respectively

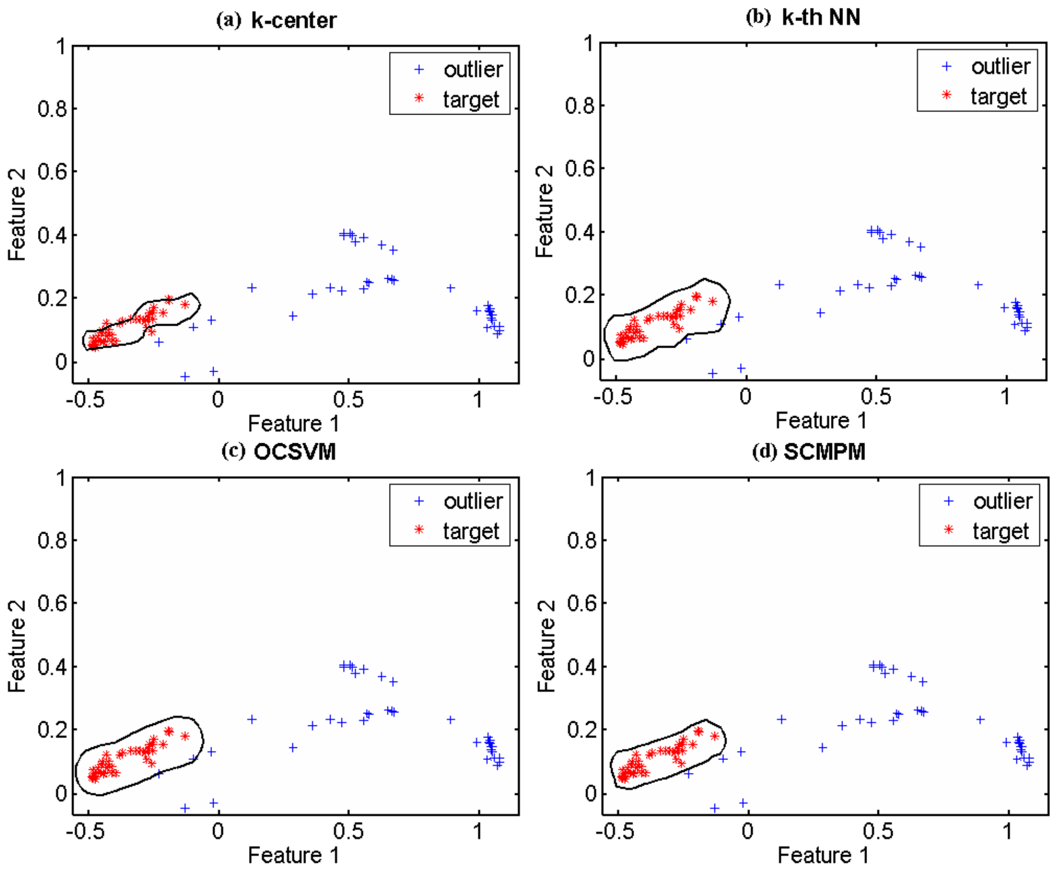

Fig. 19. The 2-D visualized plots of the test features and decision boundaries for the k-center, k-th NN, OCSVM and SCMPM respectively

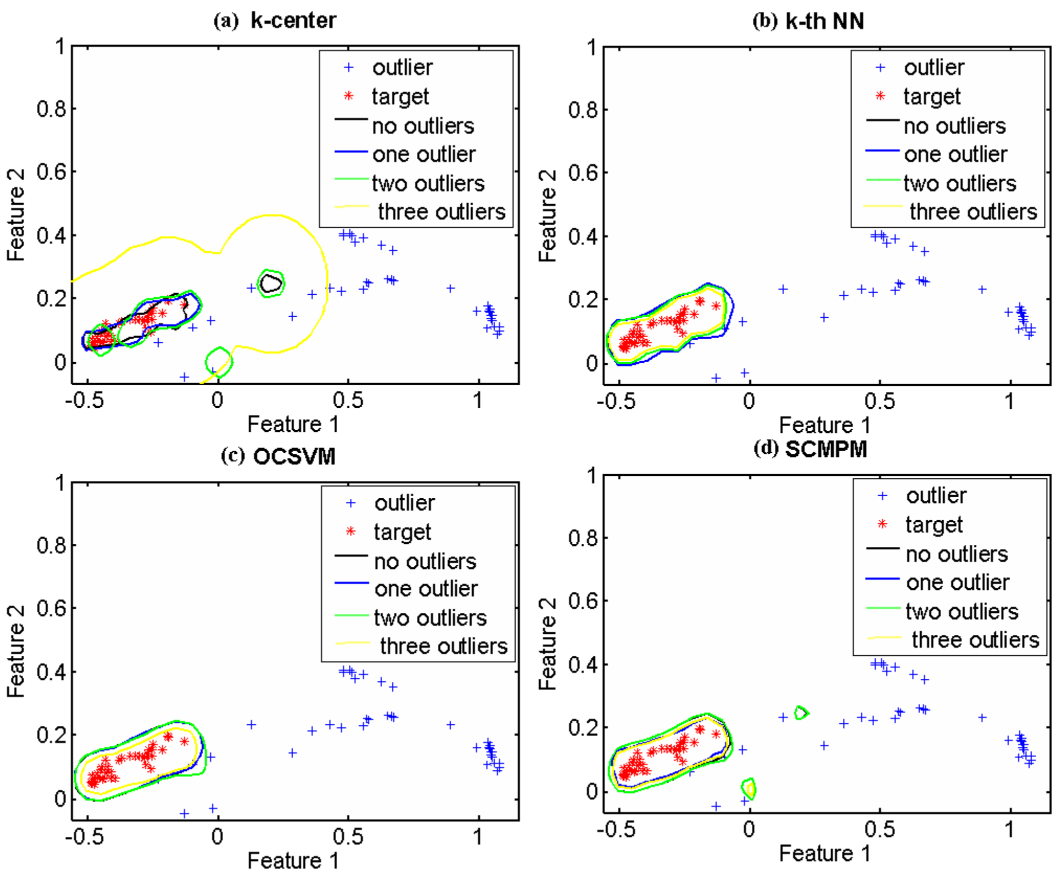

Fig. 20. The 2-D visualized plots of the test features and decision boundaries for the k-center, k-th NN, OCSVM and SCMPM respectively with different number of outliers in the training dataset