CHALMERS

Identification of human fall events from using RGB-D videos

Master's Thesis in Signals and Systems

DURGA PRIYA KUMAR

Department of Signals and Systems
Communication Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015
Master's Thesis EX071/2015

MASTER'S THESIS EX071/2015

Identification of human fall events from using RGB-D videos

DURGA PRIYA KUMAR

Supervisor and Examiner: Prof. Irene Yu-Hua Gu

Department of Signals and Systems
Division of Signal Processing
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Identification of human fall events from using RGB-D videos
DURGA PRIYA KUMAR

© DURGA PRIYA KUMAR, 2015.

Supervisor: Irene Yu-Hua Gu, Department of Signals & Systems
Examiner: Irene Yu-Hua Gu, Department of Signals & Systems

Master's Thesis EX071/2015
Department of Signals and Systems
Division of Signal Processing
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2015

Abstract

Fall detection and identification is essential for elderly care applications. In this thesis, video analysis techniques are investigated for detection of falls using RGB-D sensor. The focus is to detect the human object from videos and study the contribution of different features in fall classification. Speeded Up Robust Features detection by thresholding the difference image is used to locate the key points of the human. The key points are used to define the region of interest from which features are extracted. The spatio-temporal features studied are Histogram of Oriented Gradients, optical flow and shape analysis. The extracted features are tested using SVM classifier to distinguish fall and lying down events. The experiments are conducted on 800 RGB-D videos performed by 19 subjects. The results obtained are good for almost all features (about 95% classification rate).

Keywords: HOG, Shape analysis, Optical flow, SURF, C-SVM, fall detection, elderly care, RGB-D videos

Acknowledgements

I would like to thank my supervisor and examiner Professor. Irene Yu-Hua Gu for the valuable support she was through the working of the thesis. I would like to acknowledge the Signal Processing research group at Signals and Systems, Chalmers University of Technology for providing the video dataset which was helpful in the thesis experiments. I am also grateful to Yixiao Yun, a Ph.D. student at Signal Processing group for helping with the optical flow feature.

This publication is part of my master thesis work at Chalmers University of Technology, thanks to a Swedish Institute scholarship. Finally, I would like to extend my gratitude to my family and friends for being a strong pillar of support.

Durga Priya Kumar

Contents

1 Introduction **1**
1.1 Objective of this thesis 2
1.2 Scope 3
1.3 Outline of the report 3

2 Theory and methods **4**
2.1 Detect foreground object 4
2.1.1 Background subtraction 4
2.1.2 Morphological operations 5
2.2 SURF key points detection 5
2.3 Features 6
2.3.1 Histogram of Oriented Gradients 6
2.3.2 Optical flow 8
2.3.3 Shape feature 10
2.4 SVM classification 12

3 Work in this thesis **14**
3.1 Object detection and tracking methods 15
3.1.1 Background subtraction by using the difference image 15
3.1.2 Apply morphological operations to find object contour 15
3.1.3 Object tracking with bounding box by using SURF key points 16
3.1.4 Append zeros outside bounding box to form the region of interest of size $(a \times a)$ 16
3.2 Extract features from the region of interest 17
3.2.1 Window averaging to obtain time dependent features 17
3.2.2 HOG feature vector 19
3.2.3 Optical flow feature vector 19
3.2.4 Shape feature vector 21
3.2.5 Normalize feature values 23

3.2.6 Combine all features in series 24
3.3 Design of a classifier 25
3.3.1 Performance measures 26

4 Experimental results **28**
4.1 RGB-D video datasets 28
4.2 Setup 32
4.2.1 HOG features 32
4.2.2 Optical flow features 33
4.2.3 Shape features 35
4.3 Evaluation 36
4.3.1 Case study-1 36
4.3.2 Case study-2 37
4.4 Results from combined 3 types of feature vectors 39
4.5 Tests to evaluate parameter settings in algorithms 40
4.5.1 Comparison of optical flow features 40
4.5.2 Shape feature components evaluation 41
4.5.3 Impact of M value to compute time-dependent feature vectors 42
4.5.4 Optimal ROI size $(a \times a)$ for feature extraction 43
4.5.5 Varying metric threshold $SURF_{thres}$ in SURF 44

5 Conclusions **46**

Bibliography **49**

List of Figures **51**

List of Tables **53**

1

Introduction

Falls are a major cause of fatal injuries in elderly population (> 70 years of age) [1]. Based on WHO statistics [2], accidental falls affect 32% - 42% of the elderly population each year. Elderly people experience frequent falls as they are infirm and weak from age-related biological changes. According to UN global statistics, almost 40% of the old people currently live independently [3]. The rise in aged population is expected to be more than 2 billion people in 2050. For such a growing elderly population, assistive devices are needed to lead an independent life. Such devices help in the detection and reporting of fall accidents. Hence, healthcare and surveillance industries are investing in camera-based fall detection systems which provides timely medical assistance [4]. The elderly population also welcomes the idea of fall detection systems for their improved security and safety at home.

The current research works in the field of fall detection and classification are as follows. Rougier et al. [5] uses shape deformation during a video sequence to detect falls and shape matching to track the person's silhouette. Shape analysis methods are used to quantify the deformation and for classification Gaussian mixture model is used. Mastorakis et al. [6] uses Kinect sensor inputs to detect fall, based on the velocity and inactivity calculations. The contraction and expansion of 3D bounding box in terms of width, height and depth is used to measure the velocity. Both [5] [6] depend on the lack of inactivity after post-fall phase. Zhang et al. [7] uses 3D RGB-D videos for analysis to protect the person's identity. The person tracking switches to RGB video when the person is out of range from 3D camera. The analysis employs hierarchy classification to robustly recognize 5 activities.

Most of the studies have suggested different features with similar results, but no comparison has been made to find the best-suited method [8]. Also, the number of volunteers involved in tests are comparatively low to use the results for real-life scenarios. A robust fall detection system must classify fall as fall and any other activity as non-fall to use in real life situations. If it classifies fall as non-fall it is termed as miss and in such case the medical response cannot be made on time. Similarly, if non-fall is classified as fall it is termed as false alarm and such systems becomes undependable and useless. Some of the studies does not consider about the privacy of a person in the videos when designing systems for fall detection. Few detection systems are based on inactivity of the person in the post-fall phase. But a long lie (inactivity) is associated with high mortality rates among the elderly, hence a reliable fall detector must diminish the long lie. The upcoming fall detection systems must provide better solutions to overcome the mentioned drawbacks. It is difficult to design such reliable and robust systems. Most of the commercial products in the market are not as widely used and have no impact on elderly independent living [9].

The current project tries to find working solutions for the researched problem of fall detection and classification by using computer vision and machine learning. In this thesis, RGB-D videos which are captured from the cameras and infrared sensors are used for fall detection. The object needs to be tracked to detect falls and relevant features that describe the activity are later extracted. These extracted features are used in activity classification to determine the type of activity the feature belongs to.

1.1 Objective of this thesis

The thesis aims to detect and classify falls using videos with less false alarm and misses. The video dataset made in university campus containing the fall and lying down activities are provided by Signal Processing research group at Chalmers University of Technology. Each video contains a single person performing the activity of either fall or lying down. The long video needs to be cropped to shorter activity events, which is done manually by using video editing software. The object in the video foreground describes the detail of any activity. So it's necessary to use object detection for tracking the single person in the videos. The tracked objects are used to find the region of interest (ROI). The next step is to find prominent features that describe the activity from the detected region.

The features need to be as distinct as possible to differentiate fall from another confusion class lying down. The goal is to extract features like the Histogram of Oriented Gradients (HOG), optical flow and shape analysis for feature extraction. Finally, the features are used as inputs to a classifier for activity identification. In this thesis, support vector machine (SVM) based on supervised learning is used for the classification. The trial and error method is used to find the parameter settings with high classification rate. The process setup containing the modules of object detection, feature extraction and activity classification are implemented in MATLAB software.

1.2 Scope

The study can be extended for various other video activities (apart from fall and lying down) to observe human behavior that could help automation of elderly care. The current work focuses on different features and its respective parameters to enhance the classification rate, while for future a comparative study between different classifiers can be done. Such study will identify which classifier gives optimal learning solution for the classification of human activities.

1.3 Outline of the report

The thesis report contains the following sections: **Section 2** details basic theory and methods that are related to the thesis work. **Section 3** describes the methods and block diagram used in this thesis. **Section 4** includes the experimental results and evaluation. Finally, conclusions are given and future work is discussed.

2

Theory and methods

This section reviews some basic theory and mathematical concepts behind image processing methods that are useful in this project. The algorithms mentioned here are used for fall detection and classification. This section is divided into 3 parts detailing the algorithm for object detection and tracking, defining the features and classifying them. The first part on object detection covers background subtraction, morphological operations and Speeded-Up Robust Features (SURF) detector. The second part is about features such as HOG, Optical flow and Shape analysis. The final part covers the supervised learning algorithm of SVM to classify the features.

2.1 Detect foreground object

Background subtraction is one way of detecting the objects in the foreground. The morphological operations when used on images result in object contours which are useful for detection. To define the foreground object with boundary points, its relative interest points need to be detected. SURF detection algorithm is a useful method for key point detection of an object.

2.1.1 Background subtraction

A fixed camera captures the RGB-D video sequence of the object with a background. The difference frame I_{diff} is found between two consecutive frames to detect the foreground object (human). It is extracted from the RGB video for successive frames I_{RGB} until the last frame is reached.

$$I_{diff,n} = I_{RGB,n+1} - I_{RGB,n} \tag{2.1}$$

where n^{th} frame ranges from [1, *len*-1] and *len* - number of frames in a video. It is not always possible to have a stationary background in real environments. Factors like indoor reflections and illumination changes affect the captured frames. Yet, background subtraction is a simple method which is useful in moving object detection.

2.1.2 Morphological operations

Morphological operations are a useful mathematical tool for shape based image processing. The two basic operations are erosion and dilation [10].

Erosion and Dilation: The process of erosion shrinks the object in an image while dilation grows the object within an image. Consider $f(x,y)$ a greyscale image and $b(x,y)$ an operator used in morphological process. The erosion output $[f \ominus b]$ uses the operator $b(-x,-y)$ for filtering, while dilation output $[f \oplus b]$ uses $b(x,y)$ as the operator. The functions are given in (2.2) and (2.3).

$$[f \ominus b](x,y) = \min_{(s,t)\in b} \{f(x+s,y+t)\} \tag{2.2}$$

$$[f \oplus b](x,y) = \max_{(s,t)\in b} \{f(x-s,y-t)\} \tag{2.3}$$

Closing and Opening: Closing $[f \bullet b]$ operator fills in the holes within an image, while opening $[f \circ b]$ operator separates the connected objects in an image. These operations given in (2.5) and (2.4) are derived from the erosion and dilation process.

$$[f \bullet b] = (f \oplus b) \ominus b \tag{2.4}$$

$$[f \circ b] = (f \ominus b) \oplus b \tag{2.5}$$

2.2 SURF key points detection

Key or corner point detectors like Scale Invariant Feature Transform (SIFT) and SURF are used to find the interest points in an image. SURF detection is computationally faster than the SIFT method.

The 'Fast-Hessian' detector from SURF is based on scale-space analysis of Hessian matrix [11]. The second order derivatives of Gaussian function $L_{xx}(\mathbf{p},\sigma)$, $L_{xy}(\mathbf{p},\sigma)$, $L_{yy}(\mathbf{p},\sigma)$ defines the Hessian matrix $\mathcal{H}(\mathbf{p},\sigma)$ as shown in (2.6). The image I at point $\mathbf{p} = (x,y)$ is convolved with a given Gaussian function of scale σ to obtain the derivatives. The Gaussian derivatives are discretized and cropped for practical applications, instead the SURF detector uses filter approximation for Gaussian second order derivatives (D_{xx}, D_{yy} and D_{xy}). The Hessian determinant modified after including filter approximation is given in (2.7).

$$\mathscr{H}(\mathbf{p},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{p},\sigma) & L_{xy}(\mathbf{p},\sigma) \\ L_{xy}(\mathbf{p},\sigma) & L_{yy}(\mathbf{p},\sigma) \end{bmatrix} \tag{2.6}$$

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.7}$$

For scale-space analysis in SURF detector, it uses integral images ($I_{\sum}(\mathbf{p})$) and filter approximations to avoid the smoothing of images at every level in image pyramids. This is possible because the filter can easily be up-scaled to larger filter sizes 9×9, 15×15, 21×21, 27×27 at every scale. The Gaussian approximations (σ) of the scaled filters also scale accordingly. For example, the 4^{th} scale filter of size 27×27 has a corresponding $\sigma = 3 \times 1.2 = 3.6 = s$ [11]. The scales are grouped into octaves and at larger scales for higher octaves the filter step sizes are doubled (i.e. 15×15, 27×27, 39×39, 51×51 for 2^{nd} octave).

To detect the interest points from the image, a 3×3×3 neighbourhood non-maximum suppression is done to localize the image over scale and space. The maxima of Hessian matrix determinant is interpolated in both scale and image space. This interpolation locates the interest points as detailed in [11]. The detectors are used in applications of object recognition involving humans or face details and to track moving targets.

2.3 Features

The success of any activity classification depends on the prominence and distinctiveness of its features. For two actions to be distinct, it is mandatory to have similar intra-class features and distinctive inter-class features. The properties of such extracted features have to be robust, cost efficient, orientation invariant and insensitive to disfigurement. Lastly, feature extraction from an image helps in reducing dimensionality. The transform values of the image or newly defined feature functions achieves feature reduction. A detailed description on HOG, optical flow, shape analysis features are given in the next sections.

2.3.1 Histogram of Oriented Gradients

HOG descriptors [12] describe the pixel distribution (histogram of intensity values) based on contour orientations (edge directions).

The image processing in HOG descriptor according to [12] is given and its process flow is as shown in the Figure 2.1. RGB colour space is preferred and the pixel's gradient is assigned from the largest norm gradient between three colour channels. A simple gradient filter [-1, 0, 1] with $\sigma = 0$ is the best option for filtering, as convolving with larger or smoothing masks decreases the performance.

The method considers '*cells*' as small spatial regions that divides the image window and '*blocks*' as a larger spatial region containing the cells. The block shape is either a rectangle (R-HOG) or a circle (C-HOG), but R-HOG is chosen as the spatial subdivision into *blocks* and *cells* are easier. The R-HOG contains non-overlapping cells of size $m \times m$. An edge orientation histogram is formed for each cell, with histogram bins in the range of 0° to 180° where the gradient is unsigned. The number of bins in the histogram is 9 thus the range has a unit scale value of 20°.

Each pixel in a local cell associates itself with an orientation based on its gradient element. The vote for the histogram is a function of pixel's gradient magnitude. The cells are grouped into

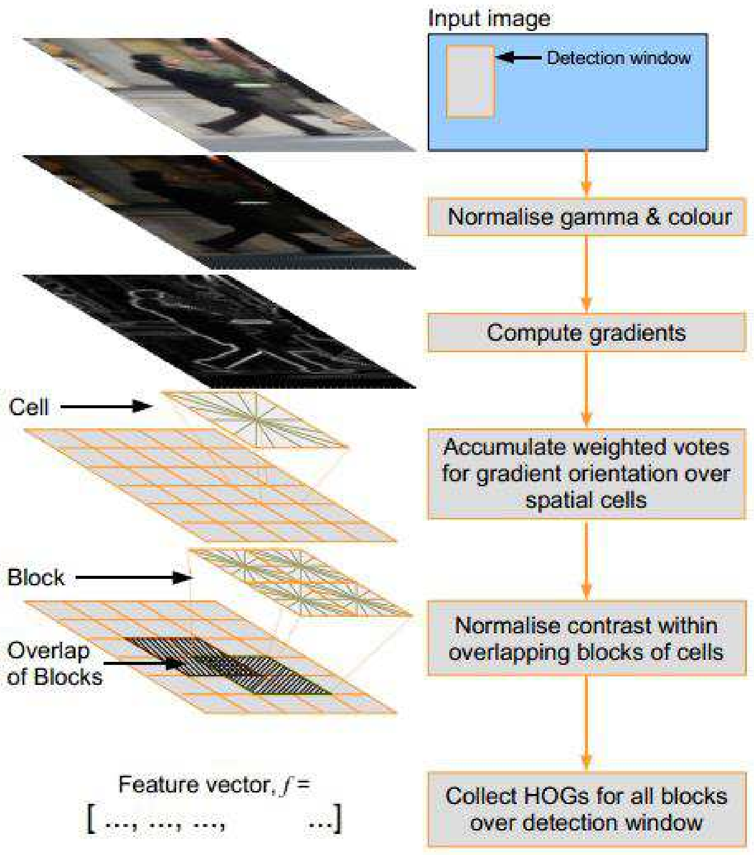

Input image
Detection window
Normalise gamma & colour
Compute gradients
Cell
Accumulate weighted votes for gradient orientation over spatial cells
Block
Overlap of Blocks
Normalise contrast within overlapping blocks of cells
Feature vector, f =
[..., ..., ..., ...]
Collect HOGs for all blocks over detection window

Figure 2.1: Steps involved in the processing of Histogram of Oriented Gradients from [13]

blocks of size $r \times r$, and the blocks are overlapped at half the block size. The optimal value for cell size is 8×8 and block size is 2×2, as 6-8 wide pixel cells imitate human anatomy in an image. A trade-off is observed when defining the block size, as large values ignore local content and smaller value suppresses global details.

Normalization is carried out to remove local range variations and it uses block-based *L2-Hys* method. It clips the L2 normalized value which is given as $\mathbf{v}/\sqrt{||\mathbf{v}||_2^2 + k^2}$, where $||\mathbf{v}||_2$ is the $L2$-norm of descriptor vector $\mathbf{v}$ and $'k'$ is a regularization constant. When descriptors are not normalized or when $L1$-norm is used instead of $L2$, detection performance reduces drastically.

A detection window is used to compute the descriptor, which also includes margin information around the object that provides extra context to the person. The reduction in margin size or an increase in the person size results in border reduction, giving loss of performance. Instead of a soft linear SVM, a Gaussian kernel SVM increases performance. The HOG descriptor is often used for human detection.

2.3.2 Optical flow

Optical flow is the pattern of apparent motion that is contained in a visual scene. To estimate the optical flow certain assumptions are made [14]. The mostly followed assumption is '*Grey value Constancy*' in which the pixel intensity does not change as it flows between frames. Consider $f(x,y,t)$ is the intensity of the pixel (x,y) at time 't', if the flow is $[u(x,y,t), v(x,y,t)]$ then the assumption is written as:

$$f(x,y,t) = f(x+u, y+v, t+1) \tag{2.8}$$

Taylor expansion is applied to linearize the above equation.

$$f(x+u, y+v, t+1) = f(x,y,t) + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + 1\frac{\partial f}{\partial t} \tag{2.9}$$

From which the '*Optical flow constraint*' is obtained, given in below equation:

$$u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} = 0 \tag{2.10}$$

But the *Grey value Constancy* assumption leads to an ill-defined problem, as two unknowns (u and v) have to be solved from a single equation. To resolve the problem, two different methods that are either local or global in approach are used. Lucas and Kanade (LK) came up with a localized solution which considers a smaller image neighbourhood of size ρ where the assumption is true. Horn and Schunuck (HS) proposed a global differential method which considers the assumption to be true for the entire image, where dense flow estimates are obtained with a regularization parameter α.

A hybrid technique that combines robustness of local LK method and the density of global HS method is used for optical flow estimation. To obtain a combined local-global (CLG) method, a multi-resolution approach is followed that adopts coarse-to-fine warping techniques [15]. The details of these solutions are unrelated to this thesis work and are defined from [14], [15]. The optical flow implementation used in this project is available from [16]. The optical flow vectors are very large for high-resolution frames. So the optical flow is defined as a characteristic distribution instead of using the entire flow vector as the motion feature descriptors.

Histogram of Oriented Optical flow: Optical flow vector uses a characteristic distribution in terms of magnitude or direction to describe its features. The orientation is based on flow vector $\mathbf{v} = [v_x, v_y]$ with respect to horizontal axis computed from $\theta = \tan^{-1}(\frac{v_y}{v_x})$. To make it invariant to direction (left to right or vice versa) the orientation range is considered to be,

$$-\frac{\pi}{2} + \pi\frac{b-1}{B} \leq \theta < -\frac{\pi}{2} + \pi\frac{b}{B} \tag{2.11}$$

where B is the number of bins used in the histogram which is user variant input. The value added to b^{th} bin ($1 \leq b \leq B$) in a histogram is the sum of vector magnitudes $\sqrt{{v_x}^2 + {v_y}^2}$ which is later normalized. This feature is called as Histogram of Oriented Optical flow (HOOF) introduced in [19].

HOG of Optical flow: The optical flow (OF) vector $\mathbf{v} = [v_x, v_y]$ is characterized as an RGB colour image. A HSV (Hue-Saturation-Value) model based color-coding converts the optical flow vector to an RGB image. At each pixel, the flow direction OF_{phase} is coded as hue while the flow magnitude OF_{mag} is coded as saturation.

$$OF_{phase} = \tan^{-1}(\frac{v_y}{v_x}) \tag{2.12}$$

$$OF_{mag} = \sqrt{{v_x}^2 + {v_y}^2} \tag{2.13}$$

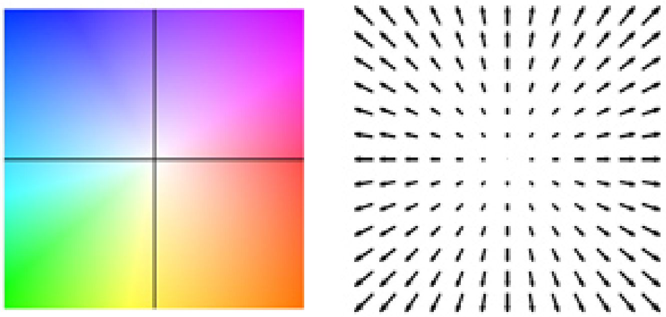


Figure 2.2: The visualization of flow fields from [17]. The color code in [18] is used to visualize a flow field: each pixel denotes a flow vector where the orientation and magnitude are represented by the hue and saturation of the pixel respectively.

The color-coding scheme is given in [18] and it is shown in the Figure 2.2. The optical flow fields are color-coded to result in an RGB image I_{HSV}. The colour of the image shows the direction of object motion and the saturation shows the magnitude of its motion. After finding the HSV based RGB image, to define its features the HOG descriptor (see Section 2.3.1) is used. This HOG feature on HSV image I_{HSV} is called as HOG of Optical flow (HOGOF).

2.3.3 Shape feature

The shape feature in [20] is described below. Shape representations such as the centre of gravity, eccentricity, orientation, aspect ratio, extrema points and the distance between boundary points and centroid are explained.

Centre of gravity: Centroid or centre of gravity is a fixed position (g_x, g_y) in the region with N number of points and is given by,

$$\begin{aligned} g_x &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ g_y &= \frac{1}{N}\sum_{i=1}^{N} y_i \end{aligned} \tag{2.14}$$

such that, $(x_i, y_i) \in \{(x_i, y_i) | f(x_i, y_i) = 1\}$. It's illustrated in the Figure 2.3.

Eccentricity: It is the ratio of lengths between the major axis and minor axis in an elliptical boundary. A minimum bounding rectangle method is used to determine the eccentricity (V) of a symmetric object as shown in the Figure 2.3.

$$V = \frac{Major\ axis\ length}{Minor\ axis\ length} = \frac{a}{b} \tag{2.15}$$

Minimum bounding rectangle: The smallest bounding box (BB) is in the shape of a rectangle

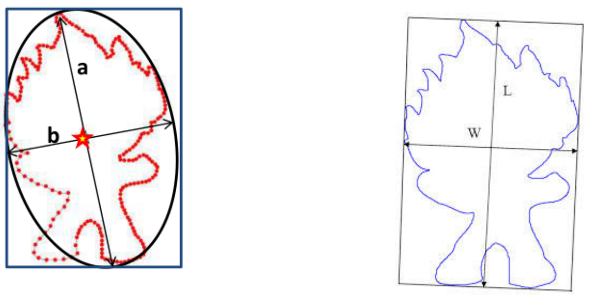


Figure 2.3: Shape feature analysis from [20]. Left: Centroid and Eccentricity. Right: Minimum bounding rectangle.

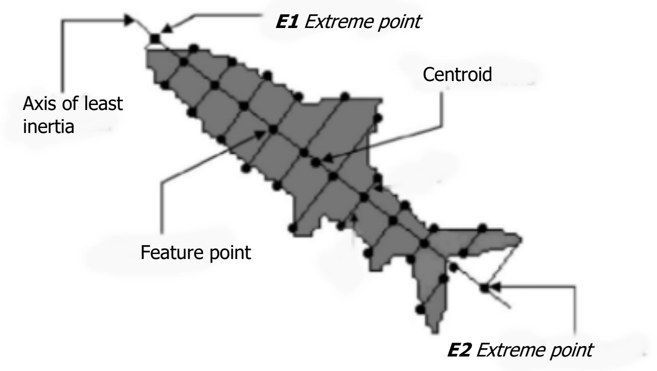


Figure 2.4: Symbolic features (Extrema points) based axis of least inertia from [20]

that contains the entire object points within it as shown in the Figure 2.3. The formulation of aspect ratio of this method is given as:

$$Aspect\ Ratio = \frac{Length}{Width} = \frac{L}{W} \tag{2.16}$$

Orientation: Orientation gives the angular value between the major axis of the ellipse to its x-axis. The values range from $-90°$ to $90°$, where the ellipse has same second moments as the bounding box.

$$\theta = \tan^{-1} \frac{Major\ axis\ length}{x\text{-}\ axis\ length} \tag{2.17}$$

The shape representations cannot be used as standalone shape descriptors. The shape descriptors also include other parameters like distance function and its derivatives to discriminate shapes.

Extrema Points: The longest axis passing through the centroid of a shape is defined as the reference, to compute squared distances between the axis and all extrema points on the outer boundary as illustrated in the Figure 2.4. Among the projected points, 8 farthest points away from the axis is selected as the Extrema points $E1$, $E2$, $E3$, $E4$, $E5$, $E6$, $E7$ and $E8$. This feature is invariant to transforms, rotation and translation. Extrema points along with other mentioned shape features are available in MATLAB function '*regionprops*'.

Boundary point to centroid distance: It is defined as the Euclidean distance between the extrema in the boundary to the centroid. It is translation invariant and is defined as:

$$d(n) = \sqrt{(x(n) - g_x)^2 + (y(n) - g_y)^2} \tag{2.18}$$

The derivative of distance function could be used to define the trajectory of the moving object and is formulated as the difference in distances between consecutive frames.

2.4 SVM classification

A Support Vector Machine is a useful method for data classification. A non-linear function is used to map the input feature vectors to a high-dimensional feature space, where the hyperplane is defined. An optimal hyperplane [21] to separate classes is a decision function $D(x)$ with the maximal margin between feature vectors as shown in Figure 2.5. A small part of the input data called *support vectors* determines the margin function that has a maximum width of separation. SVM requires training data to construct a hyperplane and using the decision function it classifies the features of testing data. The training data contains two components: one is the class label and other is the feature vector.

In the Figure 2.5 the support vectors in both classes lie on the *'dashed line'* which represent the margin function while the decision function is the *'solid line'* that divides the two classes. The decision function decides the feature class, from the figure $D(x) > 0$ is labelled as class (+1) marked in '*blue*' and $D(x) < 0$ as class (-1) marked in '*red*'. Supervised training for the SVM uses the labelled training data. The use of support vectors to find optimal hyperplane (with maximal margin) also makes the computation faster.

***C*-Support Vector Machine for Classification:** C-SVM constructs a soft margin separating hyperplane for classifying features into respective class labels. Given training vectors $\mathbf{x_i} \in \mathscr{R}^n$, $i = 1,...,l$ in two classes, and a label vector $\mathbf{y} \in \mathscr{R}^l$ such that $y_i \in \{1, -1\}$ C-SVM [21]; [23] solves for minimum generalization error,

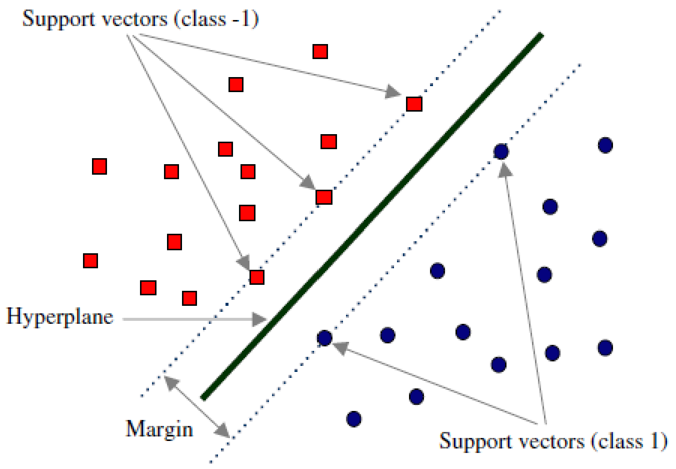


Figure 2.5: An example of a separable problem in a 2-dimensional space from [22]. The support vectors, marked on *dashed line*, define the margin of largest separation between the two classes.

$$\begin{aligned} \min_{\mathbf{w},b,\varepsilon} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1,...,l \end{aligned} \tag{2.19}$$

where $\phi(\mathbf{x}_i)$ maps $\mathbf{x_i}$ into a higher-dimensional space, $\mathbf{w}$ is a weight vector, b is a bias, $C > 0$ is a regularization coefficient, and ξ_i is a slack variable. The algorithm to find the soft margin hyperplane is detailed in [23]. The decision function $D(\mathbf{x})$ that defines the hyperplane is given in (2.20) where *sgn* is the sign function,

$$D(\mathbf{x}) = sgn(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \tag{2.20}$$

In this thesis, SVM is used as a tool for classification. To implement the classifier module a kernel function is used, to map the features into high dimensional space. *C*-Support Vector Classification from LIBSVM package [24] is used in this project.

3

Work in this thesis

This chapter describes in detail the methods and algorithms used for fall detection and classification. The project flow diagram is shown in the Figure 3.1. The process begins with RGB-D video inputs given to object detection techniques. Object detection is followed by tracking the object and later using it to find the region of interest. ROI is given as the input for feature extraction that uses spatio-temporal features. The SVM classifier takes feature inputs and classifies them to respective class labels.

In this project RGB-D videos that capture human activity is used as the input. The object detection is carried out by using difference image and object contour from morphological operations. The detected object is tracked with a bounding box using SURF key points and the BB is modified to the region of interest. The ROI is then normalized to fixed size and used as the input for feature extraction. The three features used in this thesis are HOG, optical flow and shape. The features are normalized and fixed to a constant interval. The feature vectors are used in C-SVM classifier to identify the classes by binary classification. At last, the classification performance and evaluation are given.

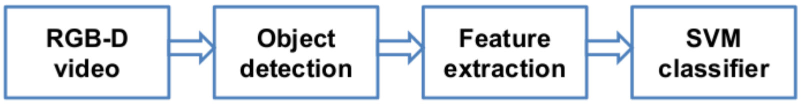


Figure 3.1: Flow diagram of modules used for fall detection and classification

3.1 Object detection and tracking methods

3.1.1 Background subtraction by using the difference image

The difference image I_{diff} defined in Section 2.1.1 is useful for object detection. When taking the frame difference between consecutive frames the stationary background is subtracted, leaving the region of a moving object. This is possible as the camera is fixed and it captures a moving object. The object needs to be moving, if the object is also stationary it results in zero detection. The foreground object detection is not always possible in real environments, as the background is affected by factors like reflections and illumination changes. In this method, RGB difference image is used as it has fewer background disturbances. The depth difference image is avoided as it includes more background noise. Though RGB difference image is better than depth, it suffers from reflection and illumination changes. Also when the person or object in the video wears black colour clothing that is similar to the background, it results in partial object detection. Hence, only either the upper or lower part of the human is detected.

3.1.2 Apply morphological operations to find object contour

The object contour is obtained using morphological operations on depth frames I_{Depth} (greyscale images). In this method, the opening operators discussed in Section 2.1.2 are used to define morphological skeleton I_{skel}. The outer boundary points of this skeleton are used as the object contour I_{shape}. The above steps are implemented by using '*bwmorph*' function in MATLAB as shown in the Algorithm 1. The depth images are considered for morphological operators in MATLAB, as the '*bwmorph*' function performs on a binary or greyscale image. The output frames containing the object contour also contains some background clutter, which needs to be removed before feature definition.

Algorithm 1 Object contour by morphological operations in Depth video

```
for every Depth video with frames $I_{Depth}$ do
    for n = 1 to len, where 'len' - number of frames in a video do
        function BWMORPH($I_{Depth,n}$, 'skel')        ▷ Image Processing Toolbox in MATLAB
            return $I_{skel,n}$
        end function
        function BWMORPH($I_{skel,n}$, 'endpoints')
            return $I_{shape,n}$
        end function
    end for
end for
```

3.1.3 Object tracking with bounding box by using SURF key points

Once the object is detected in a video, the bounding box is then allocated which encloses the detected object. To define the location of the bounding box, first the interest points of the moving object are defined using SURF discussed in Section 2.2. In this thesis work, the SURF key points are computed from the RGB difference image I_{diff} to obtain the object bounding box. It uses the function '*detectSURFFeatures*' in MATLAB described by the Algorithm 2. The detected key points are given as x-axis ($\mathbf{S}_x$) and y-axis ($\mathbf{S}_y$) coordinate vectors. These coordinates are used to define the locations of the bounding box by (3.1).

$$[x_1,x_2,y_1,y_2] = [\min(\mathbf{S}_x),\max(\mathbf{S}_x),\min(\mathbf{S}_y),\max(\mathbf{S}_y)] \tag{3.1}$$

The bounding box location 'x_1' and 'x_2' indicates the minimal and maximal point in the horizontal axis respectively while 'y_1' and 'y_2' indicates the minimal and maximal point in the vertical axis. One of the parameters used in '*detectSURFFeatures*' function is '*MetricThreshold*'. This parameter indicated as $SURF_{thres}$ is modified in this method to vary the number of key points. When the threshold is high less key points are detected and when the threshold is less more key points are detected, which can affect the bounding box size.

Algorithm 2 Object bounding box from SURF key points

```
for every RGB video with difference image $I_{diff}$ do
    for n = 1 to $len$, where '$len$' - number of frames in a video do
        function detectSURFFeatures($I_{diff,n}$, 'MetricThreshold')
                                ▷ Computer Vision System Toolbox in MATLAB
            return $[\mathbf{S}_x,\mathbf{S}_y]_n$
        end function
        Find $[x_1,x_2,y_1,y_2]_n$ by (3.1)
    end for
end for
```

3.1.4 Append zeros outside bounding box to form the region of interest of size $(a \times a)$

The region of interest is a small segment of the entire image frame. This segmenting step focuses on finding the foreground object. Consider the tracked objects with bounding box locations $[x_1,x_2,y_1,y_2]$. The 'width' $(x_2 - x_1 + 1)$ and 'height' $(y_2 - y_1 + 1)$ of the box are used to compute the aspect ratio of the object. The length l defined as $l = \max(width, height)$, varies for every frame as the tracked object's aspect ratio changes with motion. Next the tracked object with BB (bounding box) size $(width \times height)$ is defined as ROI with area $(l \times l)$. When defining ROI to maintain the aspect ratio of the object, zero pixel values are appended in the remaining area outside BB. An illustrative example of this method where background pixel values are appended

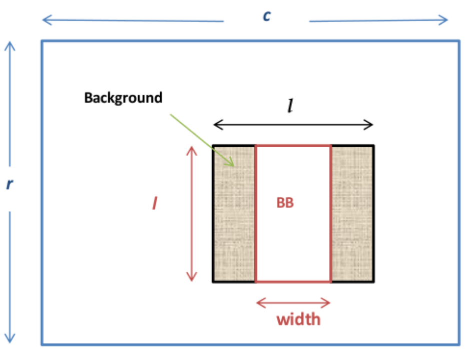


Figure 3.2: Illustration of changing the bounding box to the region of interest as defined in Section 3.1.4. Append intensity values outside BB to form ROI of area $(l \times l)$ which is later resized to size $(a \times a)$, in the illustration the appended pixels are background pixel values indicated by '*texture*' region.

is shown in the Figure 3.2. The method is adopted from [25] and uses either RGB (I_{RGB}) or Depth (I_{shape}) based tracked objects. The overall image is segmented to ROI frame I_{ROI} where only the object details are present that is useful in feature extraction. The region of interest (I_{ROI}) is then normalized to size $(a \times a)$.

3.2 Extract features from the region of interest

The features are extracted using three types of descriptors namely HOG, Optical flow and Shape. These feature methods use the region of interest frames $I_{ROI,n}$ to obtain the frame-based vectors '$\mathbf{h}_n$'. The frame-based vector contains 'L_f' number of feature components for an ROI frame. The value 'n' ranges between $[1, len]$ where 'len' denotes the number of frames in an activity video. The total number of frames for each activity video is not the same, due to differences in motion speed. But, the input of classifier requires features on fixed interval (equal feature length) for all activity.

3.2.1 Window averaging to obtain time dependent features

Consider the frame-based vector '$\mathbf{h}_n$' of a video activity, where $n = 1,...,len$. The length of a video activity 'len' is different for each video. Window averaging is defined to obtain time dependent feature vectors. A fixed number of feature vectors are obtained on a fixed interval $k = 1,...,M$. Each time dependent feature vector '$\hat{\mathbf{f}}_k$' is an average from a segment of frame-based vectors with window size $W = \text{round}\left(\frac{len}{M}\right)$. The varying averaging window W is used to

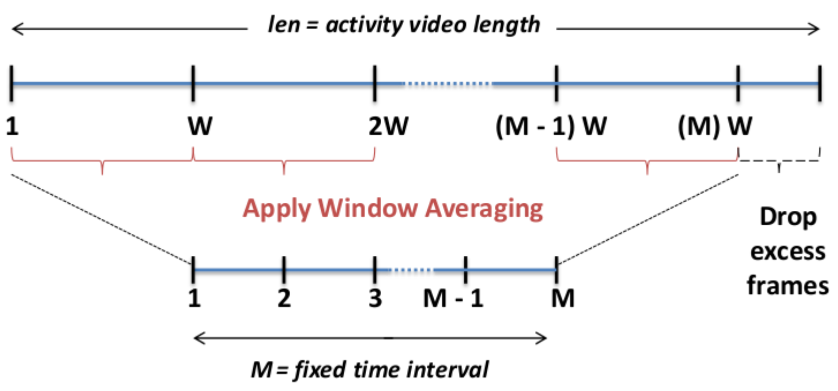


Figure 3.3: Window Averaging method on a fixed interval M. The graphical figure shows a line diagram that indicates frame-based vector. The video length 'len' is divided into windows of size 'W'. The grouped windows shown in 'red' brackets undergo window averaging to give time dependent feature vectors.

calculate the time dependent feature vector as given in (3.2). The feature vector '$\hat{\mathbf{f}}_k$' has a fixed interval 'k' ranging between $[1,M]$.

$$\hat{\mathbf{f}}_k = \frac{1}{W}\sum_{j=1}^{W}\mathbf{h}_{(k-1)W+j} \quad ; \quad k = 1,...,M \tag{3.2}$$

The excess frames outside the 'M' fixed interval $[(len - M \times W) < W]$ is dropped. Additional use of this method is to reduce the feature vector dimensionality. A pictorial representation of the window averaging method is shown in the Figure 3.3. The frame-based vectors from a video of frame length 'len' are grouped into windows of size 'W'. The time dependent feature vectors in interval M after window averaging is indicated in the figure.

For each selected feature (HOG, Optical flow and Shape), the time dependent feature vectors '$\hat{\mathbf{f}}_k$' are obtained from window averaging in a fixed interval M. The transpose of time dependent feature vectors are computed and are serially combined one after the other to give the final feature vector (row vector) denoted by '$\hat{\mathbf{f}}^{\,j}$' given in (3.3) for each j^{th} video. The length of feature vector '$\hat{\mathbf{f}}^{\,j}$' is $M \times L_f$ and it is based on fixed interval 'M' and number of feature components 'L_f', which are fixed to obtain an equal feature length for all activities. The value of 'j' ranges between $[1,N]$ where 'N' is the number of videos.

$$\hat{\mathbf{f}}^{\,j} = [\hat{\mathbf{f}}_1{}^{j} \;\; \hat{\mathbf{f}}_2{}^{j} \;\; ... \;\; \hat{\mathbf{f}}_M{}^{j}] \quad ; \quad j = 1,...,N \tag{3.3}$$

3.2.2 HOG feature vector

The HOG feature is extracted from the region of interest I_{ROI}. The definition of HOG is detailed in Section 2.3.1. In this method the MATLAB function '*extractHOGFeatures*' is used to obtain the feature vector as shown in the Algorithm 3. The HOG frame-based vector is given as '$\mathbf{h}_{HOG,n}$', with L_{HOG} as the number of HOG feature components.

Window averaging defined in (3.2) is used to obtain HOG time dependent vectors '$\hat{\mathbf{f}}_k^{\,j,HOG}$' in fixed interval M. The final HOG feature vector '$\hat{\mathbf{f}}^{\,j,HOG}$' for each j^{th} video with feature length $M \times L_{HOG}$ is obtained according to (3.3).

$$\hat{\mathbf{f}}^{\,j,HOG} = [\hat{\mathbf{f}}_1^{\,j,HOG} \ \hat{\mathbf{f}}_2^{\,j,HOG} \ \dots \ \hat{\mathbf{f}}_M^{\,j,HOG}] \tag{3.4}$$

Algorithm 3 Obtaining HOG features

```
for every RGB video with $I_{ROI}$ do
    for n = 1 to $len$, where '$len$' - number of frames in a video do
        function extractHOGFeatures($I_{ROI,n}$)
                                        ▷ Computer Vision System Toolbox in MATLAB
            return $\mathbf{h}_{HOG,n}$
        end function
    end for
end for
```

3.2.3 Optical flow feature vector

The optical flow vector between two adjacent frames is computed. The input used in optical flow is the region of interest of an image frame I_{ROI} from the RGB video. The optical flow vector $[\mathbf{v}_x, \mathbf{v}_y]$ in the '$x-axis$' and '$y-axis$' directions are obtained from adjacent ROI frames along with parameter settings in [16]. The MATLAB function '*Coarse2FineFrames*' from [16] is used to find the flow vectors. The optical flow vectors cannot be directly used as feature descriptor because of its large feature length. The flow vectors are used to define optical flow feature as in Section 2.3.2 by either the HOOF or HOG of optical flow (HOGOF) method.

Histogram of Oriented Optical flow: In HOOF feature, the optical flow magnitude and direction are used in a distribution histogram. The flow vectors $[\mathbf{v}_x, \mathbf{v}_y]$ for every frame in a video is given as input along with the number of bin specification (*binSize*). The constant bin number is used to specify the orientation range for the histogram. The output of this function is HOOF frame-based vector '$\mathbf{h}_{OF,n}$' with 'L_{HOOF}' number of HOOF feature components for n^{th} frame. This method is available as a MATLAB function '*gradientHistogram*' from [19].

Algorithm 4 Obtaining Optical flow features

```
for every RGB video with $I_{ROI}$ do
    for n = 1 to len, where 'len' - number of frames in a video do
        function Coarse2FineTwoFrames($I_{ROI,n}$, $I_{ROI,n+1}$, 'para')
                                                        ▷ Optical flow code from [16]
            return $[\mathbf{v}_x, \mathbf{v}_y]_n$
        end function
        HOOF Feature
        function gradientHistogram($[\mathbf{v}_x, \mathbf{v}_y]_n$, 'binSize')
                                                        ▷ HOOF code from [19]
            return $\mathbf{h}_{OF,n}$
        end function
        HOGOF Feature
        function flowToColor($[\mathbf{v}_x, \mathbf{v}_y]_n$)
                                                        ▷ Flow color coding from [18]
            return $I_{HSV,n}$
        end function
        function extractHOGFeatures($I_{HSV,n}$)
                                        ▷ Computer Vision System Toolbox in MATLAB
            return $\mathbf{h}_{OF,n}$
        end function
    end for
end for
```

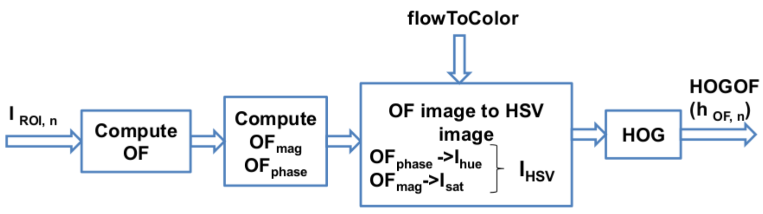


Figure 3.4: Flow diagram of HOGOF feature extraction module

HOG of Optical flow: The HOG of Optical flow feature (HOGOF) uses optical flow vectors $[\mathbf{v}_x, \mathbf{v}_y]$ from the region of interest as input. The optical flow magnitude (OF_{mag}) and direction (OF_{phase}) are computed from (2.13) and (2.12) respectively. Based on HSV model, for each frame the optical flow magnitude is coded as saturation image 'I_{sat}' and direction is coded as hue image 'I_{hue}'. The combined HSV image I_{HSV} is in RGB colour space. The magnitude and direction values of flow vectors are colour-coded using existing MATLAB function '*flowToColor*' from [18].

The output $I_{HSV,n}$ frames are used as input to HOG to yield HOG of Optical flow feature (HOGOF). This is achieved by using '*extractHOGFeatures*' function from MATLAB. The output from this function is the HOGOF frame-based vector $\mathbf{h}_{OF,n}$ for each n^{th} frame, with L_{HOGOF} number of HOGOF feature components. The process flow is as shown in the Figure 3.4.

The steps involved to find optical flow features are given in Algorithm 4. The optical flow frame-based vector '$\mathbf{h}_{OF,n}$' uses window averaging in (3.2) to obtain time dependent feature vector '$\hat{\mathbf{f}}_k^{\,j,OF}$'. The final optical flow feature vector '$\hat{\mathbf{f}}^{\,j,OF}$' for each j^{th} video with feature length $M \times L_{OF}$ is computed following (3.3).

$$\hat{\mathbf{f}}^{\,j,OF} = [\hat{\mathbf{f}}_1^{\,j,OF} \;\; \hat{\mathbf{f}}_2^{\,j,OF} \;\; \ldots \;\; \hat{\mathbf{f}}_M^{\,j,OF}] \tag{3.5}$$

3.2.4 Shape feature vector

Depth videos are used to extract shape features. Shape feature analysis is done on depth video I_{ROI} frames from object detection to give the feature descriptors in two parts. The two parts contain the form and motion features as given in (3.6) and the process flow is shown in the Figure 3.5.

$$\mathbf{h}_{SHAPE} = [\mathbf{h}_{contour} \;\; \mathbf{h}_{motion}] \tag{3.6}$$

Form features of shape descriptor: The shape feature with form components is defined in (3.7). The form features follows shape parameters in Section 2.3.3 and are defined below.

$$\mathbf{h}_{contour,n} = [\mathbf{p}_g, \mathbf{p}_{Extrema,1}, \ldots, \mathbf{p}_{Extrema,8}, \theta, AR, V, d_1, \ldots, d_8]_n \tag{3.7}$$

The proposed descriptor at every n^{th} frame of the video has the centre of gravity of ROI ($\mathbf{p}_g$) given by (2.14). The i-Extrema points ($\mathbf{p}_{Extrema,i}$) where $i = 1, \ldots, 8$, are obtained from the outer boundary of the region of interest I_{ROI}. The coordinate vector of a pixel point in 2D space is given as $\mathbf{p} = (p_x, p_y)$, so each spatial point has two feature values. The spatial points contribute 18 shape feature components.

Orientation (θ) is the angle between the x-axis and the major axis of the bounding box ellipse given in (2.17). Aspect ratio (AR) is the ratio of width to height obtained from BB locations $[x_1, x_2, y_1, y_2]$ given by (3.8).

$$AR = \frac{x_2 - x_1 + 1}{y_2 - y_1 + 1} \tag{3.8}$$

I ROI, n with depth image
Shape feature extraction
h SHP, n

Figure 3.5: Flow diagram of shape feature extraction module

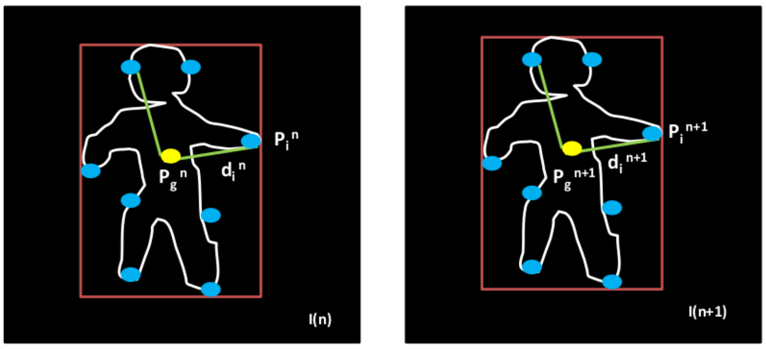


Figure 3.6: Form feature components of shape descriptor. An illustration of form features between adjacent frames n and $n+1$ is shown. The spatial points used to determine pixel velocity are indicated by colour '*blue*' (Extrema) and '*yellow*' (centroid). The Euclidean distances d_i are indicated by '*green*' solid lines.

Eccentricity (V) is the ratio of lengths between the major and minor axis of the object in an elliptical bounding box given by (2.15).

The Euclidean distance (d_i) between each extrema coordinate ($\mathbf{p}_{Extrema,i}$) and centroid ($\mathbf{p}_g$) is defined by (3.9). There are a total of 29 form feature components defined and some of them are illustrated in the Figure 3.6 between adjacent frames. The shape feature components are obtained using the MATLAB function '*regionprops*' as shown in Algorithm 5.

$$d_i = \|\mathbf{p}_{Extrema,i} - \mathbf{p}_g\|_2 \quad ; \quad i = 1,...,8 \tag{3.9}$$

Motion features of shape descriptor: The motion features are grouped as shape descriptor which is mathematically defined below:

$$\mathbf{h}_{motion,n} = [\nabla d_1,...,\nabla d_8, k_{\mathbf{p}_{Extrema,1}},..,k_{\mathbf{p}_{Extrema,8}}, k_{\mathbf{p}_g}]_n^{n+1} \tag{3.10}$$

The final part of shape descriptor consists of 17 motion features which contain the distance gradients and pixel velocity due to object movement between adjacent frames 'n' and '$n+1$'.

The first 8 motion features indicates the distance gradients ∇d_i between two adjacent frames. The gradients find the difference in Euclidean distances from 8 extrema points. The ∇d_i^n for n^{th} frame is defined by (3.11) taking distance values between frames 'n' and '$n+1$', where d_i value is defined by (3.9).

$$\nabla d_i^n = d_i^{n+1} - d_i^n \quad ; \quad i = 1,...,8 \tag{3.11}$$

The next 9 motion features indicate pixel velocity of 9 spatial points (8-extrema and centroid). The pixel velocity is defined as the Euclidean distance between respective spatial point locations in adjacent frames 'n' and '$n+1$'. Let's consider $k_{\mathbf{p}}^n$ as the pixel velocity of spatial point '$\mathbf{p}$' at n^{th} frame,

$$k_{\mathbf{p}}^n = \|\mathbf{p}^{n+1} - \mathbf{p}^n\|_2 \tag{3.12}$$

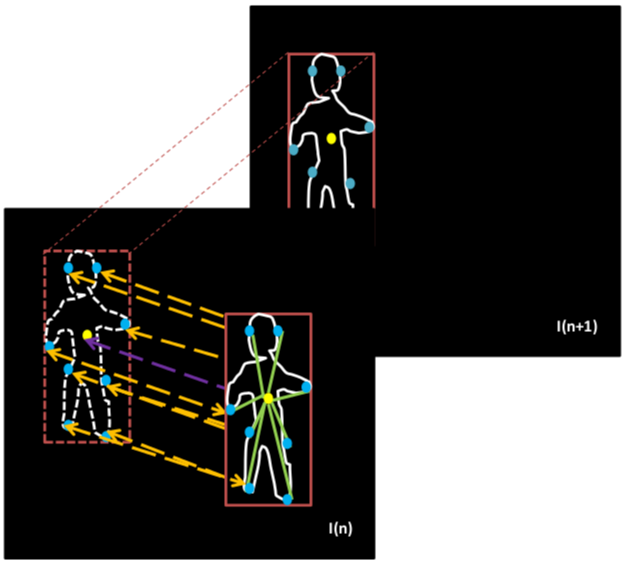


Figure 3.7: An illustration of Shape descriptors with spatio-temporal shape features. The spatial points are indicated by colour '*blue*' (Extrema) and '*yellow*' (centroid). The Euclidean distances d_i are indicated by '*green*' solid lines. The pixel velocity indicated by dashed lines with arrows. The colour '*orange*' for the extrema points and '*violet*' for the centroid.

A visual illustration of the defined shape descriptors with 46 features are partially shown in the Figure 3.7. The spatio-temporal features are indicated as markers and lines of different colours to identify the shape descriptors.

The frame-based vector '$\mathbf{h}_{SHAPE,n}$' for shape feature is computed from (3.6), with $L_{SHAPE} = 46$ number of shape feature components. To find time dependent feature vector '$\hat{\mathbf{f}}_k^{\,j,SHAPE}$', window averaging is done on a fixed interval M by (3.2). The final shape feature vector '$\hat{\mathbf{f}}^{\,j,SHAPE}$' for each j^{th} video with feature length $M \times L_{SHAPE}$ is followed from (3.3).

$$\hat{\mathbf{f}}^{\,j,SHAPE} = [\hat{\mathbf{f}}_1^{\,j,SHAPE} \quad \hat{\mathbf{f}}_2^{\,j,SHAPE} \quad \dots \quad \hat{\mathbf{f}}_M^{\,j,SHAPE}] \tag{3.13}$$

3.2.5 Normalize feature values

The normalization of each given feature (HOG, Optical flow and Shape) is defined after window averaging on time dependent feature vectors '$\hat{\mathbf{f}}_k^{\,j}$' given in (3.3). Let $\boldsymbol{\mu}_k$ denote the mean feature vector and $\boldsymbol{\sigma}_k$ denote the variance feature vector computed over $k = 1,...,M$ fixed interval and $j = 1,...,N$ number of training videos. The definition of $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$ are given in (3.15) and (3.16) respectively. Each vector '$\hat{\mathbf{f}}_k^{\,j}$', $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$ has L_f number of feature components. The feature

Algorithm 5 Shape descriptors defined by form features

```
for every Depth video with $I_{ROI}$ do
    for n = 1 to $len$, where '$len$' - number of frames in a video do
        function REGIONPROPS($I_{ROI,n}$, 'Centroid', 'Extrema', 'Orientation')
                                                  ▷ Image Processing Toolbox in MATLAB
            return $[\mathbf{p}_g, \mathbf{p}_{Extrema,1}, ..., \mathbf{p}_{Extrema,8}, \theta]_n$
        end function
        function REGIONPROPS($I_{ROI,n}$, 'MajorAxisLength', 'MinorAxisLength')
                                                  ▷ Image Processing Toolbox in MATLAB
            $V$ = MajorAxisLength / MinorAxisLength
            return $[V]_n$
        end function
        function ASPECTRATIO($[x_1, x_2, y_1, y_2]_n$)
            $AR = (x_2 - x_1 + 1)/(y_2 - y_1 + 1)$
            return $[AR]_n$
        end function
        for i = 1 to 8 do
            function EUCLIDIST($\mathbf{p}_{Extrema,i}, \mathbf{p}_g$)
                return $d_{i,n}$
            end function
        end for
        $\mathbf{h}_{contour,n} = [\mathbf{p}_g, \mathbf{p}_{Extrema,1}, ..., \mathbf{p}_{Extrema,8}, \theta, AR, V, d_1, ..., d_8]_n$
    end for
end for
```

normalization '$\mathbf{f}_k{}^j$' is defined on time dependent feature vectors as given in (3.14).

$$\mathbf{f}_k{}^j = \frac{\hat{\mathbf{f}}_k{}^j - \mu_k}{\sigma_k} \quad ; \; j = 1,..,N \quad ; \; k = 1,...,M \tag{3.14}$$

$$\mu_k = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{M}\sum_{k=1}^{M}\hat{\mathbf{f}}_k{}^j \tag{3.15}$$

$$\sigma_k = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\hat{\mathbf{f}}_k{}^j - \mu_k)^2} \tag{3.16}$$

3.2.6 Combine all features in series

To obtain most of the spatio-temporal details in a single feature vector, the serial combination of feature vectors is done. In this thesis, the row vector of one feature is appended along with the previous feature's row vector. The feature vector $\mathbf{f}^{\,j,COMBINE}$ defined in (3.17) is a serially combined vector (row vector), which has feature length $M \times (L_{HOG} + L_{OF} + L_{SHP})$ for each j^{th} video. The HOG feature $\mathbf{f}^{\,j,HOG}$ provides spatial details, the optical flow feature $\mathbf{f}^{\,j,OF}$ gives the motion details and the shape feature $\mathbf{f}^{\,j,SHAPE}$ has both form and motion features. A feature vector with three type of features is useful for classifying an activity.

$$\mathbf{f}^{\,j,COMBINE} = [\mathbf{f}^{\,j,HOG} \quad \mathbf{f}^{\,j,OF} \quad \mathbf{f}^{\,j,SHAPE}] \tag{3.17}$$

3.3 Design of a classifier

The final part is to design a classifier that separates the extracted features to its related classes. The classifier used in this thesis is C-SVM for classification as discussed in section 2.4.

Activity Classification for Binary Class: Binary classification means two-class classification, where the C-SVM classifier separates the final feature vector by applying a soft margin. The RBF kernel function is used in C-SVM to map the features into a higher dimensional space. The RBF function with kernel parameter γ is given as,

$$K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0. \tag{3.18}$$

Cross-validation and Grid search: The cross-validation followed in this design is n-fold cross-validation, where the dataset is randomly divided into n equal subsets. Out of the n subsets, one subset is tested each time against rest of $n-1$ training data. This is repeated 'n' times till each subset is tested only once. The mean value from the n-fold results gives a single estimation. Grid search identifies the parameter with best estimates by using an extensive search. The search is made from coarse to the fine range of values. Grid search and cross validation are used together to determine the (C, γ) values for C-SVM. The C-SVM classifier from LIBSVM package [24] is used for experiments.

The Algorithm 6 details the steps involved in using the designed C-SVM classifier. First the input feature vectors ($\mathbf{f}^{\,j,HOG}$,$\mathbf{f}^{\,j,OF}$,$\mathbf{f}^{\,j,SHAPE}$,$\mathbf{f}^{\,j,COMBINE}$) for all 'j' videos are labelled based on their classes. The entire set of feature vectors is divided into training and testing data matrix to be used in the classifier. The data is converted to the sparse matrix form to be used in LIBSVM. The function used for classification is '*easy.py*' from LIBSVM package which takes training and testing data as inputs. The features are normalized for training and testing set respectively by an in-built function. The parameters 'C' and 'γ' are determined by grid-search and 10-fold cross validation. The classifier is trained with the input to produce support vectors. The support vectors are later used in testing data to find decision values used for classifying the features. Later, the testing data with classified labels are used to find the performance measures including classification rate.

Algorithm 6 C-SVM classification to find the class labels for test features

```
for each feature do
    Label the features by class
    Separate and group them into training and testing data
    Convert the grouped data in sparse matrices
    function easy.py(training matrix, testing matrix)
                                        ▷ Function from LIBSVM package [24]
        Normalize the matrix values
        Find C and $\gamma$ by grid search and 10-fold cross validation
        Training the classifier with RBF kernel
        Use C and $\gamma$ to find decision values for testing matrix
        return Classify the labels using decision function D(x)
    end function
end for
```

3.3.1 Performance measures

The definition of binary classification performance measures [26] are based on following four parameters TP - True Positive, FP - False Positive, TN - True Negative and FN - False Negative given by confusion matrix in Table 3.1. The classified labels from C-SVM classification are compared with the original class labels of testing data. The classification parameter for each testing sample is assigned based on the confusion matrix.

Class	Classified as (+1)	Classified as (-1)
Label (+1)	TP	FN
Label (-1)	FP	TN

Table 3.1: Confusion matrix to determine parameters used in binary classification performance measures

After assigning each testing data with classification parameters, the overall performance measures are computed for the total testing dataset. The six basic measures of classification analysis for C-SVM are TPR - True Positive Rate, TNR - True Negative Rate, FPR - False Positive Rate, FNR - False Negative Rate, Classification rate also referred to as Accuracy (ACC) and Miss rate. All the measures are in percentage (%) and are computed as follows:

$$\textit{True Positive Rate} \ \ (TPR\%) = \frac{TP}{TP+FN} \tag{3.19}$$

$$\textit{True Negative Rate} \ \ (TNR\%) = \frac{TN}{TN+FP} \tag{3.20}$$

$$\textit{False Positive Rate} \ \ (FPR\%) = \frac{FP}{TN+FP} \tag{3.21}$$

$$\textit{False Negative Rate}\ \ (FNR\%) = \frac{FN}{TP + FN} \tag{3.22}$$

$$\textit{Classification rate}\ \ (\%) = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.23}$$

$$\textit{Miss rate}\ \ (\%) = \frac{FP + FN}{TP + FN + TN + FP} \tag{3.24}$$

4

Experimental results

The videos used in the experiments are provided by Signal Processing research group from Chalmers University of Technology. The actions are captured as RGB-D videos using Kinect. The Kinect is an electronic device with infrared (IR) motion sensing technology that provides depth data indicating object's relative location with respect to the camera in addition to RGB visual data [27]. Kinect videos have a default 24-bit RGB video stream with resolution 640 $\times$ 480 pixels. The greyscale depth video is also of same (640 $\times$ 480) resolution with 11-bit depth data. The IR sensor has a location ranging limit of 1 to 6 m approximately and the camera pivot can be either tilted or rotated. The camera is fixed during the experiment to obtain still background frames. RGB videos are made up of 3 channels (Red, Blue and Green) each with 8-bit data and an intensity range [0, 255]. The combined 24-bit value gives pixel intensity used in color computation analysis. The RGB-D videos are preferred as they contain depth video with greyscale frames which are useful in shape feature extraction.

4.1 RGB-D video datasets

The two main activities fall and lying down are considered for the experiments. The activities are performed by 19 different persons and are recorded as a single long video that is segmented manually using video editing software. The final activity videos are made up of 30-60 frames with a runtime of 1s-3s, making them independent of long lie after the activity. The dataset is comprised of 400 falls and 400 lying down videos. Each fall video consists of one person walking in front of the camera and suddenly falling down arbitrarily. Each lying down video captures the person as he or she sits down to lay flat on the floor.

For classification, the fall video features are labelled as (+1) and lying down video features are labelled as (-1). The sequence of frames involved in each activity is shown in the Figures 4.1 and 4.2 for RGB and depth videos respectively. The rows (1-3) shows fall frames, rows (4-6) shows lying down frames and rows (7-9) shows the intermediate position of sitting down.

To use in experimental analysis the video datasets for both fall and lying down is divided into two study cases:

- **Case study-1** uses 50% of total videos as training data and the rest 50% as testing data.

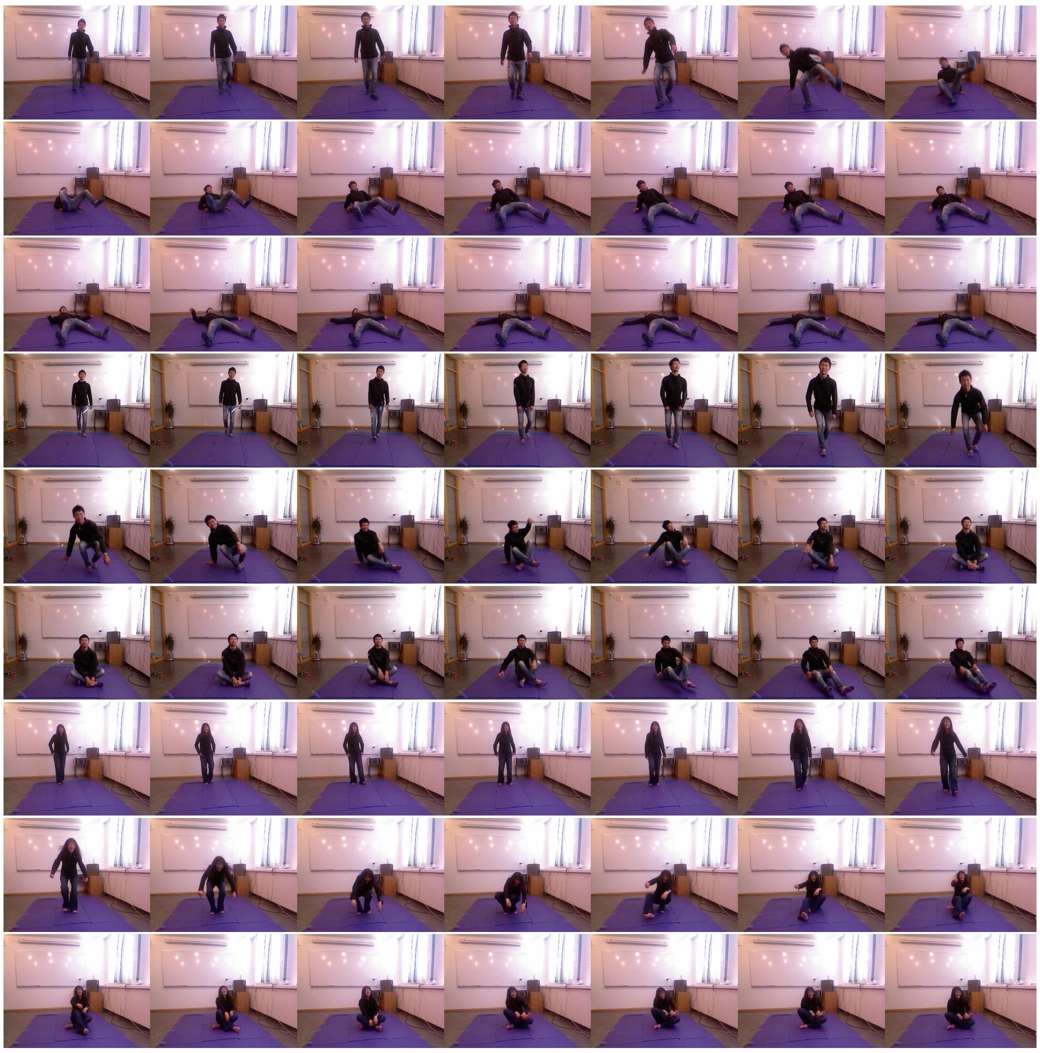


Figure 4.1: Frame sequence from the RGB video dataset. Row 1-3: Frame sequence for falling down activity in RGB video. Row 4-6: Frame sequence for lying down activity in RGB video. Row 7-9: Frame sequence for sitting down activity in RGB video.

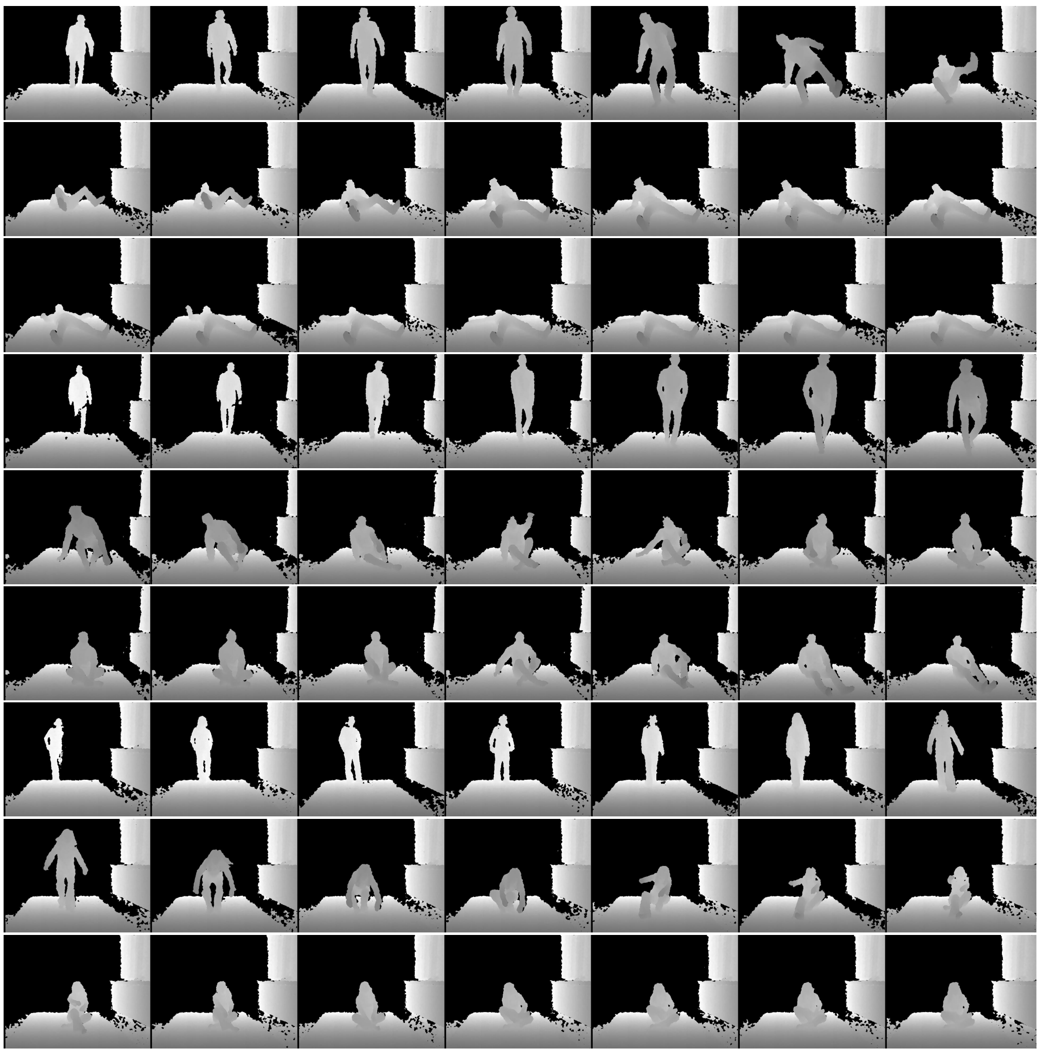


Figure 4.2: Frame sequence from the Depth video dataset. Row 1-3: Frame sequence for falling down activity in depth video. Row 4-6: Frame sequence for lying down activity in depth video. Row 7-9: Frame sequence for sitting down activity in depth video.

The first 200 videos are used for training while the rest 200 videos are used for testing in both classes of activity.

- **Case study-2** uses 80% of total videos as training data and the rest 20% as testing data. The first 320 videos are used for training and the remaining 80 videos are used for testing in fall and lying down activity.

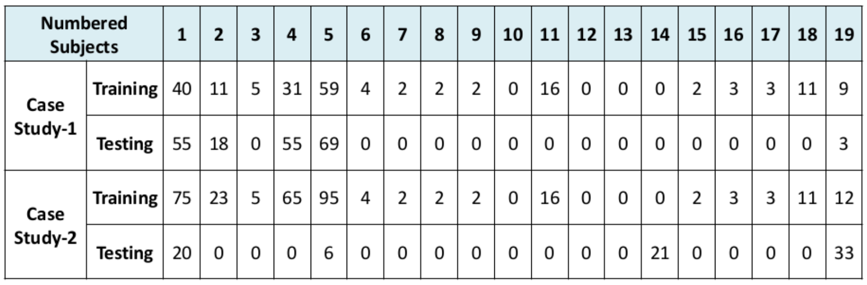
Classifier Input	No. of Fall Videos	No. of Lying Down Videos
Training	200 (50%)	200 (50%)
Testing	200 (50%)	200 (50%)
Total Number of Videos	400	400

Table 4.1: Case study-1 with first 50% of total videos used for training data and the rest 50% used for testing data

Classifier Input	No. of Fall Videos	No. of Lying Down Videos
Training	320 (80%)	320 (80%)
Testing	80 (20%)	80 (20%)
Total Number of Videos	400	400

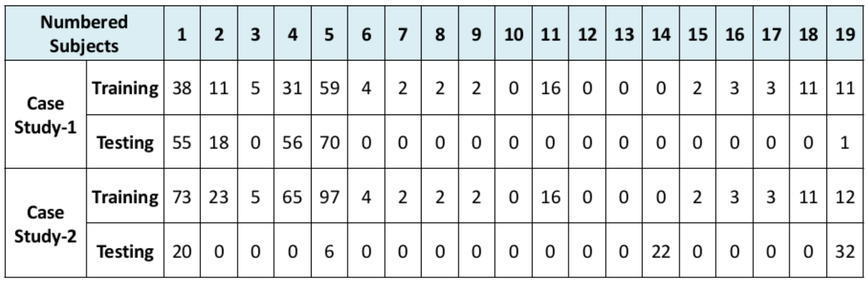
Table 4.2: Case study-2 with first 80% of videos used for training data and the rest 20% used for testing data

The experiments are performed on a randomized dataset. This is shown in a distribution table where the contribution of each member is indicated for fall and lying down activity. The data also shows the contribution of each member for training and testing data in both case studies. Among the 19 member group each subject appearing on the video is visually recognized and identified with a number. The tables shown in the Figures 4.3 and 4.4 is included respectively for both fall and lying down activities.



Numbered Subjects		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Case Study-1	Training	40	11	5	31	59	4	2	2	2	0	16	0	0	0	2	3	3	11	9
	Testing	55	18	0	55	69	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Case Study-2	Training	75	23	5	65	95	4	2	2	2	0	16	0	0	0	2	3	3	11	12
	Testing	20	0	0	0	6	0	0	0	0	0	0	0	0	21	0	0	0	0	33

Figure 4.3: Distribution of number of fall activities performed by each subject in Case study-1 and Case study-2. The data is further divided into training and testing data for both case studies.



Numbered Subjects		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Case Study-1	Training	38	11	5	31	59	4	2	2	2	0	16	0	0	0	2	3	3	11	11
	Testing	55	18	0	56	70	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Case Study-2	Training	73	23	5	65	97	4	2	2	2	0	16	0	0	0	2	3	3	11	12
	Testing	20	0	0	0	6	0	0	0	0	0	0	0	0	22	0	0	0	0	32

Figure 4.4: Distribution of number of lying down activities performed by each subject in Case study-1 and Case study-2. The data is further divided into training and testing data for both case studies.

Tests on fall classification are conducted by using the three normalized feature vectors HOG $\mathbf{f}^{\,j,HOG}$ given by (3.4), HOG of optical flow (HOGOF) $\mathbf{f}^{\,j,OF}$ given by (3.5) and shape feature $\mathbf{f}^{\,j,SHAPE}$ given by (3.13). They are tested separately and in combined form $\mathbf{f}^{\,j,COMBINE}$ given by (3.17), where the 'j' value ranges between $[1,N]$, with N number of videos. The tests are conducted to study the contribution of each feature for activity classification based on classification rate. The experiment and evaluations are carried out in MATLAB (R2014b) student software with Image Processing and Computer Vision System Toolboxes.

4.2 Setup

4.2.1 HOG features

The HOG feature used in the experiments are obtained from RGB video inputs. This section mentions the parameter settings used to extract the HOG feature vector $\mathbf{f}^{\,j,HOG}$. The RGB video frames are given as input and for every successive frame difference image is determined by background subtraction. The object detected by difference frames are given as input to SURF detector (Algorithm 2). The metric threshold is set to $SURF_{thres}$ = 50 and the detected SURF key points are used to define bounding box locations $[x_1,x_2,y_1,y_2]$ for object tracking. The bounding box locations and RGB frames are taken as inputs to find the ROI. The ROI is formed by appending zero values outside BB as shown in the Figure 3.2. For the last step of foreground human detection, the ROI frames I_{ROI} are normalized to size 32×32.

The normalized ROI frames I_{ROI} is given as input to HOG feature extraction in Algorithm 3. The default parameters of HOG given in Section 2.3.1 are set as function inputs. The function gives HOG frame-based vector $\mathbf{h}_{HOG,n}$ for each frame with number of HOG feature components L_{HOG} = 324. The number of frames in a video for each input video is different at this point, so a fixed interval for time dependent feature vector is set to M = 10. The window averaging is

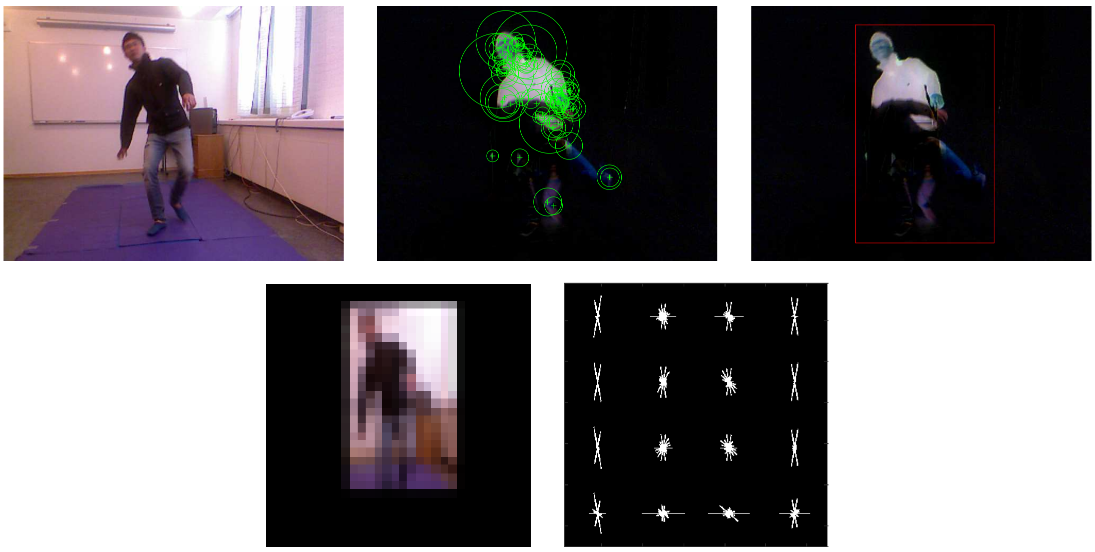


Figure 4.5: Output frames at each step of the HOG feature setup. Top Frame 1: Fall RGB frame used as input. Top Frame 2: SURF key points in '*green*' marked on RGB difference frame. Top Frame 3: The '*red*' bounding box marked in difference frame based on SURF key points. Bottom Frame 1: The ROI after appending zeros outside BB and resized to area 32×32. Bottom Frame 2: HOG feature visualization for ROI input, in which each white plot shows the distribution of gradient orientations within the HOG cell.

obtained by following (3.2). The HOG feature vector $\hat{\mathbf{f}}^{j,HOG}$ given by (3.4) is given as output with feature length 3240 (i.e. 324×10) for each 'j' videos. The feature values are normalized by using (3.14) on time dependent feature vectors.

The Figure 4.5 shows the output frames at each step of HOG feature extraction. The figures in the top row are results of object detection and tracking. In Row 1: Frame 1 indicates the RGB input for fall activity, Frame 2 indicates the SURF key points on RGB difference frame and Frame 3 shows the bounding box marked from SURF point locations. The figures in the bottom row are from feature extraction. In Row 2: Frame 1 shows the resized ROI used as HOG input and Frame 2 indicates the visualization of HOG feature.

4.2.2 Optical flow features

The methods and parameter settings for optical flow features HOOF and HOGOF are given as follows. The RGB videos are taken as input, where the object is detected by background subtraction from difference frames. This is followed by the object tracking with a bounding box, where the BB locations are obtained from SURF detector. The SURF takes difference frames as input with metric threshold $SURF_{thres}$ = 50 as mentioned in Algorithm 2. To find the region of interest I_{ROI} the method shown in the Figure 3.2 is used. The object detected RGB frames are appended with background pixel values outside BB, which are later normalized to fixed size 32×32.

Next step is the optical flow feature extraction by either HOOF or HOGOF descriptors. Initially, the optical flow vectors between adjacent frames I_{ROI} are computed using Algorithm 4. The parameter settings for '*Coarse2FineFrames*' are the default values from [16] package. The obtained flow vectors $[\mathbf{v}_x, \mathbf{v}_y]$ are given as inputs to either of the two feature descriptors.

Histogram of Oriented Optical flow: For HOOF feature along with flow vectors the parameter number of bins is set to *binSize* = 32 in the function *gradientHistogram* [19]. This gives output HOOF frame-based vector $\mathbf{h}_{OF,n}$ with number of HOOF feature components L_{HOOF} = 32. The time dependent feature vectors from window averaging using (3.2) has a fixed interval value M = 10. The feature values are normalized by using (3.14) on time dependent feature vectors. The HOOF feature vector $\hat{\mathbf{f}}^{j,OF}$ given by (3.5) is of feature length 320 (i.e. 32×10) for each 'j' videos.

HOG of Optical flow: The HSV images I_{HSV} of resolution 32×32 are obtained by colour-coding optical flow vectors as shown in Algorithm 4. It uses the function *flowToColor* from [18]. The HSV-based RGB frames I_{HSV} are used to extract HOG features by Algorithm 3. The extracted frame-based vector HOG of optical flow (HOGOF) '$\mathbf{h}_{OF,n}$' has number of HOGOF feature components L_{HOGOF} = 324.

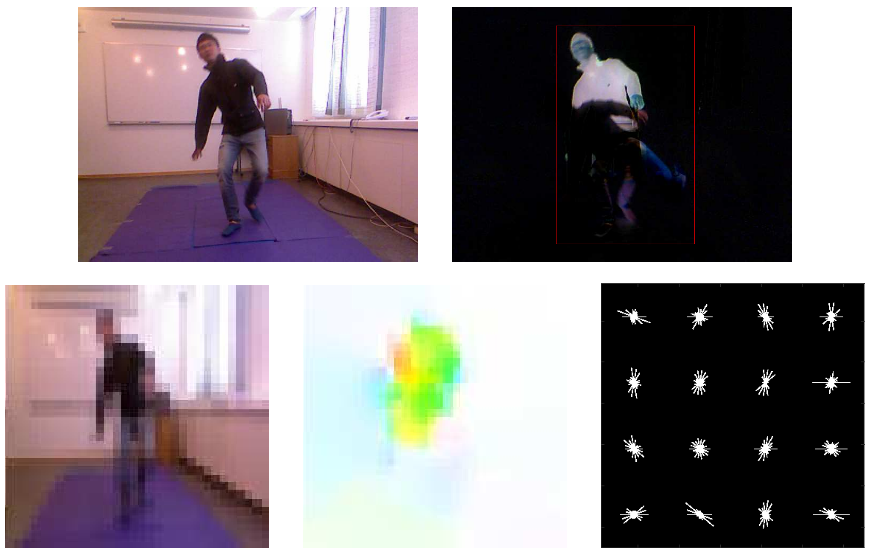


Figure 4.6: Output frames at intermediate steps of Optical flow (HOGOF) setup. Top Frame 1: Input RGB frame for the fall activity. Top Frame 2: The tracked bounding box (*red*) defined using SURF key points over the difference frame. Bottom Frame 1: The region of interest (I_{ROI}) resized to size 32 × 32. Bottom Frame 2: The HSV image frame I_{HSV} after colour-coding by [18]. Bottom Frame 3: HOGOF feature visualization for I_{HSV} input, in which each white plot shows the distribution of gradient orientations within the HOG cell.

The time dependent feature vectors from window averaging using (3.2) has a fixed interval value $M = 10$. The HOGOF feature vector $\hat{\mathbf{f}}^{j,OF}$ given by (3.5) is of feature length 3240 (i.e. 324×10) for each '*j*' videos. The feature values are normalized by using (3.14) on time dependent feature vectors. The final experimental setup with optical flow feature uses HOGOF descriptors.

The Figure 4.6 shows intermediate frame outputs from the optical flow experimental setup. The setup focuses on HOGOF feature, where the top row of figures indicates object detection and tracking. In Row 1: Frame 1 shows the fall activity from RGB video and Frame 2 indicates the SURF key points based BB ('*red*') on RGB difference frame. The figures in the bottom row are part of optical flow feature extraction. In Row 2: Frame 1 shows normalized ROI frame which is used to find the flow vectors, Frame 2 shows HSV image frame I_{HSV} of size 32×32 from colour-coding the flow vector and Frame 3 indicates the visualization of HOGOF feature for I_{HSV} frame.

4.2.3 Shape features

The shape feature extracted from Depth videos uses the setup parameters as given. Initially, object detection by background subtraction is done to result in difference frames of RGB videos. The difference frame is used in object tracking to define bounding box by SURF key points. The settings for the detector are metric threshold $SURF_{thres} = 50$. The tracked bounding box locations $[x_1, x_2, y_1, y_2]$ are used in the ROI definition.

Since for shape feature depth videos are used as input, initially morphological operations are applied on depth videos to get contour image frames I_{shape} by Algorithm 1. This depth contour frame is used along with the BB locations to get the region of interest as shown in the Figure 3.2. The depth details inside the bounding box locations are defined for the region of interest. The region outside of the bounding box is appended with zero values to give I_{ROI} and are normalized to size 32×32.

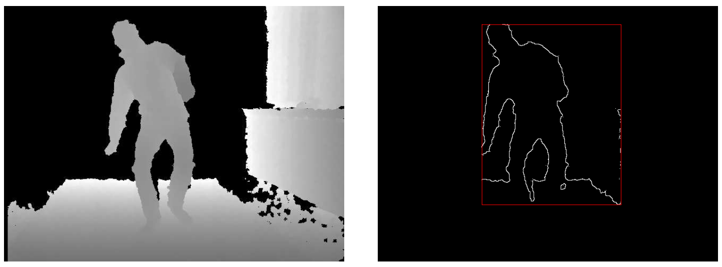


Figure 4.7: The frame outputs at each level of the experimental setup in shape feature extraction. Frame 1: Depth video frame for the fall activity. Frame 2: The object contour frame I_{shape} and the bounding box locations.

The normalized ROI frames are used in shape feature extraction following (3.6). The shape feature extraction is obtained using '*regionprops*' function in MATLAB Image Processing Toolbox. The shape frame-based vector ($\mathbf{h}_{SHAPE,n}$) for every frame contains form and motion descriptors with number of shape feature components $L_{SHAPE} = 46$. To obtain time dependent features window averaging is computed on a fixed interval $M = 10$ using (3.2). The shape feature vector $\hat{\mathbf{f}}^{\,j,SHAPE}$ given by (3.13) is with feature length 460 (i.e. 46×10) for each 'j' videos. The feature values are normalized by using (3.14) on time dependent feature vectors.

In the Figure 4.7 the output frames at intermediate steps of shape feature extraction is shown. The figures from left to right shows object tracking in shape features. Frame 1 shows the fall activity in depth video used as the input and Frame 2 shows the contour frame I_{shape} and bounding box marked in '*red*' used for the region of interest.

4.3 Evaluation

4.3.1 Case study-1

The contribution of each extracted features (HOG, Optical flow and Shape) for activity classification is studied in terms of its performance measures (see Section 3.3.1). For the experimental setup, the individual features HOG, HOGOF and Shape are extracted by following the steps in Sections 4.2.1, 4.2.2 and 4.2.3 respectively. The final feature vectors are indicated by $\mathbf{f}^{\,j,HOG}, \mathbf{f}^{\,j,OF}, \mathbf{f}^{\,j,SHAPE}$ for all input RGB-D videos.

The extracted feature vectors for both activities are labelled into respective classes fall (+1) and lie down (-1). The dataset is now divided into training and testing data to be used in the classifier. For Case study-1, the first 50% of total feature dataset is used as training data by combining both fall and lying down activities. According to table 4.1, the 200 feature vectors from fall and lying down are combined to form the training data with $N = 400$ labelled feature vectors. In the same way, the rest 50% of feature dataset is used as testing data. The remaining 200 feature vectors in fall and lying down are combined to form the testing data with $N = 400$ labelled feature vectors.

To classify the features into respective activity C-SVM classifier from LIBSVM package is used. The processing steps are as mentioned in Algorithm 6. The 'C' and 'γ' parameters determined by grid search varies for each feature HOG, HOGOF and Shape. The testing data is classified into certain labels based on its feature vector, this classified labels is compared with the original labels to determine the performance measures from (3.19) to (3.24). The results of the test are presented in Table 4.3 for the three features based on Case study-1 testing data.

It can be observed that all the three features have classification rate $\geq 93\%$. The HOG and HOGOF features have highest classification rate about 94%. A direct relation between number

Type of Features	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
HOG	93.75	6.25	92.50	95.00	5.00	7.50
HOGOF	94.00	6.00	92.00	96.00	4.00	8.00
Shape	92.75	7.25	94.50	91.00	9.00	5.50

Table 4.3: The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOG, Optical flow (HOGOF) and Shape. The evaluation is done on Case study-1 based on the testing dataset.

of feature components L_f and classification rates can also be seen. The HOG and Optical flow (HOGOF) features have similar classification rate about 94%, with higher number of distinct features components $L_{HOG} = L_{HOGOF} = 324$. The Shape feature has relatively a lesser classification rate at 92.75%, yet it is obtained at low feature cost L_{SHAPE} = 46.

In terms of FPR and FNR all the three features have values between 5% and 10%. An asymmetry is observed where either FNR or FPR is of higher value than the other. The measures true positive rate (TPR), true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR) are inter-related. A very less FPR (about 1%) is good to avoid faulty classification of any activity as fall. But the FNR must also be of lesser value (<5%) to prevent not classifying genuine fall activity as fall.

4.3.2 Case study-2

In this test each extracted features (HOG, Optical flow and Shape) are analyzed individually for fall classification. The test is compared in terms of its performance measures (see Section 3.3.1). For the experimental setup, the individual features HOG '$\mathbf{f}^{\,j,HOG}$', HOGOF '$\mathbf{f}^{\,j,OF}$' and Shape '$\mathbf{f}^{\,j,SHAPE}$' are extracted from input RGB-D videos by following the steps in Sections 4.2.1, 4.2.2 and 4.2.3 respectively.

The feature vectors are labelled into respective classes, based on the type of activity fall (+1) and lie down (-1). The feature dataset is separated into training and testing data according to Case study-2, to be later used in the classifier. In Case study-2, the first 80% of total feature dataset is used as training data from both fall and lying down activities. According to table 4.2, the 320 feature vectors from fall and lying down are combined to form the training data with $N = 640$ labelled feature vectors. The rest 20% of the dataset is used as testing data. The remaining 80 feature vectors in fall and lying down are combined to form the testing data with $N = 160$ labelled feature vectors.

Type of Features	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
HOG	96.88	3.12	96.25	97.50	2.50	3.75
HOGOF	96.88	3.12	95.00	98.75	1.25	5.00
Shape	91.88	8.12	100.00	83.75	0.00	16.25

Table 4.4: The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOG, Optical flow (HOGOF) and Shape. The evaluation is done on Case study-2 based on the testing dataset.

The C-SVM classifier from LIBSVM package follows the steps in Algorithm 6 to classify testing features into respective class labels. The 'C' and 'γ' parameters determined by grid search varies for each test based on the type of features used (HOG, HOGOF and Shape). The testing data with classified labels are compared with original labels to determine the performance measures in (3.19) to (3.23). In Table 4.4 the Case study-2 testing data results are presented for the three features.

From the results, it can be seen that the HOG and HOGOF feature has highest classification rate (96.88%) among the three. When comparing the results between two case studies, it is observed that Case study-2 is better than Case study-1. The classification rate for HOG and HOGOF features are above 95%. This can be reasoned out as in Case study-2 the number of feature details L_f along with the number of training data (80% of total dataset) is relatively high.

The FPR and FNR are less than 5% for the given two features. It can be seen in Case study-2 that the FPR and FNR have relatively reduced from the results in Case study-1. They also have symmetrical values at $< 5\%$ for HOG and HOGOF, which is better than the performance in Case study-1. It can be concluded that fall identification with high classification rate and better FPR and FNR is possible when large detailed feature vectors are combined over a large number of training data.

The shape feature has a classification rate of 91.88% with a very high FNR (16.25%). Shape feature has lesser performance than Case study-1 with 92.75% classification rate. An asymmetry is still observed in shape feature where FNR (16.25%) is very high than FPR (0%). The reason behind lesser performance can be more than the contribution number of feature components. This can be due to the selection of (C,γ) values used in classification. The extensive grid search obtains best (C,γ) for better TPR than the overall classification rate.

Some of the technical problems faced while extracting features from the region of interest can also play a key role in its classification. The contour obtained from morphological operations contains corner points from the background as well, which are not completely removed while appending zero values outside BB. Such faulty contour might result in Extrema point detection outside the human form. In some cases, the Extrema points are plotted on the same location or even left undetected, which is due to partial contour obtained from limb occlusions. Such technical difficulties when corrected while extracting features, might improve the performance of a classifier.

4.4 Results from combined 3 types of feature vectors

The three types of features HOG, Optical flow (HOGOF) and Shape are extracted from RGB-D videos following the experimental setup mentioned in Sections 4.2.1, 4.2.2 and 4.2.3. In this evaluation for optical flow feature, HOG of optical flow (HOGOF) is used. The serially combined feature vectors $\mathbf{f}^{\,j,COMBINE}$ is given by (3.17) for each j videos. This serial combination is carried out for all videos in the dataset for both fall and lying down activities.

This long feature vectors are labelled into respective classes fall (+1) and lie down (-1). The labelled feature vectors are divided into training and testing data according to Case study-1 and Case study-2. Case study-1 has 50% of training and testing data while Case study-2 has 80% training and 20% testing data. The training and testing data follows the steps in Algorithm 6 when used in C-SVM classifier from LIBSVM package. The 'C' and 'γ' parameters determined by grid search is different for each Case Study 1 and 2. The classified labels of testing data are compared with its original labels to compute the performance measures in (3.19) to (3.23).

The test results achieved on testing data of Case study-1 and 2 datasets for serially combined features is shown in Table 4.5. The classification rate is high ($> 95\%$) in both case studies, compared to the individual features contribution as seen between Table 4.3 and 4.4. As expected Case study-2 has higher performance than Case Study-1. The FPR and FNR have also reduced $< 5\%$ and are more similar in values.

It can be observed that a very high classification rate >95% is possible because the serial combination attempts to include most of the spatio-temporal details from each of the studied features

Type of Case Study	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
Case study-1	95.25	4.75	95.50	95.00	5.00	4.50
Case study-2	97.50	2.50	97.50	97.50	2.50	2.50

Table 4.5: The performance measures in % specified by (3.19) to (3.24) for evaluating serially combined feature vector. The evaluation is done on both Case study-1 and 2 based on the testing datasets.

(HOG, optical flow and shape). In both case studies, a reduction in FPR and FNR is also achieved while for the same case study the individual features had very high FNR and FPR $> 5\%$.

It can be concluded that combination of feature vectors with multiple spatio-temporal feature details will improve the FNR, FPR and hence make fall identification better. When feature vector is combined serially verification must be done for the limit of feature length, as long feature vector tends to take more computation time when used in the classifier. In this evaluation the serially combined feature vector is of length 6940 (i.e. $3240+3240+460$). The given feature length becomes a relatively lesser problem when using the LIBSVM tool, as it can handle feature length in the range of thousands for 800 videos.

4.5 Tests to evaluate parameter settings in algorithms

From the results discussed in both Case study-1 and Case study-2 each feature individually and in combined form achieves a classification rate of $\geq 93\%$. This is possible because the initial settings fixed in each method or algorithm is evaluated before using in the final experimental setup. This section details the test cases carried out to find optimal parameters for some of the detection and extraction methods.

4.5.1 Comparison of optical flow features

The experimental setup for optical flow feature uses two type of descriptors: Histogram of Oriented Optical flow (HOOF) and Histogram of Oriented Gradients for Optical flow (HOGOF). To evaluate which of the two optical flow features achieve high classification rate, a comparative study is done. Initially, the HOOF and HOGOF features are obtained by following the experimental settings mentioned in Section 4.2.2. Later the features are evaluated for classification by using C-SVM classifier on Case study-2 testing dataset mentioned in Section 4.3.2. The performance measures for each feature is given in Table 4.6.

Type of Features	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
HOGOF	96.88	3.12	95.00	98.75	1.25	5.00
HOOF	83.75	16.25	78.75	88.75	11.25	21.25

Table 4.6: The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOGOF and HOOF. The evaluation is done on Case study-2 based on the 20% testing data.

From the results, it can be observed that HOGOF feature has higher classification rate than HOOF feature. The FPR and FNR values are $\leq 5\%$ for HOGOF feature, while for HOOF feature the FPR and FNR are $>10\%$. Let's consider the number of feature components in each case: HOOF feature L_{HOOF} = 32 and HOGOF feature L_{HOGOF} = 324. It can be seen that the relation between the number of feature components $L_{HOGOF} > L_{HOOF}$ can be directly related with classification results.

When a given feature has a higher number of descriptors to describe an activity, the better is the feature's classification ability. While both the optical flow features HOOF and HOGOF captures the motion detail of the activity using flow vectors, the way in which each descriptor represents the dense flow vector is different. While HOOF uses a standard orientation histogram, the HOG uses a dense estimate by dividing the ROI size to smaller '*blocks*' and '*cells*' and combining their histogram values. Hence, the HOGOF feature is better than the HOOF feature.

4.5.2 Shape feature components evaluation

The shape feature extraction used in this project includes both form and motion features as given in (3.6). To study the contribution of motion features in shape descriptor a comparative evaluation is done where shape feature vector with only form feature components $L_{SHAPE} = 29$ given in (4.1) is evaluated initially.

$$\mathbf{h}_{SHAPE} = [\mathbf{h}_{contour}] \tag{4.1}$$

Later, both the form and motion feature components for shape feature vector with $L_{SHAPE} = 46$ from (3.6) is evaluated.

The experimental setup used to obtain the shape feature is given in Section 4.2.3, which is evaluated based on Case study-2 testing dataset with classifier settings mentioned in Section 4.3.2. The performance measures of both the shape feature vectors are presented in Table 4.7.

From the results, it can be observed that the detection rate TPR values are higher for the shape feature with both form and motion components. This increase in detection rate for motion feature

Type of Shape feature	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
$\mathbf{h}_{SHAPE} = [\mathbf{h}_{contour}]$	94.38	5.62	97.50	91.25	8.75	2.50
$\mathbf{h}_{SHAPE} = [\mathbf{h}_{contour}\ \mathbf{h}_{motion}]$	91.88	8.12	100.00	83.75	0.00	16.25

Table 4.7: The performance measures in % specified by (3.19) to (3.24) for evaluating two variation of shape feature in (3.6). The evaluation is done on Case study-2 based on the 20% testing data.

components can be reasoned out two ways: First the increase in the number of descriptors, i.e. the number of shape feature components increases from L_{SHAPE} =29 to 46. An increase in the number of features can include more details of the activity and thus help to classify better. Secondly, the added shape components are motion features that include the temporal details of the activity.

Though the overall classification rate is less, as mentioned before in Case study-2 it is due to the technical problems in shape feature classification. It can be concluded from the above tests that the shape features with more number of descriptors and with motion feature details perform better with improved detection rates (TPR).

4.5.3 Impact of M value to compute time-dependent feature vectors

For a given jth video containing 'len' frames, the averaging window length W for computing time-dependent features changes depending on the M value (i.e, $W = \text{round}\left(\frac{len}{M}\right)$). This subsection will study the impact on classification performance while changing M value, hence change of averaging window length in extracting time-dependent features.

The captured RGB-D videos for any activity either fall or lying down is made up of varying length $'len'$ for a video. Since each subject can perform the same activity in a different manner and it is also impossible for the same person to repeat the activity similarly. Thus, the activity videos end up with varying video length. But the classifier requires feature vector inputs of fixed interval value to perform activity classification. Window averaging given by (3.2) is used to find time dependent feature vector in an interval M.

A test is done to find the optimal M setting for combined feature '$\mathbf{f}^{\,j,COMBINE}$' that gives high classification rate. In this test the combined feature is obtained following (3.17) for different fixed interval values M = [5, 10, 15]. The evaluation is carried out on Case study-2 testing

Fixed time interval (M)	Window Size Mean (W_{avg})	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
M =5	6	98.75	1.25	100.00	97.50	2.50	0.00
M = 10	3	97.50	2.50	97.50	97.50	2.50	2.50
M =15	2	96.88	3.12	96.25	97.50	2.50	3.75

Table 4.8: The performance measures in % specified by (3.19) to (3.24) for different fixed interval $M = [5, 10, 15]$ using combined feature vector '$\mathbf{f}^{\,j,COMBINE}$'. The evaluation is done on Case study-2 based on the 20% testing data.

dataset (see Section 4.3.2) using C-SVM classifier. The results are given in Table 4.8 with performance measures for each fixed interval M.

The peak classification rate of 98.75% is observed at M = 5. Consider on an average the activity videos are made up ≈35 frames, the window sizes W = [5 and 7] for given $len = 35$ does not drop any frames. The time limit M for window averaging must be fixed to achieve window sizes that prevents the drop of excess frames. When a lot of excess frames in the end are dropped the activity details are lost partially and hence result in faulty classification.

Another reason for high classification rate at $M = 5$ is the overfitting problem in the classifier, as with increase in 'M' value the dimensionality of the feature also increases ($L_f \times M$). A trade-off can be observed between classification rate and computation time for higher M, as it eventually increases the final feature length (i.e. $L_f \times M$). At last, the fixed interval selected for each feature HOG, optical flow and shape is M =10 which reduces computation time as well as achieves better classification rate.

4.5.4 Optimal ROI size $(a \times a)$ for feature extraction

The region of interest I_{ROI} obtained from modifying bounding box is normalized to fixed size $a \times a$ as shown in the Figure 3.2. This step is included in all feature extraction methods as explained in Sections 4.2.1, 4.2.2 and 4.2.3 respectively. In HOG and HOGOF the feature components (L_{HOG}, L_{HOGOF}) are dependent on the ROI size. Hence, it's fixed to a constant size $a \times a$ to have same number of feature components for all frames in an RGB-D video. The computation of features is also faster and easier when the size is small.

A test study is done to find an optimal size ($a \times a$) which has higher performance for the combined feature '$\mathbf{f}^{\,j,COMBINE}$'. The combined feature is obtained following (3.17), where different ROI sizes $a = [16, 24, 32]$ are tested. The classification rate is measured by evaluating Case study-2 test dataset in C-SVM classifier (see Section 4.3.2). The results of the evaluation are

ROI size ($a \times a$)	Feature Used	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
a =16	$\mathbf{f}^{\,j,HOG}$	96.25	3.75	96.25	96.25	3.75	3.75
a =24	$\mathbf{f}^{\,j,HOG}$	97.50	2.50	96.25	98.75	1.25	3.75
a = 32	$\mathbf{f}^{\,j,HOG}$	96.88	3.12	96.25	97.50	2.50	3.75

Table 4.9: The performance measures in % specified by (3.19) to (3.24) for HOG feature vector '$\mathbf{f}^{\,j,HOG}$' evaluated with different sizes ($a \times a$), where a = [16, 24, 32]. The evaluation is done on Case study-2 based on the 20% testing data.

ROI size ($a \times a$)	Feature Used	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
a =16	$\mathbf{f}^{\,j,COMBINE}$	97.50	2.50	96.25	98.75	1.25	3.75
a =24	$\mathbf{f}^{\,j,COMBINE}$	97.50	2.50	96.25	98.75	1.25	3.75
a = 32	$\mathbf{f}^{\,j,COMBINE}$	97.50	2.50	97.50	97.50	2.50	2.50

Table 4.10: The performance measures in % specified by (3.19) to (3.24) for combined feature vector '$\mathbf{f}^{\,j,COMBINE}$' evaluated with different sizes ($a \times a$), where a = [16, 24, 32]. The evaluation is done on Case study-2 based on the 20% testing data.

listed in Tables 4.9 and 4.10 for ROI size a = [16, 24, 32].

It can be observed that the classification rate and TPR increases as the I_{ROI} size 'a' increases (see Figure 3.2 for definition of 'a'), because the number of feature components L_f is directly related with 'a'. The combined feature with more components includes more detail of the activity and thus has a better classification rate. The HOG feature has best performance at $a = 32$ with classification rate 97.50% and TPR 97.50%.

The test is not evaluated for the ROI of size less than $a = 16$, as the default '*cell*' size is 8×8 and HOG feature cannot be obtained with the assumed '*block*' settings. A trade-off can be observed between classification rate and computation time for higher size $a \times a$. Finally, an optimal size 32×32 is fixed for each feature as better classification rate is achieved with less computation.

4.5.5 Varying metric threshold $SURF_{thres}$ in SURF

The SURF detector is used on RGB difference frames to find the human in an activity video. The key points from SURF detection are used to define the bounding box locations which is helpful in object tracking. The SURF detection can be modified by varying the metric threshold '$SURF_{thres}$' parameter, which defines the strongest feature threshold. The strongest feature threshold indicates the maxima of Hessian matrix determinant given in (2.7). For a larger value of $SURF_{thres}$ less SURF key points are detected, while for smaller $SURF_{thres}$ more SURF key points are obtained.

To evaluate the impact of $SURF_{thres}$ in the combined feature '$\mathbf{f}^{\,j,COMBINE}$' classification, tests are conducted on different metric thresholds $SURF_{thres}$ = [50, 100, 500]. The feature extraction for the combined feature is from (3.17) and are evaluated on Case study-2 (see Section 4.3.2) testing dataset using the C-SVM classifier. The classification results are listed in Table 4.11 for the HOG feature with varying SURF metric thresholds.

Metric Threshold ($SURF_{thres}$)	Feature Used	Classification Rate (%)	Miss Rate (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
50	$\mathbf{f}^{j,COMBINE}$	97.50	2.50	97.50	97.50	2.50	2.50
100	$\mathbf{f}^{j,COMBINE}$	98.13	1.87	97.50	98.75	1.25	2.50
500	$\mathbf{f}^{j,COMBINE}$	97.50	2.50	96.25	98.75	1.25	3.75

Table 4.11: The performance measures in % specified by (3.19) to (3.24) for the combined feature '$\mathbf{f}^{j,COMBINE}$' evaluated with different SURF metric threshold $SURF_{thres}$ = [50, 100, 500]. The evaluation is done on Case study-2 based on the 20% testing data.

The observations show that for the combined feature with $SURF_{thres} = 100$ a high classification rate 98.13% is achieved, and then the classification rate decreases with increase in $SURF_{thres}$. A low SURF metric threshold results in a large bounding box with background details around the object, while a high metric threshold with less key points results in a tight bounding box around the object.

A tight bounding box is useful when it eliminates the background clutter for defining the region of interest. Though for the HOG feature according to Section 2.3.1, the detection window should also include margin information around the object for improved performance. The margin information provides extra context to the person and reducing the border results in the performance loss. So for the final setup, $SURF_{thres} = 50$ is fixed for all the features HOG, optical flow and shape.

5

Conclusions

The thesis on fall identification uses three types of features HOG, optical flow and shape, to obtain a classification rate of about 95%. To define the region of interest given algorithms are used, along with existing methods on object detection. The three features are studied for fall activity classification using RGB-D videos. The object detection and feature extraction modules use MATLAB based algorithms. The optimal parameters required for the above algorithms are evaluated such that the classification rate is high. Shape feature at very low feature cost results in good classification rate. The shape feature is based on depth videos so it is useful to overcome the problem of subject's privacy. The fall detection is independent of the long lie as it uses short activity videos. The combination of three feature vectors serially results in high classification rate along with low FPR and FNR. The given methods can be used for identifying accidental falls, in health care applications.

Though a wide range of algorithms is available for object detection and feature extraction, in this thesis it is restricted to the given methods because of time constraint. The problems encountered while using the methods are as follows: The difference image detects a partial object when the subject in the video is wearing black clothing. In shape analysis, the contour obtained from morphological operation contains background objects. The defined region of interest for shape partially removes the background details. This partial removal results in the definition of extrema points outside the contour. The extrema points are also lost when human contour undergoes limb occlusions.

Bibliography

[1] C. Griffiths, C. Rooney, A. Brock, Leading causes of death in england and wales how should we group causes, Health Statistics Quarterly 2005 28 (9).

[2] W. H. O. Ageing, L. C. Unit, WHO global report on falls prevention in older age, World Health Organization, 2008.

[3] U. N. D. of Economic, S. A. P. D. (2013), World population ageing 2013.

[4] M. Mubashir, L. Shao, L. Seed, A survey on fall detection: Principles and approaches, Neurocomput. 100 (2013) 144–152.
URL `http://dx.doi.org/10.1016/j.neucom.2011.09.037`

[5] C. Rougier, J. Meunier, A. St-Arnaud, J. Rousseau, Robust video surveillance for fall detection based on human shape deformation, Circuits and Systems for Video Technology, IEEE Transactions on 21 (5) (2011) 611–622.

[6] G. Mastorakis, D. Makris, Fall detection system using kinect's infrared sensor, J. Real-Time Image Process. 9 (4) (2014) 635–646.
URL `http://dx.doi.org/10.1007/s11554-012-0246-9`

[7] C. Zhang, Y. Tian, E. Capezuti, Privacy preserving automatic fall detection for elderly using rgbd cameras, in: Proceedings of the 13th International Conference on Computers Helping People with Special Needs - Volume Part I, ICCHP'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 625–633.
URL `http://dx.doi.org/10.1007/978-3-642-31522-0_95`

[8] I. P. Raul I., Carlos M., Challenges issues and trends in fall detection systems, BioMedical Engineering OnLine 2013.

[9] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, J. Lundy, Fall detection - principles and methods, in: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, 2007, pp. 1663–1666.

[10] W. K. Pratt, Digital Image Processing (2Nd Ed.), John Wiley & Sons, Inc., New York, NY, USA, 1991.

[11] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
URL `http://dx.doi.org/10.1016/j.cviu.2007.09.014`

[12] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, 2005, pp. 886–893 vol. 1.

[13] N. Dalal, Finding people in images and videos, Ph.D. thesis, Institut National Polytechnique de Grenoble-INPG (2006).

[14] A. Bruhn, J. Weickert, C. Schnörr, Lucas/kanade meets horn/schunck: Combining local and global optic flow methods, International Journal of Computer Vision 61 (2005) 211–231.

[15] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, Springer, 2004, pp. 25–36.

[16] C. Liu, Beyond pixels: Exploring new representations and applications for motion analysis, Ph.D. thesis, Cambridge, MA, USA, aAI0822221 (2009).

[17] C. Liu, J. Yuen, A. Torralba, Sift flow: Dense correspondence across scenes and its applications, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (5) (2011) 978–994.

[18] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, R. Szeliski, A database and evaluation methodology for optical flow, Int. J. Comput. Vision 92 (1) (2011) 1–31.
URL `http://dx.doi.org/10.1007/s11263-010-0390-2`

[19] R. Chaudhry, A. Ravich, G. Hager, R. Vidal, Histograms of oriented optical flow and binetcauchy kernels on nonlinear dynamical systems for the recognition of human actions, in: in In IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2009.

[20] Y. Mingqiang, et al., A survey of shape feature extraction techniques (2008).

[21] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, ACM, New York, NY, USA, 1992, pp. 144–152.
URL `http://doi.acm.org/10.1145/130385.130401`

[22] P. Hiremath, J. R. Tegnoor, Follicle detection and ovarian classification in digital ultrasound images of ovaries, Advancements and Breakthroughs in Ultrasound Imaging, InTechOpen, UK (2013) 167–199.

[23] C. Cortes, V. Vapnik, Support-vector networks, in: Machine Learning, 1995, pp. 273–297.

[24] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using second order information for training support vector machines, J. Mach. Learn. Res. 6 (2005) 1889–1918.
URL `http://dl.acm.org/citation.cfm?id=1046920.1194907`

[25] Y. Yun, I. Y. Gu, Human fall detection via shape analysis on riemannian manifolds with applications to elderly care, in: IEEE int'l conf. on image processing, 27-30 Sept. 2015, 2015, p. 5.

[26] Evaluation of binary classifiers (12 Nov. 2015).
URL `https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers`

[27] Kinect for windows, `https://en.wikipedia.org/wiki/Kinect` (12 Nov. 2015).

List of Figures

2.1 Steps involved in the processing of Histogram of Oriented Gradients from [13] 7

2.2 The visualization of flow fields from [17]. The color code in [18] is used to visualize a flow field: each pixel denotes a flow vector where the orientation and magnitude are represented by the hue and saturation of the pixel respectively. . 9

2.3 Shape feature analysis from [20]. Left: Centroid and Eccentricity. Right: Minimum bounding rectangle. 10

2.4 Symbolic features (Extrema points) based axis of least inertia from [20] 11

2.5 An example of a separable problem in a 2-dimensional space from [22]. The support vectors, marked on *dashed line*, define the margin of largest separation between the two classes. 12

3.1 Flow diagram of modules used for fall detection and classification 14

3.2 Illustration of changing the bounding box to the region of interest as defined in Section 3.1.4. Append intensity values outside BB to form ROI of area $(l \times l)$ which is later resized to size $(a \times a)$, in the illustration the appended pixels are background pixel values indicated by '*texture*' region. 17

3.3 Window Averaging method on a fixed interval M. The graphical figure shows a line diagram that indicates frame-based vector. The video length 'len' is divided into windows of size 'W'. The grouped windows shown in '*red*' brackets undergo window averaging to give time dependent feature vectors. 18

3.4 Flow diagram of HOGOF feature extraction module 20

3.5 Flow diagram of shape feature extraction module 21

3.6 Form feature components of shape descriptor. An illustration of form features between adjacent frames n and $n+1$ is shown. The spatial points used to determine pixel velocity are indicated by colour '*blue*' (Extrema) and '*yellow*' (centroid). The Euclidean distances d_i are indicated by '*green*' solid lines. 22

3.7 An illustration of Shape descriptors with spatio-temporal shape features. The spatial points are indicated by colour '*blue*' (Extrema) and '*yellow*' (centroid). The Euclidean distances d_i are indicated by '*green*' solid lines. The pixel velocity indicated by dashed lines with arrows. The colour '*orange*' for the extrema points and '*violet*' for the centroid. 23

4.1 Frame sequence from the RGB video dataset. Row 1-3: Frame sequence for falling down activity in RGB video. Row 4-6: Frame sequence for lying down activity in RGB video. Row 7-9: Frame sequence for sitting down activity in RGB video. 29

4.2 Frame sequence from the Depth video dataset. Row 1-3: Frame sequence for falling down activity in depth video. Row 4-6: Frame sequence for lying down activity in depth video. Row 7-9: Frame sequence for sitting down activity in depth video. 30

4.3 Distribution of number of fall activities performed by each subject in Case study-1 and Case study-2. The data is further divided into training and testing data for both case studies. 31

4.4 Distribution of number of lying down activities performed by each subject in Case study-1 and Case study-2. The data is further divided into training and testing data for both case studies. . . 32

4.5 Output frames at each step of the HOG feature setup. Top Frame 1: Fall RGB frame used as input. Top Frame 2: SURF key points in '*green*' marked on RGB difference frame. Top Frame 3: The '*red*' bounding box marked in difference frame based on SURF key points. Bottom Frame 1: The ROI after appending zeros outside BB and resized to area 32×32. Bottom Frame 2: HOG feature visualization for ROI input, in which each white plot shows the distribution of gradient orientations within the HOG cell. 33

4.6 Output frames at intermediate steps of Optical flow (HOGOF) setup. Top Frame 1: Input RGB frame for the fall activity. Top Frame 2: The tracked bounding box (*red*) defined using SURF key points over the difference frame. Bottom Frame 1: The region of interest (I_{ROI}) resized to size 32×32. Bottom Frame 2: The HSV image frame I_{HSV} after colour-coding by [18]. Bottom Frame 3: HOGOF feature visualization for I_{HSV} input, in which each white plot shows the distribution of gradient orientations within the HOG cell. 34

4.7 The frame outputs at each level of the experimental setup in shape feature extraction. Frame 1: Depth video frame for the fall activity. Frame 2: The object contour frame I_{shape} and the bounding box locations. 35

List of Tables

3.1 Confusion matrix to determine parameters used in binary classification performance measures . . . 26

4.1 Case study-1 with first 50% of total videos used for training data and the rest 50% used for testing data . . . 31
4.2 Case study-2 with first 80% of videos used for training data and the rest 20% used for testing data . . . 31
4.3 The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOG, Optical flow (HOGOF) and Shape. The evaluation is done on Case study-1 based on the testing dataset. . . . 37
4.4 The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOG, Optical flow (HOGOF) and Shape. The evaluation is done on Case study-2 based on the testing dataset. . . . 38
4.5 The performance measures in % specified by (3.19) to (3.24) for evaluating serially combined feature vector. The evaluation is done on both Case study-1 and 2 based on the testing datasets. . . . 39
4.6 The performance measures in % specified by (3.19) to (3.24) for evaluating three extracted features HOGOF and HOOF. The evaluation is done on Case study-2 based on the 20% testing data. . . . 40
4.7 The performance measures in % specified by (3.19) to (3.24) for evaluating two variation of shape feature in (3.6). The evaluation is done on Case study-2 based on the 20% testing data. . . . 41
4.8 The performance measures in % specified by (3.19) to (3.24) for different fixed interval $M = [5, 10, 15]$ using combined feature vector '$\mathbf{f}^{j,COMBINE}$'. The evaluation is done on Case study-2 based on the 20% testing data. . . . 42
4.9 The performance measures in % specified by (3.19) to (3.24) for HOG feature vector '$\mathbf{f}^{j,HOG}$' evaluated with different sizes ($a \times a$), where a = [16, 24, 32]. The evaluation is done on Case study-2 based on the 20% testing data. . . . 43

4.10 The performance measures in % specified by (3.19) to (3.24) for combined feature vector '$\mathbf{f}^{j,COMBINE}$' evaluated with different sizes ($a \times a$), where a = [16, 24, 32]. The evaluation is done on Case study-2 based on the 20% testing data. 44
4.11 The performance measures in % specified by (3.19) to (3.24) for the combined feature '$\mathbf{f}^{j,COMBINE}$' evaluated with different SURF metric threshold $SURF_{thres}$ = [50, 100, 500]. The evaluation is done on Case study-2 based on the 20% testing data. 45