Optimised spatio-temporal descriptors for real-time fall detection : comparison of SVM and Adaboost based classification

Charfi I.[a,b], Miteran J.[a], Dubois J.[a], Atri M.[b] and Tourki R.[b]

[a]Laboratoire d'électronique, d'informatique et de l'image, Le2i, Faculté Mirande, Université de Bourgogne, 21000 Dijon, France;
[b]Laboratoire d'électronique et de microelectronique, LEME, Faculté des sciences de Monastir, Tunisie

ABSTRACT

We propose a supervised approach to detect falls in home environment using an optimised descriptor adapted to real-time tasks. We introduce a realistic dataset of 191 videos, a new metric allowing to evaluate fall detection performance in a video stream, and an automatically optimised set of spatio-temporal descriptors which fed a supervised classifier. We build the initial spatio-temporal descriptor named $STHF$ using several combinations of transformations of geometrical features (height and width of human body bounding box, the user's trajectory with her/his orientation, projection histograms and moments of order 0, 1 and 2). We study the combinations of usual transformations of the features (Fourier Transform, Wavelet transform, first and second derivatives), and we show experimentally that it is possible to achieve high performance using Support Vector Machine and Adaboost classifiers. Automatic feature selection allows to show that the best tradeoff between classification performance and processing time is obtained combining the original low-level features with their first derivative. Hence, we evaluate the robustness of the fall detection regarding location changes. We propose a realistic and pragmatic protocol which enables performance to be improved by updating the training in the current location with normal activities records.

Keywords: Video surveillance, Fall detection, real-time processing, spatio-temporal descriptor, Fast Fourier Transform, Wavelet , Support Vector Machine, Adaboost.

1. INTRODUCTION

Automatic detection of falls using artificial vision is a particular case of human activities recognition, and can be useful for helping elderly people : according to the Center for Research and Prevention of Injuries report[1] , fall-caused injuries of elderly people in UE-27 are five times as frequent as other injuries causes which reduces considerably their mobility and independence. Among the diverse applications of computer vision systems, object detection and event recognition are of the most prominent related recognition and motion analysis, that is, researchers had the idea to spread it in fall detection. The fall event, extracted automatically from the video scene represents itself, crucial information that can be used to alert emergency. In this context, visual information on the corresponding scene is highly important in order to take the "right" decision. Therefore, video compression may be included into the acquisition system to reduce data-bandwidth. Meanwhile, detecting such particular situations allows the video compression to be controlled. For instance, the compression rate can be reduced after a fall, to provide more details on the scene or the different compression rates can be applied on the background and the regions of interest. Detecting falls requires two main tasks. First, we have to use a

Further author information: (Send correspondence to Charfi I.)
Charfi I.: E-mail: imen.charfi@u-bourgogne.fr
Miteran J.: E-mail: miteranj@u-bourgogne.fr
Dubois J.: E-mail: Julien.Dubois@u-bourgogne.fr
Atri M.: E-mail: mohamed.atri@fsm.rnu.tn
Tourki R.: E-mail: Rached.Tourki@fsm.rnu.tn

robust method for human body tracking. Second, a robust feature extraction method should be proposed which describes the user's behavior to discriminate falls from other activities.

In this paper, we aim a fall detection system that not only insures privacy protection but insures real-time and adaptive video compression. Our objective is to design a fall detector system that acquires realistic video data, detects falls and alerts emergency by sending the relevant part of the video. Thus, the main contributions of this paper are (i) to propose to build experimentally some spatio-temporal human fall descriptors, named $STHF$, finding the best combination between several features and transformations of features, using a robust human body tracking algorithm, and a supervised based classification system, and (ii) to compare the classification results obtained using standard SVM and Adaboost based algorithm, which is more suitable to real-time embedded applications.

This paper is organised as follows. In section 2, we review the related works. In section 3, we present the dataset and the experimental protocols. The section 4 proposes an overview of the selected fall detection method and is followed by the description of the extracted features and the transformations used for the spatio-temporal descriptor definition, the classification methods and the new proposed evaluation metric. In the section 5, the experimental results and discussions are presented.

2. RELATED WORK

Numerous methods have been developed recently for camera based human action recognition, such as by Liu[2] who used Affine-SIFT based descriptors combined with supervised classification. Jain[3] developed a novel SVM tree applied to gesture recognition, and Wang[4] combined large-scale global features and local patch features with specific SVM classification. However, we will focus in this study only on human fall detection, in order to build a simple descriptor optimised for fall, and suitable for real-time embedded applications.

Numerous existing fall detection systems are based on sensor devices such as accelerometers, microphones and cameras and where summarised by Noury[5] . The camera-based methods for which the recorded videos can be used for outlying and post verification and analysis, are less cumbersome because they are installed in the buildings and not worn by users. Since our aim is to combine event detection and image compression, we will focus on fall detection systems using machine learning methods based only on image sensor.

Most of the image based fall detection methods start by the tracking of the body and the spatial features extraction at each time step which are used to distinguish falls from other actions. The analysis of the bounding box representing the person is one of the simplest and commonly used techniques. In the vision part of Toreyin and al. work[6] the aspect ratio of the moving region detected with a standard camera is analysed by the motion model with Hidden Markov Model (HMM).

Anderson and al.[7] adopted the width to height ratio of the silhouette bounding box and the off-diagonal term from the covariance matrix as the features to determine whether fall incident occurs. These features need to be extracted from the silhouette to train and perform classification stage with HMMs for temporal pattern recognition. Vishwakarma and al.[8] proposed an analytic method for video fall detection by analysing the bounding box representing the person. Same features are extracted by Willems and al.[9] after a background subtraction method for object segmentation.

Khan[10] estimated the magnitude and the orientation of the movement using the Gradient Image. They also analysed the bounding box changes. Shortcoming of these methods is that they did not consider falls in the direction of the camera axis. Moreover, Wu and al.[11] studied unique features of the velocity during normal and abnormal activities. Nevertheless, the 2-D velocity seemed inefficient when the person is close to the camera.

Nait-Charif[12] tracked the head movement and detected falls with an omni-directional camera in an overhead view for automatically extracting motion trajectory. The tracking process is performed by a particle filter estimating ellipse parameters describing human posture. These existing fall detection methods[13] showed a weakness to distinguish between a simple lying situation and a real fall scene.

Foroughi et al.[14] proposed a fall detection method combining the variations of the best-fit approximation of an ellipse around the human silhouette, the projection histograms and the changes of head pose as features for a Support Vector Machine based classifier (SVM).

Rougier and al.[15] analysed with the Gaussian Mixture Models classification method the shape's deformations through video sequences acquired from only one uncalibrated camera and they improve the performance by combining results of four uncalibrated cameras mounted in different points of view. The same authors[16] tracked the head represented as a 3D ellipsoid, with hierarchical particle filter based on color histograms and shape information. They used the vertical velocity of the head centroid during the critical phase (500 ms) in addition to the height of the head relatively to the ground to detect falls.

Liu and Lee[17] detected falls by adopting the human body silhouette to improve privacy and vertical projection histograms and statistical scheme to reduce human body upper limbs activities. The kNN classification algorithm is used to classify postures using the ratio and difference of human body silhouette bounding box height and width. Even though they demonstrate their effectiveness, these works are based on the presumption that the lighting conditions stay relatively uniform which does not always remain true in daily life.

Recently, Liao and al.[18] proposed a method able to detect both fall and slip only events based on the motion activity measure and human silhouette shape variation. The motion measure is obtained by analysing the energy of the motion active area allowing to detect falls parallel to the optical axis.

These previously cited works used the selection of interest points in the spatial domain and none made use of local features such local spatio-temporal descriptor to detect falls, even though they were used for other several human actions recognition. The main idea presented by De Souza[19] is to consider local spatio-temporal features showing that motion patterns are efficient to classify the video elements to violent or non-violent. The authors detected violent scenes characterised by motion variations of image structures over time using the concept of visual codebooks and SVM classifier, where the relevant information is retained describing the spatial interest with respect to derivatives in space and time directions.

We will focus in this paper on a mono camera fall detection system, showing that it is interesting to extend simple spatial features set in the spatio-temporal domain using standard well known transformations and a powerful classifier able to take into account temporal variations in order to obtain a useful detector, regardless the direction of the fall. From a large initial set of features and thanks to a large video dataset, we will build experimentally an optimised spatio-temporal fall descriptor suitable to real-time fall detection.

3. THE DATASET AND THE EVALUATION PROTOCOLS

3.1 The Dataset

When we started this study, there was only few available datasets dedicated to fall detection solution. Moreover, the authors of the related works (for example Rougier[15]) used generally the same location for testing and training which does not enable to evaluate the robustness of the method to the location change. Thus, we build our own dataset named S for the problem of fall detection in realistic videosurveillance setting and in several environments using a single camera.

We acquired a total number of 191 video sequences including 143 containing falls and 48 containing several normal activities, motions, body transfers, for instance from a chair to a sofa. It was necessary to include more videos containing falls than video without fall, since the average duration of a fall is only 14 frames, and since the decision of the classifier is made at the frame level. In these sequences, nine different characters are acting, nevertheless only one at a time is present in the video. The frame rate is 25 frames/s and the original resolution (640x480) is downsampled to 320x240 pixels for the next experiments.

The video data illustrates the main difficulties of realistic video sequences that we can find at an elderly home environment, as well as in a simplest office room. Our video sequences contain variable illumination as well as shadows and reflections that can be detected as moving objects, and typical difficulties like occlusions or cluttered and textured background. The actors wearing different clothes with different colors and texture performed various normal daily activities (walking in different directions, sitting down, standing up, crouching down, housekeeping, moving a chair) and falls (forward falls, falls when inappropriate sitting-down, loss of balance). All activities are taken in different directions without taking in account the camera point of view as shown in figure 1.

The dataset contains videos noted v_i, that we annotated with extra information representing the ground-truth of the fall position in the image sequence. This annotation is used to evaluate the fall detection method. The fall position is defined for each video by the two frame numbers which respectively correspond to the beginning b_i and to the end e_i of the fall ($b_i = e_i = 0$ if the video does not contain any fall). Then, each frame of each video is annotated: the localization of the body is manually defined using bounding boxes. This manually defined bounding box, noted β, allows to evaluate the classification features independently from the automatic body detection. Moreover, a label y_t is associated to each frame I_t, such as $y_t = 1$ if $b_i \leqslant t \leqslant e_i$ and $y_t = 0$ otherwise. Thus, a video v_i is a set of labeled and annotated frames we can note as $v_i = \{I_t, y_t, \beta_t\}_{t=1}^{\gamma}$ where γ is the frame number of the video sequence.

The video sequences of the dataset S have been grabbed into different locations listed in Table 1, defining four subsets, in order to evaluate the robustness of our fall detection method against location change. S is publicly available following the link : http://le2i.cnrs.fr/Fall-detection-Dataset.

Table 1. The Dataset.

"Coffee room" (c)	$S_c = \{v_i, b_i, e_i\}_{i=1}^{n_c}, n_c = 70$
"Home" (h)	$S_h = \{v_i, b_i, e_i\}_{i=1}^{n_h}, n_h = 60$
"Lecture room" (l)	$S_l = \{v_i, b_i, e_i\}_{i=1}^{n_l}, n_l = 28$
"Office" (o)	$S_o = \{v_i, b_i, e_i\}_{i=1}^{n_o}, n_o = 33$
The full dataset	$S = S_h \cup S_c \cup S_o \cup S_l$

3.2 Definition of the experimental protocols

In order to build the fall descriptors and to evaluate the performance of our method, we defined three protocols P_1, P_2 and P_3 that we built using training sets L and testing sets T coming from the subsets described before, assuming that $\forall s = \{h, o, c, l\}$, we have $L_s \cap T_s = \emptyset$ and $S_s = L_s \cup T_s$.

- For P_1, the training and the test subsets were built with videos from the locations "Home" and "Coffee room" *i.e.* from the subsets S_h and S_c : $L_{P1} = L_h \cup L_c$ and $T_{P1} = T_h \cup T_c$. This protocol was used first for low-level features and transformations selection, which will experimentally determine the final fall descriptors, and secondly for evaluation of the automatic tracking of the moving human body.
- For P_2, the training set was built using the videos from "Coffee room" while the test set was built from videos coming from different locations, using "Office" and "Lecture room" subsets : $L_{P2} = L_c$ and $T_{P2} = T_o \cup T_l$.

  This configuration was used in order to evaluate robustness of the fall detection system against the location change.
- For P_3, the training set was built using the videos from "Coffee room" and some videos without any fall (nf) of "Office" ($L_o^{nf} = \{v_i, b_i = e_i = 0\}_{i=1}^{\eta_1}$) and "Lecture room" ($L_l^{nf} = \{v_i, b_i = e_i = 0\}_{i=1}^{\eta_2}$). The test set was built from "Office" and "Lecture room" : $L_{P3} = L_h \cup L_o^{nf} \cup L_l^{nf}$ and $T_{P3} = T_o \cup T_l$.

  We defined this realistic protocol P_3 in order to show that it is possible to improve the performance by updating the training in the current location, with normal activities records, and without the need to record new falls.

4. THE FALL DETECTION METHOD

4.1 Overview

Our method consists in building the STHF descriptors of the fall containing both spatial and temporal information, and to use these descriptors at the input level of a supervised classification system.

The main steps of our detection system, described in details in next sections, are as follows (Figure 2) :

- Motion detection and tracking of the most important moving part in the image, using foreground/background segmentation. This step ends using morphological operators (erosion and dilatation) allowing to remove segmentation artifacts. The result is a binary image and the coordinates of the bounding box of the human body.

(a) Fall samples

(b) Normal activities samples

Figure 1. Samples from the Dataset S

- Feature extraction from the binary image and the bounding box coordinates, such as aspect ratio of the bounding box, ellipse orientation, moments, etc. These features contain only spatial information.
- Feature transformations, capturing temporal variations of previous features, since a fall is characterized by large movement and change of the human shape. The combination of all the feature transformations is the $STHF$ descriptor.
- Image level fall detection using supervised classification : we compared Support Vector Machine[20] and Adaboost[21] based systems to classify each image. The decision of the classifier is followed by a filter (majority vote in a moving window of size $m_w = 5$) removing isolated decisions. The final decision is given at the frame rate.
- Slot image level fall detection using a novel metric to evaluate the performances introducing tolerance in the final classification.

4.2 Feature extraction

From each video v of γ images from S, we extract γ spatio-temporal descriptors noted $STHF_t$, t=1,...,γ, built from spatial low-level features combined with standard transformations allowing the temporal information to be

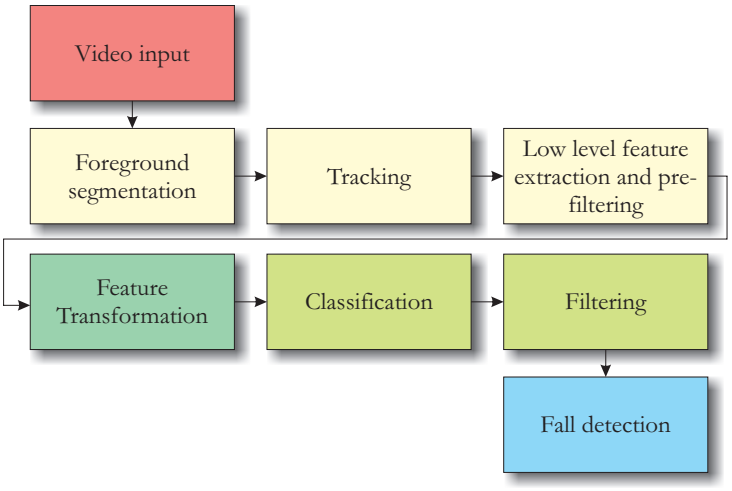


Figure 2. Overview of the fall detection method.

exploited.

4.2.1 Low-level features

For moving object detection, we compared the background subtraction method from L. Li[22] and a simple background subtraction updated using circular buffer. The method of Li is based on the Bayesian decision that performs effectively on many difficult videos of both indoor and outdoor scenes. The algorithm consists of four parts: moving pixels detection, moving pixels classification, foreground object segmentation and background learning and maintenance. However, we evaluated in the preliminary studies of this work that the performance using simple background subtraction were, in our case, at least as good as the Li based method, and 6 times faster. Thus, the results presented in this paper will be presented only using simple background subtraction. The result is a binary image as shown on Figure 3. From this moving object detection, we defined an initial set F of $\varphi = 14$ features that we present in Figure 3, containing height and width of the bounding box (B_h, B_w), aspect ratio of the bounding box (B_r), coordinates of the center of the bounding box (C_x, C_y), coordinates of the center of the best fitting ellipse (E_x, E_y), horizontal and vertical projection histograms (H_{ph}, V_{ph}),[23] moments of order 0, 1 and 2 (m_{00},m_{11},m_{02},m_{20}) of the moving parts of the image, and orientation of the Ellipse (E_o). The moving object detection and feature extraction are implemented in C++ using the OpenCV Library[24] .

The full set of low-level features is then :

$$F = \{B_h, B_w, B_r, C_x, C_y, E_x, E_y, V_{ph}, H_{ph}, m_{00}, m_{11}, m_{02}, m_{20}, E_o\}\,. \tag{1}$$

These features capture spatial and temporal information, since they are determined from the moving object detection. However, a fall, as other human actions, is characterised by a variation of motion, so it is necessary to capture more information about this variation taking into account a group of frames. Then, the basic spatio-temporal descriptor of one frame is made of φ sets of w_1 values : $X = \left\{f_i = \left\{f_t^i\right\}_{t=1}^{w_1}\right\}_{i=1}^{\varphi}$, where f_t^i if the value of the feature number i computed at the frame number t. X is built only from the original low-level features throughout w_1 successive frames.

Some examples of variations of these features are depicted on Figure 4 (a) for a video without any fall and (b) for a video with a fall. More generally, these two sequences illustrate several activities : scene entry, walk, fall, immobility, sitting down, standing up and walk away from the camera, walk towards the camera. From this curves, it is possible to suppose that the more significant features characterizing the fall (part numbered 3 in the figure) are the horizontal projection histograms H_{ph}, the coordinate of the bounding box center, C_y, the moment m_{02}. The automatic feature selection method applied in section 5.1.1 will provide the final feature set.

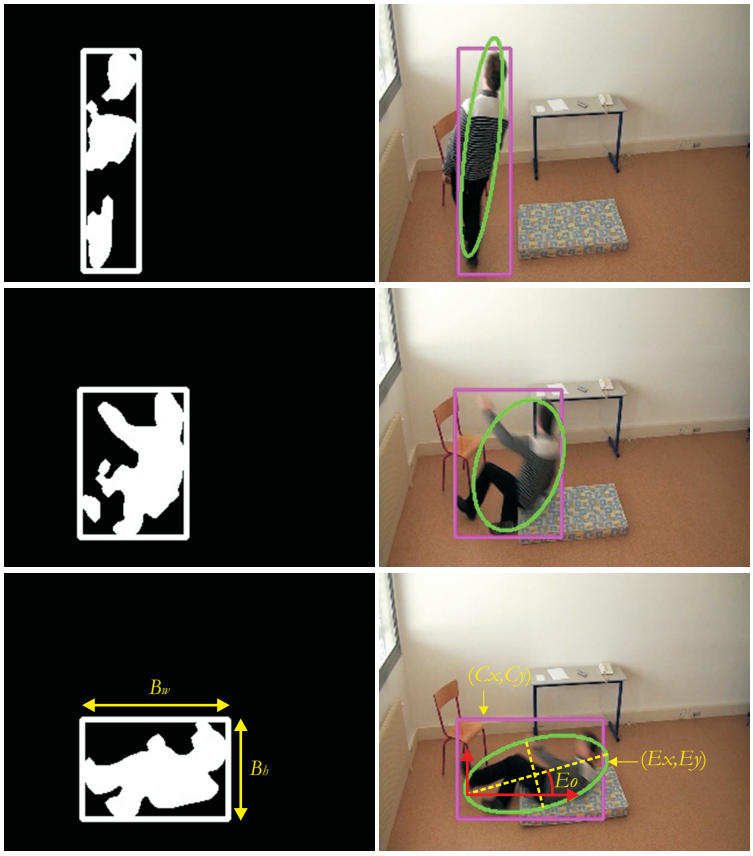


Figure 3. Initial segmentation and low-level feature extraction samples.

4.2.2 Transformations and combinations of low-level features

The descriptor X can be seen as an input vector of the classification method as well as a set of time-dependent signals. In order to include the variations of these signals in the classification process, it is possible to use several well known transformations. The velocity can be approximated using the discrete first derivative. The acceleration is approximated using the discrete second derivative. It is also known that the windowed Fourier Transform combined with supervised classifier has been used with success in several cases, such as 2D object recognition by Smach[25] cancer detection by S. Parfait[26] . It is also well known that the wavelets coefficients can be used as features for pattern recognition[27] . The result of the first derivative applied to X (set of φ signals) during w_1 frames is a set of $w_1 \times \varphi$ values. We note this operator FD, where:

$$FD(X) = \left\{ \alpha_i = \left\{ (f_t^i - f_{t+1}^i) \right\}_{t=1}^{w_1} \right\}_{i=1}^{\varphi}. \tag{2}$$

The same principle is applied for the second derivative. The operator is noted SD. The result is also a set of $w_1 \times \varphi$ values.

$$SD(X) = \left\{ \delta_i = \left\{ (\alpha_t^i - \alpha_{t+1}^i) \right\}_{t=1}^{w_1} \right\}_{i=1}^{\varphi}. \tag{3}$$

The standard FFT algorithm is applied on the signals of X. The module is computed, and the result is a set of $w_{tf} \times \varphi$ coefficients noted ξ_f. We note this operator TF.

$$TF(X) = \left\{ \xi_i = \left\{ \xi_f^i \right\}_{f=1}^{w_{tf}} \right\}_{i=1}^{\varphi}. \tag{4}$$

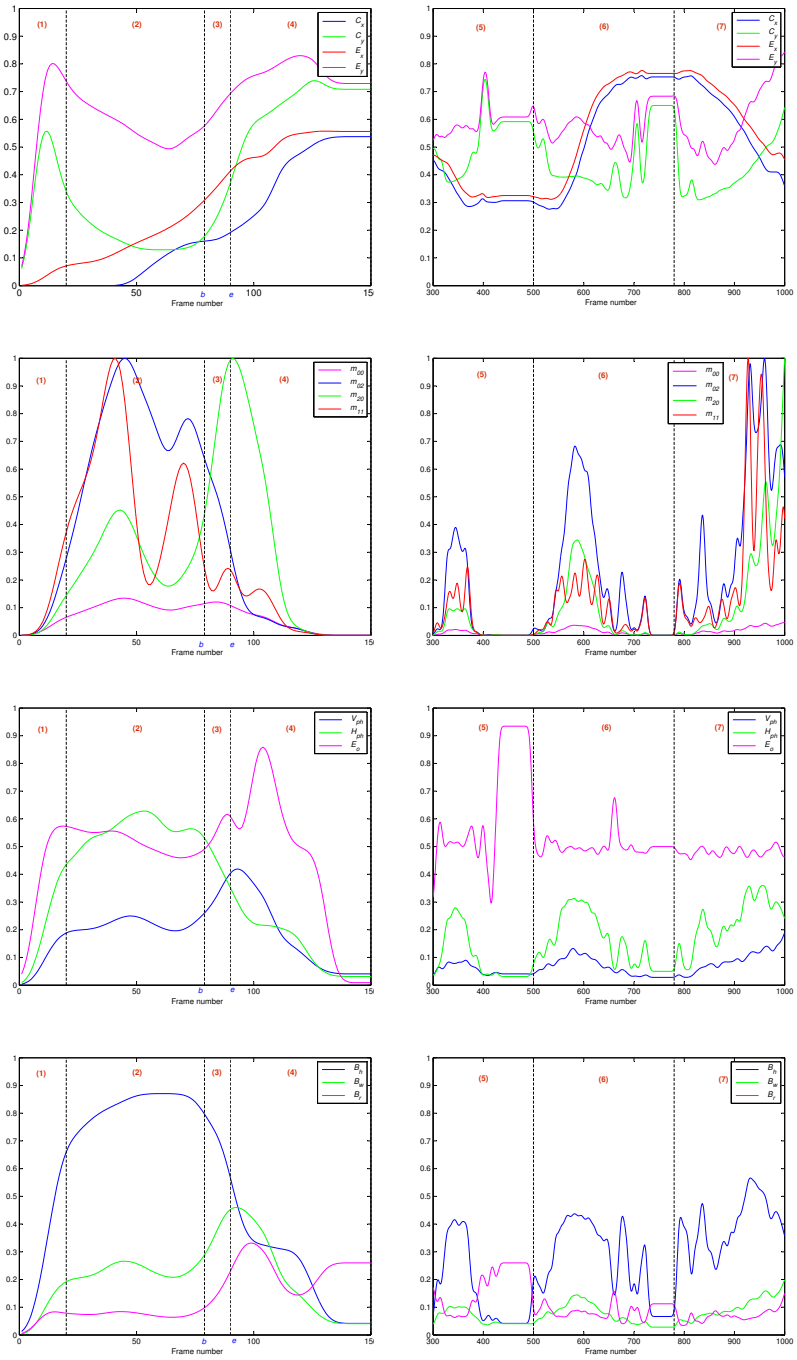


Figure 4. Example of low-level features variations. (left column) Video without fall, (right column) Video containing fall. Activities numbered on figures are (1)scene entry, (2) walk, (3) fall, (4) immobility, (5) walk and sitting down (6) standing up and walk away from the camera, (7) walk towards the camera.

Finally, we used in the next experiments D_4 Daubechies orthogonal wavelets[28] to compute the last set of $w_W \times \varphi$ values, which are the wavelets coefficients C_k. We noted this operator W :

$$W(X) = \left\{ C_i = \left\{ C_k^i \right\}_{k=1}^{w_W} \right\}_{i=1}^{\varphi}. \tag{5}$$

The whole spatio-temporal descriptor is as follows:

$$STHF = \{\{f_i, \alpha_i, \delta_i, \xi_i, C_i\}_{i=1}^{\varphi}\}. \tag{6}$$

$STHF$ can be seen as a vector of dimension $d = \varphi \times (3w_1 + w_{tf} + w_W)$, which can be of high value (typically $d = 14 \times 32 \times 5 = 2240$ in this study). Moreover, important redundancy can be present between the several used operators. Therefore, a part of this study, presented in the section 5 will be dedicated to automatic and experimental selection of features in order to reduce the final feature vector dimension d. The influence of some parameters of the descriptor, such as w_1 and w_{tf} will also be studied in this section. One can note that since we used a relatively low sampling frequency (30 frames/s), and that artifacts in segmentation can occur due to automatic tracking errors, we filtered the signals of each feature using oversampling, Gaussian filtering and subsampling. The influence of this step will be also evaluated in the section 5.

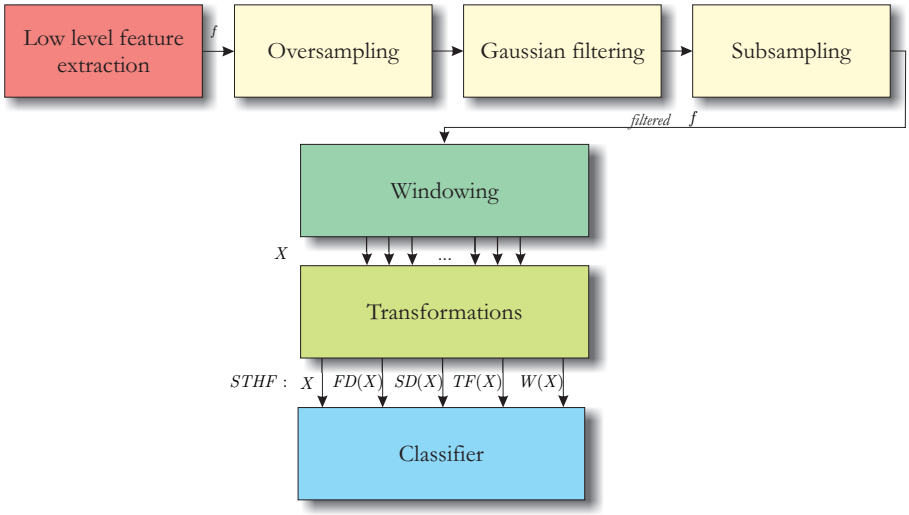


Figure 5. Spatio-temporal feature vector construction.

4.3 The classification methods

We compared in this study the classification result obtained using two supervised methods : SVM and Adaboost. Both methods are well known and were used successfully to solve numerous pattern recognition problems such as anomalies detection on complex objects under manufacturer production in the industrial domain[29] . SVM usually performs better in terms of classification error, whereas Adaboost is faster during the decision step. Moreover, Adaboost is also more suitable to hardware implementation, which can be a critical criterion for real-time embedded application of fall detection.

4.3.1 Support Vector Machine

SVM is a universal learning machine developed by Vladimir Vapnik[20] in 1979. The SVM performs a mapping of the input vectors from the input space (initial feature space) R^d into a high dimensional feature space Q; the mapping is determined by a kernel function K. It finds a linear decision rule in the feature space Q in the form of an optimal separating boundary, which leaves the widest margin between the decision boundary and the input vector mapped into Q. A Radial Basis Function SVM (RBF) was used in this study:

$$K(u, v) = \exp\left(\frac{-\|u - v\|^2}{2\sigma^2}\right) \tag{7}$$

Mapping the separating plane back into the input space R_d, gives a separating surface which forms the following nonlinear decision rules:

$$C(x) = Sgn\left(\sum_{i=1}^{N_v} y_i \epsilon_i \cdot K(s_i, x) + b\right) \tag{8}$$

where ϵ_i are the Lagrange coefficient obtained during the optimisation process. The separating plane is constructed from those N_v input vectors, for which $\epsilon_i \neq 0$. These vectors $s_i, i = 1, ..., N_v$ are called *support vectors* and reside on the boundary margin. All results with SVM were obtained with a home made software based on the LIBSVM library[30] . We used the default values of the SVM parameters defined in this implementation, excepted for the RBF parameter σ which was automatically tuned in order to optimize the classification rate.

4.3.2 Adaboost

AdaBoost is a machine learning algorithm formulated by Freund and Schapire[21] in 1995. It is widely used in the machine learning community and its application showed good performance in many important subtasks of computer vision such as face detection[31] . The basic idea of the Boosting classification is to build "strong" classifier from "weak" ones, focusing at each iteration on misclassified samples. One particularity of Adaboost is to provide a simple decision function and to perform a feature selection in the same training process, when the weak classifier depends only on a subset of feature (usually only one). The algorithm described below consists of learning weak classifiers h_t given a training weighted and labeled data set D_L of p samples in order to build the final classifier by adding the weighted weak classifiers. The weight of each sample $\mathbf{x}_i$ is noted g. In our case, the weak classifier is defined as follow: $h_t(\mathbf{x}) = 1 \Leftrightarrow x_k < \rho_k$ and $h_t(\mathbf{x}) = -1$ otherwise, where ρ_k is a threshold applied on the component x_k of the feature vector $\mathbf{x}$.

- Input $\mathrm{D_L} = \{\mathbf{x}_i, y_i\}_{i=1}^{p}$, maximum number of iteration T_{max}
- Initialize $g_i^{(t)} = 1/p, \forall i = \{1, ..., p\}$
- Do for $t = 1, ..., T_{max}$:
  - Train the classifier with respect to the weighted samples set $\left\{\mathrm{D_L}, g^{(t)}\right\}$ and obtain hypothesis h_t : $\mathbf{x} \rightarrow \{-\mathbf{1}, +\mathbf{1}\}$.
  - Calculate the weighted error e_t of h_t:
    $e_t = \sum_{i=1}^{p} g_i^{(t)} \mathrm{I}(y_i \neq h_t(\mathbf{x}_i))$
  - Compute the coefficient λ_t
    $\lambda_t = \frac{1}{2} \log\left(\frac{1-e_t}{e_t}\right)$
  - Update the weights
    $g_i^{(t+1)} = \frac{g_i^{(t)}}{Z_t} \exp\{-\lambda_t y_i h_t(\mathbf{x}_i)\}$
- Stop if $e_t = 0$ or $e_t \geq \frac{1}{2}$ and set $T = t - 1$
- Output: $y(\mathbf{x}) = sgn\left(\sum_{t=1}^{T} \lambda_t h_t(\mathbf{x})\right)$

Note that we developed a software tool allowing Adaboost evaluation as well as automatic hardware implementation of the decision function. This tool generates VHDL code which can be embedded in highly parallel programmable device such as FPGA. This work is described by Miteran[32] and can be used in the future evolution of this work, *i.e.* the hardware implementation of the full fall detection process.

4.4 Evaluation metrics

We evaluate the fall detection process firstly at the classifier output level (measuring the error rates, computed from the well and misclassified frames) and secondly after the filtering, computing the sensitivity, the specificity, the accuracy, the recall and the final classification error rate as follows:

- true positives (TP): number of falls correctly detected,

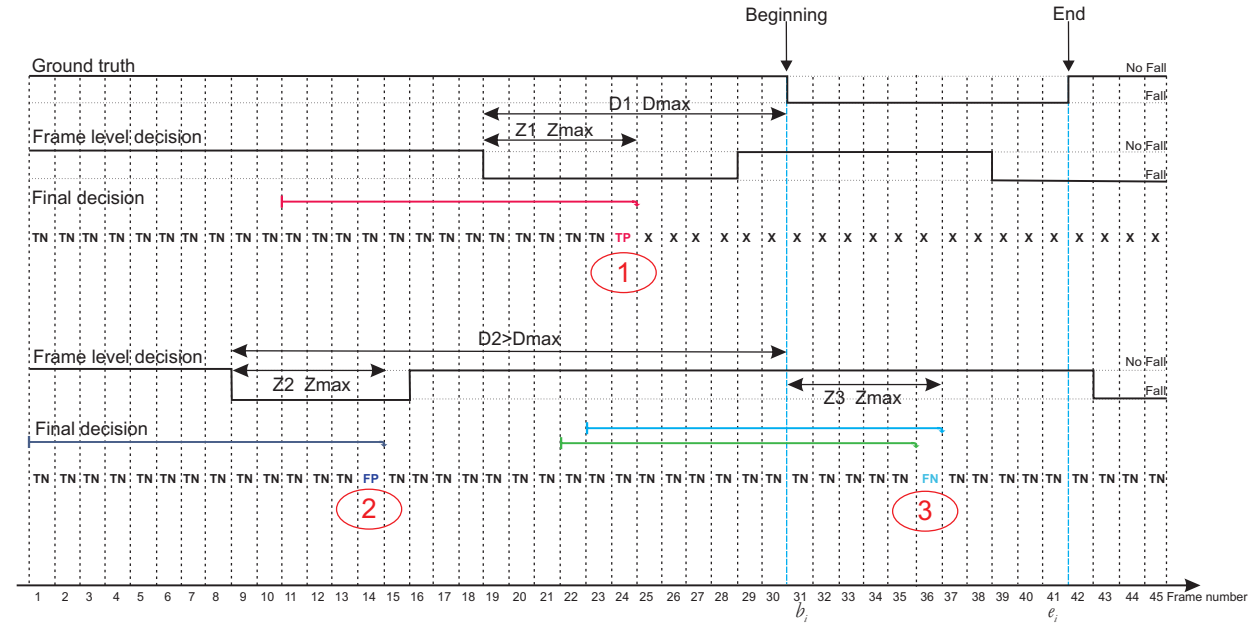
- false negatives (FN): number of falls not detected,
- false positives (FP): number of images detected as a fall,
- true negatives (TN): number of images of normal activities not detected as a fall,
- precision: $Pr = \frac{TP}{TP+FP} \times 100\%$,
- specificity: $Sp = \frac{TN}{TN+FP} \times 100\%$,
- accuracy: $Ac = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$,
- recall : $Re = \frac{TP}{TP+FN} \times 100\%$,
- classification error rate: $E = \frac{FN+FP}{TP+TN+FP+FN} \times 100\%$.

Since the goal of our work is to perform the fall detection in real-time and in a continuous way, we defined an evaluation protocol based on a moving analysis window that we applied to the classifier output decision. The window size w has been fixed regarding the average fall duration, which is 14 images for the whole dataset S, with a standard deviation of 1.4. We fixed then $w = 18$ images. All images of the whole video sequences of test subsets are classified, regarding information contained in the last w images. The resulting latency of a real time implementation would be less than 1 s, which is acceptable for a real time fall detection system. And since it is not critical to detect the fall few frames before or after the fall ground truth (which is not easily to define at the frame level precision), we defined a delay or tolerance factor D between the theoretical fall and the detected fall, measuring the number of images between the beginning of the ground truth fall and the detected fall. We depicted the most important cases in the Figure 6 split into part (a) and (b) for better visibility.

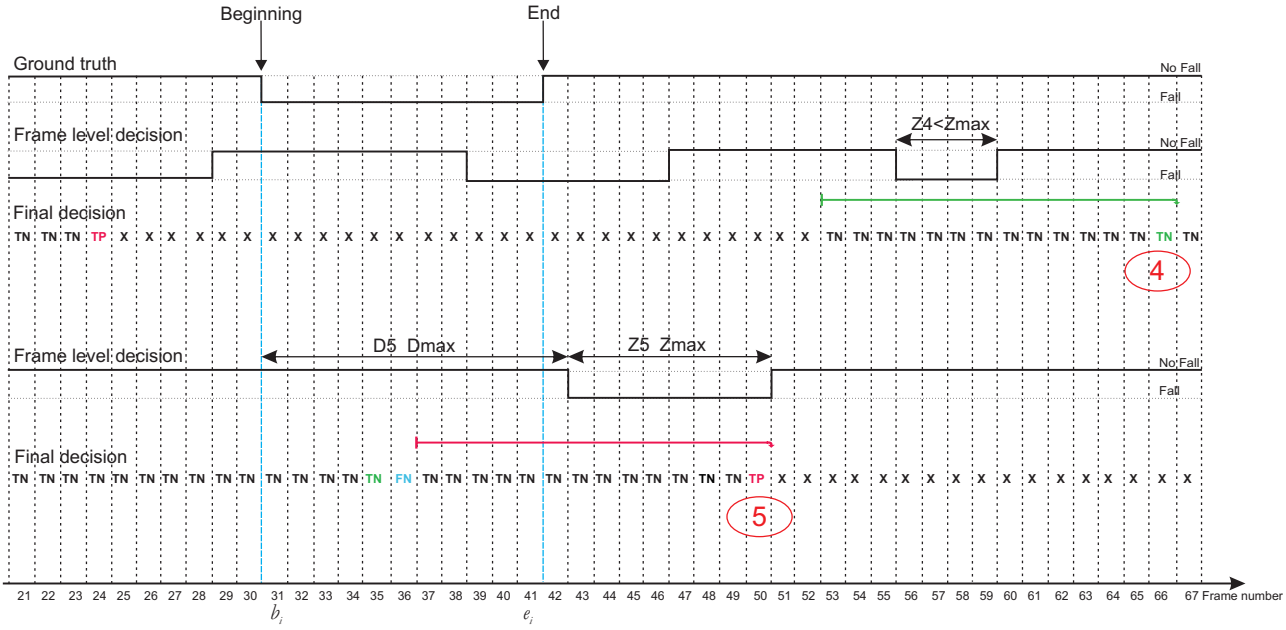
If a beginning of a fall is detected, the system waits w images before to continue the analysis. A TP (or TN) occurs when the number Z of successive decisions "Fall" (or "No fall") of the classifier is higher than a threshold Z_{max}, and if this detection occurs close to the ground truth fall.

If $D < D_{max}$ and $Z > Z_{max}$, where D_{max} is a parameter of the system, a true positive is considered. This situation is depicted in the Figure 6 (b). It is obvious that a too high value of the D_{max} parameter could lead to an non significant classification error (always equal to zero). It seems realistic that the tolerance allowed for the position evaluation of the fall in the video stream is lower than one second (30 frames). The final value has been experimentally fixed to $D_{max} = 15$, as described in the next section. The main cases depicted in Figure 6 (a) and (b) are summarized as follows :

- Case 1 : at the frames number 24, a TP is generated ($Z1 > Z_{max}$ and $D1 < D_{max}$). Note that a good detection will generate only one true positive (TP) for the whole set of fall frames. If a fall is well detected, a TP is generated and we wait $2w$ frames assuming that in a realistic configuration it is not possible to have more than one fall in a short time duration (2 seconds),
- Case 2 : at the frame number 14, a false alarm (FP) is generated when $Z2 \geq Z_{max}$. This is logical since the classifier output is wrong for a high number of successive frames, and the beginning of this detection is too far from the real fall ($D2 > D_{max}$).
- Case 3, at the frame number 35, a TN is generated despite of being inside the interval of fall ground truth. However, when $Z3 \geq Z_{max}$, a FN is generated, frame 36. If no fall is detected, the classification process continues, analyzing the next frame. A non detected fall will generate several false negatives (FN) depending on the duration of the fall.
- Case 4 : the fall detection of the classifier occurs late after the ground-truth fall (from frame 56), and the duration of this detection is short. However, no FP occurs : we filtered the false alarms generated at the classification output thanks to the Z_{max} parameter which allows to generate a TN at the frame number 66 ($Z4 < Z_{max}$).
- Case 5 : the fall detection occurs after the ground truth fall, but since $Z5 > Z_{max}$ and $D5 < D_{max}$, a TP is generated for the frame 50, and the system wait $2w$ frames before to continue the analysis.



(a)



(b)

Figure 6. Final decision : (a) case 1 to 3 (b) case 4 to 5.

5. EXPERIMENTAL RESULTS

We evaluated the robustness of the fall detection system thanks to the three protocols defined in section 3.2 for both SVM and Adaboost classification methods. The SVM-based fall detection evaluation includes the feature selection step in order to remove the non-significant features and builds the optimum $STHF$ descriptor. The fall detection using Adaboost classifier makes use of the full $STHF$ descriptor. For all the experiments, the manually defined bounding box has been used, in order to consider only the intrinsic discriminative capacity of

each feature, independently from the body detection. So far, performance of the system including the automatic body detection are presented.

5.1 SVM-based fall detection

During this section, we evaluated the fall detection performance using SVM and the protocol $P1$. We applied feature selection methods and we evaluated the robustness of the automatic annotation.

5.1.1 Feature selection

In order to optimise the final classification rate and to build an optimised descriptor, we evaluated several combinations of transformations using Protocol $P1$ and manual annotation. We evaluated the result at the SVM output level (so the error rate is computed at image level), after a simple majority vote filtering, and at the final decision level using the previously described metric. For computational time reasons, it is not possible to use an exhaustive feature selection method (the total number of low-level features is $d = 2240$). We decided to select firstly the best combinations of transformations and secondly the subset of low-level features using selected combinations to build the final descriptor. Since the number of combinations of transformations is relatively low (31 possible combinations), it was possible to use an exhaustive approach for this step, where the initial number of low-level features is fixed to $\varphi = 14$.

For each step of this selection process, we tuned the parameters of the method (the width w_1 of the first and the second derivatives, the width w_{tf} of the Fourier Transform and the N_W, number of wavelet level of decomposition) and we retained the optimum values for each combination. Examples illustrating the influence of these parameters are depicted in Figure 7 and 8.

We tuned also the impact of the parameters of the protocol of tolerance at the final decision level and we fixed for all the next experiments $D_{max} = 15$ and $Z_{max} = 9$ giving the best performance. An example of this process is illustrated in Figure 9 at the SVM output level, after the majority vote filtering and at the final decision.

The majority vote filtering applied to the SVM decisions at the image level improved the classification performance. This enables the isolated decisions in the temporal domain to has a better consideration. However, the filter does not affect the performance at the final decision, since our metric removes also the isolated decisions.

The results are summarised in the Table 2, together with the results obtained applying the final decision stage previously described in section 4.4 which was proposed aiming to tolerate delay between the detected fall and its corresponding ground truth. Using this new metric, four combinations allow us to have global error equal to zero : $STHF_a = \{X, FD(X)\}$, $STHF_b = \{W, FD(X)\}$, $STHF_c = \{X, W(X), FD(X)\}$, $STHF_d = \{TF(X), W(X), FD(X)\}$.

In order to optimise the global computation time, we used in the next experiments these four best combinations. For a real-time implementation, the simplest combination can be retained, *i.e.* the original features values combined with the first derivative transformation : $STHF_a = \{X, FD(X)\}$. Indeed, at the SVM output level, the recall rate is better for $STHF_b$, $STHF_c$ and $STHF_d$ which use the Fourier and Wavelet transforms, but this difference is not significant at the final decision level. For real-time hardware implementation of the full process, the computation of transformations can be then avoided without affecting significantly the classification performances.

Then, we selected the features from the set F using a standard SBFS approach[33] minimising the error at the SVM output level. The minimum error at the majority vote level is obtained for $\varphi = 7$ features (see Figure 10). The final set F_s of features used for next experiments is thus:

$$F_s = \{C_y, m_{00}, E_x, B_r, E_o, m_{02}, H_{ph}\} \tag{9}$$

Using these $\varphi = 7$ low-level features selected after applying the two transformations, we build the descriptor $STHF_{a\_SBFS} = \{X_s, FD(X_s)\}$ where X_s represents the basic spatio-temporal descriptor constituted only of the F_s selected features. The final dimension of the feature space is $d = 32 \times 7 \times 2 = 448$.

Table 2. Combinations performance for manual annotation with pre-filtering at the SVM output and final levels using the protocol $P1$.

	SVM level performance					Final level performance				
Combination	E	Sp	Ac	Pr	Re	E	Sp	Ac	Pr	Re
X	3.51	97.01	96.39	92.21	94.75	0.29	99.79	99.70	96.07	98.01
TF	5.05	94.82	94.77	86.61	94.63	0.67	99.98	99.32	97.87	88.46
W	2.56	98.29	97.40	95.42	95	0.38	99.79	99.61	96.07	96.07
FD	3.1	97.29	96.84	92.59	95.56	0.29	99.79	99.71	96.07	98.03
SD	7.6	94.14	92.85	86.07	89.66	1.37	98.75	98.62	80.05	96.04
X+FD	**2.5**	**98.57**	**97.47**	**96.12**	**94.54**	**0**	**100**	**100**	**100**	**100**
$X+TF$	3.42	97.23	96.53	92.74	94.64	0.39	99.59	99.61	92.59	100
$X+SD$	2.82	98.61	97.16	96.18	93.28	0.29	99.79	99.70	96.07	98.02
$X+W$	3.11	97.50	96.84	93.42	95.10	0.19	99.79	99.80	96.15	100
$FD+SD$	3.75	97.58	96.22	93.48	92.58	0.58	99.48	99.41	90.74	98.01
$TF+FD$	3.8	96.83	96.14	91.74	94.29	0.48	99.69	99.51	94.11	96.08
$TF+SD$	3.61	97.45	96.34	93.20	93.39	0.38	99.79	99.61	96.00	96.03
$TF+W$	3.33	97.18	96.62	92.67	95.10	0.19	99.89	99.80	98.01	98.07
W+FD	**2.91**	**97.24**	**97.08**	**93.40**	**96.67**	**0**	**100**	**100**	**100**	**100**
$W+SD$	3.08	98.59	96.92	96.39	92.76	0.29	99.69	99.70	94.34	100
$X+TF+FD$	2.67	98.08	97.30	94.88	95.20	0.09	99.89	99.90	98.03	100
$X+TF+SD$	2.65	98.43	97.33	95.51	94.20	0.09	99.89	99.90	98.03	100
$X+TF+W$	3.26	96.74	96.66	91.71	96.46	0.29	99.79	99.70	96.07	98.05
X+W+FD	**2.67**	**97.72**	**97.32**	**94.47**	**96.33**	**0**	**100**	**100**	**100**	**100**
$X+W+SD$	3.01	98.01	96.98	95.05	94.42	0.29	99.69	99.70	94.34	100
$FD+SD+X$	3.26	97.94	96.70	94.44	93.39	0.19	99.89	99.80	98.08	98.01
$FD+SD+TF$	3.73	97.41	96.35	93.11	93.52	0.19	99.89	99.80	98.04	98.00
$FD+SD+W$	3.58	97.36	96.44	93.03	93.97	0.29	99.79	99.71	96.07	98.00
TF+W+FD	**3.04**	**96.71**	**96.93**	**92.28**	**97.50**	**0**	**100**	**100**	**100**	**100**
$TF+W+SD$	3.35	97.82	96.64	94.56	93.70	0.29	99.79	99.71	96.07	98.03
$X+TF+W+FD$	3.01	97.05	96.97	92.97	96.80	0.09	99.89	99.90	98.03	100
$X+TF+FD+SD$	3.66	95.93	96.32	90.61	97.27	0.09	100	99.90	100	98.02
$X+TF+W+SD$	3.35	97.82	96.64	94.56	93.70	0.38	99.69	99.61	94.23	98.05
$X+W+FD+SD$	2.97	97.09	97.01	93.08	96.80	0.38	99.79	99.61	96.01	96.08
$TF+W+FD+SD$	3.56	97.34	96.43	93.46	94.20	0.19	99.89	99.80	98.00	98.04
$X+TF+W+FD+SD$	3.29	97.58	96.66	93.59	94.19	0.29	99.79	99.70	96.07	98.03

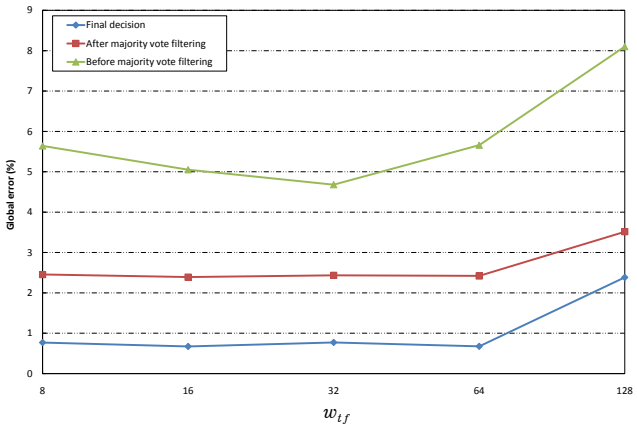


Figure 7. Influence of the width of the TF window.

5.1.2 Performance of the automatic tracking and annotation

We evaluated our fall detection method using SVM and the selected $STHF_{a\_SBFS}$ descriptor including the automatic annotation, for the four combinations giving best performance in the previous experiment, *i.e* using manual annotation). We presented some examples of automatic and manual annotations (bounding boxes and ellipses) in Figure 11. According to the Table 3, at the SVM level, the classification errors are slightly higher using the automatic annotation than manual annotation, for the four combinations previousely selected. However, at the final decision, which includes tolerance, the performance are close to the performance of the manual annotation showing that the automatic annotation is approved (classification errors are lower than 1%). Between

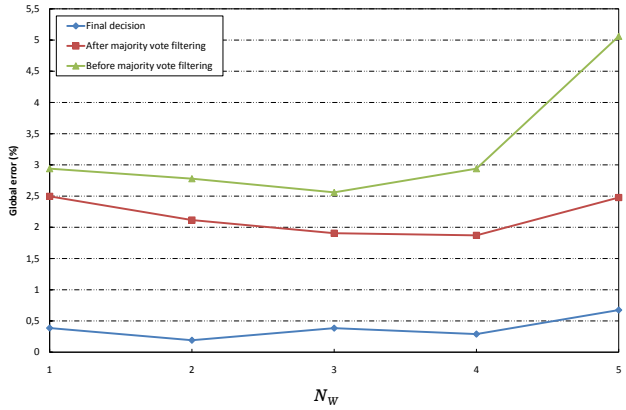


Figure 8. Influence of the Wavelet transform level.

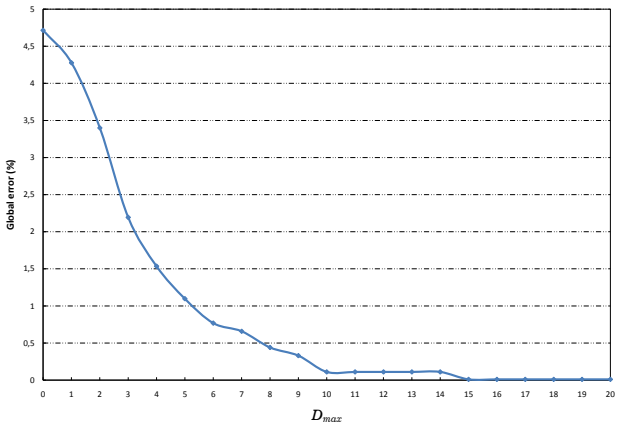


Figure 9. D_{max} influence for the combination $STHF_a = \{X, FD(X)\}$.

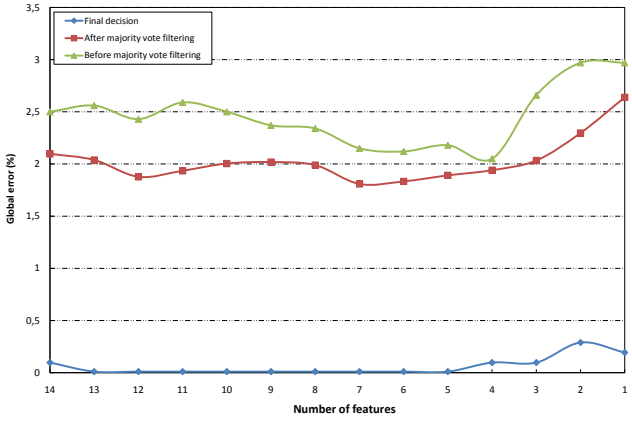
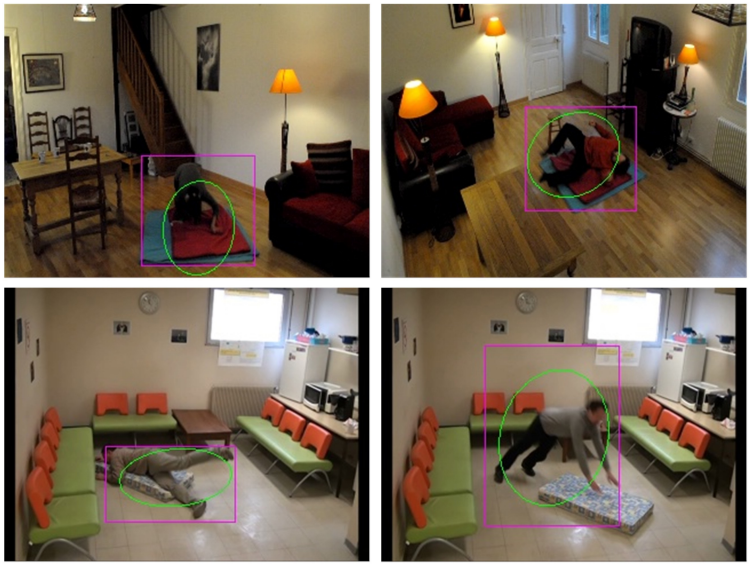


Figure 10. SBFS-based low-level feature selection for the combination $STHF_a = \{X, FD(X)\}$.

these four combinations, $X + FD$ applied to F_s is still giving the best performance at the final decision level as well as at the SVM output which confirms the result of the previous combinations selection procedure.

Still in the context of automatic annotation validation, we evaluated the pre-filtering step which consists to apply oversampling and Gaussian filtering to the feature vector samples. This approach is standard in pyramidal

(a) Manual annotation



(b) Automatic annotation

Figure 11. Dataset with manual (a) and automatic (b) annotations.

Table 3. Fall detection performance (%) using SVM and $STHF_{a\_SBFS}$ descriptor - Protocol $P1$

	Manual annotation										Automatic annotation									
	SVM level performance					Final level performance					SVM level performance					Final level performance				
Combination	*E*	*Sp*	*Ac*	*Pr*	*Re*	*E*	*Sp*	*Ac*	*Pr*	*Re*	*E*	*Sp*	*Ac*	*Pr*	*Re*	*E*	*Sp*	*Ac*	*Pr*	*Re*
$X + FD$	**2.50**	**98.57**	**97.47**	**96.12**	**94.54**	**0**	**100**	**100**	**100**	**100**	**4.94**	**87.36**	**95.22**	**87.36**	**96.41**	**0.38**	**99.61**	**94.23**	**98.00**	**99.69**
$W + FD$	2.91	97.24	97.08	93.40	96.67	0	100	100	100	100	5.04	87.51	94.93	87.51	96.08	0.48	99.51	94.34	96.15	99.69
$X + W + FD$	2.67	97.72	97.32	94.47	96.33	0	100	100	100	100	5.11	87.56	94.86	87.56	95.73	0.77	99.22	93.87	90.19	99.69
$TF + W + FD$	3.04	96.71	96.93	92.28	97.50	0	100	100	100	100	5.11	87.56	94.86	87.56	95.73	0.48	94.11	99.51	96.00	99.79

multiscale signal analysis. As an important result, we got to improve the detection rates. Indeed, at the final decision the recall was clearly increased and the global error was decreased to half when we performed the method based SVM and $STHF_{a\_SBFS}$ using the automatic annotation. These simple operations not only smoothed the automatic annotation errors but also allowed to have better performance when the manual annotation is used (the recall raised from 98% to 100%). Table 4 illustrates the impact of the pre-filtering operations to filter out

detection errors and hence the classification errors and showed the robustness of the automatic annotation as well (the global error is 0,38%). In the best case, using the automatic annotation, only one fall was not detected from a total of 65 videos, regardless the direction of the fall or the position of the body in the scene, and there were three false positives.

Table 4. Fall detection performance (%) using SVM and the $STHF_{a\_SBFS}$ descriptor at the final decision level- Protocol $P1$

	Manual annotation		Automatic annotation	
	Without pre-filtering	With pre-filtering	Without pre-filtering	With pre-filtering
Sp	100	**0**	99.69	**99.69**
Ac	99.90	**100**	99.32	**99.61**
Pr	100	**100**	94.00	**94.23**
Re	98.00	**100**	92.157	**98.03**
E	0.097	**0**	0.67	**0.38**

5.2 Adaboost-based fall detection

Since Adaboost algorithm provides classification rule together with feature selection, the original descriptor used is the full $STHF$, *i.e.* the total number of $d = 32 \times 14 \times 5 = 2240$ values. We evaluated the performance of the method using manual and automatic annotations with and without pre-filtering operations (protocol $P1$). Results are presented in the Table 5.

Using the automatic annotation, performance were slightly lower than the manual ones. Thus, we manage to confirm the robustness of the automatic annotation used for our fall detection system where the error is 0,33% for the manual annotation while it is 0,58% for the automatic tracking. From this table, we noted that the performance were significantly improved thanks to the Gaussian filtering applied to the oversampled low-level features. These operations implied a considerable gain in recall (roughly 2% for manual and automatic annotations). Using manual annotation, the global error rate increased from 0,48% to 0,33% and from 1,07% to 0,58% using automatic annotation. Thus, we experimentaly shown the impact of these operations to filter out the errors of the automatic annotation.

An interesting result is that Boosting classification, despite the simplicity of the decision function, gave good performance. It showed the effectiveness of our spatio-temporal descriptor to distinguish between falls and others activities. Using Boosting classification, from a total of 65 videos, 50 containing falls (Protocole $P1$) only two falls were non detected, without any false positive.

At the final decision level and with pre-filtering operator, the fall detection performance using Adaboost, compared with the performance of the method using SVM classification, decreased for manual (error increased from 0% to 0,33%) and automatic annotations (error increased from 0,38% to 0,58%) . This loss of performance is acceptable considering the possibility of computing time gain (see section 5.4).

Table 5. Fall detection performance using Adaboost and $STHF$ descriptor at the final decision level- Protocol $P1$.

	Manual annotation		Automatic annotation	
	Without pre-filtering	With pre-filtering	Without pre-filtering	With pre-filtering
Sp	99.69	**99.76**	99.38	**99.79**
Ac	99.51	**99.66**	98.92	**99.42**
Pr	94.11	**96.07**	88.46	**95.91**
Re	96.00	**98.00**	90.19	**92.15**
E	0.48	**0.33**	1.07	**0.58**

5.3 Comparison with state-of-the-art

There is unlikely no common fall database available, and so the most of previously reported works create their own databases for evaluation aims, and so it is hard to compare them with our method directly. Rougier[15] obtained a global error rate of 4.6% whereas our error rate is 0.38%, however these rates are not really comparable since the test protocols are different: we evaluated the performance at the slot level, and Rougier evaluated the performance at the video level, which is not really applicable in the real world where video is acquired in a continuous way.

Thome[34] obtained for real falling cases 82% true positives using a multi-view method whereas we had 92% and 90% true positives using respectively SVM and Boosting clasifications, noting that we acquired video data from single camera independently of the camera axis.

Moreover, the choice of the orientation of the camera of Rougier (very wide angle lens and camera fixed on the ceiling) is not suitable to our feature extraction. We evaluated our algorithm using their database, and obtained a global error rate of 4%,with a recall of 73% and a precision of 97.7%. The difference of performance comes mainly from the difference of camera setup : our camera was set at only 2 meters from the ground and used a narrower angle lens. Our setup better captures vertical motion which characterizes falls. This lead us to evaluate the robustness of our method to location change between training and testing, since in real application the final user it is not always able to train the system using fall simulation.

5.4 Robustness to location change

We evaluated our fall detection method using the images acquired in several locations. The three protocols $P1$, $P2$ and $P3$ were used in order to evaluate the robustness of the automatic fall detection method to location change, using Adaboost and SVM classifiers. The results are presented in the Table 6, in terms of classification performance and in terms of computation time of the decision function, per frame. The computation time is here evaluated using C++ implementation of the decision function and a 2.6 GHz based standard PC, without specific optimisation.

For all experiments, the classification results obtained using the protocol $P2$ (training using videos of "Coffee Room" and testing with videos from "office" and "Lecture Room") depicted a significant loss of recall and precision performance in comparison with performance using the protocol $P1$ where all the videos were acquired in the same location.

Indeed, using $P2$, the system has to classify some data acquired using a setup very different from the training, which causes an important number of false positives and false negatives. However, the results of protocol $P3$ show that it is beneficial to add in the training set some videos without fall, acquired in the current location allowing to get close to $P1$ performance. We selected only videos without fall which is realistic since final user can easily record some videos of normal activities at home (the final location) and then update the training. Using this protocol we obtained better performance of the fall detection system. Thus, we prove that it is robust to the location change. These observations are suitable for the five configurations (listed in the table below) that we used in order to find the optimum descriptor in terms of recognition rates and processing time.

Using the $STHF_{a\_SBFS}$ descriptor, the results of protocol $P1$ using SVM (global error rate of 0.38% and recall of 98%) are better than the results of the same protocol $P1$ using SVM and the full descriptor $STHF$ (global error rate of 0.77% and recall of 88.23%). This is the same for the protocol $P3$, using selected features, the global error decreased from 0.86% to 0.38% and the recall increased from 77.41% to 90.62%. As a finding, the optimised descriptor $STHF_{a\_SBFS}$ allows to have better performance by removing the irrelevant features. In the context of robustness to location change, the detection results are very motivating especially for the "SVM after selection" configuration.

Table 6. Automatic fall detection robustness to location change at the final decision level - Descriptor optimisation

	SVM before selection			SVM after selection			Boosting			SVM after Boosting			Boosting after selection		
Initial descriptor	$STHF$			$STHF_{a\_SBFS}$			$STHF$			$STHF_{boost}$			$STHF_{a\_SBFS}$		
Protocol	$P1$	$P2$	$P3$	$P1$	$P2$	$P3$	$P1$	$P2$	$P3$	$P1$	$P2$	$P3$	$P1$	$P2$	$P3$
Sp	99.79	99.05	99.63	99.69	99.70	99.8	99.79	99	99.80	99.88	98.69	99.70	99.69	99.35	99.85
Ac	99.23	98.24	99.13	99.61	99.54	99.61	99.42	98.33	98.42	99.56	97.57	99.28	99.51	98.03	99.35
Pr	95.74	86.84	82.75	94.23	84.84	90.62	95.91	56.41	89.28	97.91	82.92	85.71	94.11	87.87	92.30
Re	88.23	86.84	77.41	98.00	90.32	90.62	92.15	68.75	80.64	94.03	82.92	80.00	96.01	76.31	77.41
E	0.77	1.75	0.86	0.38	0.45	0.38	0.58	1.55	0.57	0.43	2.42	0.71	0.48	1.96	0.64
Time(μs)	8590.47	7901.39	11932.41	248.61	128.16	216.08	3.63	3.27	3.64	42.31	33.90	40.91	2.63	2.56	2.62

The decision time per frame using the $STHF_{a\_SBFS}$ is then decreased (about 40 times) in comparison with the $STHF$ descriptor for the three protocols. Thus, we got an optimised descriptor able to distinguish falls from others activities in a relatively short time.

The performance obtained using Adaboost and $STHF$ descriptor are close to those obtained using SVM and $STHF_{a\_SBFS}$. Even if the recall is only 80,64% for the Adaboost-based method, the performance still promising especially regarding the processing time. Indeed, the processing time using SVM is about 30 times longer than Adaboost decision. This difference is due to decision function complexities of these two algorithms, given in section 4.3. It should also be noted that the $STHF$ descriptor used for Boosting based fall detection requires the computation of all transformations and all the low-level features. This may slow the global system down (the computation time of these transformations is 0.56 ms/frame for $STHF$ and 0.3 ms/frame for $STHF_{a\_SBFS}$). We defined then the configurations "SVM after Boosting" and "Boosting after selection" using respectively $STHF_{boost}$ (the descriptor built using the feature selection step performed in the Adaboost training process) and the $STHF_{a\_SBFS}$ descriptor. The general performance (protocol $P1$) using Adaboost and the optimised descriptor $STHF_{a\_SBFS}$ are close to the performance of SVM fed by the full descriptor, especially for the protocol $P1$. Using this configuration (Boosting after selection), the method is still robust to location change (protocol $P3$).

6. CONCLUSION

We presented in this paper a method for real-time fall detection of one elderly person living alone, using only one camera, based on a spatio-temporal descriptor.

We evaluated the robustness of our method using a realistic dataset and we evaluated the ability of some standard transformations to improve the classification performance. We built a high-dimensional spatio-temporal descriptor named $STHF$ in order to raise maximum informations about events and to use a powerful classifier which enables the variation of these features to be taken into account.

We also introduced a new metric allowing to evaluate the event detection at the slot level, taking into account a tolerance (0.5 s) on the instant of the detection which is not critical for real application.

We evaluated SVM and Adaboost algorithms for classification followed by final decision step. During this study, we were brought to build experimentally an optimised descriptor that allows to have good performance in terms of classification rates and time processing.

For the SVM-based method, we built the descriptor $STHF_{a\_SBFS}$ after feature and combination of transformations selection methods. This descriptor is obtained combining seven low-level features with their first derivative, *i.e.* their velocity, automatically extracted and tracked using robust algorithm.

Using Adaboost classification, we reached promising tradeoff between classification performance and time processing. This method is more suitable for real-time (eventually hardware) fall detection application thanks to its simple decision function.

Hence, we evaluated the robustness of our method regarding location changes. We proposed a realistic and pragmatic protocol which enables performance to be improved by updating the training in the current location, with normal activities records.

Evaluating the method using $P1$ and $P3$ protocols, the global error rate is lower than 1%. This error seems acceptable to real applications of fall detection using only one camera.

We plan in the next future to implement the full process in a smart camera, as well as the adaptive compression scheme described in the introduction.

REFERENCES

1. C. for Research and P. of Injuries-CEREPRI, *Fact sheet: Prevention of Falls among Elderly*, Europeen Network for Safety among Elderly. http://www.euroipn.org/eunese/factsheets.htm.
2. J. Liu and J. Yang, "Action recognition using spatiotemporal features and hybrid generative/discriminative models," *Journal of Electronic Imaging* **21**(2), pp. 023010–1–023010–10, 2012.
3. H. Jain, A. Chatterjee, S. Kumar, and B. Raman, "Recognizing human gestures using a novel svm tree," pp. 83000M–83000M–9, 2012.

4. Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic vs. max-margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(7), pp. 1310–1323, 2011.
5. N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. Laighin, V. Rialle, and J. Lundy, "Fall detection - principles and methods," in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2007)*, pp. 1663–1666, 2007.
6. B. Toreyin, Y. Dedeoglu, and A. Cetin, "Hmm based falling person detection using both audio and video," *Signal Processing and Communications Applications, 2006 IEEE 14th* , pp. 1–4, 2006.
7. D. "Anderson, J. Keller, M. Skubic, X. Chen, and Z. He, ""recognizing falls from silhouettes"," in *"Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE"*, pp. 6388–6391, 2006.
8. V. Vishwakarma, C. Mandal, and S. Sural, "Automatic detection of human fall in video," in *Proceedings of the 2nd international conference on Pattern recognition and machine intelligence*, *PReMI'07*, pp. 616–623, Springer-Verlag, (Berlin, Heidelberg), 2007.
9. J. Willems, G. Debard, B. Vanrumste, and T. Goedem, "A video-based algorithm for elderly fall detection," in *World Congress on Medical Physics and Biomedical Engineering*, **25/5**, pp. 312–315, 2009.
10. M. Khan and H. Habib, "Video analytic for fall detection from shape features and motion gradients," in *Lecture Notes in Engineering and Computer Science*, **2179**, pp. 1311–1316, 2009.
11. G. Wu, "Distinguishing fall activities from normal activities by velocity characteristics," *Journal of Biomechanics* **33**(11), pp. 1497–1500, 2000.
12. H. Nait-Charif and S. McKenna, "Activity summarisation and fall detection in a supportive home environment," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, *ICPR '04*, pp. 323–326, IEEE Computer Society, (Washington, DC, USA), 2004.
13. J. Tao, M. Turjo, M. Wong, M. Wang, and Y. Tan, "Fall incidents detection for intelligent video surveillance," *Information, Communications and Signal Processing, 2005 Fifth International Conference on* , pp. 1590–1594, 2005.
14. H. Foroughi, A. Rezvanian, and A. Paziraee, "Robust fall detection using human shape and multi-class support vector machine," *Sixth Indian Conference on Computer Vision, Graphics and Image Processing* , pp. 413–420, 2008.
15. C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *Circuits and Systems for Video Technology, IEEE Transactions on* **21**, pp. 611 – 622, 2011.
16. C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "3d head tracking for fall detection using a single calibrated camera," *Image and Vision Computing* , 2012.
17. C. Liu, C. Lee, and L. P.M., "A fall detection system using k-nearest neighbor classifier," *Expert systems with applications* **37**, pp. 7174 – 7181, 2010.
18. Y. Liao, C. Huang, and S. Hsu, "Slip and fall event detection using bayesian belief network," *Pattern recognition* **45**, pp. 24 – 32, 2012.
19. F. D. M. de Souza, G. C. Chvez, E. A. do Valle Jr., and A. de Albuquerque Arajo, "Violence detection in video using spatio-temporal features," in *SIBGRAPI'10*, pp. 224–230, 2010.
20. V. Vapnik, ed., *The Nature of Statistical Learning Theory*, Springer-Verlag, springer-verlag ed., 1995.
21. Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," 1996.
22. L. Li, W. Huang, G. I., and Q. Tian, "Foreground object detection from videos containing complex background," in *In MULTIMEDIA 03: Proceedings of the eleventh ACM international conference on Multimedia*, pp. 2–10, ACM Press, 2003.
23. Y. Yong and A. B. S.A., "Integration of projection histograms and linear prediction for object tracking," *Jurnal Teknologi* **39(D)**, pp. 57–68, 2003.
24. G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools* , 2000.
25. F. Smach, J. Miteran, M. Atri, J. Dubois, M. Abid, and J. Gauthier, "An fpga-based accelerator for fourier descriptors computing for color object recognition using svm," *Journal of Real-Time Image Processing* **2**(4), pp. 249–258, 2007.

26. S. Parfait, P. Walker, G. Crehange, X. Tizon, and J. Miteran, "Classification of prostate magnetic resonance spectra using support vector machine," *Biomedical Signal Processing and Control* **2011**(0), pp. 1–8, 2011.
27. S. Chaplot, L. M. Patnaik, and N. Jagannathan, "Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network," *Biomedical Signal Processing and Control* **1**(1), pp. 86–92, 2006.
28. I. Daubechies, *Ten lectures on wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.
29. J. Miteran, S. Bouillant, M. Paindavoine, F. Meriaudeau, and J. Dubois, "Real-time flaw detection on a complex object: comparison of results using classification with a support vector machine, boosting, and hyperrectangle-based method," *Journal of Electronic Imaging* **15**(1), pp. 013018–013018–9, 2006.
30. C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology* **2**, pp. 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
31. P. Viola and M. Jones, "Robust real time face detection," *International Journal of Computer Vision* **57**(2), pp. 137–154, 2004.
32. J. Miteran, E.-B. Matas, J. an Bourennane, M. Paindavoine, and J. Dubois, "Automatic hardware implementation tool for a descrete adaboost-based decision algorithm," *EURASIP Journal on Applied Signal Processing* **7**, pp. 1035–1046, 2005.
33. A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **19**, pp. 153 –158, feb 1997.
34. 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* **18**, pp. 1522–1532, 2008.