Computer Vision Based Techniques for Fall Detection with Application Towards Assisted Living

by

Miao Yu

A doctoral thesis submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy (PhD)

January 2013

Advanced Signal Processing Group,
School of Electronic, Electrical and Systems Engineering,
Loughborough University, Loughborough,
Leicestershire, UK, LE11 3TU

CERTIFICATE OF ORIGINALITY

I dedicate this thesis to everyone who has helped in my development,

particularly my Mum and Dad.

Abstract

In this thesis, new computer vision based techniques are proposed to detect falls of an elderly person living alone. This is an important problem in assisted living.

Different types of information extracted from video recordings are exploited for fall detection using both analytical and machine learning techniques. Initially, a particle filter is used to extract a 2D cue, head velocity, to determine a likely fall event. The human body region is then extracted with a modern background subtraction algorithm. Ellipse fitting is used to represent this shape and its orientation angle is employed for fall detection. An analytical method is used by setting proper thresholds against which the head velocity and orientation angle are compared for fall discrimination. Movement amplitude is then integrated into the fall detector to reduce false alarms.

Since 2D features can generate false alarms and are not invariant to different directions, more robust 3D features are next extracted from a 3D person representation formed from video measurements from multiple calibrated cameras. Instead of using thresholds, different data fitting methods are applied to construct models corresponding to fall activities. These are then used to distinguish falls and non-falls.

In the final works, two practical fall detection schemes which use only one un-calibrated camera are tested in a real home environment. These approaches are based on 2D features which describe human body posture.

These extracted features are then applied to construct either a supervised method for posture classification or an unsupervised method for abnormal posture detection. Certain rules which are set according to the characteristics of fall activities are lastly used to build robust fall detection methods. Extensive evaluation studies are included to confirm the efficiency of the schemes.

Contents

Statement of Originality

The contributions of this thesis are mainly on the system integration aspects, different features and algorithms are proposed to achieve efficient fall detection methods, which have potential to be used in real application for remote health caring of elderly people. These methods are tested in the experimental lab and real home environment and the novelty of the contributions is supported by the following international journal and conference papers:

Chapter 3, proposes a simple and robust fall detection system by using head velocity cue, shape cue and movement amplitude cue, the head velocity is estimated from the particle based head tracking results and the fitted ellipse on the extracted human body region using codebook background subtraction provides the human body shape information. The head velocity and shape change value are compared with proper thresholds for fall detection; the movement amplitude is analyzed to confirm further whether fall activities happen or not in order to reduce false alarms. The results have been published in:

1. M. Yu, S. Naqvi and J. Chambers , "Fall detection in the elderly by head tracking", in Proc. IEEE Statistical Signal Processing (SSP), Cardiff, UK, 2009.

2. M. Yu, S. Naqvi and J. Chambers, "A robust fall detection system for the elderly in a Smart Room", In Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Dallas, USA, 2010.

Chapter 4 proposes a fall detection scheme based on data fitting models for 3D features data fitting. Two cameras are initially calibrated by Tsai camera calibration method and a 3D person is then constructed from the background subtraction results of two calibrated cameras. Some 3D features (including the 3D position, velocity and orientation information) corresponding to fall activities are extracted to build a model for distinguishing fall activities and non-fall activities using a single threshold. Different types of models are compared in this Chapter. These works have been presented in:

3. M. Yu, S. Naqvi, A. Rhuma and J. Chambers, " One class boundary method classifiers for application in a video-based fall detection system", IET Proc. Computer Vision, Volume 6, Issue 2, pages: 90-100, 2012.

4. M. Yu, S. Naqvi, A. Rhuma and J. Chambers, "Fall detection in a smart room by using a fuzzy one class support vector machine and imperfect training data", In Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Prague, Czech Republic, 2011.

Chapter 5, proposes two effective fall detection methods for a real home application, which are based on supervised and unsupervised techniques respectively. The codebook background subtraction is used to extract the human body postures and some post-processing technique is applied to solve the background subtraction errors cased by some environmental changes in a real home environment. Some features which can describe postures in details are extracted and used to construct the corresponding supervised or unsupervised classifiers. The results of constructed classifiers, together with some rules determined from the fall characteristics, are used to distinguish fall activities and non-fall activities. The novelty of this work is supported by the following publications:

5. M. Yu, A. Rhuma, S. Naqvi, L. Wang and J. Chambers , "Posture Recognition Based Fall Detection System For Monitoring An Elderly Person In A Smart Home Environment", IEEE Transactions on Information Technology in Biomedicine, Volume 16, Issue 6, pages: 1274-1286, 2012.

6. M. Yu, A. Rhuma, S. Naqvi, L. Wang and J. Chambers, "An unsupervised online one class support vector machine based person-specific fall detection system for monitoring an elderly individual in a room environment", submitted to IEEE Transactions on Information Technology in Biomedicine.

Acknowledgements

I AM DEEPLY INDEBTED to my supervisor Professor Jonathon A. Chambers for his kind interest, generous support and constant advice throughout the past three years. I have benefitted tremendously from his rare insight, his ample intuition and his exceptional knowledge. This thesis would never have been written without his tireless and patient mentoring. It is my very great privilege to have been one of his research students. I wish that I will have more opportunities to work with him in the future.

I am extremely thankful to Professor L. Wang in the National Lab of Pattern Recognition (NLPR), China. Due to his kind help, I have learnt much from his patient advice.

I am also grateful to Dr. Mohsen and June for their support and encouragement.

I express my gratitude to all my friends Gao Jie, Yanfeng, Jie Tang, Abdulla, Adel, Mustafa, Lu Ge, Ziming, Yu Wu and Juncheng in the Advanced Signal Processing Group for providing a stable and cooperative environment within the Advanced Signal Processing Group.

Last, but most importantly, I really can not find appropriate words or suitable phrases to express my deepest and sincere heartfelt thanks, appreciations and gratefulness to my parents, my grandmothers, my brothers, my sisters and my previous girl friend for their constant encouragement, attention, prayers and their support in innumerable ways throughout my PhD and before. I would like to dedicate this thesis to all of them.

List of Acronyms

ACSS	Accelerometer Smart Sensor
ADM	Analysis and Decision Module
AR	Area Ratio
AUC	Area Under Curve
BDT	Binary Decision Tree
BIC	Bayes Information Criterion
DAGSVM	Directed Acyclic Graph Support Vector Machine
DBTC	Distance Between Two Classes
FNR	False Negative Rate
HPF	High Pass Filter
IAU	Intelligent Accelerometer Unit
KKT	Karush-Kuhn-Tucker
KNN	K-Nearest Neighbour
LPF	Low Pass Filter
MEI	Motion Energy Image
MER	Minimum Enclosing Rectangle

MFCC	Mel Frequency Cepstral Coefficient
MHI	Motion History Image
MLPNN	Multi-Layer Perceptron Neural Network
MMSE	Minimum Mean Square Error
MNRL	Maximum Negative Run-Length
MoG	Mixture of Gaussians
NBC	Naive Bayes Classifier
OCSVM	One Class Support Vector Machine
OM	Optimization Module
PCA	Principle Component Analysis
PIP	Personal Intelligent Platform
PSE	Personal SErver
RKHS	Reproducing Kernel Hilbert Space
ROC	Receiver Operating Characteristic
SIR	Sampling Importance Resampling
SIS	Sequential Importance Sampling
SoM	Sensor of Movements
SRP-PHAT	Steered Response Power with PHAse Transform
SVM	Support Vector Machine
TPR	True Positive Rate

List of Symbols

Some frequently used notations are as follows:

$\mathrm{E}(\cdot)$	Statistical expectation
$(\cdot)^T$	Transpose
$\parallel \cdot \parallel$	Euclidean norm
$\max(\cdot)$	Maximum value
$\min(\cdot)$	Minimum value
$\sum$	Summation
ρ	Bhattacharyya coefficient
$colordist(\cdot,\cdot)$	Chromatic difference
$f(x,y)$	Binary image
$(\bar{x},\bar{y})$	Centroid
R	Rotation matrix
T	Translation vector
$//$	Paralell
$P(\cdot)$	Distribution
$\ln(\cdot)$	log-liklihood

$\subset$	Subset
$\| \cdot \|_H^2$	Distance in the reproducing kernel Hilbert space
$(\cdot)_+$	$\max(\cdot, 0)$
$exp(\cdot)$	Exponential value
$D_{Euclidean}(\cdot,\cdot)$	Euclidean distance
$D_{Bhattacharyya}(\cdot,\cdot)$	Bhattacharyya distance

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 The significance of fall detection

Due to the development of health care technology in modern society, human life expectancy has grown and there continues to be a concomitant trend for an increase in the population of old people. According to the US bureau of statistics, as reported by the Guardian [2] newspaper in the UK, within 10 years old people will outnumber children for the first time. Over the next 30 years it is forecast that the number of over-65s across the world is expected to almost double, from 506 million in 2008, to 1.3 billion, a leap from 7% of the world's population to 14%. Already, the number of people in the world aged 65 and over, is increasing at an average of 870,000 each month. And among old people, a large percent of them live alone at home according to the research reported in [3]. So, caring for old people living alone is very important and is a global challenge.

There are many issues related to the care of old people living alone, and one of the most important is fall detection. According to [4], [5] and [6], falls occur commonly in the old person community and can lead to serious damage, such as broken bones, connective and soft tissue damage, even death; as such the problem is responsible for considerable medical costs, morbidity and mortality among the elderly population. As reported in [4], in the United States, falls happen among the elderly with a median age of 79 and commonly result in fractures (primarily hip and femur), estimated at 155,000 to 200,000 each year. Cost estimates range from a present annual amount of 750,000,000 to one billion US dollars, and a similar situation exists in other developed countries. Unlike when monitoring young children, for example in a nursery, it is unrealistic to assign nurses to take care of old people in their home in a 24/7 manner. So, instead of human resources, new technologies are required to detect falls, as part of the area of assisted living, with the target of reducing the tremendous costs incurred by falls in a home environment.

The governments in many developed countries have increased investment to push development of fall detection technology. As it is reported in [7], a conference held in Singapore, SiCEX 2008, promoted concepts, products and technologies related to healthcare for elderly and patients, especially fall detection. And in the USA, many research institutes, which include interdisciplinary groups of faculty, staff, and students, are being built to investigate, develop, and evaluate technology to serve the needs of older adults and others with physical and cognitive challenges. A representative one is the center for eldercare and rehabilitation technology (CERT), Missouri, where one important project is Passive Fall Detection and Gait Analysis for Fall Risk Assessment, which investigates a non-intrusive method to detect falls in a home environment [8].

1.2 The characteristic of fall activities

The definition of a fall and its characteristics is firstly summarized before introducing different fall detection techniques. According to [9], a fall is defined as “unintentionally coming to the ground or some lower level other than as a consequence of sustaining a violent blow, loss of consciousness, sudden onset of paralysis as in stroke”. Some other researchers have used a broader definition to include those falls which occur as a result of dizziness and syncope as in [10]. Based on the particular definition, falls can be further divided into different categories according to different criteria:

1) According to the orientation:

a) Frontal fall–a person falls towards his/her frontal direction, mostly with his/her face impacting with the floor.

b) Backward fall–a person falls towards his/her backward direction, mostly with the back of their head impacting with the floor.

c) Side fall–a person falls towards his/her side direction.

2) According to the amplitude:

a) Fast fall–a person falls fast, the amplitude of the body movement is large, the duration is short (1-2s).

b) Slow fall–a person falls slowly, the amplitude of the body movement is comparatively small and the duration is comparatively long.

3) According to the transition of postures:

a) Fall from standing – a person falls from an initial standing posture, this type of fall occurs when an old person walks or stands still due to slipping or unconsciousness. Both the head and center of gravity move towards one direction and their height reduces (normally to the plane of the ground). Typically, this type of fall belongs to the category of fast fall with large movement amplitude.

b) Fall from sitting – a man falls from an initial sitting posture, this type of fall occurs when an old person slips from a chair due to his/her unconsciousness. Similarly, the head and center of gravity move towards one direction with a reduced height; however, compared with fall from standing, this type of fall has a smaller movement amplitude.

c) Fall from lying – a person falls from an initial lying posture, this type of falls means that an old person rolls to the floor from the bed during sleep. Initially, the person lies on the bed, when the fall happens, the body reduces its height from the bed to the floor plane, with the final body position being near the bed. This type of falls usually happens when an old person sleeps and his/her body rolls out of the bed while the person remains unconscious.

d) Fall from other postures – a person falls from an initial bending/crouching or other postures; this type of fall happens for example when an old person ties his/her shoe lace or does other activities and suddenly becomes unconscious.

Compared with non-fall activities, fall is an unconscious activity or an activity happening beyond an old person's control (such as he/she slips and falls). And normally, fall activity ends with a lying posture on the floor, as presented in Figure 1.1. After introducing the definition and characteristics of falls, the complete scheme of a fall detection system is presented in the next section.

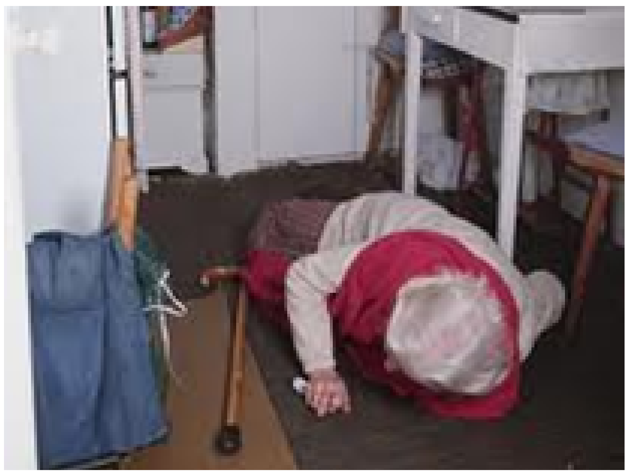


Figure 1.1. An old person falls ending with a lying posture on the floor.

1.3 Scheme of a fall detection system

A complete fall detection system is proposed in Figure 1.2. Initially, signals are acquired from different sensors (including wearable sensors such as accelerometers, cameras) and the acquired signals are then processed and corresponding information is extracted, which is then fed into a fall detection system to detect falls with the aid of certain algorithms (typically analytical algorithms or machine learning algorithms). When the fall activity of an elderly person is detected, an alarm signal is generated and this signal will be either sent to his/her family members, or some care suppliers (including a hospital, or monitoring center) by modern communication technique (such as the wire or wireless communication network presented in Figure 1.2). After receiving the alarm signal, the care staff will swiftly come to assist the elderly person.

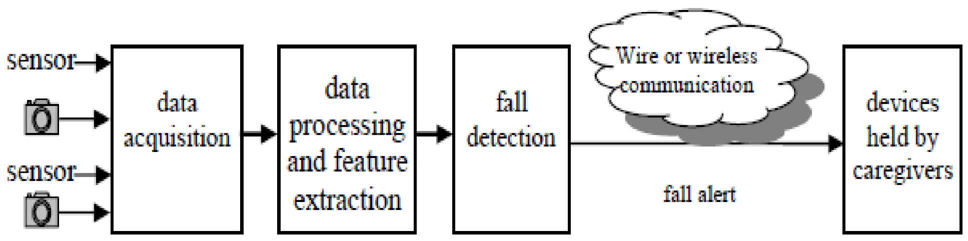

sensor
sensor
data acquisition
data processing and feature extraction
fall detection
Wire or wireless communication
fall alert
devices held by caregivers

Figure 1.2. The diagrammatic representation of a complete fall detection system.

Some examples of the sensors and extracted information used to detect falls are presented in Figure 1.3, either non-computer vision based sensors (wearable device or ambience device) or computer vision based sensors (cameras) are applied to capture certain types of signals, and corresponding information (posture information, motion information, body shape change information and so on) is then extracted from the acquired signals for fall detection purpose.

This thesis focuses on fall detection in a room environment by using computer vision techniques, thereby effective computer vision methods are proposed for detecting fall activities by using one or multiple video cameras. This approach has the major advantage that it avoids the requirement for wearing a sensor which becomes difficult for elderly people, particularly those

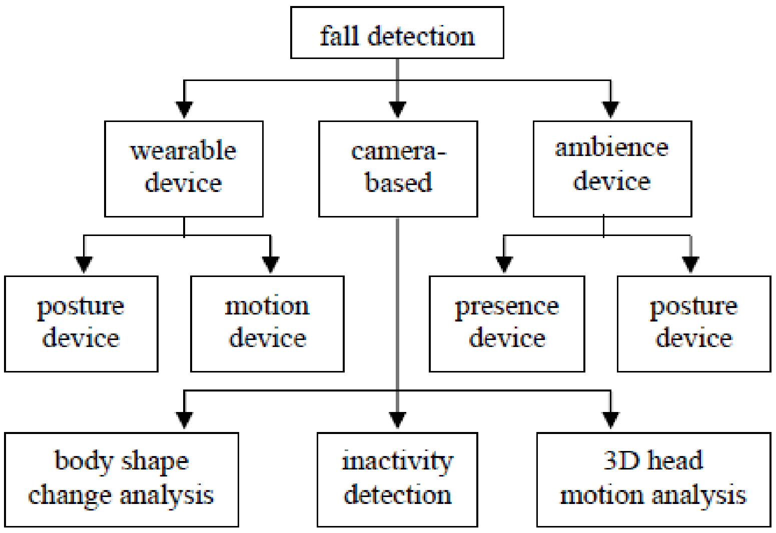


Figure 1.3. Classification of fall detection devices.

suffering from conditions such as dementia.

1.4 Aims and objectives

The aims of this thesis are to:

• Exploit state-of-the-art computer vision methods in the development of fall detection systems for potential application in assisted living.

• Use head tracking and feature extraction to characterize the position and pose of a human target.

• Employ support vector machine-based single and multi-class classifiers to perform robust fall detection.

• Introduce appropriate rules to minimize false alarms in the detection of falls.

• Evaluate methods on extensive datasets measured in laboratory and real room environments.

At the end of the study the objectives are to have:

• Demonstrated the feasibility of fall detection with the proposed systems using datasets with volunteers who attempt to mimic the movements of an elderly person in a room environment.

• Published the research finding in the leading international conferences and journals.

1.5 Organization of this thesis

This thesis is organized as follows:

Chapter 2, reviews some state-of-the-art fall detection methods by using either non-computer vision techniques or computer vision based techniques, which provide the background for the later work within the thesis.

A simple and robust fall detection method by using three cues: head velocity cue, shape cue and movement amplitude cue, which are extracted from the recorded video information is proposed in Chapter 3. A particle filtering technique is used for head tracking which exploits gradient and colour information to obtain head velocity. The codebook background subtraction method and moment-based ellipse fitting are applied to fit a human body region to an ellipse; the change in the orientation angle and the ratio of the two axes of an ellipse are used to reflect the variation of the human body shape. Finally, the movement amplitude of a person estimated from MHI is used for a further confirmation of a fall when both the head velocity and shape change exceed certain threshold values. The experiments were done in a simulated lab environment, six people were invited to attend the experiment to simulate fall activities in different directions and a series of non-fall activities, which were then used to test the proposed fall detection method.

In Chapter 4, a fall detection scheme is proposed based on 3D features and a data fitting model scheme. Two cameras are initially calibrated by the popular Tsai camera calibration method and a 3D person is then constructed from the obtained codebook background subtraction results from two calibrated cameras. Some 3D features (including the 3D position, velocity and orientation information) corresponding to fall activities are extracted to build a model for distinguishing fall activities and non-fall activities. Three types of models–single Gaussian model, mixture of Gaussians model and OCSVM model are compared in this chapter. The experiments were performed in a simulated lab environment and eight people were invited to participate in the experiment by simulating different fall activities and non-fall activities, which were used to construct the training dataset for model construction and testing dataset for performance evaluation by ROC analysis.

Two effective fall detection methods for a real home environment are presented in Chapter 5, which are respectively based on supervised and unsupervised learning techniques. The codebook background subtraction method is again used to extract the postures and certain post-processing techniques are applied to reduce the background subtraction errors cased by some environmental changes in a real home environment. Some features (projection histogram and shape-structure features) which can describe postures in detail are extracted and used to construct the corresponding supervised DAGSVM or unsupervised OCSVM classifiers. The classification results for the DAGSVM or online OCSVM, together with some rules determined from the fall characteristics, are used to distinguish fall activities and non-fall activities. The experiments were done in a real-home environment, for the supervised learning based fall detection methods, 15 people were invited to simulate different postures, which construct a dataset used for training DAGSVM and a series of simulated fall and non-fall activities by different people were recorded for testing purpose; for the unsupervised learning based fall detection method, a man simulated the normal activities like the elderly person, from which postures were extracted to build the corresponding OCSVM model to abnormal postures and normal postures, and another set of activities (including both fall activities and non-fall activities) is simulated for evaluating the performance of the unsupervised learning based method.

Chapter 6, concludes the thesis and includes suggestions for future work.

Chapter 2

LITERATURE REVIEW OF FALL DETECTION METHODS

2.1 Introduction

As mentioned in Chapter 1, efficient fall detection techniques are needed to detect fall activities for an elderly person living alone at home and corresponding alarm signals can be sent to the hospital or care monitoring centre for attention. In this Chapter, different fall detection methods are summarized in an organized way as shown in Figure 1.3. The fall detection methods can be divided into two main divisions: non-computer vision based methods (ambience and wearable devices) and computer-vision based methods (camera-based).

2.2 Non-computer vision based methods

There are many non-computer vision based methods for fall detection. For these methods, different sensors (acceleration sensor, floor vibration sensor and acoustic sensor) are used to capture the sound, vibration and human body movement information to determine a fall.

In [11], Karantonis et al. proposed a simple real-time human movement classification system by using a single, waist-mounted triaxial accelerometer unit, which is a small,wireless and low-power-consuming device. Acceleration signals generated due to gravity and body motion were sampled and processed by certain types of digital filters, such as median filter, third-order elliptical low pass filter (LPF) and seventh-order elliptical high pass filter (HPF). A hierarchical binary structure classifier was then applied on the processed data for classifying different types of movements and detecting falls based on a second-by-second scheme. The vast majority of operations was performed online in an embedded low-power microcontroller, and when a possible fall was detected, more data was sent to the receiver (local computer in this paper) for further analysis. According to the experimental results tested on six subjects, an accuracy of 95.6% was achieved for the detection of possible falls. However, this system failed to distinguish falls with fast sitting/lying due to the similar generated acceleration signals by these activities. Instead of fixing the accelerometer at the waist position, Kangas et al. [12] tested the performance of a triaxial accelerometer attached to the subject's body in different positions: head, waist and wrist to detect fall activities. The acceleration information measured by the accelerometer in different positions was compared with a proper threshold to determine a fall. The results showed that fall detection using a triaxial accelerometer worn at the waist or head is efficient, with a sensitivity of $97 - 98\%$ and specificity of 100% by using the simple threshold-based algorithm.

An acceleration sensor can be used with other devices to achieve a comprehensive fall detection system. As presented in [13], an acceleration sensor was used as one important component for a multidevice personal intelligent platform for fall detection. A multilayer process architecture was proposed in this paper; the first layer is an intelligent accelerometer unit (IAU), which is composed of several acceleration sensors and captures the acceleration data. The second layer is a personal intelligent platform (PIP), which is in charge of managing the communications with peripherals (such as multi-person intelligent platform (MIP), which is an access point to the telehealthcare center) and performing the fall detection algorithm based on the acceleration data obtained from the first layer. The proposed fall detection algorithm is based on the comparisons of the vertical angle variation and the linear AR-Burg spectrum estimate calculated from the segmented acceleration data with proper thresholds, which is a double-threshold analysis as mentioned in the paper. Two examples of magnitude threshold and angle threshold were presented to illustrate the fall detection algorithm; however, no detailed statistical analysis was presented in this work and some problems (such as the choice of threshold) need to be solved for improving the performance of this system. An extension of this work was proposed in [14], improvements have been made in both hardware and algorithm aspects: a new small-sized and low-power waterproof biocompatible accelerometer smart sensor (ACSS) was applied and an additional user interface module was integrated in the second layer (denoted as personal server (PSE) in this paper) to allow the elderly person to access some of the most important data being processed; from the algorithm aspect, an additional time analysis was used by convoluting the obtained acceleration data segment with certain defined waveforms, to detect some problematic fall events such as a knee fall. 332 samples of fall and non-fall activities simulated by 31 young and healthy males and females were tested, 100% sensitivity and 95.68% specificity were obtained and a further reduction of false positives can be obtained by manually canceling the fall alarm through the user interface. In [15], a fall detection system with a distributed processing architecture that explicitly integrated capabilities for continuous adaptation to the medium, the context and the user, was presented. The accelerometer based smart sensor, referred to as the Sensor of Movements (SoM), performed the first energy or impact event detection. When an impact was detected, 4s of accelerometer data around the impact instant was transmitted as an activity pattern to an analysis and decision module (ADM) for fall detection by using the double-threshold analysis proposed in [13] and [14]. The most important contribution in this work, an optimization module (OM), was designed to find the optimum operation parameters for the impact detection with the SoM and the fall detection with the ADM. Personalization and adaptation can be achieved with the OM using the new obtained activity patterns for operation parameters updating or updating the firmware implemented in the processing modules of the SoM. A set of experiments developed by a cohort of young volunteers has demonstrated the feasibility of the proposed fall detection system: with a 100% success for impact detection (under the optimal parameter set obtained by the OM), 100% sensitivity and 95.68% specificity rates for fall detection.

Besides the accelerometer, some other sensors such as acoustic and vibration sensors were also applied for the non-computer vision based methods. Li et al. [16] developed an acoustic fall detection system, which automatically detected a fall and reported it to the care giver. The study used an 8-microphone circular array which provided a better 3-D estimation of the sound location by using the steered response power with phase transform (SRP-PHAT) algorithm [17], and the sound signal was then enhanced by a beamforming technique by the aid of the obtained location information. Mel frequency cepstral coefficient (MFCC) features were extracted from the enhanced sound signal and the k-th nearest neighbor method was applied to discriminate a fall from a non-fall activity. A pilot experiment on a dataset containing 30 fall activities and 120 non-fall activities was performed, all the falls were detected and only 6 non-fall activities were taken as fall activities. An improvement of [16] was proposed in [18] by introducing the sound source's height information; if the sound source's height is larger than a particular threshold, then it is unlikely to be a fall. In this way, the false alarms due to background noise were reduced to a large extent. A larger dataset which contained 120 simulated fall sounds and 120 simulated non-fall sounds by three stunt actors was used for evaluation. A good performance was obtained with 100% fall detection rate and 3% false detection rate. In order to solve the problem that it is not easy to obtain the realistic fall sound for training, a one class classifier technique was proposed in [19] using only the non-fall sound for classifier construction. The sound signals were initially preprocessed by a Wiener filter for noise removal and the extracted MFCC features were then used to build the corresponding classifiers. In this work, three types of one class classifier: nearest neighbor classifier, the one class support vector machine classifier and mixture of Gaussians classifier were tested. From the preliminary results from the experimental part, it was found that the fall detection results by the three one class classifiers were comparable with those by using the popular two-class support vector machine from the ROC curve analysis.

Alwan et al. [20] proposed a design for a floor vibration-based fall detection system that was completely passive and unobtrusive to the resident. The system used a special piezoelectric sensor coupled to the floor surface by means of mass and spring arrangement. Successful differentiation between the vibration patterns of a human fall from other activities of daily living and from the falls of other objects were achieved. Laboratory tests were conducted using anthropomorphic dummies. The results showed 100% fall detection rate with minimum potential for false alarms. The drawback of this approach was the very limited range of the vibration sensor; i.e. only 20 feet. Moreover, the vibrations couldn't be detected on all kinds of floor materials.

The acoustic and floor vibration sensors can be used together for a robust fall detection system. Zigel et al. in [21] proposed a fall detection system based on floor vibration and sound sensing; temporal and spectral features were extracted from the obtained vibration and sound signals, and a Bayes' classifier was then applied to classify fall and nonfall activities based on the extracted features. In their work, a doll which mimicked a human was used to simulate falls and their system detected such falls with a fall detection rate of 97.5% and a false detection rate of 1.4%.

Although non-computer vision based methods may appear to be suitable for wide application in the fall detection field, several problems do exist; they are either inconvenient (elderly people have to wear acceleration sensors) or easily affected by noise in the environment (acoustic sensors and floor vibration sensors). For the non-computer vision based methods, additional hardware (such as the data acquisition card) needs to be applied to convert the obtained analogue signals into digital signals for the PCs to process, which adds extra costs. In order to overcome these problems, computer vision based fall detection techniques are adopted. For the computer vision based method, there is no need for the elderly people to wear certain equipment and it is not affected by the background noises; besides, only cheap USB cameras are needed for the computer vision based method so that the constructed fall detection system is economical. Infringement of personal privacy is a concerning issue for computer vision based fall detection systems and elderly people may worry that they are being 'watched' by cameras. However, in most computer vision based fall detection systems, only the alarm signal (sometimes with a short video clip as further confirmation of whether an elderly person has fallen or not) will be sent to the caregivers or family members when a fall is detected; additionally, the original video recordings of an elderly person's normal activities will not be stored, nor transmitted.

2.3 Computer vision based methods

In the last 10 years, there have been many advances in computer vision and camera/video and image processing techniques that use real time movement of the subject, which opens up a new branch of methods for fall detection. Compared with non-computer vision based methods, computer vision based methods have the following advantages: (1) they are non-intrusive, an elderly person need not wear some special equipment such as an accelerometer; (2) they are not easily affected by noise in the environment (suffered by floor vibration and acoustic sensors based methods). Based on the analysis of algorithms for fall detection, the computer vision based methods are divided into two categories: analytical methods and machine learning methods.

2.3.1 Analytical methods

For analytical methods, certain types of video features are extracted and these features are analyzed empirically to determine whether falls happen or not. The most popular used analytical method is the threshold-based algorithm; a threshold is set empirically and the extracted features are compared with the threshold to determine whether a fall happens or not.

Miaou et al. [22] and [23] proposed a detection system consisting of an omni-dimensional camera and a computer server, which had the advantage of capturing 360° simultaneously in a single shot to remove blind spots. In this approach, a clean background was first obtained. After that, the foreground of interest was obtained by subtracting the background model from the current image. After removing noise from the picture, a rectangle enclosing the object was created. The height to width ratio of this rectangle was compared with a particular threshold to detect falls. The threshold value in this system was customizable depending on personal physique. The experimental results showed a detection rate of 78% without personal information that increased to 90% with personal information.

Rougier et.al in [24] proposed a fall detection system based on the motion history image and some changes in the shape of the person. The movement amplitude was measured by the motion history image (MHI) obtained from the frame differencing results and when a large amplitude movement was detected, the shape change information (such as the changes of the aspect ratio and the orientation angle of the fitted ellipse) was compared with proper thresholds for fall detection. The threshold values were chosen empirically and the experimental results showed a good rate of fall detection with the sensitivity of 88%, and an acceptable rate of false detection with a specificity of 87.5% being obtained, assuming fixed threshold. In another of their works [25], a fall detection system based on the analysis of the human shape deformation during the video sequence was proposed. The elderly person's silhouette was tracked along the video sequence using shape context matching, and the Procrustes shape analysis was then applied to estimate the shape deformation value based on the shape context matching results. A fall was confirmed if the shape deformation value was larger than a preset threshold, and an inactivity period was then detected by analyzing the mean and the standard deviation of the shape deformation value during a period of time. ROC analysis was applied in the experiment and a sensitivity of 95.5% and a specificity of 96.4% were obtained under the optimal threshold set.

Shoaib et al. [26] proposed a novel context based human fall detection mechanism in a real home environment by using a threshold method. The image was divided into small blocks, the centroid location of the head and feet from each frame were used to learn a context model consisting of normal head and floor blocks. For each floor block, an associated Gaussian distribution representing a set of head blocks was also trained and that Gaussian distribution was used to define the head mean location (the mean of the centroid positions of the head blocks) for that particular floor block. To detect a fall, head and feet locations were initially obtained; the vertical distance between the head location and the head mean location corresponding to the floor block at the feet location was compared. If the distance was larger than a particular threshold, then a fall was detected. As presented in the conclusion, the method achieved a high fall recognition rate of about 96% in an unconstrained real world home environment. The core of this system was the location of the head's position and it failed when the head was invisible in the image.

Instead of 2D features, 3D features are extracted and compared with proper threshold for fall detection in some works. Leone et al. [27] proposed a similar approach that reconstructed 3D bodies of the subjects. In their work, the mixture of Gaussian background subtraction method was used initially with a depth-image to obtain the foreground region. The 3-D position of the centroid of the foreground region was obtained by a special TOF (time-of-flight) camera, which was self-calibrated initially by automatic floor detection. The distance from the 3-D centroid position of the person and the floor plane was compared with a threshold to detect a fall. Good fall detection performance can be obtained as presented in the experimental part, 100% fall detection rate is obtained with a very small false detection rate of 2.7% under a height threshold of 0.4m.

Rougier [28] developed an approach to detect a fall using monocular head tracking in real time. The head was tracked by a particle filter and its 3D position was determined. The head's 3D velocity was then calculated and compared with a particular threshold to determine whether a fall happens or not. The proposed fall detection system can achieve perfect fall detection rate (100%); however, this system will easily mistake some non-fall activities (such as fast sitting) as fall activities. A. Nghiem in [29] also proposed a fall detection algorithm by detecting the head position, but using a Kinect camera for depth video. Moving regions were initially detected using a background subtraction algorithm, in each detected moving region; pixels were clustered according to their depth (provided by the Kinect camera), which is able to separate moving objects even if they overlap each other. The possible head positions for each moving object was then determined and the head position for the particular moving object which was classified as the person was taken as the final head position. The fall detection algorithm was based on comparing the speed of the head and the body centroid, as well as their distances to the ground with proper thresholds. Experimental results on a particular dataset showed that 29 out of 30 falls were detected and no false detections were generated for the 31 non-fall activities.

Another threshold based method on 3D features was proposed in [30]. Calibrated cameras were used to reconstruct the three-dimensional shape of a person. Fall events were detected by analyzing the volume distribution along the vertical axis, and an alarm was triggered when the major part of this distribution was abnormally near the floor over a predefined period of time, which implied that a person had fallen on the floor. The experimental results showed good performance of this system (achieving 99.7% fall detection rate or better with four cameras or more) and a graphic processing unit (GPU) was applied for efficient computation.

The performance of the threshold-based algorithm is strongly related to the chosen thresholds. For obtaining a good fall detection performance, proper thresholds need to be chosen for the fall detection systems; however, in the real scenario, sometimes it is difficult to choose the proper thresholds for different persons to be monitored. In order to solve this problem, machine leaning algorithms can be applied.

2.3.2 Machine learning algorithms

Machine learning algorithms, as proposed in [31], have been used in a wide range of areas and many researchers have applied these machine learning algorithms for fall detection application. For machine learning algorithms, different types of video features are extracted from video signals, and these features are applied to train either classifiers by supervised methods (such as k-nearest neighbor, neural network, support vector machine and Hidden Markov Model (HMM)) to classify different types of postures or activities for fall detection, or a general normal model by unsupervised methods (such as single Gaussian, mixtures of Gaussians, Parzen window and one class support vector machine) to distinguish normal activities and falls.

Supervised methods

For the supervised methods based fall detection system, some video features are extracted from postures or short video sequences. and the extracted features are then used to construct a particular supervised classifier for distinguishing different postures or activities to detect falls.

Posture recognition based fall detection methods are proposed in [32], [33] and [34]. In [32] and [34], projection histogram features were extracted from the segmented human body region from the background subtraction method. And different types of supervised classifiers such as a neural fuzzy network in [32] and posture probabilistic template in [34] were constructed from the projection histogram features for classifying different postures. And if the detected posture changed from ‘stand’ to ‘lie’ in a short time [32] or the ‘lie’ posture stays for a long time [34], fall activities are reported. A posture recognition rate of 97.8% was achieved in [32] and four sequences were shown to illustrate falls can be successfully detected by the posture recognition results combined with the rule set in this work (a fall is confirmed when the posture changed from ‘stand’ to ‘lie’ in a short time), and a fairly robust posture recognition result (about 90% for three different datasets) was reported in [34].

Similar projection histogram features were also used in [33] with an improvement by using a statistical scheme to reduce the effect of human body upper limb activities, and a more common k-nearest neighbour classifier was applied for posture classification purpose. A fall activity was then confirmed if the time difference between a 'stand' posture and 'lie' posture is less than a threshold; which is determined by statistical hypothesis testing for distinguishing a fall event from lying down. In the experiment, it was presented that the obtained threshold for fall confirmation is 0.4s and a correct detection rate of 84.44% is obtained on fall detection and lying down event detection according to their experimental results.

Projection histogram features can be combined with other features to achieve a more robust system for posture recognition and fall detection. In [35], projection histogram features combined with ellipse features were extracted for posture classification, the features were used to train a directed acyclic graph support vector machine (DAGSVM) classifier and four different types of postures (stand, bend, sit and lie) were classified, the classification results together with the floor region detected during a floor detection phase were applied to detect falls. The experimental results showed that the posture recognition rate of combining ellipse features and projection histogram features is better than that of using either separately, and the final fall detection system was tested on a 15 person dataset, a high fall detection rate (97.08%) and very low false detection rate (0.8%) were achieved.

Besides posture features, some features can also be extracted from short video sequences, Hazelhoff et al. [36] adopted a system with two uncalibrated cameras. The human body regions were firstly extracted from both cameras; then two features, the direction of the main axis of the body and the ratio of the variances in the x and y directions were obtained by principle component analysis (PCA); a Gaussian multi-frame classifier was used to recognize fall activities based on the obtained features and the robustness of the system was increased by a head-tracking module, which can reject false positives. The performance of this system was evaluated in different situations (including some challenging situations such as a person carrying objects and occlusions) and more than 85% fall detection rate can be achieved in real-time.

In [37], Ni et al. proposed a computer vision based fall prevention system for hospital ward application. A Microsoft sensor which can obtain the colour and depth information was applied; motion features and shape features, such as motion history image (MHI), histogram of oriented gradients (HOG) and histogram of optic flows (HOF) from both colour and depth image sequences were extracted, which were then fused via a multiple kernel learning framework for training the fall event detector. Experimental results demonstrated the high accuracy that can be achieved by the proposed system with an activity recognition accuracy of 98.76%.

Mirnahboub et al. [38] proposed a view-invariant fall detection system by using a single camera. The silhouette area extracted from background subtraction combined with inclination angle were extracted from a video sequence as features. And these were then fed into the popularly-used support vector machine (SVM) for classifying fall activities and non-fall activities. Different kernels were tested in this work and the experimental results on a public dataset showed that the polynomial kernel of 2^{nd} degree can achieve the best performance with 100% fall detection rate and less than 1% of mistaking non-fall activities as falls.

The extracted features from short video sequences can be used to build an HMM for activity recognition to detect fall activities. For [39], a bounding box and motion information were extracted from consecutive silhouettes as features. These features were then used to train HMMs for classifying fall and non-fall activities. Preliminary results were presented by constructing three HMMs for walking, kneeling and falling activities from several training sequences and the most likely state sequence for a particular test sequence can be successfully estimated by the corresponding HMM. In [40] a method was presented based on short video sequence activity classification. In this work, a novel method was proposed to extract a person's three-dimensional orientation information from multiple uncalibrated cameras. From extracted orientation information from a short video sequence, an improved version of HMM—-layered hidden Markov model (LHMM) was trained and used to detect falls. The experimental results on falling and walking sequences showed that a fall detection rate of 98% can be achieved by using two cameras, with no walking activities mistaken as falls. Htike et al. [41] presented a vision-based framework that could detect falls using a single camera, irrespective of the viewpoint of the camera with respect to the subjects. The proposed system made use of invariant pose models which performed view-invariant human pose recognition by using the chord distribution of the resampled points along the contour of the extracted foreground region. Based on the chord distribution information, inference with an expectation-maximization algorithm was performed on an ensemble of pose models and the probability value that the given frame contained a corresponding pose was then calculated. The system finally detected falls by analyzing a sequence of frames using a fuzzy hidden Markov model (FHMM) based on the estimated pose probability values for every frame and this system achieved a 94.1% success rate when it was tested on a challenging multiple view dataset. A multi-camera based HMM approach was proposed in [42], wherein projection histogram features were extracted from every single frame for posture recognition by a posture probabilistic template, the results were then fed into an HMM model which exploited the temporal coherence of the postures for detecting falls for an acquired sequence. Multiple calibrated cameras were used to transfer the appearance information to solve the initial occlusion problem when the person passes to another monitored room.

Besides the features extracted from postures or short video sequences, some other features can also be applied to construct the corresponding supervised classifiers. In [43], Mihailidis et al. used a single camera to classify fall and non-fall activities. Carefully engineered features, such as silhouette features, lighting features and flow features were extracted to allow the system to be robust to lighting, environment and the presence of multiple moving objects. Three pattern recognition methods were compared (logistic regression, neural network and support vector machine) and the neural network achieved the best performance with a fall detection rate of 92% and a false detection rate of 5%. Foroughi et al. [44] proposed a new method for fall detection using a multi-class support vector machine (MCSVM). A combination of best-fit approximated ellipse around the human body, projection histograms of the segmented silhouette and temporal changes of head pose was extracted as features and the extracted feature vectors were fed to an MCSVM for precise classification of motions and determination of fall events. A reliability rate of 88.08% was achieved in the experimentation.

3D features were applied in [39] by constructing a 3D voxel person from multiple calibrated cameras. Based on the extracted 3D features (including the 3D centroid and orientation information), Anderson proposed a fuzzy logic based linguistic summarization for fall detection. A hierarchy of fuzzy logic was used, where the output from each level was summarized and fed into the next level for inference. Corresponding fuzzy rules were designed under the supervision of nurses to ensure that they reflect the manner in which elderly people perform their activities. The proposed framework was extremely flexible and rules can be modified, added, or removed to allow for per-resident customization. This system was tested on a dataset which contained 14 fall activities and 32 non-fall activities, all the fall activities were correctly detected and only two non-fall activities were mistaken as fall activities (100% fall detection rate and 6% false detection rate), which showed an acceptable level of performance.

The main problem for supervised fall detection methods is that they do not provide a person-specific solution for individuals. A large dataset needs to be constructed initially (which should contain the data collected from many people in different views) for a supervised fall detection system, if a person does not fit the dataset very well (such as if he/she is obese), a good performance can definitely not be obtained for this specific person. Moreover, supervised fall detection methods will be affected by occlusions which happen in a real home environment. In order to solve these problems, unsupervised methods can be exploited.

Unsupervised methods

As described in [31], an unsupervised learning method solves the problem of finding the hidden structure in unlabeled data or the normal model which unlabeled data follow. Some data (such as features extracted from postures or short video clips) can be collected from a particular elderly person's normal activity video stream and these data can be used to construct the normal activity model by some unsupervised learning methods, which can then be used to distinguish falls and normal activities. The representative unsupervised fall detection methods include [45] and [46]. In [45], a single camera was used for video recording and the particle filter technique was applied to track the human body with an ellipse model. From the tracking results, they obtained the position information and for normal activities, this was used to find the "usual activity region" by using an expectation maximization (EM) method. A fall was detected when a person's position was outside the "usual activity region" for a certain time longer than the preset time threshold. In [46], a shape matching technique was used to track the person's silhouette through the video sequence. The shape deformation was then calculated from shape-context information extracted from the obtained silhouettes. The calculated shape deformation along with the inactivity time of an elderly person were used as features to construct a Gaussian mixture model to describe a person's normal activity, which was then used to detect falls and a multiple cameras scheme was applied to guarantee good performance. As presented in the experimental results, a perfect fall detection performance 100% fall detection rate and 0% false detection rate can be obtained by using four cameras together.

For the unsupervised methods, a person-specific solution can be obtained and there is no need to obtain a large dataset collected from different persons for training a classifier. However, a particular time period is needed to collect sufficient video features from that particular person for model construction, during which the fall detection can not be operated.

In summary, a taxonomy of non-computer vision based methods and computer vision based methods is given in Table 2.1.

Table 2.1. Comparison of non-computer vision based methods and computer vision based methods for fall detection

Methods description	Equipment	Merits	Demerits
Non-computer vision based methods	User-worn accelerometer devices, sound or vibration sensors	No infringement of personal privacy	High rate of false alarm; inconvenient to wear, easily affected by background noise
Computer vision based methods	Single or multiple digital cameras	Cheap, convenient to use and high accuracy	Some infringement of personal privacy, and the performance is strongly linked with illumination level

Summary

In this chapter, some popular fall detection methods based on both non-computer vision techniques and the computer vision technique were reviewed. For the non-computer vision techniques, some non-computer vision sensors (acceleration sensor, floor vibration sensor and acoustic sensor) were used to capture the sound, vibration and human body movement information for fall detection. The non-computer vision based methods are used widely in the fall detection field; however, they are either inconvenient to use or easily affected by noise in the environment. For the computer vision based methods, one or multiple cameras are applied to extract the video information. The extracted video information is either compared with certain thresholds, or used to construct supervised classifiers or an unsupervised model for fall detection. Compared with the non-computer vision based methods, computer vision based methods are not affected by the background noise and an elderly person need not wear any equipment. Although infringement of personal privacy is a concerning issue for computer vision based fall detection systems, this can be solved by sending only the alarm signal to the caregivers or family members when a fall is detected, without recording or transmitting the original video recordings of an elderly person's normal activities.

For the current computer-vision based techniques, four main problems exist:

1. For most of the threshold-based methods, only one particular feature (such as the ratio between the fitted rectangle or 3D head velocity) is used for comparison with the threshold, which is not enough and may generate high false detection rate. For a robust fall detection system, different types of features (cues) should be extracted, and a decision should then be made by comparing different features with proper thresholds in an organized way.

2. Most of the 2D features used in the fall detection works are not

invariant to directions; either direction invariant 3D features need to be used or 2D features captured from different directions need to be used to build a supervised classifier or unsupervised model which is invariant to directions.

3. For the posture classification methods for fall detection, an improved classifier should be applied for achieving a better posture classification performance. Besides, the current posture classification based methods are not easy to distinguish fall activities from fast lying activities. Additional information, such as floor region information is needed to distinguish these two activities.

4. For the unsupervised fall detection methods in either [45] or [46], not enough information is exploited (only the position information is used in [45] and two-dimensional feature vectors are applied to train the GMM model in [46]) thus different types of activities can not be distinguished efficiently. More information is needed either from postures or video sequences for a robust model construction. Additionally, certain types of rules can also be used together with the constructed model to achieve a better fall detection performance.

Different techniques are proposed in the next three contribution chapters to solve these four problems to achieve more robust fall detection methods for better detection performance.

Chapter 3

FALL DETECTION BY USING THREE CUES: HEAD VELOCITY, SHAPE CHANGE AND MOVEMENT AMPLITUDE

3.1 Introduction

To cope with the problem that only one particular cue is applied in most of the threshold-based methods for fall detection, in this chapter, three cues obtained from head velocity, shape and movement information are used for achieving a simple and robust fall detection system. The head is tracked by a particle filter using gradient and colour information; the velocity of the head is then estimated from the head tracking results. A codebook background subtraction method is applied to extract the moving object and moment-based ellipse fitting is applied to fit the human body region with an ellipse. Changes of the orientation angle and ratio between the two axes of the ellipse are used to reflect the shape change. A large head velocity and a large shape change indicate a possible fall and finally, the movement amplitude of a person is estimated for a further confirmation of a fall activity. Experiments are presented to show that by properly combining these three cues: head velocity, shape change and movement amplitude, a simple but robust fall detection system can be achieved which can distinguish falls from different types of non-fall activities.

3.2 Particle filter for head tracking

In this section, the particle filtering scheme for head tracking is introduced, which is exploited to estimate the head velocity for detecting falls. The aim of the particle filter is to estimate the assembled state vectors $\mathbf{X}_t = \{\mathbf{x}_j, j = 0, ...t\}$ at a time instance t based on the observations $\mathbf{Z}_t = \{\mathbf{z}_j, j = 0, ...t\}$ using the sampling-based method [47]; the assembled state vectors $\mathbf{X}_t$ and observations $\mathbf{Z}_t$ have different meanings for different applications. The particle filter is developed in the general sense then specialized to head tracking.

3.2.1 Minimum Mean Square Error (MMSE) estimator

Assuming at a time instance t, a number of observations $\mathbf{Z}_t = \{\mathbf{z}_j, j = 0, ...t\}$ is available, where $\mathbf{z}_i$ is an observation vector at the discrete time instance i, and the target is to estimate the assembled state vectors $\mathbf{X}_t$ based on observations $\mathbf{Z}_t$. One way is to minimize the Bayesian mean square error defined by:

$$\begin{aligned} Bmse(\hat{\mathbf{X}}_t) &= E_{p(\mathbf{X}_t,\mathbf{Z}_t)}[\| \mathbf{X}_t - \hat{\mathbf{X}}_t \|^2] \\ &= \int\int \| \mathbf{X}_t - \hat{\mathbf{X}}_t \|^2 \, p(\mathbf{X}_t, \mathbf{Z}_t) d\mathbf{X}_t d\mathbf{Z}_t \end{aligned} \quad (3.2.1)$$

where $\hat{\mathbf{X}}_t$ is the estimate of $\mathbf{X}_t$, and the multi-dimensional definite integrals in (3.2.1) are defined between $-\infty$ and $+\infty$ but the limits are not shown in the development.

By conditional probability:

$$p(\mathbf{X}_t, \mathbf{Z}_t) = p(\mathbf{Z}_t)p(\mathbf{X}_t|\mathbf{Z}_t) \tag{3.2.2}$$

and equation (3.2.1) can be rewritten as:

$$Bmse(\hat{\mathbf{X}}_t) = \int [\int \parallel \mathbf{X}_t - \hat{\mathbf{X}}_t \parallel^2 p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t]p(\mathbf{Z}_t)d\mathbf{Z}_t \tag{3.2.3}$$

In order to minimize equation (3.2.3) with respect to $\hat{\mathbf{X}}_t$, the term in the middle bracket of equation (3.2.3) is differentiated with respect to $\hat{\mathbf{X}}_t$ to obtain:

$$\begin{aligned}\frac{\partial}{\partial \hat{\mathbf{X}}_t}\int \parallel \mathbf{X}_t - \hat{\mathbf{X}}_t \parallel^2 p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t &= \int \frac{\partial}{\partial \hat{\mathbf{X}}_t} \parallel \mathbf{X}_t - \hat{\mathbf{X}}_t \parallel^2 p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t \\ &= \int -2(\mathbf{X}_t - \hat{\mathbf{X}}_t)p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t \qquad (3.2.4) \\ &= -2\int \mathbf{X}_t p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t + 2\hat{\mathbf{X}}_t \int p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t\end{aligned}$$

which when equated to zero results in:

$$\begin{aligned}\hat{\mathbf{X}}_t &= \int \mathbf{X}_t p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t \\ &= E(\mathbf{X}_t|\mathbf{Z}_t) \qquad (3.2.5)\end{aligned}$$

$E(\mathbf{X}_t|\mathbf{Z}_t)$ is called the MMSE estimator of $\mathbf{X}_t$ based on the observations $\mathbf{Z}_t$. By letting $\hat{\mathbf{X}}_t = E(\mathbf{X}_t|\mathbf{Z}_t)$, the Bayesian mean square error is thereby minimized. Evaluation of the integral for calculating $E(\mathbf{X}_t|\mathbf{Z}_t)$ is considered next.

3.2.2 Sequential Importance Sampling Particle Filtering

Importance Sampling for Monte Carlo integration

From equation (3.2.5), it is clear the integration operation is required to obtain the MMSE. One simple and efficient way to calculate this integration is to use the Monte Carlo method for numerical integration [47]. Assuming N samples $\{\mathbf{X}_t^i | i = 1, ..., N\}$ are drawn from the distribution $p(\mathbf{X}_t|\mathbf{Z}_t)$, where N is large enough, the distribution can be approximated as:

$$p(\mathbf{X}_t|\mathbf{Z}_t) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{X}_t - \mathbf{X}_t^i) \tag{3.2.6}$$

Then the MMSE estimator can be approximated as:

$$\begin{aligned} E(\mathbf{X}_t|\mathbf{Z}_t) &\approx \int \mathbf{X}_t(\frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{X}_t - \mathbf{X}_t^i))d\mathbf{X}_t \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_t^i \end{aligned} \tag{3.2.7}$$

However, in most cases $p(\mathbf{X}_t|\mathbf{Z}_t)$ is unknown so that it is not possible to sample directly from this posterior distribution. Instead, a technique called importance sampling [47] is applied. Instead of directly sampling on the distribution $p(\mathbf{X}_t|\mathbf{Z}_t)$, another distribution $q(\mathbf{X}_t|\mathbf{Z}_t)$ can be used which is comparatively easy to obtain, then equation (3.2.5) can be rewritten by introducing $q(\mathbf{X}_t|\mathbf{Z}_t)$ as:

$$\begin{aligned} E(\mathbf{X}_t|\mathbf{Z}_t) &= \int \mathbf{X}_t p(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t \\ &= \int \mathbf{X}_t \frac{p(\mathbf{X}_t|\mathbf{Z}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)} q(\mathbf{X}_t|\mathbf{Z}_t)d\mathbf{X}_t \end{aligned} \tag{3.2.8}$$

A sufficient number of N samples $\{\mathbf{X}_t^i | i = 1, ..., N\}$ is then drawn from $q(\mathbf{X}_t|\mathbf{Z}_t)$, so that equation (3.2.8) can be written as:

$$\begin{aligned} E(\mathbf{X}_t|\mathbf{Z}_t) &= \int \mathbf{X}_t \frac{p(\mathbf{X}_t|\mathbf{Z}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)} q(\mathbf{X}_t|\mathbf{Z}_t) d\mathbf{X}_t \\ &\approx \int \mathbf{X}_t \frac{p(\mathbf{X}_t|\mathbf{Z}_t)}{q(\mathbf{X}_t|\mathbf{Z}_t)} (\frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{X}_t - \mathbf{X}_t^i)) d\mathbf{X}_t \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_t^i \frac{p(\mathbf{X}_t^i|\mathbf{Z}_t)}{q(\mathbf{X}_t^i|\mathbf{Z}_t)} \end{aligned} \tag{3.2.9}$$

As in [47], w_t^i is used to replace $\frac{p(\mathbf{X}_t^i|\mathbf{Z}_t)}{q(\mathbf{X}_t^i|\mathbf{Z}_t)}$, and equation (3.2.9) becomes:

$$E(\mathbf{X}_t|\mathbf{Z}_t) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_t^i w_t^i \tag{3.2.10}$$

Sequential Importance Sampling and Weight Calculation

From equation (3.2.10), it is evident that in order to find the MMSE estimate of $\mathbf{X}_t$, the samples $\{\mathbf{X}_t^i, i = 1, ..., N\}$ and the weights $\{w_t^i, i = 1, ..., N\}$ are required. Assuming at time instance $t-1$, the values of $\{\mathbf{X}_{t-1}^i, i = 1, ..., N\}$ and $\{w_{t-1}^i, i = 1, ..., N\}$ are known and by using conditional probability, the following equation holds for the distribution of $q(\mathbf{X}_t|\mathbf{Z}_t)$:

$$q(\mathbf{X}_t|\mathbf{Z}_t) = q(\mathbf{x}_t|\mathbf{X}_{t-1}, \mathbf{Z}_t) q(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1}) \tag{3.2.11}$$

so that samples $\mathbf{X}_t^i \sim q(\mathbf{X}_t|\mathbf{Z}_t)$ can be obtained by augmenting each of the samples $\{\mathbf{X}_{t-1}^i, i = 1, ..., N\}$ (which are already obtained) with the new state vector $\mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{X}_{t-1}^i, \mathbf{Z}_t)$. In this way, the samples $\{\mathbf{X}_t^i, i = 1, ..., N\}$ are obtained.

For calculating $\{w_t^i, i = 1, ..., N\}$, it is observed that:

$$\begin{aligned} p(\mathbf{X}_t|\mathbf{Z}_t) &= \frac{p(\mathbf{z}_t|\mathbf{X}_t, \mathbf{Z}_{t-1})p(\mathbf{X}_t|\mathbf{Z}_{t-1})}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})} \\ &= \frac{p(\mathbf{z}_t|\mathbf{X}_t, \mathbf{Z}_{t-1})p(\mathbf{x}_t|\mathbf{X}_{t-1}, \mathbf{Z}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1})}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})} \\ &= \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{z}_t|\mathbf{Z}_{t-1})}p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1}) \\ &\propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{t-1}) \end{aligned} \tag{3.2.12}$$

in which the first-order Markov assumption: $p(\mathbf{x}_t|\mathbf{X}_{t-1}, \mathbf{Z}_{t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the assumption that $\mathbf{z}_t$ only relies on $\mathbf{x}_t$:$p(\mathbf{z}_t|\mathbf{X}_t, \mathbf{Z}_{t-1}) = p(\mathbf{z}_t|\mathbf{x}_t)$ are used.

From equations (3.2.11) and (3.2.12), the following can be obtained:

$$\begin{aligned} w_t^i &= \frac{p(\mathbf{X}_t^i|\mathbf{Z}_t)}{q(\mathbf{X}_t^i|\mathbf{Z}_t)} \\ &\propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)p(\mathbf{X}_{t-1}^i|\mathbf{Z}_{t-1})}{q(\mathbf{x}_t^i|\mathbf{X}_{t-1}^i, \mathbf{Z}_t)q(\mathbf{X}_{t-1}^i|\mathbf{Z}_{t-1})} \\ &= w_{t-1}^i \frac{p(\mathbf{z}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)} \end{aligned} \tag{3.2.13}$$

For the commonly used SIR (sampling importance resampling) particle filter [47], the term $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$ is chosen as the transitional prior: $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, so the weight for the SIR particle filter is calculated as:

$$w_t^i = w_{t-1}^i p(\mathbf{z}_t|\mathbf{x}_t^i) \tag{3.2.14}$$

After obtaining $\{w_t^i, i = 1, ..., N\}$, a normalized weights set can be obtained by:

$$\tilde{w}_t^i = \frac{w_t^i}{\sum_{j=1}^N w_t^i}, \quad i = 1, ...N \tag{3.2.15}$$

and the normalized weights set $\{\tilde{w}_t^i, i = 1, ..., N\}$ is then used to replace the original weights set $\{w_t^i, i = 1, ..., N\}$:

$$w_t^i = \tilde{w}_t^i, \quad i = 1, ...N \tag{3.2.16}$$

in order to make the integration of $p(\mathbf{X}_t|\mathbf{Z}_t) \approx \sum_{i=1}^N w_t^i \delta(\mathbf{X}_t - \mathbf{X}_t^i)$ be unity, i.e. exploiting $\sum_{i=1}^N w_t^i = 1$.

Resampling

After the sequential importance sampling and weight calculation at time t, a new set of samples with corresponding weights: $\{\mathbf{x}_t^i, w_t^i, i = 1, ..., N\}$ is obtained. Sometimes, the variance of the estimated weights will be high and this will lead to the degeneracy problem [48], which means that all but one particle will have negligible normalized weights and a large computational effort is devoted to updating particles whose contribution to the approximation of the MMSE estimator $E(\mathbf{X}_t|\mathbf{Z}_t)$ is almost zero. In this case, a resampling scheme is applied to eliminate samples with low importance weights and multiple samples with high importance weights.

As in [48], a measurement of degeneracy is introduced as:

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^N (w_t^i)^2} \tag{3.2.17}$$

where $1 \leq \hat{N}_{eff} \leq N$; if the weights are uniform then $\hat{N}_{eff} = N$ and if all but one particle have negligible normalized weights (assuming zeros) then $\hat{N}_{eff} = 1$. So, small $\hat{N}_{eff}$ indicates a severe degeneracy. An approximate threshold is set and if $\hat{N}_{eff}$ falls below this threshold, resampling is applied.

Resampling generates a new set of samples $\{\mathbf{x}_t^i, i = 1, ..., N\}$ by sampling N times from an approximate representation of $p(\mathbf{x}_t|\mathbf{Z}_t)$ described as:

$$p(\mathbf{x}_t|\mathbf{Z}_t) \approx \sum_{i=1}^N w_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) \tag{3.2.18}$$

and each of the obtained sample is assigned to a new weight 1/N, the details of how to sample from the distribution of $p(\mathbf{x}_t|\mathbf{Z}_t)$ are shown in [47]. By using resampling, the degeneracy problem is avoided, each of the samples will have the same non-zero weight 1/N to guarantee the non-negligible contribution to the MMSE estimators for the current and ensuing time instances.

The Framework of the Sequential Importance Sampling Particle Filter

From the above discussions, the framework of a SIR particle filter is listed in Table 3.1, from which it can be seen that the new samples $\{\mathbf{x}_t^i|i = 1 : N\}$ and weights $\{w_t^i|i = 1 : N\}$ can be obtained from the previous samples $\{\mathbf{X}_{t-1}^i|i = 1 : N\}$ and weights $\{w_{t-1}^i|i = 1 : N\}$:

Table 3.1. The procedure of a SIR particle filter.

Given $\{\mathbf{X}_{t-1}^i|i = 1 : N\}$ and $\{w_{t-1}^i|i = 1 : N\}$: I. Obtain the new samples $\mathbf{x}_t^i$ from the proposal distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$ for every i II. Calculate the weights w_t^i from $w_t^i = w_{t-1}^i p(\mathbf{z}_t|\mathbf{x}_t^i)$ III. Normalize the w_t^i to make $\sum_{i=1}^N w_t^i = 1$ IV. If $\hat{N}_{eff}$ is less than a threshold, a resampling procedure is operated. Finally, the new samples $\{\mathbf{x}_t^i|i = 1 : N\}$ and new weights $\{w_t^i|i = 1 : N\}$ are obtained at time t.

The MMSE estimation of $\mathbf{x}_t$ (state vector at time t) can then be calculated by:

$$E(\mathbf{x}_t|\mathbf{Z}_t) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_t^i w_t^i \tag{3.2.19}$$

using the weights and samples obtained at time t as shown in Table 3.1.

3.2.3 Particle filter for head tracking

From Table 3.1, it can be seen that to estimate the new weights and samples, two types of distribution are needed, they are $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{z}_t|\mathbf{x}_t)$.

For the head tracking problem, the head is modeled as an ellipse for simplicity (as shown in [49], [50] and [51]) and the state vector $\mathbf{x}_t$ can be represented as a four-by-one vector $\mathbf{x}_t = [x_t, y_t, l_t, \theta_t]^T$, where (x_t, y_t) is the center of the ellipse representing a head at time t; l_t is the length of the minor semi-axis at time t and θ_t is the orientation angle of the ellipse at time t.

In this work, the relationship between $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ is modeled as a first-order random walk model with additive Gaussian noise [49], which has the following form:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{A} \cdot \mathbf{n}_t \tag{3.2.20}$$

where $\mathbf{A}$ is a four-by-four diagonal matrix and $\mathbf{n}_t$ is a four-by-one Gaussian noise vector, each component of $\mathbf{n}_t$ has zero mean and unity variance. So, the distribution of $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is in a Gaussian form, with mean $\mathbf{x}_{t-1}$ and its covariance matrix is determined by $\mathbf{A}$.

For calculating $p(\mathbf{z}_t|\mathbf{x}_t)$, two types of measurements are needed, they are the gradient measurement and colour measurement. And $p(\mathbf{z}_t|\mathbf{x}_t)$ can be represented as:

$$p(\mathbf{z}_t|\mathbf{x}_t) = p(\mathbf{z}_{t,gradient}|\mathbf{x}_t)p(\mathbf{z}_{t,colour}|\mathbf{x}_t) \tag{3.2.21}$$

where $p(\mathbf{z}_{t,gradient}|\mathbf{x}_t)$ and $p(\mathbf{z}_{t,colour}|\mathbf{x}_t)$ are the probabilities obtained from the gradient and colour measurements.

Gradient Measurement

For a particular $\mathbf{x}_t^i$, the summation of the gradient magnitude along the contour of the ellipse that $\mathbf{x}_t^i$ corresponds to can be estimated. A higher summation value means that $\mathbf{x}_t^i$ is more likely to represent the head region, considering that the contrast between the head's edge and background is always large (as shown in 3.1 (b)).

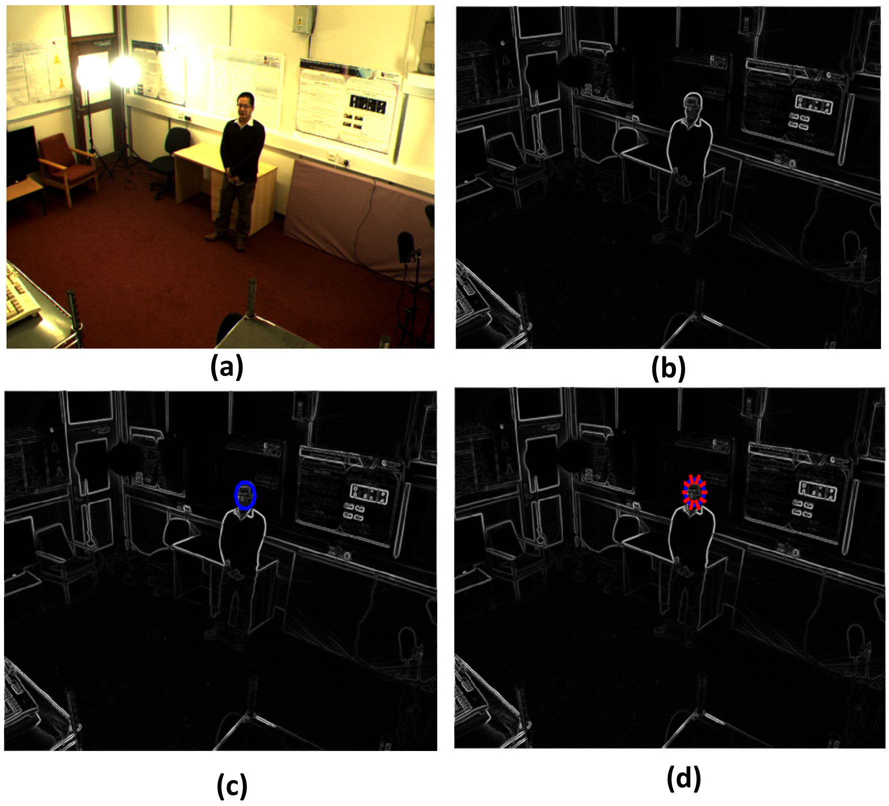


Figure 3.1. Illustration of the gradient magnitude. (a) is the original image; (b) shows the gradient magnitude image, the brighter pixels have larger gradient magnitude values; (c) shows that the head is modeled by a blue ellipse and (d) shows the normal lines (in red) along the ellipse for more accurate estimation of the summation of the gradient magnitude along the contour of the ellipse.

The normalized summation of the gradient magnitude around the ellipse boundary is calculated in the following way [51]:

$$\psi_g(\mathbf{x}_t^i) = \frac{1}{N} \sum_{i=1}^{N} g(x_i, y_i) \tag{3.2.22}$$

where $g(x_i, y_i)$ is the intensity gradient magnitude of pixel (x_i, y_i) located at the boundary of the ellipse and N is the number of pixels on the perimeter of the ellipse.

A simple operator is used to compute the gradient in the x-direction and y-direction gradients for pixel (x_n, y_n):

$$\begin{aligned} g_x(x_n, y_n) &= I(x_n - 2, y_n) + 2I(x_n - 1, y_n) \\ &\quad -2I(x_n + 1, y_n) - I(x_n + 1, y_n) \\ g_y(x_n, y_n) &= I(x_n, y_n - 2) + 2I(x_n, y_n - 1) \\ &\quad -2I(x_n, y_n + 1) - I(x_n, y_n + 1) \end{aligned} \tag{3.2.23}$$

Finally, $g(x_n, y_n)$ can be calculated as

$$g(x_n, y_n) = \sqrt{g_x^2(x_n, y_n) + g_y^2(x_n, y_n)} \tag{3.2.24}$$

However, it should be noticed that an ellipse does not accurately model the contour of the head. Therefore, given the inaccurate modeling, a more robust way to calculate the intensity gradient magnitude is proposed as:

$$g(x_i, y_i) = \max_{(x_n, y_n) \in L_n} \{g(x_n, y_n)\}, \quad \text{where} \quad \sqrt{(x_n - x_i)^2 + (y_n - y_i)^2} \leq c \tag{3.2.25}$$

where L_n represents the normal line along the point (x_i, y_i) (as shown in Figure 3.1 (d)), c is a small constant value and the condition $\sqrt{(x_n - x_i)^2 + (y_n - y_i)^2} \leq c$ implies (x_n, y_n) is in the nearby region of (x_i, y_i).

The normalized summation of the gradient magnitude $\psi_g(\mathbf{x}_t^i)$ corresponding to the sample $\mathbf{x}_t^i$ is taken as the sample's gradient measurement $\mathbf{z}_{t,gradient}$ and according to [49], $\frac{1}{\psi_g(\mathbf{x}_t^i)}$ is substituted into a Gaussian distribution function to estimate $p(\mathbf{z}_{t,gradient}|\mathbf{x}_t^i)$ as:

$$\begin{aligned} p(\mathbf{z}_{t,gradient}|\mathbf{x}_t^i) &= p(\psi_g(\mathbf{x}_t^i)|\mathbf{x}_t^i) \\ &= \frac{1}{\sqrt{2\pi}\sigma_g} e^{-\frac{\left(\frac{1}{\psi_g(\mathbf{x}_t^i)}\right)^2}{2\sigma_g^2}} \end{aligned} \tag{3.2.26}$$

A large $\psi_g(\mathbf{x}_t^i)$ value means a high $p(\mathbf{z}_{t,gradient}|\mathbf{x}_t^i)$, which indicates a high probability that the sample $\mathbf{x}_t^i$ corresponds to the head in terms of the gradient aspect.

Colour Measurement

For calculating the colour measurement $p(\mathbf{z}_{t,colour}|\mathbf{x}_t)$, a histogram-based colour model is adopted. Different from the model introduced in [52], whose colour distribution is represented by a colour histogram calculated in the RGB space, a normalized r-g-b colour space is applied to avoid the change of illumination [53]. The colour model is constructed from some reference image patches (in this work, the captured patches are the head region captured from different views, as shown in Figure 3.2).

For a particular image patch R centered at $\mathbf{o}$, its colour histogram, denoted by p, is calculated as [52]:

$$p^{(u)} = f \sum_{x_i \in R} k(\frac{\|\mathbf{o} - \mathbf{x}_i\|}{a}) \delta[h(\mathbf{x}_i) - u] \tag{3.2.27}$$

where $\mathbf{x}_i$ is the position of the i-th pixel in R, u is the bin index, δ is the Kronecker delta function and $h(\mathbf{x}_i)$ gives the index of the bin to which the colour of the pixel at $\mathbf{x}_i$ belongs, f is the normalization factor, $\mathbf{o}$ is the center of the image patch, a is the size of the image patch R and $k(\cdot)$ is defined by:

$$k(r) = \begin{cases} 1 - r^2 & r < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2.28}$$

which gives higher weights to pixels closer to the region center $\mathbf{o}$ (r represents the variable of the function $k(\cdot)$). Figure (3.2) shows the colour histogram of the head images with different views.

Assuming N reference patches are available (in this work, the head's image is captured from eight different directions and N is equal to eight), for

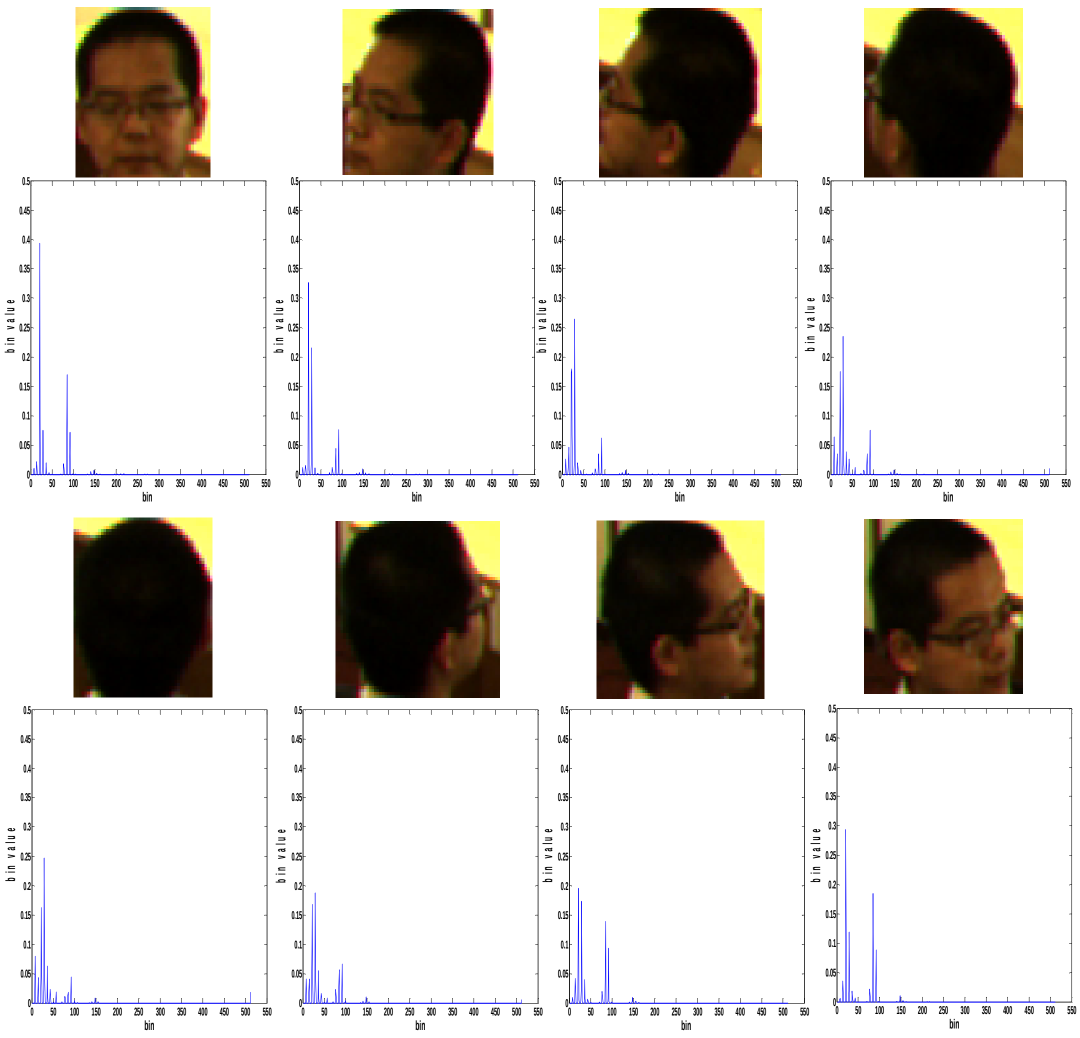


Figure 3.2. The captured images of heads from different views and their corresponding histograms.

each reference patch the colour histogram is obtained according to equation (3.2.27) and the colour histogram set $\{p_j, j = 1, ..., N\}$ is obtained. For a particular sample $\mathbf{x}_t^i$, the colour histogram corresponding to the region that sample represents is also calculated, and denoted as $q_{\mathbf{x}_t^i}$. The largest Bhattacharyya coefficient (which is used to measure the similarity between two histograms) between $q_{\mathbf{x}_t^i}$ and the colour histogram set is calculated as:

$$\rho = max_j \{\sum_{u=1}^{m} \sqrt{p_j^{(u)} q_{\mathbf{x}_t^i}^{(u)}}\} \tag{3.2.29}$$

where u is the bin index of a colour histogram and j is from 1 to N.

The calculated Bhattacharyya coefficient ρ is taken as the colour measurement $\mathbf{z}_{t,colour}$ for the sample $\mathbf{x}_t^i$ and similar to the case of the gradient model, the $p(\mathbf{z}_{t,colour}|\mathbf{x}_t^i)$ is also assumed to be a Gaussian form by substituting $(1-\rho)$ into a Gaussian distribution function to obtain:

$$\begin{aligned} p(\mathbf{z}_{t,colour}|\mathbf{x}_t^i) &= p(\rho|\mathbf{x}_t^i) \\ &= \frac{1}{\sqrt{2\pi}\sigma_c} e^{-\frac{(1-\rho)^2}{2\sigma_c^2}} \end{aligned} \tag{3.2.30}$$

From this equation, it can be seen that a large value ρ means a high $p(\mathbf{z}_{t,colour}|\mathbf{x}_t^i)$, which indicates a higher probability that the sample $\mathbf{x}_t^i$ corresponds to the head in terms of the colour aspect.

Importantly, the reference patches need to be updated to cope with the changes in the head region image. An update criterion with a properly chosen *threshold* value is:

$$p(\mathbf{z}_{t,colour}|\hat{\mathbf{x}}_t) > threshold \tag{3.2.31}$$

where $\hat{\mathbf{x}}_t$ is the MMSE estimate of $\mathbf{x}_t$ and this criterion can prevent the model updating when the object is out of track.

If the criterion is met, the colour histogram of the reference image patch whose Bhattacharyya coefficient with $q_{\hat{\mathbf{x}}_t}$ (which represents the colour histogram for the region represented by the MMSE estimator $\hat{\mathbf{x}}_t$) is largest, is updated as: $p_t^{imax} = (1-\alpha)q_{\hat{\mathbf{x}}_t} + \alpha p_{t-1}^{imax}$, where $imax$ is the index of the reference patch whose Bhattacharyya coefficient with $q_{\hat{\mathbf{x}}_t}$ is the largest.

After obtaining $p(\mathbf{z}_{t,gradient}|\mathbf{x}_t^i)$ and $p(\mathbf{z}_{t,colour}|\mathbf{x}_t^i)$, then $p(\mathbf{z}_t|\mathbf{x}_t^i)$ can be obtained according to equation (3.2.21), together with new samples obtained from equation (3.2.20), Table 3.1 is thereby finalized for the head tracking problem and equation (3.2.19) is then used to estimate the MMSE estimation of $\mathbf{x}_t$, including the positions, axis length and orientation angle of the ellipse model representing a head.

The head velocity can then be obtained from the head tracking results, assuming for the frame instances $t-1$ and t, the head tracking results $\hat{\mathbf{x}}_t = [\hat{x}_t, \hat{y}_t, \hat{l}_t, \hat{\theta}_t]^T$ and $\hat{\mathbf{x}}_{t-1} = [\hat{x}_{t-1}, \hat{y}_{t-1}, \hat{l}_{t-1}, \hat{\theta}_{t-1}]^T$ are obtained and the head velocity v_t can then be estimated as:

$$v_t = \sqrt{(\hat{x}_t - \hat{x}_{t-1})^2 + (\hat{y}_t - \hat{y}_{t-1})^2} \tag{3.2.32}$$

Normally, the head velocity of an elderly person is low (considering the movement of an elderly person is usually slow) and a large value of v_t is usually obtained for some abnormal activities such as falling, in this way v_t can be taken as a cue to detect falls and a large v_t indicates a possible fall activity.

3.3 Ellipse Fitting for Shape Change Analysis

Although the head velocity discussed in Section 3.2 can be used to distinguish fall and non-fall activities; using only the head velocity will cause many false detections (such as fast walking or sudden sitting will be mistaken as a fall). In order to solve this problem, shape change is analyzed by ellipse fitting to provide other cues for a more accurate fall detection system.

3.3.1 Codebook Background Subtraction

The first step for ellipse fitting is to extract the object from the background. In visual surveillance, a common approach for discriminating moving objects from the background is detection by background subtraction. Currently, there are many background subtraction algorithms, which include the single-mode model background subtraction method [54] and [55], the mixture of Gaussians (MoG) background subtraction method [56], the non-parametric density estimation based method [57] and the codebook background subtraction method [58]. Here, the codebook method is applied because of its advantages. There is no parametric assumption on the codebook model and it shows the following merits as proposed in [58]: (1) resistance to artifacts of acquisition, digitization and compression, (2) capability of coping with illumination changes, (3) adaptive and compressed background models that can capture structural background motion over a long period of time under limited memory, (4) unconstrained training that allows moving foreground objects in the scene during the initial training period.

As for the single-mode model background subtraction method in [54] and [55] and the mixture of Gaussians (MoG) background subtraction method in [56], the codebook background subtraction algorithm is also a pixel-wise method, and each pixel is modelled by a number of codewords. One codeword **c** is composed of an RGB vector $\mathbf{v} = (\bar{R}, \bar{G}, \bar{B})$ and a 6-tuple $\mathbf{aux} = \langle \check{I}, \hat{I}, f, \lambda, p, q \rangle$. The meaning of the elements in the 6-tuple **aux** is shown in Table 3.2:

Table 3.2. The meaning of the elements in the tuple **aux**.

$\check{I}, \hat{I}$	the min and max brightness of all pixels assigned to this codeword
f	the frequency with which the codeword has occurred
λ	the maximum negative run-length (MNRL) defined as the longest interval during the training period that the codeword has not recurred
p,q	the first and last access times, respectively, that the codeword has occurred

The codebook background subtraction algorithm is divided into three main steps: model training, testing and model updating. For model training, the codewords used to model every single pixel will be obtained from a training sequence. The procedure for constructing the codewords for a particular pixel is shown in Table 3.3.

Initially, the codewords set for a pixel is set to be empty so that the

Table 3.3. The training procedure for constructing the codewords for a pixel.

```
I.L$\leftarrow$0, $\Phi \leftarrow \phi$(empty set)

II.for t=1 to N do

(i) $\mathbf{x}_t$=(R,G,B),$I \leftarrow \sqrt{R^2+G^2+B^2}$

(ii) Find the codeword $\mathbf{c}_m$ in $\Phi$ (codewords class for a pixel) $= \{\mathbf{c}_i | 1 \leq \mathbf{c}_i \leq L\}$
matching to $\mathbf{x}_t$ based on:

(a) colourdist($\mathbf{x}_t$,$\mathbf{v}_m$)$\leq \varepsilon_t$

(b) brightness(I,$\langle \hat{I}_m, \check{I}_m \rangle$)=true

(iii) If $\Phi = \phi$ or there is no match,then L$\leftarrow$L+1.
Create a new codeword $\mathbf{c}_L$ by setting:

•$\mathbf{v}_L \leftarrow (R, G, B)$

•$\mathbf{aux}_L \leftarrow \langle I, I, 1, t-1, t, t \rangle$

(iv) Otherwise, update the matched codeword $\mathbf{c}_m$, consisting of
$\mathbf{v}_m = (\bar{R}_m, \bar{G}_m, \bar{B}_m)$ and
$\mathbf{aux}_m = \langle \check{I}_m, \hat{I}_m, f_m, \lambda_m, p_m, q_m \rangle$, by setting:

•$\mathbf{v}_m \leftarrow (\frac{f_m \bar{R}_m + R}{f_m+1}, \frac{f_m \bar{G}_m + G}{f_m+1}, \frac{f_m \bar{B}_m + B}{f_m+1})$

•$\mathbf{aux}_m \leftarrow \langle min\{I, \check{I}_m\}, max\{I, \hat{I}_m\}, f_m+1, max\{\lambda_m, t-q_m\}, p_m, t \rangle$.

end for

III. For each codeword $\mathbf{c}_i$, i=1,.......,L, wrap around $\lambda_i$ by setting
$\lambda_i \leftarrow max\{\lambda_i, (N - q_i + p_i - 1)\}$, removing the codewords whose
$\lambda s$ are larger than a particular threshold.
```

number is zero. Codewords are constructed and updated by matching the existing codewords with the incoming pixel in the training set. If matched, the matched codeword will be updated and a new codeword will be constructed if there is no match. Finally, the codeword set is refined by deleting the codewords which don't recur for a certain interval (measured by the

MNRL value λ) to form a more compact codebook model. For a particular codeword $\mathbf{c}$, it is said to match the incoming pixel $\mathbf{x}_t$ if the following two conditions are met:

$$\begin{aligned} colordist(\mathbf{x}_t, \mathbf{c}) &\le \varepsilon \\ brightness(I, \langle \hat{I}, \check{I} \rangle) &= true \end{aligned} \tag{3.3.1}$$

where ε is a preset threshold value for comparison, I represents the norm of $\mathbf{x}_t$, $\hat{I}$ and $\check{I}$ are the first two parameters of the 6-tuple **aux** vector of the codeword $\mathbf{c}$.

The $colordist(\mathbf{x}_t, \mathbf{c})$ measures the chromatic difference between two colour vectors, which can be calculated as:

$$colordist(\mathbf{x}_t, \mathbf{c}) = \sqrt{\| \mathbf{x}_t \|^2 - \frac{\mathbf{x}_t \cdot \mathbf{v}}{\| \mathbf{v} \|^2}} \tag{3.3.2}$$

where $\mathbf{v}$ represents the RGB vector $\mathbf{v} = (\overline{R}, \overline{G}, \overline{B})$ of codeword $\mathbf{c}$.

The $brightness(I, \langle \hat{I}, \check{I} \rangle)$ is defined as:

$$brightness(I, \langle \hat{I}, \check{I} \rangle) = \begin{cases} true & if \;\; I_{low} \le I \le I_{hi} \\ false & \text{otherwise} \end{cases} \tag{3.3.3}$$

where $I_{low} = \alpha \hat{I}$ and $I_{hi} = min\{\beta \hat{I}, \frac{\check{I}}{\alpha}\}$. In the experimental studies, α and β are fixed to be 0.5 and 2 respectively for background subtraction.

The codebook model training procedure is applied for every pixel and codewords for every pixel are constructed, the trained codebook models are then used for background subtraction, the procedure is shown in Table 3.4. A background subtraction result is shown in Figure 3.3, the black and white object extraction result is presented in Figure 3.3 (c).

Sometimes, the background will change after the training process (due to

Table 3.4. The codebook background subtraction procedure.

```
Step I. For each pixel $\mathbf{x}_t$=(R,G,B) (assuming the time instance of the frame is t),
calculate the intensity from the (R,G,B) value of a colour image by
I$\leftarrow \sqrt{R^2+G^2+B^2}$

Step II. Find the first codeword $\mathbf{c}_m$ from the corresponding codebook
matching to $\mathbf{x}_t$ based on two conditions:

1. colourdist($\mathbf{x}_t$,$\mathbf{c}_m$)$\leq \varepsilon_2$

2. brightness(I,$\langle \hat{I}_m, \check{I}_m \rangle$)=true

Update the matched codeword

Step III. If there is no match, then the pixel $\mathbf{x}_t$ is categorized as foreground;
otherwise, it is regarded as a background pixel.
```

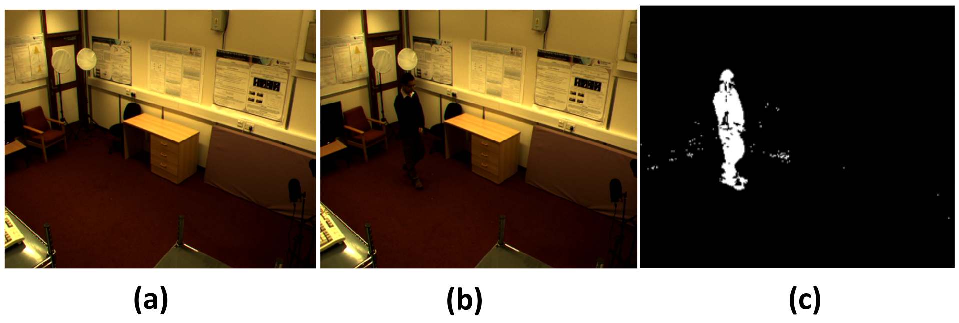


Figure 3.3. Example background subtraction result. (a) the original background (b) the person enters into the scene (c) background subtraction result

the movement of furniture, etc.) and therefore the corresponding codebook model for every pixel should be updated. For the model updating, an additional model $\hbar'$ called a cache and three parameters— $T_{\hbar'}$, T_{add} and T_{delete} are defined. The updating procedure is then described as in Table 3.5.

The codebook model is updated so any changes in the background will be taken as the new background after certain iterations of the above steps. One example is shown in Figure 3.4, from which it can be seen that the moved chair and the generated ghost region (where the chair was previously) is absorbed into the background by using the updating procedure as shown in

Table 3.5. The updating procedure for the codebook model.

```
Step I. After training, the background model $\hbar$ for a pixel is obtained.
Create an empty model $\hbar'$ as a cache.

Step II. For an incoming pixel value $\mathbf{x}_t$, find a matching codeword in $\hbar$.
If found, update the codeword.

Step III. Otherwise, try to find a matching codeword in $\hbar'$ and update it. For no
matching, a new codeword $\mathbf{h}$ is created and added to $\hbar'$.

Step IV. Filter out the cache codewords based on $T_{\hbar'}$:

$\hbar' \leftarrow \hbar' - \{\mathbf{h}_i | \mathbf{h}_i \in \hbar', \lambda \ of \ \mathbf{h}_i \ is \ longer \ than \ T_{\hbar'}\}$

Step V. Move to the cache codewords staying for enough time, to $\hbar$:

$\hbar \leftarrow \hbar \cup \{\mathbf{h}_i | \mathbf{h}_i \in \hbar', \mathbf{h}_i \ stays \ longer \ than \ T_{add}\}$

VI. Delete the codewords not accessed for a long time from $\hbar$:

$\hbar \leftarrow \hbar - \{\mathbf{c}_i | \mathbf{c}_i \in \hbar, \mathbf{c}_i \ not \ accessed \ for \ T_{delete}\}$

VII. Repeat the process from the Step II.
```

Table 3.5.

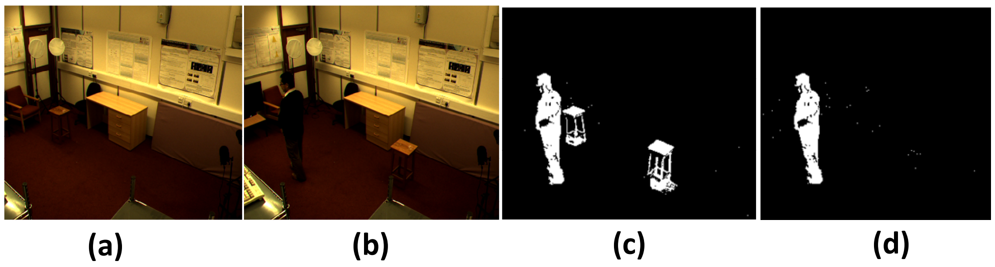


Figure 3.4. The background subtraction with background model updating. (a) the original background image (b) the person moves the chair to a different place (c) the background subtraction result without background model updating (d) the background subtraction result with background model updating.

Postprocessing

Although background subtraction can obtain the moving object, however, the obtained result is not always perfect and it can contain noise (as shown in Figure 3.5 (a)). In order to obtain a better background subtraction result, some postprocessing technique can be applied. Here two types of post-processing technique are applied to improve the final results:

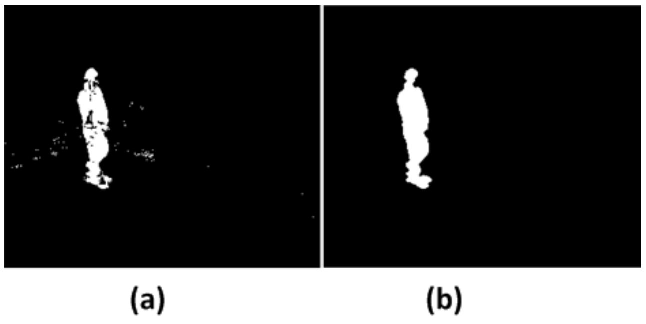


Figure 3.5. The post-processing for removing noises and filling holes. (a) original background subtraction result (b) background subtraction result with post-processing.

I. Blob operation: Different groups of foreground pixels which are 4-connected or 8-connected (shown in in Figure 3.6) form different blobs. A threshold is set and the blobs whose pixel numbers are less than the set threshold are removed and the pixels in these blobs are taken as background pixels. In this way, most of the noise (salt-and-pepper noise and small noisy blobs) can be removed.

II. Filling of small holes: Small holes can be filled by a 'CLOSE' operation, which is composed of erosion and dilation. As mentioned in [59], erosion and dilation are two morphological techniques. Morphological techniques typically probe an image with a small shape called a structuring element. Two structuring element examples are presented in Figure 3.7 with the square-shape and cross-shape respectively.

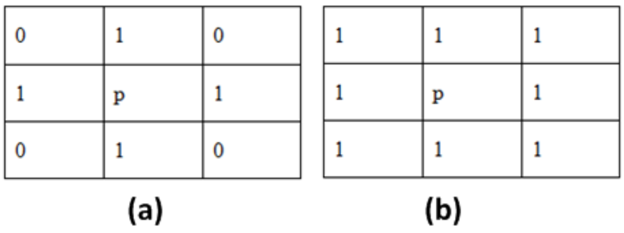


Figure 3.6. 4-connection and 8-connection for a pixel p. (a) the 4-connection (b) the 8-connection. The pixels which are connected to p are marked 1.

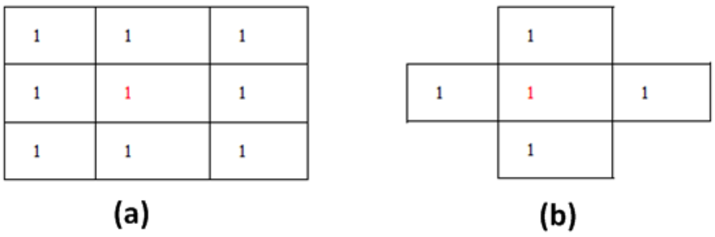


Figure 3.7. Two structure elements. (a) square structure element (b) cross structure element. The center of the structure element is marked by red 1 and neighboring pixels are marked by black 1s.)

The structure element is positioned at all possible locations in the image and compared to the neighboring pixels (the pixels covered by the black '1s' are shown in Figure 3.7). The morphological operations differ in how they carry out this comparison. For the binary-image erosion operation, the center pixel of the structure element will be set to 0 (black) if any of its neighboring pixels is zero. For the binary-image dilation operation, the center pixel of the structure element will be set to 1 (white) if any of its neighboring pixels is 1. For the image processing, the size of the structure element needs to be chosen properly to obtain good postprocessing results and in this work, the square element is applied with the size 3 pixels for erosion operation and 5 pixels for dilation operation.

3.3.2 Moments Based Ellipse Fitting

After applying the post-processing technique to obtain a better object extraction result, ellipse fitting is applied to find the ellipse which best fits the object. A moments based technique for ellipse fitting is presented.

For the obtained binary image $f(x, y)$, the moments can be calculated by:

$$m_{pq} = \sum_{(x,y)\in Pixels} x^p y^q f(x, y) \tag{3.3.4}$$

where p, q = 0, 1, 2.

The centroid $(\bar{x}, \bar{y})$ of the fitted ellipse can then be obtained by: $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$.

After obtaining the centroid, central moments can be calculated in the following way:

$$u_{pq} = \sum_{(x,y)\in Pixels} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \tag{3.3.5}$$

With the aid of central moments, the orientation of the ellipse can be calculated as:

$$\theta = \frac{1}{2} \arctan(\frac{2u_{11}}{u_{20} - u_{02}}) \tag{3.3.6}$$

The major semi-axis a and the minor semi-axis b of the fitted ellipse are calculated in the following formula:

$$\begin{aligned} a &= (4/\pi)^{1/4} \left[\frac{(I_{max})^3}{I_{min}} \right]^{1/8} \\ b &= (4/\pi)^{1/4} \left[\frac{(I_{min})^3}{I_{max}} \right]^{1/8} \end{aligned} \tag{3.3.7}$$

where I_{max} and I_{min} are the larger and smaller eigenvalues of J, where

$$J = \begin{pmatrix} u_{20} & u_{11} \\ u_{11} & u_{02} \end{pmatrix} \tag{3.3.8}$$

An ellipse can then be fitted when the necessary parameters (center, orientation angle, major and minor axes) are calculated. Figure 3.8 shows the comparison of the moments-based ellipse fitting with the simple minimum enclosing rectangle (MER) fitting for a stretching person, from which it can be seen that the moments-based ellipse fitting method obtains a better result with less affect when the arms of the human are stretched.

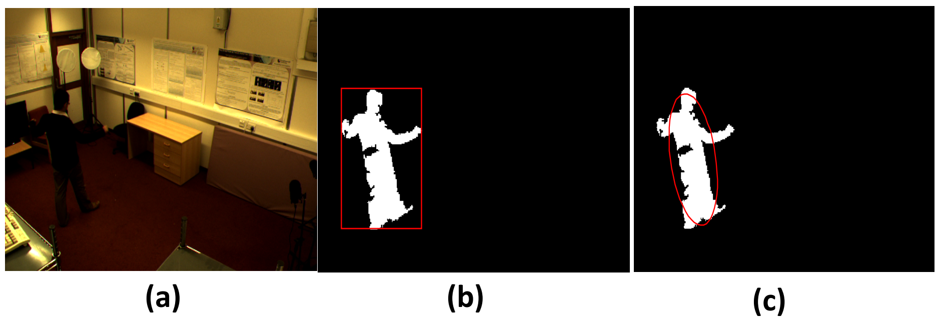


Figure 3.8. The comparison of the MER fitting result and moment-based ellipse fitting result. (a) original image with a person stretching the arms (b) MER fitting result (c) moment-based ellipse fitting result.

The fitted ellipse approximately reflects the shape of a human body; the changes of the ellipse's orientation angle and ratio between two axes during a time period can be used to reflect the shape change and a large shape change indicates a possible fall.

3.3.3 Movement Amplitude Detection

The last cue used for detecting falls is the movement amplitude of a person. It is noticed that after a person falls, a person will be always static (considering for the general cases, elderly persons fall due to loss of consciousness and usually lie on the floor for a certain time) and the movement amplitude is low. So, whether the movement amplitude of a person is low for a certain time or not can be used for the confirmation of a fall activity. To estimate the movement amplitude, the concept of motion history image (MHI) is proposed. MHI is first introduced by Bobick and Davis [60], which is an image where the pixel intensity represents the recency of motion in an image sequence. The motion history image shows the tendency of a person's movement by the magnitudes of its pixels so that it is commonly used for activity recognition. The definition of the MHI is given in equation (3.3.9) and one example of MHI is given in Figure 3.9.

$$MHI_\tau(x,y,t) = \begin{cases} \tau & \text{if D(x,y,t)=1} \\ max(0, MHI_\tau(x,y,t-1)-1) & \text{otherwise} \end{cases} \tag{3.3.9}$$

where τ is a parameter called duration time defined in [60] and $D(x,y,t)$ represents the pixel value at the position (x,y) of a binary image D at time t, which is '1' if the absolute value of the difference between consecutive frames $I(x,y,t)$ and $I(x,y,t-1)$ is larger than a threshold. The resulting $MHI_\tau(x,y,t)$ is a scalar-valued image in which more recently moving pixels are brighter.

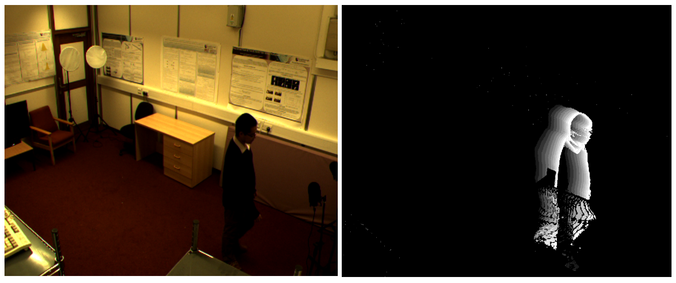


Figure 3.9. Demonstration of the MHI image. (a) a person walks in the room (b) the corresponding MHI of a person's walking activity.

After the MHI is obtained, whether the movement is large or small can be determined by calculating a parameter C_{motion} defined by [24]. The formula to calculate C_{motion} is given as:

$$C_{motion} = \frac{\sum_{pixel(x,y)\in blob} MHI_{\tau}(x,y,t)}{255 * \sharp pixels \in blob} \tag{3.3.10}$$

where $blob$ represents the region of the person extracted using the code-book background subtraction and $\sharp pixels \in blob$ means the number of pixels in the extracted person's region. The value of C_{motion} is between 0 and 1, and a large value of C_{motion} means a large movement amplitude and vice verse.

All the obtained three cues–head velocity, shape change and movement amplitude are combined to construct a robust fall detection system, which is shown in the experimental results.

3.4 Experimental Evaluation

The experiments were undertaken in a simulated home environment in the Intelligent lab, Loughborough University, as shown in Figure 3.10. A Basler Af312c colour camera together with the Streampix 5 software were used to record the video sequences with a frame rate of 8 frames/s. And the recorded sequence are then processed by VC++ 6.0 (with OpenCV 1.0) and MATLAB R2010.

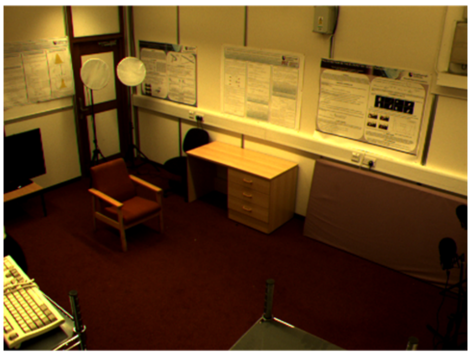


Figure 3.10. The experimental mock room environment.

3.4.1 Head Tracking Results

Figure 3.11 shows the comparison of head tracking results using gradient measurement, colour measurement and the combined measurement, which shows that by using only the gradient and colour measurement, the head is out of track after a certain number of frames and accurate tracking results can be obtained by combining the gradient measurement and colour measurement.

A fall activity is simulated and the corresponding head velocity curve is presented in Figure 3.12, it can be observed that for a fall activity, high head velocity is obtained. Initially, when a person walks slowly, the head velocity is low (less than 15 pixels/frame); when the person falls around the 50th frame, the head velocity suddenly increases to be above 35 pixels/frame. After the person falls (from around the 65th frame), the head velocity is low again because the person lies on the floor with little movement after falling.

3.4.2 Ellipse Fitting Results

Figure 3.13 presents the ellipse fitting results for four activities (standing, sitting, bending and lying) and Figure 3.14 and 3.15 show the orientation angle (denoted as *theta*) and ratio between two axes (denoted as *rho*) velocities for two fall activities in two different directions (one is parallel to the optical axis of the camera and the other is perpendicular to it). From Figure 3.14 and 3.15 it can be seen that large shape change occurs when a person falls; either the orientation angle or the ratio between two axes changes dramatically (at around the 55th frame in Figure 3.14 and the 45th frame in Figure 3.15 when a person falls).

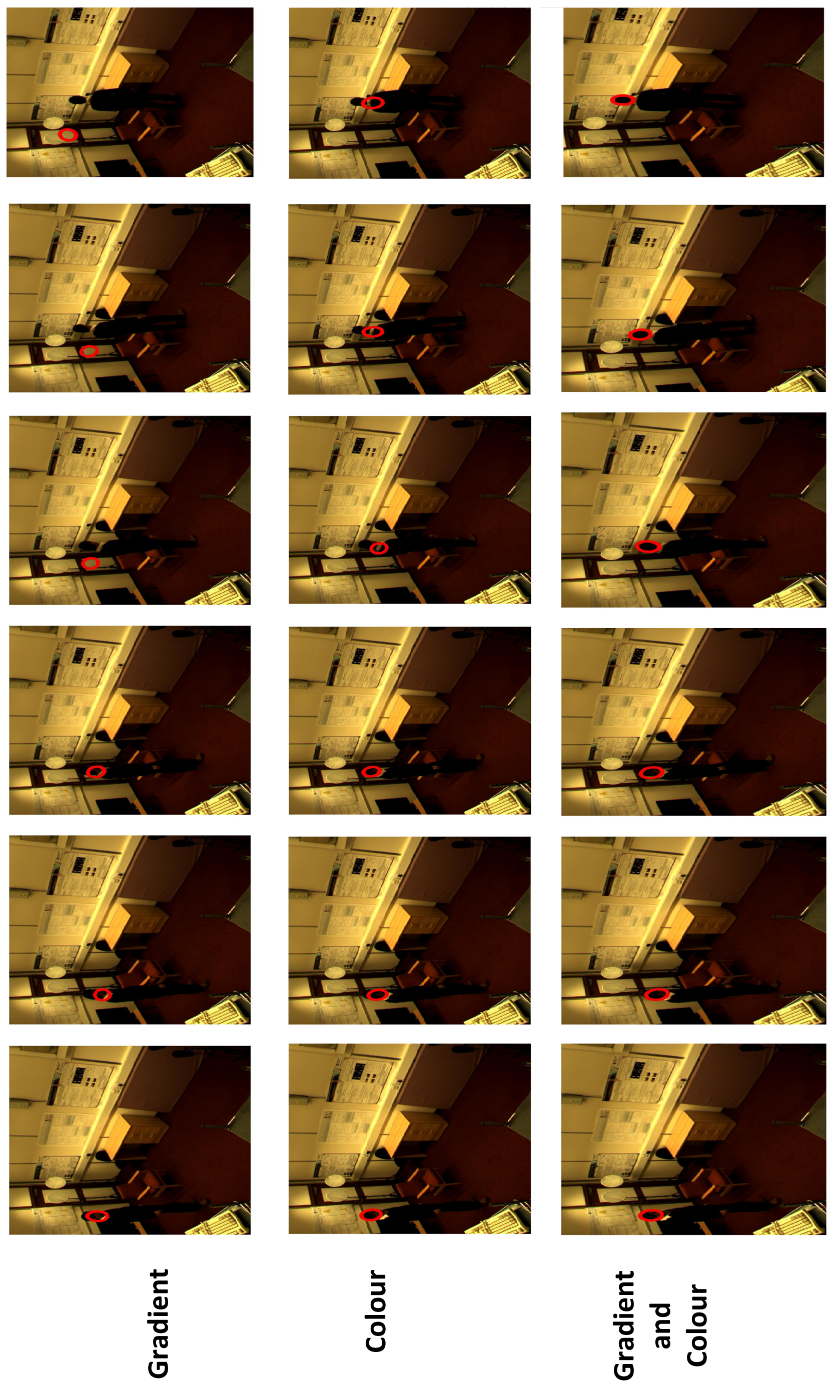


Figure 3.11. The comparison results of head tracking for selected frames. The first line shows the tracking results using only the gradient measurement, the second line shows the tracking results using only the colour measurement and the third line shows the results by using the combined gradient and colour measurement. Accurate results can be obtained by combining the gradient and colour measurement together.

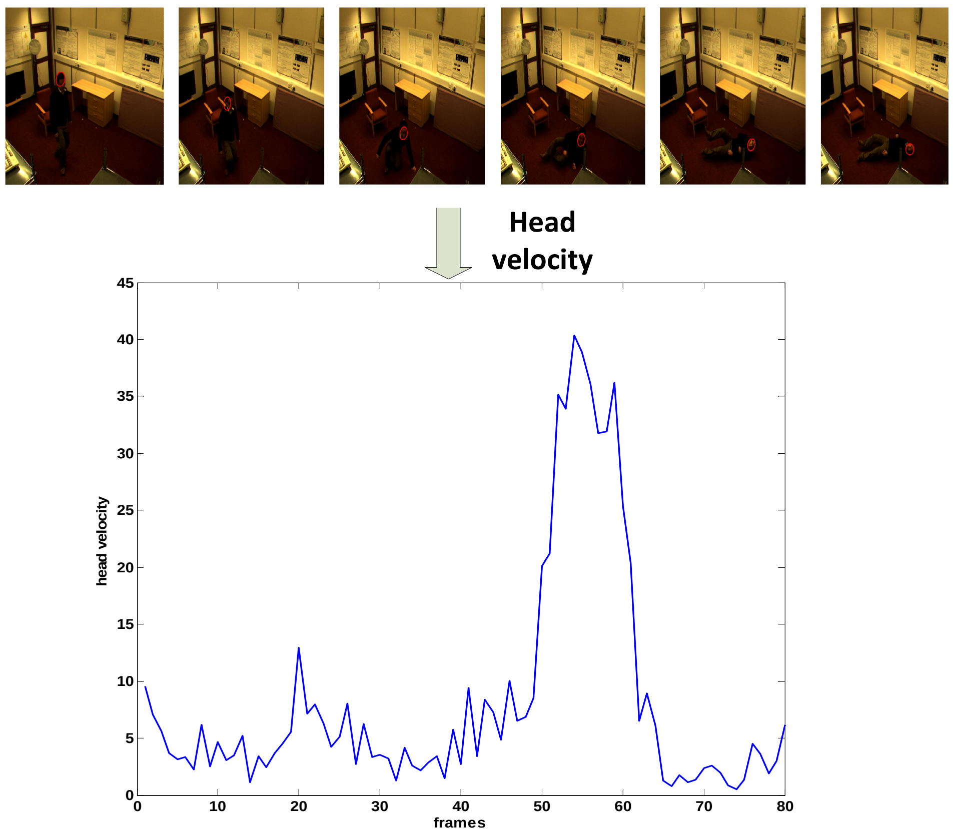


Figure 3.12. The head velocity for a fall activity, which is calculated as the difference between the head's center positions for two consecutive frames. The x-axis indicates the frame number and the y-axis indicates the head velocity (with the unit pixels/frame). The frames are captured at a frame rate of 8 frames/second.

3.4.3 Movement Amplitude Detection Results

Figure 3.16 shows the MHI results and C_{motion} values for four different activities and a large C_{motion} value is obtained for a fall activity. Figure 3.17 and 3.18 show the C_{motion} curves for two falls in two different directions. It can be observed that although the variation patterns of C_{motion} are different for falling in different directions, there is one thing in common after a fall happens, the C_{motion} values drops to be very low for a certain number of frames (the person lies almost statically on the floor).

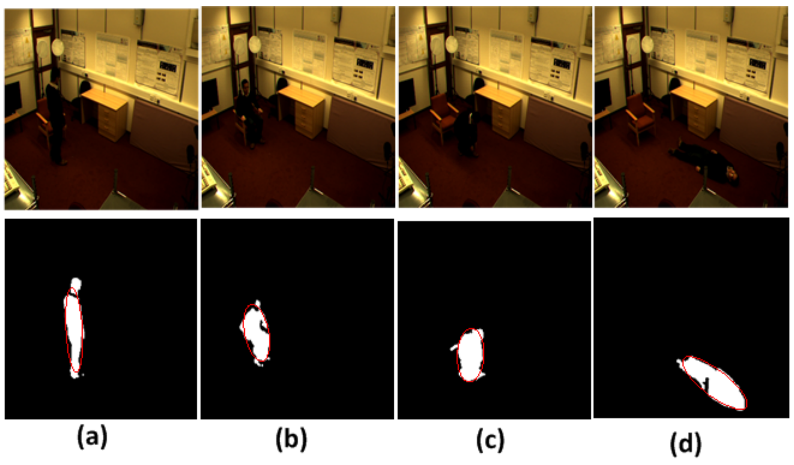


Figure 3.13. The ellipse fitting for for four activities ((a) standing, (b) sitting, (c) bending and (d) lying).

3.4.4 Fall Detection Scheme by fusing three cues

From the above analysis, it can be concluded that for a fall activity:

1. The head velocity is high.

2. The shape change (either the *theta* or *rho* velocity) is large.

3. The movement amplitude (measured by C_{motion}) is low for a certain number of frames after a fall happens.

If ALL of the three conditions are met, then a fall is determined; otherwise, no falls are reported. Different thresholds, typically found empirically, are set to head velocity, theta velocity, rho velocity movement amplitude velocity and the time period for low C_{motion}, which are denoted as: th_{head}, th_{theta}, th_{rho}, $th_{movement}$ and th_{period}. And the flowchart of the proposed fall detection system is presented as Figure 3.19. The head is tracked by particle filter and an ellipse is fitted to the human body according to the codebook background subtraction result. If either the head velocity or the shape change value is less than the corresponding threshold, then no falls are reported; otherwise, a further confirmation is made by the C_{motion} value

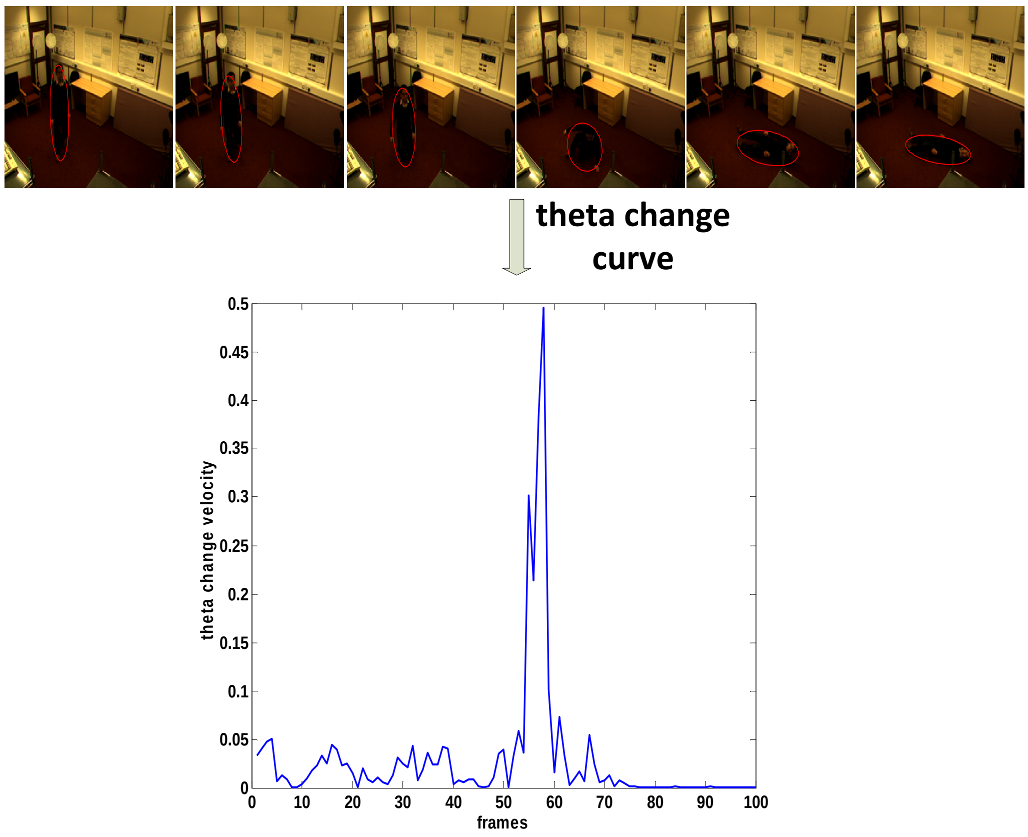


Figure 3.14. The velocity of *theta* for a fall activity which is perpendicular to the optical axis of the camera. The x-axis indicates the frame number and the y-axis indicates the *theta* change velocity (with the unit radians/frame).

and if C_{motion} is larger than a threshold for a certain time, then a fall is confirmed.

To validate the proposed system, six persons were invited to participate in the experiment. Each person simulated a series of common daily activities (including walking, walking behind furniture, sitting/standing and bending to tie the shoe belts) and two fall activities (in directions which are parallel and perpendicular to the camera's optical axis). In total, 12 fall activities and 24 non-fall activities (with each activity lasts about 30 seconds) were recorded. The system shown in Figure 3.19 is tested with different threshold values and the result with the optimal threshold is shown in Table 3.6, in which it can be seen that fall and non-fall activities can be successfully

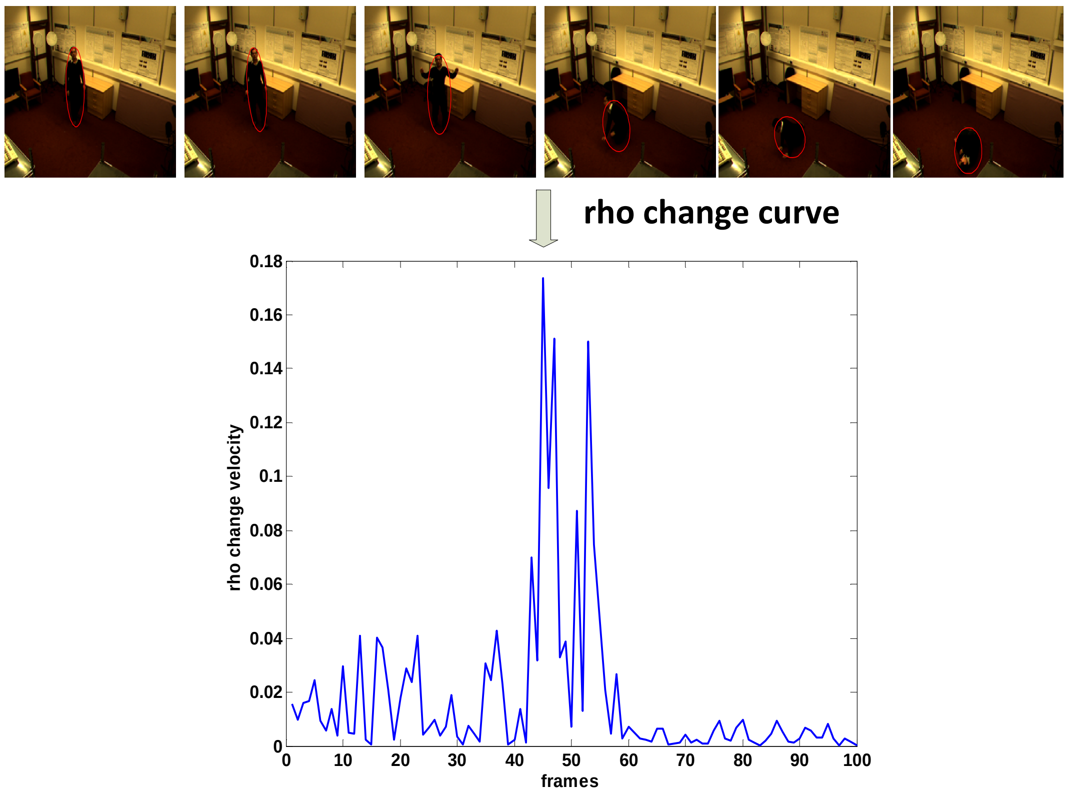


Figure 3.15. The velocity of rho for a fall activity which is parallel to the optical axis of the camera.

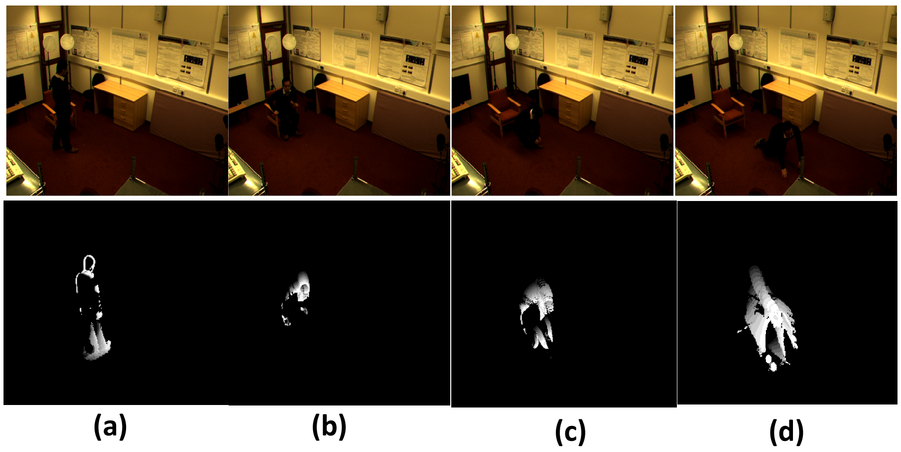


Figure 3.16. MHI for four different activities. (a) Walking, $C_{motion} = 0.2147$ (b) Sitting, $C_{motion} = 0.1286$ (c) Bending, $C_{motion} = 0.2252$ (d) Falling, $C_{motion} = 0.5227$

distinguished when proper thresholds are set.

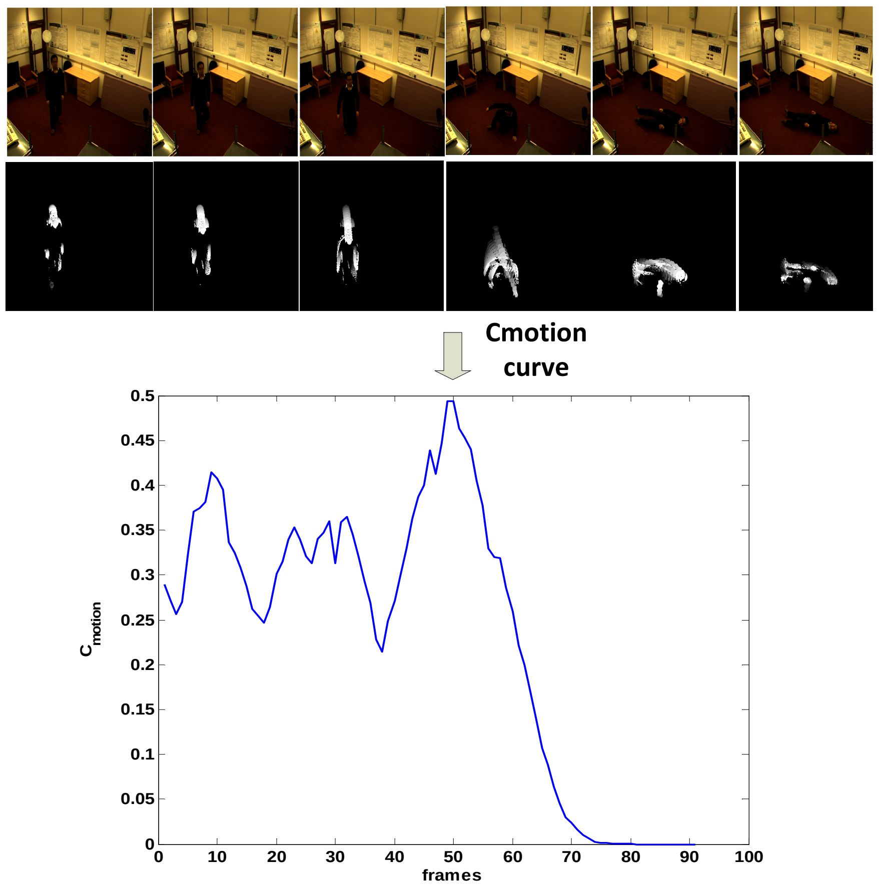


Figure 3.17. C_{motion} curve for a fall activity which is perpendicular to the optical axis of the camera. The x-axis indicates the frame number and the y-axis indicates the change of C_{motion} value per frame.

Table 3.6. Detection results of the proposed fall detection system with the optimal threshold set (chosen by trial-and-error from different threshold values): $th_{head} = 35$, $th_{theta} = 0.3$, $th_{rho} = 0.15$, $th_{movement} = 0.05$ and $th_{period} = 200$ (in frames).

	No. of Activities	Detected as falls	Detected as non-falls
Falling	12	12	0
Walking	6	0	6
Walking behind furniture	6	0	6
Sitting/Standing	6	0	6
Bending	6	0	6

A particular example is given to show how the proposed fall detection system operates; Figure 3.20 shows the variations of head velocity, theta velocity, rho velocity and C_{motion} for a sequence containing different types of activities. The threshold values are chosen as the optimal ones shown in

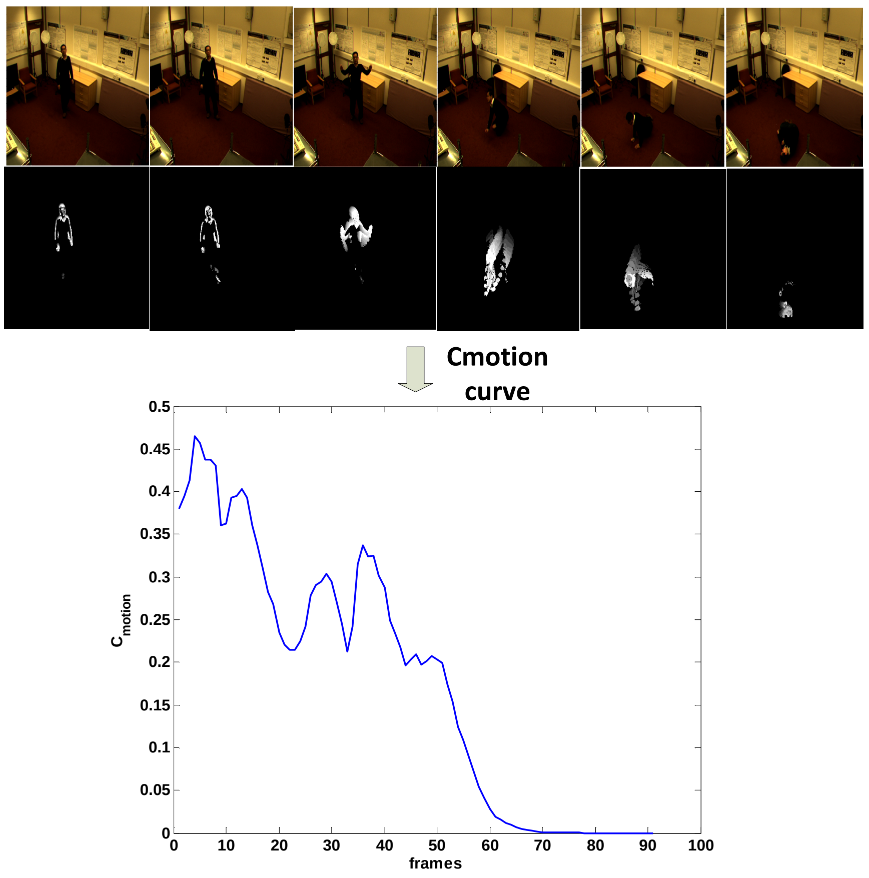


Figure 3.18. C_{motion} curve for a fall activity which is parallel to the optical axis of the camera. The x-axis indicates the frame number and the y-axis indicates the change of C_{motion} value per frame.

Table 3.6. A person enters the scene at around the 600th frame and in total more than 1200 frames are recorded.

This sequence is divided into five phases: in Phase I, a person walks in the room without any occlusion and none of the head velocity, *theta* velocity and *rho* velocity exceed the threshold, so no falls are reported.

In Phase II, the person walks behind a chair. Due to the occlusion of the chair, large changes of the *rho* value and the *rho* velocity exceed the thresholds. However, the head velocity is still far below the threshold value. So, no falls are reported because the condition for the head velocity is not met.

In Phase III, this person sits quickly down on the chair and the head

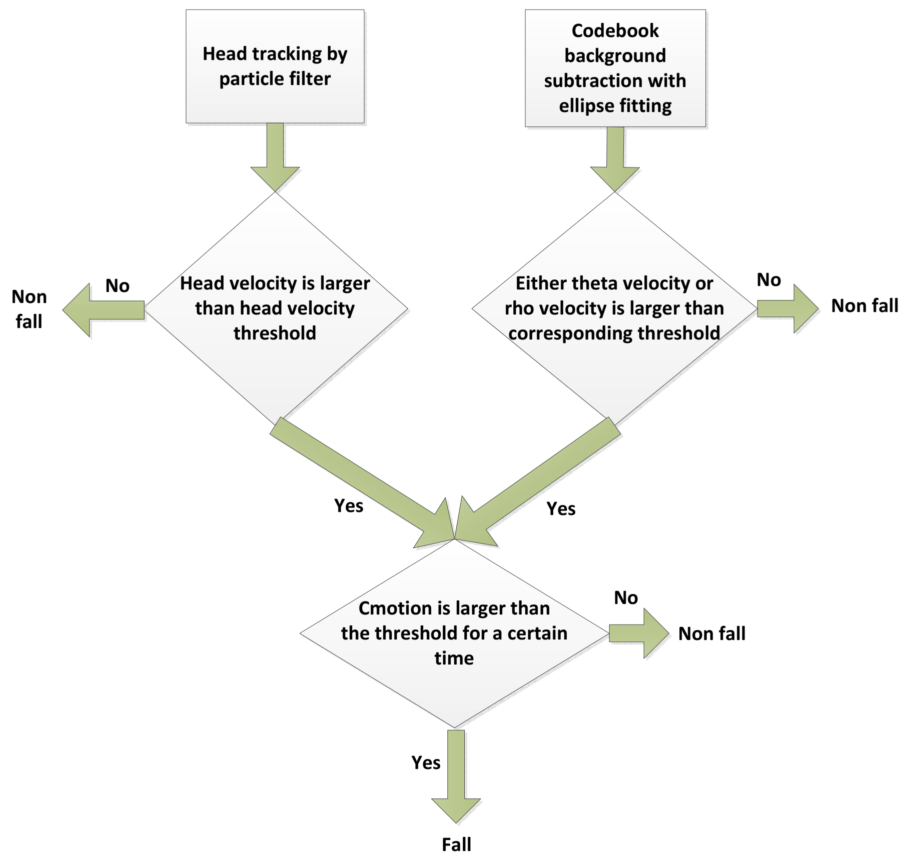

Head tracking by particle filter
Codebook background subtraction with ellipse fitting
Head velocity is larger than head velocity threshold
Either theta velocity or rho velocity is larger than corresponding threshold
No
Non fall
No
Non fall
Yes
Yes
Cmotion is larger than the threshold for a certain time
No
Non fall
Yes
Fall

Figure 3.19. The flowchart of the proposed fall detection system.

velocity exceeds the threshold. Although the head velocity condition for falling is met for this case, neither the *rho* velocity nor the *theta* velocity exceeds the corresponding threshold. So no falls are reported because no large shape change is detected.

Phase IV shows a person bends to fasten his shoe belts. In this case, both the head velocity and *rho* velocity exceed the corresponding thresholds. So both the head velocity condition and shape change conditions are met; however, the C_{motion} curve shows that the length of the low movement amplitude (the C_{motion} value is less than the threshold) period is short and less than 200 frames (because the person only bends for a short time and then continues to walk). So, no falls are reported because the third condition is not met.

In Phase V, a person falls on the ground. All of the three conditions

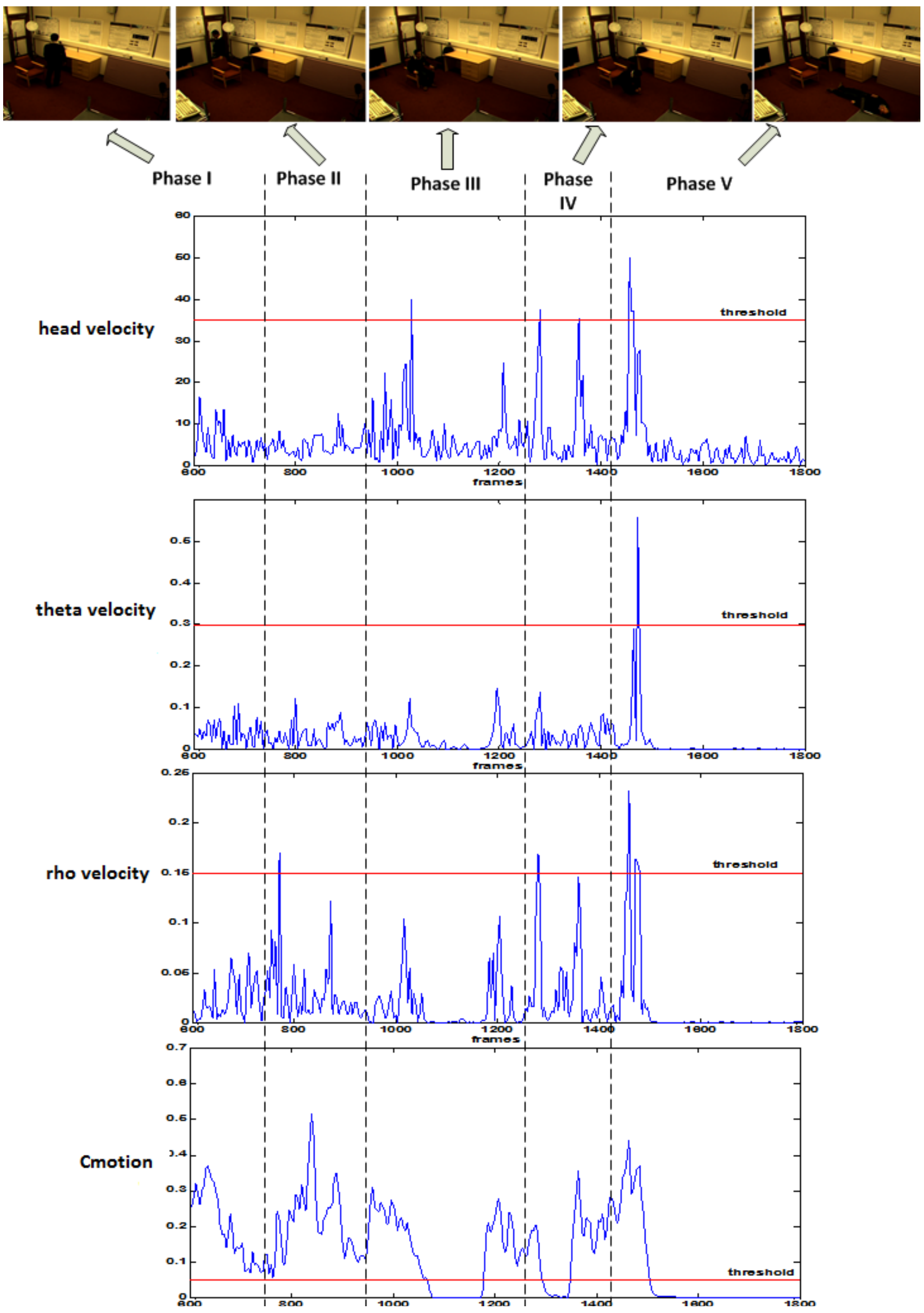


Figure 3.20. The variations of head velocity, theta velocity, rho velocity and C_{motion} for a sequence containing different types of activities.

are met–the head velocity, *rho* velocity and *theta* velocity exceed the corresponding thresholds, and the C_{motion} value is below the threshold for a long time. So, a fall is detected.

3.5 Summary

In this chapter, a simple and robust fall detection system has been proposed by using three cues: head velocity cue, shape cue and movement amplitude cue. The head was tracked by the particle filter using the gradient and colour information; the velocity of the head between consecutive frames was then estimated from the head tracking results. Codebook background subtraction method and the moment-based ellipse fitting were then applied to fit the human body region by an ellipse. And the shape change can be reflected by the velocities of the orientation angle and two axes' ratio of an ellipse. Finally, the movement amplitude of a person was estimated by a C_{motion} value obtained from MHI for a further confirmation of a fall when both the head velocity and shape change exceed certain threshold values. Experiment on a comparatively simple dataset was presented to show that by combining these three cues–head velocity, shape change and movement amplitude, a simple but robust fall detection system can be achieved which can distinguish falls with different types of non-fall activities under a proper threshold set.

However, this proposed system has its limitations from the following aspects:

1. The 2D head velocity cue commonly generates false alarms; fast walking, sudden sitting or bending, even fast nodding will generate high head velocity, which is similar to the case of falling.

2. The orientation angle and the two axes' ratio obtained from the fitted ellipse are not invariant to directions, as mentioned in [38]. For the same activity in different directions, the variations of orientation angle and ratio are different. Although good results are obtained in distinguishing two-direction falls from some non-fall activities performed in certain directions;

however, these features are not adequate for distinguishing falls and non-falls in all different directions.

3. For this system, many threshold values (th_{head}, th_{theta}, th_{rho}, $th_{movement}$ and th_{period}) need to be chosen carefully, which is inconvenient for the practical use.

In order to solve these problems, more elegant 3D features which are both distinguishable between falls and non-falls and invariant to different directions need to be considered; instead of using a simple analytic method by comparing the extracted cues with different thresholds, a model based method is applied to represent fall activities by certain models which are then used for distinguishing fall activities and non-fall activities by only one single threshold value. The details of the 3D features and model based method are discussed in the next chapter. Besides, another dataset which contains more types of fall activities and non-fall activities will be recorded for a better validation.

Chapter 4

DATA FITTING MODEL BASED FALL DETECTION METHOD BY USING 3D FEATURES

4.1 Introduction

For solving the problem that 2D features used in earlier fall detection works are not invariant to directions, in this chapter, a fall detection method is proposed on the basis of data fitting schemes, which model fall activities based on obtained 3D features. A 3D person representation is initially constructed by multiple calibrated cameras, from which 3D features are then extracted. The resulting 3D features corresponding to fall activities are used to build a particular model for fitting the feature dataset and the model is then applied to distinguish fall activities and non-fall activities. Three forms of models: single Gaussian model, mixture of Gaussians model and one class support vector machine model (OCSVM) are tested and the corresponding comparison results in the context of fall detection are presented in the experimental section.

4.2 Camera Calibration

The first step in reconstructing the 3D person and obtaining corresponding 3D features is the camera calibration, which involves estimating the external parameters and internal parameters of a camera based on certain camera models. Some well-known camera calibration methods include: the DLT method [61], Zhang method [62] and Tsai method [63]. In this work, the Tsai camera calibration method is adopted. Compared with the DLT method and Zhang method, the Tsai camera calibration method offers the possibility to calibrate internal and external parameters separately, which is particularly useful since it gives the possibility to fix the internal parameters of the camera, when known, and carry out only pose estimation.

4.2.1 Tsai Camera Model

The Tsai camera model assumes a four step procedure of transformation from 3D world coordinates to image coordinates captured by a camera, which is based on the camera geometry as shown in Figure 4.1.

Step I. For a 3D point in a scene (denoted as P as shown in Figure 4.1), it has a corresponding 3D world coordinate (x_w, y_w, z_w) (based on the 3D world coordinate system (O_w, x_w, y_w, z_w) as shown in Figure 4.1). The camera itself also has its own coordinate system, which is called the camera coordinate system denoted as (O, x, y, z) in Figure 4.1 (with the origin being the center of the lens of the camera). The world coordinate system can be aligned to the camera coordinate system by certain rotation and translation operations. So, the 3D world coordinate (x_w, y_w, z_w) and camera coordinate (x, y, z) of the point P have the following relationship:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \tag{4.2.1}$$

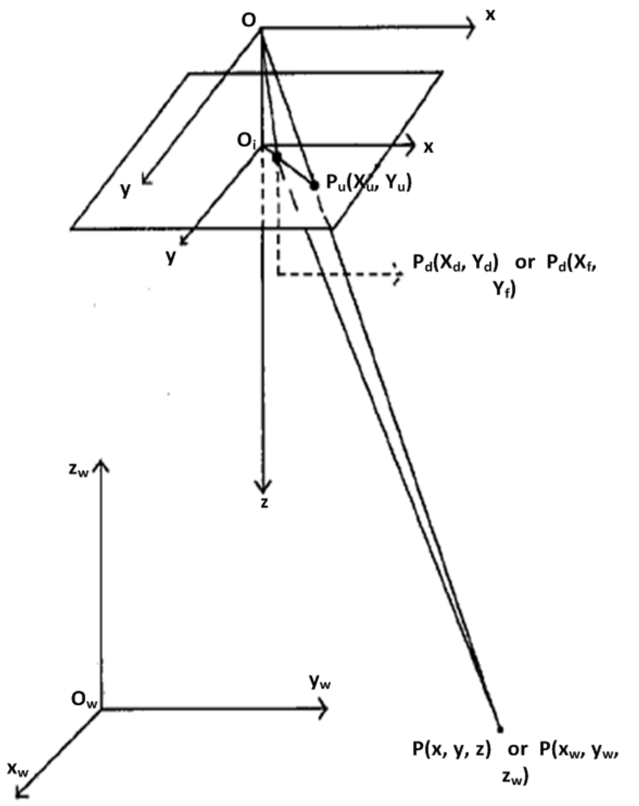

O
x
O$_i$
x
y
P$_u$(X$_u$, Y$_u$)
y
P$_d$(X$_d$, Y$_d$) or P$_d$(X$_f$, Y$_f$)
z
z$_w$
O$_w$
y$_w$
x$_w$
P(x, y, z) or P(x$_w$, y$_w$, z$_w$)

Figure 4.1. The camera geometry corresponding to the Tsai model. (O_w, x_w, y_w, z_w) is the 3D world coordinate system and (O, x, y, z) is the camera coordinate system, one 3D point P is converted to the point P_d on the image plane (O_i, x, y) according to the four step procedure for coordinates transformation.

where R is a 3×3 rotation matrix which can be represented by three rotation angles in 3D space (Euler angles yaw θ, pitch ϕ and tilt ψ) as:

$$R = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\ -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \cos(\psi)\cos(\phi) + \sin(\psi)\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) \\ \sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\phi) & -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{4.2.2}$$

or in a simpler form:

$$R = \begin{bmatrix} r1 & r2 & r3 \\ r4 & r5 & r6 \\ r7 & r8 & r9 \end{bmatrix} \tag{4.2.3}$$

and T is a 3×1 vector with:

$$T = \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix} \tag{4.2.4}$$

Step II. After the camera coordinate (x, y, z) of the point P is obtained, according to the pinhole camera model [63], the image plane coordinate (X_u, Y_u) of the ideally projected point P_u can be obtained as:

$$\begin{aligned} X_u &= f\frac{x}{z} \\ Y_u &= f\frac{y}{z} \end{aligned} \tag{4.2.5}$$

where f is the focal length of the camera, which measures the distance between the center of the lens and image plane.

Step III. Rather than the perfect pinhole camera model, for the real camera there are inevitably certain types of distortion. One important distortion which has the largest effect is the radial lens distortion. Due to radial lens distortion, the actual point on the image plane (denoted as P_d) is different from the ideally projected point P_u and the image plane coordinate of P_d is denoted as (X_d, Y_d). As proposed in [63], the relationship between (X_u, Y_u) and (X_d, Y_d) is:

$$\begin{aligned} X_u &= X_d(1 + kr^2) \\ Y_u &= Y_d(1 + kr^2) \end{aligned} \tag{4.2.6}$$

where $r = \sqrt{X_d^2 + Y_d^2}$ and k is the coefficient of the radial lens distortion.

Step IV. The last step is to convert the image plane coordinate (X_d, Y_d) to the image plane coordinate (X_f, Y_f) measured by pixels with the following transformation:

$$\begin{aligned} X_f &= s_x \dot{d}_x^{-1} X_d + C_x \\ Y_f &= \dot{d}_y^{-1} Y_d + C_y \end{aligned} \tag{4.2.7}$$

where C_x and C_y are the center coordinates of the captured image, s_x is the uncertainty scaling factor due to camera scanning and acquisition time error, $\dot{d}_x$ and $\dot{d}_y$ represent the corresponding sizes of a pixel, which are calculated as:

$$\begin{aligned} \dot{d}_x &= d_x \frac{N_{cx}}{N_{fx}} \\ \dot{d}_y &= d_y \frac{N_{cy}}{N_{fy}} \end{aligned} \tag{4.2.8}$$

where d_x and d_y are the center-to-center distance between adjacent CCD sensor elements in the X and Y directions, N_{cx} and N_{cy} are the numbers of CCD sensor elements in the X and Y directions and N_{fx} and N_{fy} are the numbers of image pixels in the X and Y directions.

The above four steps show the correspondence between the 3D real world coordinate and 2D image pixel coordinate for a particular point. From the above four steps, a 3D point is converted to a 2D point in the image.

4.2.2 Parameter Estimation

Different camera parameters are employed in the above four steps and in camera calibration these parameters are estimated. Figure 4.2 shows the parameters needed to be estimated for each step.

These parameters are divided into two groups:

External parameters: rotation matrix R and translation vector T.

Internal parameters: focal length–f, radial lens distortion coefficient–k and uncertainty scale factor–s_x.

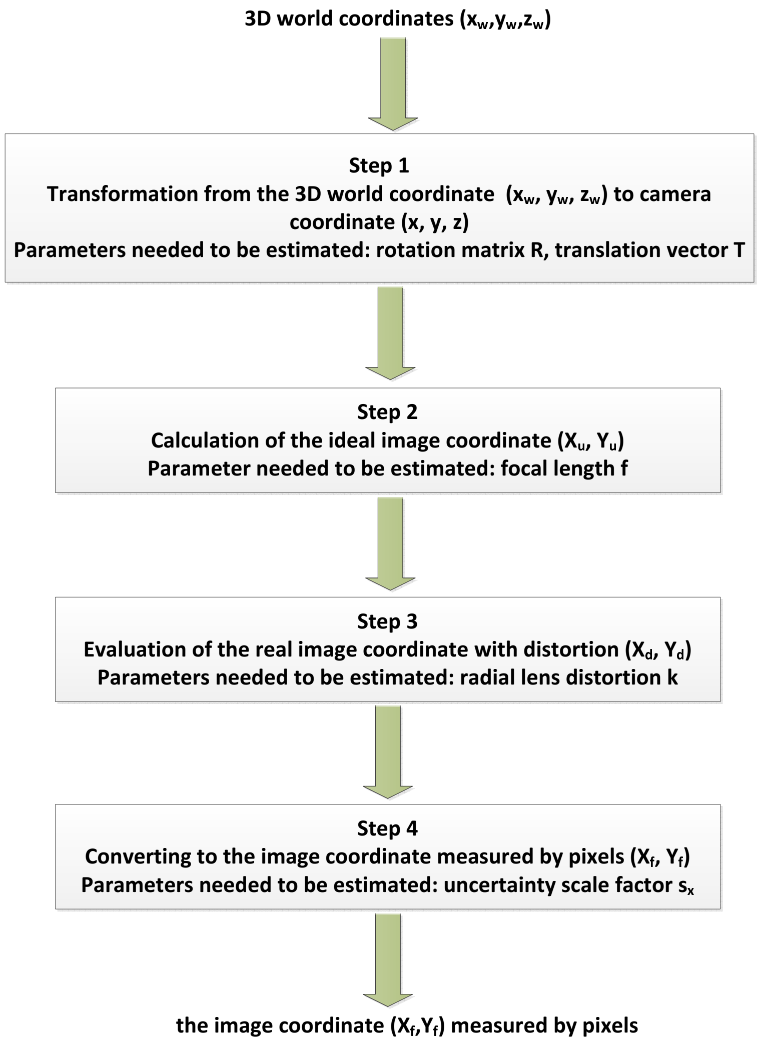

3D world coordinates (xw,yw,zw)
Step 1
Transformation from the 3D world coordinate (xw, yw, zw) to camera coordinate (x, y, z)
Parameters needed to be estimated: rotation matrix R, translation vector T
Step 2
Calculation of the ideal image coordinate (Xu, Yu)
Parameter needed to be estimated: focal length f
Step 3
Evaluation of the real image coordinate with distortion (Xd, Yd)
Parameters needed to be estimated: radial lens distortion k
Step 4
Converting to the image coordinate measured by pixels (Xf, Yf)
Parameters needed to be estimated: uncertainty scale factor sx
the image coordinate (Xf,Yf) measured by pixels

Figure 4.2. The flowchart of converting from the 3D world coordinate to the 2D image pixel coordinate and the parameters that must be estimated.

The procedures to estimate the external and internal parameters are shown in the remaining parts of the section.

Estimations of $T_y^{-1}s_xr1$, $T_y^{-1}s_xr2$, $T_y^{-1}s_xr3$, $T_y^{-1}s_xT_x$, $T_y^{-1}r4$, $T_y^{-1}r5$ and $T_y^{-1}r6$

If points $P(x, y, z)$ and $P_{oz}(0, 0, z)$ are connected (as shown in Figure 4.3), $\overline{O_iP_d}//\overline{P_{oz}P}$ (// means parallel) can be obtained by the fact that $\overline{O_iP_d}$ and $\overline{P_{oz}P}$ are the intersections of the plane (O, P, P_{oz}) and two parallel planes $((O_i, x, y)$ and $(P_{oz}, x, y))$.

As $\overline{O_iP_d}//\overline{P_{oz}P}$, and $\overline{O_iP_d} \times \overline{P_{oz}P} = 0$ ('$\times$' denotes the cross product),

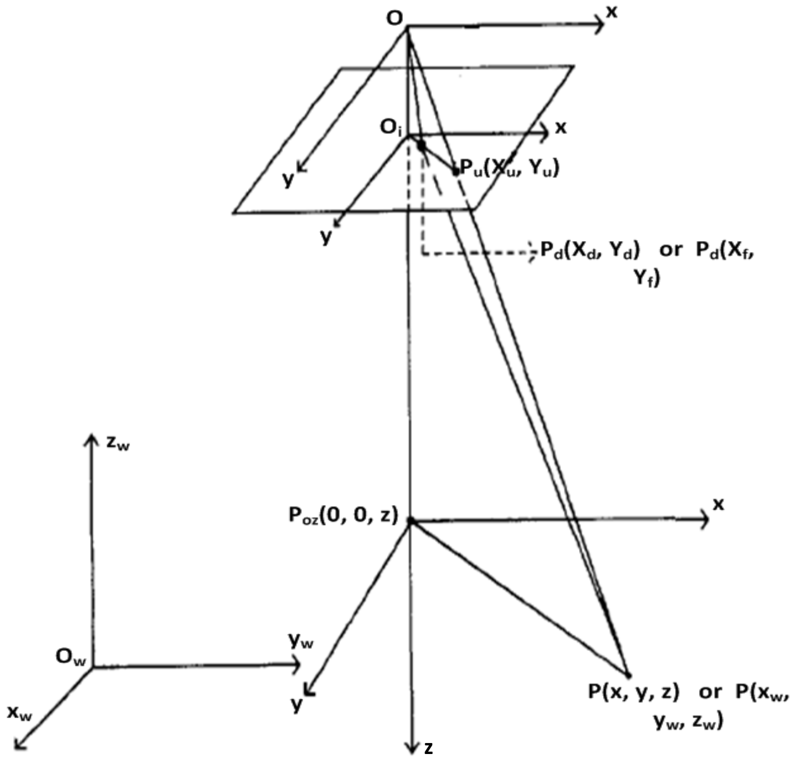

O
x
O_i
x
y
P_u(X_u, Y_u)
y
P_d(X_d, Y_d) or P_d(X_f, Y_f)
z_w
P_oz(0, 0, z)
x
y_w
O_w
y
x_w
P(x, y, z) or P(x_w, y_w, z_w)
z

Figure 4.3. $\overline{O_iP_d}//\overline{P_{oz}P}$ by connecting P and P_{oz} by the fact that $\overline{O_iP_d}$ and $\overline{P_{oz}P}$ are the intersections of the plane (O, P, P_{oz}) with two parallel planes (O_i, x, y) and (P_{oz}, x, y).

then:

$$(X_d, Y_d) \times (x, y) = 0 \tag{4.2.9}$$

from which it yields $X_dy - Y_dx = 0$.

If equations (4.2.3) and (4.2.4) are substituted into equation (4.2.1), the following can be obtained:

$$\begin{aligned} x &= r1x_w + r2y_w + r3z_w + T_x \\ y &= r4x_w + r5y_w + r6z_w + T_y \end{aligned} \tag{4.2.10}$$

and from equation (4.2.9), it can be derived that:

$$X_d(r4x_w + r5y_w + r6z_w + T_y) - Y_d(r1x_w + r2y_w + r3z_w + T_x) = 0 \tag{4.2.11}$$

By dividing each side by T_y and after some algebraic operations, equation (4.2.11) can be obtained:

$$\begin{bmatrix} Y_d x_w & Y_d y_w & Y_d z_w & Y_d & -X_d x_w & -X_d y_w & -X_d z_w \end{bmatrix} \cdot \begin{bmatrix} T_y^{-1} r1 \\ T_y^{-1} r2 \\ T_y^{-1} r3 \\ T_y^{-1} T_x \\ T_y^{-1} r4 \\ T_y^{-1} r5 \\ T_y^{-1} r6 \end{bmatrix} = X_d \tag{4.2.12}$$

Both sides are then multiplied by s_x and $\dot{X}_d$ is used to replace $X_d s_x$, equation (4.2.12) can be written as:

$$\begin{bmatrix} Y_d x_w & Y_d y_w & Y_d z_w & Y_d & -\dot{X}_d x_w & -\dot{X}_d y_w & -\dot{X}_d z_w \end{bmatrix} \cdot \begin{bmatrix} T_y^{-1} s_x r1 \\ T_y^{-1} s_x r2 \\ T_y^{-1} s_x r3 \\ T_y^{-1} s_x T_x \\ T_y^{-1} r4 \\ T_y^{-1} r5 \\ T_y^{-1} r6 \end{bmatrix} = \dot{X}_d \tag{4.2.13}$$

Initially, a set of points is chosen and for each point i, its 3D world coordinate (x_{wi}, y_{wi}, z_{wi}) and 2D image pixel coordinate (X_{fi}, Y_{fi}) are known. By using equation (4.2.7), $\dot{X}_{di}$ and Y_{di} can be calculated as:

$$\begin{aligned} \dot{X}_{di} &= (X_{fi} - C_x)\dot{d}_x \\ Y_{di} &= (Y_{fi} - C_y)\dot{d}_y \end{aligned} \tag{4.2.14}$$

and equation (4.2.13) can be denoted as:

$$\mathbf{a}_i \cdot \mathbf{x} = b_i \tag{4.2.15}$$

where

$\mathbf{a}_i = \begin{bmatrix} Y_{di}x_{wi} & Y_{di}y_{wi} & Y_{di}z_{wi} & Y_{di} & -\dot{X}_{di}x_{wi} & -\dot{X}_{di}y_{wi} & -\dot{X}_{di}z_{wi} \end{bmatrix}$,

$\mathbf{x} = \begin{bmatrix} T_y^{-1}s_x r1 \\ T_y^{-1}s_x r2 \\ T_y^{-1}s_x r3 \\ T_y^{-1}s_x T_x \\ T_y^{-1}r4 \\ T_y^{-1}r5 \\ T_y^{-1}r6 \end{bmatrix}$ and $b_i = \dot{X}_{di}$, each element of the vector $\mathbf{a}_i$ and b_i are all known.

If there are N points, then equation (4.2.16) holds:

$$\begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_i \\ \vdots \\ \mathbf{a}_N \end{bmatrix} \mathbf{x} = \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{bmatrix} \tag{4.2.16}$$

and if $\mathbf{A}$ is used to represent $\begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_i \\ \vdots \\ \mathbf{a}_N \end{bmatrix}$ which is a $N \times 7$ matrix, and $\mathbf{b}$ is used to represent $\begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{bmatrix}$ which is an $N \times 1$ vector then equation (4.2.17) can be obtained as:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{4.2.17}$$

and the least squares solution for equation (4.2.17) is:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{4.2.18}$$

from which the seven parameters:
$T_y^{-1}s_xr1$, $T_y^{-1}s_xr2$, $T_y^{-1}s_xr3$, $T_y^{-1}s_xT_x$, $T_y^{-1}r4$, $T_y^{-1}r5$ and $T_y^{-1}r6$ are estimated.

Determination of T_y and s_x

The solution of equation (4.2.18) can be used to determine $|T_y|$ and if the i-th element of $\mathbf{x}$ is denoted as x_i, $|T_y|$ can be calculated as:

$$|T_y| = (x_5^2 + x_6^2 + x_7^2)^{-1/2} \tag{4.2.19}$$

by using $r_4^2 + r_5^2 + r_6^2 = 1$ according to the correspondence between equation (4.2.2) and (4.2.3). The uncertainty scale factor s_x can be determined from $|T_y|$ as:

$$s_x = (x_1^2 + x_2^2 + x_3^2)^{-1/2}|T_y| \tag{4.2.20}$$

by using $r_1^2 + r_2^2 + r_3^2 = 1$.

To determine the sign of T_y an object point i whose 2D image pixel coordinate (X_{fi}, Y_{fi}) is away from the image center (C_x, C_y) is selected; its 3D world coordinate is denoted as (x_{wi}, y_{wi}, z_{wi}). Initially, the sign of T_y is assumed as +1 and this point's 3D camera coordinate (x_i, y_i) is computed:

$$x_i = r1x_{wi} + r2y_{wi} + r3z_{wi} + T_x \quad y_i = r4x_{wi} + r5y_{wi} + r6z_{wi} + T_y \tag{4.2.21}$$

where $r1$, $r2$, $r3$, $r4$, $r5$, $r6$ and T_x can be calculated as:

$$\begin{aligned}
r1 &= (T_y^{-1}s_xr1) \cdot T_ys_x^{-1} = x_1T_ys_x^{-1} \quad r2 = (T_y^{-1}s_xr2) \cdot T_ys_x^{-1} = x_2T_ys_x^{-1} \\
r3 &= (T_y^{-1}s_xr3) \cdot T_ysx^{-1} = x_3T_ys_x^{-1} \quad r4 = (T_y^{-1}r4) \cdot T_y = x_5T_y \\
r5 &= (T_y^{-1}r5) \cdot T_y = x_6T_y \quad r6 = (T_y^{-1}r6) \cdot T_y = x_7T_y \\
T_x &= (T_y^{-1}T_x) \cdot T_y = x_4T_y
\end{aligned} \tag{4.2.22}$$

Moreover, by using equation (4.2.7), the distorted image coordinate (X_{di}, Y_{di}) can be obtained from (X_{fi}, Y_{fi}). From the relationship between (X_{di}, Y_{di}) and (x_i, y_i) as shown in equations (4.2.5) and (4.2.6), it can be concluded that X_{di} and x_i, together with Y_{di} and y_i should have the same signs. So, if the calculated (X_{di}, Y_{di}) and (x_i, y_i) meet this condition, the initial assumption is right and the sign of T_y is +1; otherwise, the sign of T_y is −1.

Computations of R and T_x

After obtaining T_y, s_x and the solution $\mathbf{x} = [T_y^{-1}s_xr1,\ T_y^{-1}s_xr2,\ T_y^{-1}s_xr3,\ T_y^{-1}s_xT_x,\ T_y^{-1}r4,\ T_y^{-1}r5,\ T_y^{-1}r6]$, some elements of the rotation matrix $(r1, r2, r3, r4, r5, r6)$ and T_x can be obtained in the same way as equation (4.2.22).

For the remaining three elements $r7, r8, r9$ of the rotation matrix R, they can be estimated by using the fact that the third row of R can be computed as the cross product of the first two rows according to the form of R (equation

(4.2.2)), which becomes:

$$\begin{aligned} r7 &= r2 * r6 - r3 * r5 \\ r8 &= r3 * r4 - r1 * r6 \\ r9 &= r1 * r5 - r2 * r4 \end{aligned} \tag{4.2.23}$$

Computations of f, T_z and k

The remaining parameters needed to be calculated include focal length–f, T_z and radial lens distortion coefficient–k. In order to obtain these parameters, a two-step procedure is applied:

Step I. For a particular point P_i in a set of N points $\{P_1, ..., P_N\}$, by combining equations (4.2.5), (4.2.6) and (4.2.10), the following can be obtained:

$$\begin{aligned} X_{di}(1 + kr^2) &= f\frac{r1x_{wi} + r2y_{wi} + r3z_{wi} + T_x}{r7x_{wi} + r8y_{wi} + r9z_{wi} + T_z} \\ Y_{di}(1 + kr^2) &= f\frac{r4x_{wi} + r5y_{wi} + r6z_{wi} + T_y}{r7x_{wi} + r8y_{wi} + r9z_{wi} + T_z} \end{aligned} \tag{4.2.24}$$

if k is set to zero, then equation (4.2.25) can be obtained:

$$\begin{aligned} \begin{bmatrix} x_i & -X_{di} \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} &= X_{di}w_i \\ \begin{bmatrix} y_i & -Y_{di} \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} &= Y_{di}w_i \end{aligned} \tag{4.2.25}$$

where $x_i = r1x_{wi} + r2y_{wi} + r3z_{wi} + T_x$, $y_i = r4x_{wi} + r5y_{wi} + r6z_{wi} + T_y$ and $w_i = r7x_{wi} + r8y_{wi} + r9z_{wi}$.

Extending this to N points, then:

$$\begin{bmatrix} x_1 & -X_{d1} \\ y_1 & -Y_{d1} \\ & \vdots \\ x_i & -X_{di} \\ y_i & -Y_{di} \\ & \vdots \\ x_i & -X_{dN} \\ y_N & -Y_{dN} \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} = \begin{bmatrix} X_{d1}w_1 \\ Y_{d1}w_1 \\ \vdots \\ X_{di}w_i \\ Y_{di}w_i \\ \vdots \\ X_{dN}w_N \\ Y_{dN}w_N \end{bmatrix} \tag{4.2.26}$$

Similarly, a least squares solution of $\begin{bmatrix} f \\ T_z \end{bmatrix}$ can be obtained by using equation (4.2.18), with $A = \begin{bmatrix} x_1 & -X_{d1} \\ y_1 & -Y_{d1} \\ & \vdots \\ x_i & -X_{di} \\ y_i & -Y_{di} \\ & \vdots \\ x_i & -X_{dN} \\ y_N & -Y_{dN} \end{bmatrix}$ which is a $2N \times 2$ matrix and

$b = \begin{bmatrix} X_{d1}w_1 \\ Y_{d1}w_1 \\ \vdots \\ X_{di}w_i \\ Y_{di}w_i \\ \vdots \\ X_{dN}w_N \\ Y_{dN}w_N \end{bmatrix}$ which is a $2N \times 1$ vector.

Step II. In order to obtain more precise estimates of f, T_z and k, an error function $e(f, T_z, k)$ with respect to f, T_z and k is minimized, which becomes:

$$\begin{aligned} e(f, T_z, k) &= \sum_{i=1}^{N} \left(X_{di}(1 + kr^2) - f \frac{r1x_{wi} + r2y_{wi} + r3z_{wi} + T_x}{r7x_{wi} + r8y_{wi} + r9z_{wi} + T_z} \right)^2 \\ &+ \sum_{i=1}^{N} \left(Y_{di}(1 + kr^2) - f \frac{r4x_{wi} + r5y_{wi} + r6z_{wi} + T_y}{r7x_{wi} + r8y_{wi} + r9z_{wi} + T_z} \right)^2 \end{aligned} \tag{4.2.27}$$

$e(f, T_z, k)$ is a non-linear function with respect to f, T_z and k and in order to minimize it, some non-linear optimization methods, such as the steepest gradient method, Gaussian Newton method and Levenberg-Marquardt method (damped Gaussian-Newton method) [64] can be applied. In this work, the Levenberg-Marquardt method is applied, which obtains its operating stability from the steepest descent method and gains its accelerated convergence in the minimum vicinity from the Newton method as shown in [64] and the f, T_z and k values obtained in Step I are used as the initial values for the Levenberg-Marquardt method. By minimizing $e(f, T_z, k)$, the difference between the image plane coordinate obtained from 2D image pixel and that obtained from 3D real world coordinate will be small, which indicates an accurate camera model for the correspondence between the 2D image pixel coordinate and the 3D real world coordinate.

Until now, all the camera parameters are estimated and the estimated camera parameters can then be applied to reconstruct the approximate 3D person.

4.3 3D person reconstruction and 3D features extraction

4.3.1 3D person reconstruction

For 3D person reconstruction, as presented in [39], the room space (assumed to be cubic) is divided into non-overlapping blocks called ‘voxels’ as shown in Figure 4.4. Similar to Section 4.1, (O_w, x_w, y_w, z_w) and (O, x, y, z) represent the 3D world coordinate system and 3D camera coordinate system respectively.

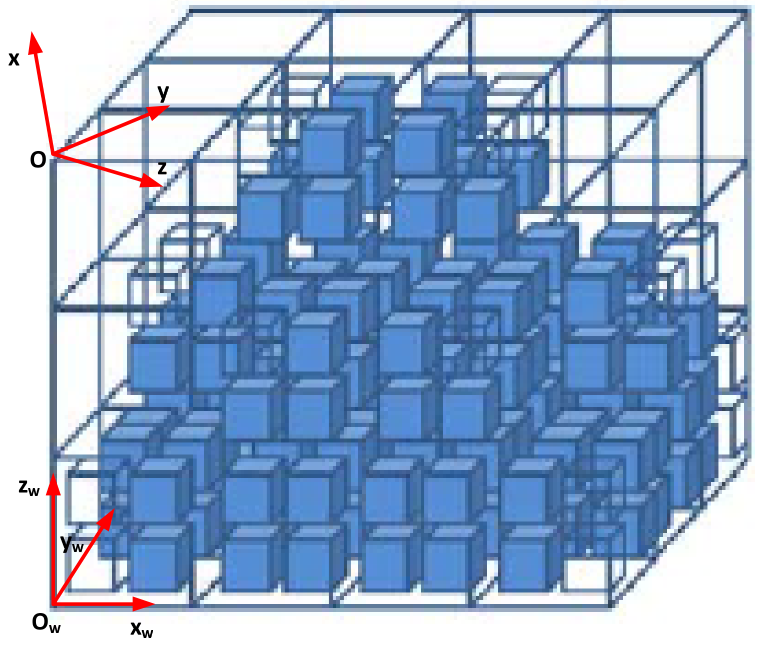


Figure 4.4. A 3D room space which is composed of non-overlapping voxels (marked as blue blocks). (O_w, x_w, y_w, z_w) and (O, x, y, z) represent the 3D world coordinate system and 3D camera coordinate system respectively

For a 2D image pixel P with the 2D image pixel coordinate (X_f, Y_f), from equations (4.2.6) and (4.2.7) the image plane coordinate (X_u, Y_u) of the corresponding ideally projected point P_u on the image plane can be obtained from (X_f, Y_f) as:

$$\begin{aligned} X_u &= \frac{(X_f - C_x)\dot{d}_x}{s_x}(1 + kr^2) \\ Y_u &= (Y_f - C_y)\dot{d}_y(1 + kr^2) \end{aligned} \tag{4.3.1}$$

For every point on the image plane, its coordinate in the z-axis of the 3D camera coordinate system is the focal length–f, so the 3D camera coordinate of the point P_u is (X_u, Y_u, f), which can be converted to the 3D world coordinate by equation (4.2.1), denoted as (Px_w, Py_w, Pz_w). Similarly, the original point $O(0,0,0)$ of the 3D camera coordinate system (center of the camera lens) can also be converted to the 3D world coordinate by equation (4.2.1), denoted as (Ox_w, Oy_w, Oz_w).

Points (Px_w, Py_w, Pz_w) and (Ox_w, Oy_w, Oz_w) can determine a 3D line in the 3D world coordinate system (x_w, y_w, z_w) as:

$$\frac{x_w - Ox_w}{Px_w - Ox_w} = \frac{y_w - Oy_w}{Py_w - Oy_w} = \frac{z_w - Oz_w}{Pz_w - Oz_w} = t \tag{4.3.2}$$

which intersects with voxels in the 3D space.

In this work a simple and efficient strategy is applied for obtaining the voxels with which the 3D line intersects. Assuming the size of a voxel is $l \times l \times l$ and the numbers of voxels along the x_w, y_w and z_w directions (which coincide with the length, width and height of the 3D space as shown in Figure 4.4) in 3D room space are Nx_w, Ny_w and Nz_w respectively. The 3D room space can then be divided into Nx_w, Ny_w and Nz_w bins along the three axes of the 3D world coordinate system, with each bin's length being l.

For the i-th bin along the x_w direction, its x_w coordinate range is $[(i-1) * l, i * l]$. As such $(i-1) * l$ and $i * l$ can be substituted into equation (4.3.2) to obtain the y_w range of the line segment in that bin, denoted as $[y_{w_{low}}, y_{w_{high}}]$ with

$$\begin{aligned} y_{w_{low}} &= \min\{Oy_w + \frac{(i-1) * l - Ox_w}{Px_w - Ox_w}(Py_w - Oy_w), Oy_w + \frac{i * l - Ox_w}{Px_w - Ox_w}(Py_w - Oy_w)\} \\ y_{w_{high}} &= \max\{Oy_w + \frac{i * l - Ox_w}{Px_w - Ox_w}(Py_w - Oy_w), Oy_w + \frac{(i-1) * l - Ox_w}{Px_w - Ox_w}(Py_w - Oy_w)\} \end{aligned} \tag{4.3.3}$$

which can be converted to the bin-index range $[I^{y_w}_{low}, I^{y_w}_{high}]$ along the y_w axis as:

$$I^{y_w}_{low} = int(\frac{y_{w_{low}}}{l}) + 1$$
$$I^{y_w}_{high} = int(\frac{y_{w_{high}}}{l}) + 1 \qquad (4.3.4)$$

where $int(\cdot)$ represents the operation of getting the integer part of the result.

Finally, $[I^{y_w}_{low}, I^{y_w}_{high}]$ intersects with $[1, Ny_w]$ to obtain the bin-index range confined by the 3D room space, denoted as $[\tilde{I}^{y_w}_{low}, \tilde{I}^{y_w}_{high}]$ if the intersection result is non-empty.

In a similar way, for every bin index from $\tilde{I}^{y_w}_{low}$ to $\tilde{I}^{y_w}_{high}$, the corresponding z_w direction's bin-index range confined by the 3D room space (denoted as $[\tilde{I}^{z_w}_{low}, \tilde{I}^{z_w}_{high}]$) can be calculated. In the end, the bin indices of voxels which are intersected with the 3D line are obtained in the i-th bin along the x_w direction.

Figure 4.5 shows the procedure of obtaining the intersection of voxels in the i-th bin along the x_w direction with the line $\overline{OP_u}$, the same procedure is applied for every bin along this direction and to obtain all the voxels with which the 3D line intersects in the 3D room space. For every pixel on the image, the voxels intersected by the corresponding 3D line are obtained which form a voxel set corresponding to that pixel.

Multiple cameras are applied for 3D person reconstruction (as shown in Figure 4.6 for a two-camera case). For each camera the voxel set which corresponds to every pixel on the image plane is initially obtained and the codebook background subtraction method as shown in Section 3 is applied for extracting the moving object. The union of voxels corresponding to pixels in the moving object for the i-th camera is then calculated, denoted as $\mathbf{V}^i_t = \{\mathbf{V}^i_{t,1}, ..., \mathbf{V}^i_{t,P_i}\}$, where t is the captured time, i is the index of the camera, $\mathbf{V}^i_{t,i}$ represents a voxel and P_i is the number of voxels. The voxel person can finally be obtained by intersecting the union sets of multiple

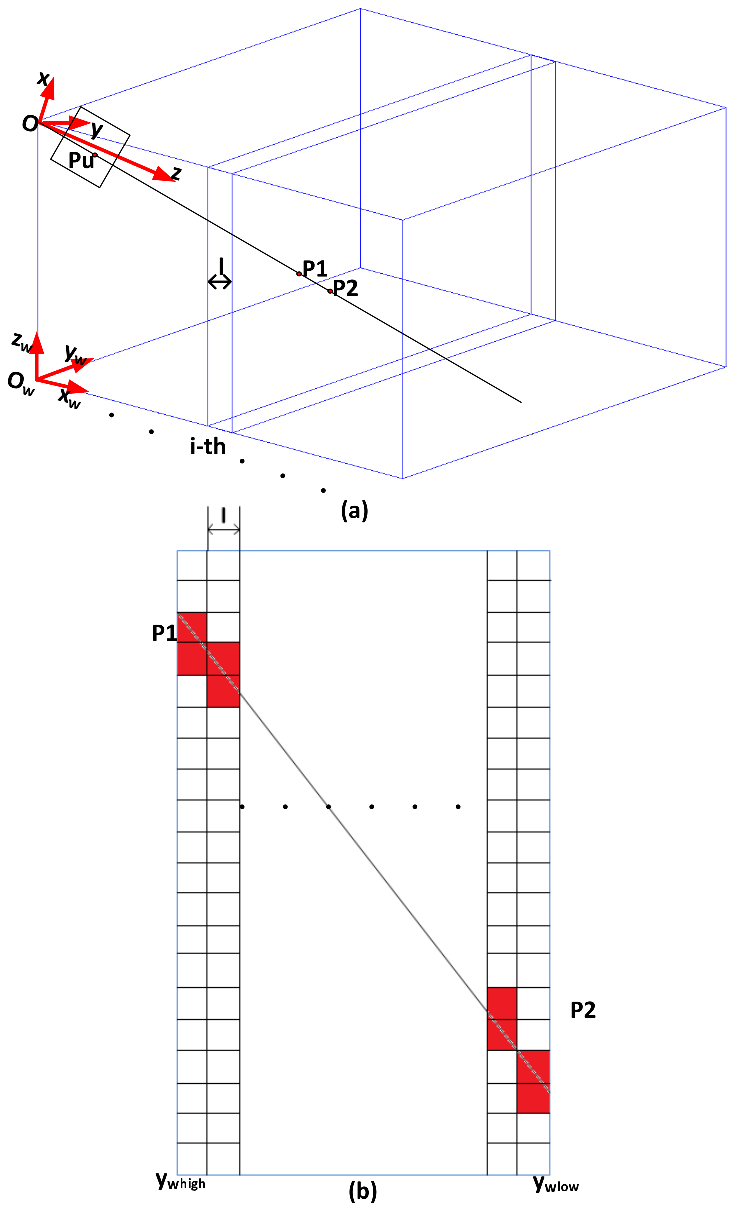


Figure 4.5. The procedure of obtaining the intersected voxels for the i-th bin along the x_w axis. (a) The 3D line connecting the camera coordinate origin O and point P_u intersects with the i-th bin along the x_w axis, the intersected line segment is P_1P_2. (b) The coordinate range of P_1P_2 in y_w direction is denoted as $[y_{w_{low}}, y_{w_{high}}]$, every bin in $[y_{w_{low}}, y_{w_{high}}]$ is tested to obtain the final intersected voxels (marked in red).

cameras as: $\mathbf{V}'_t = \bigcap_{i=1}^{C} \mathbf{V}^i_t$, where $\mathbf{V}'_t$ denotes the voxel set corresponding to the 3D reconstructed person and C is the number of cameras.

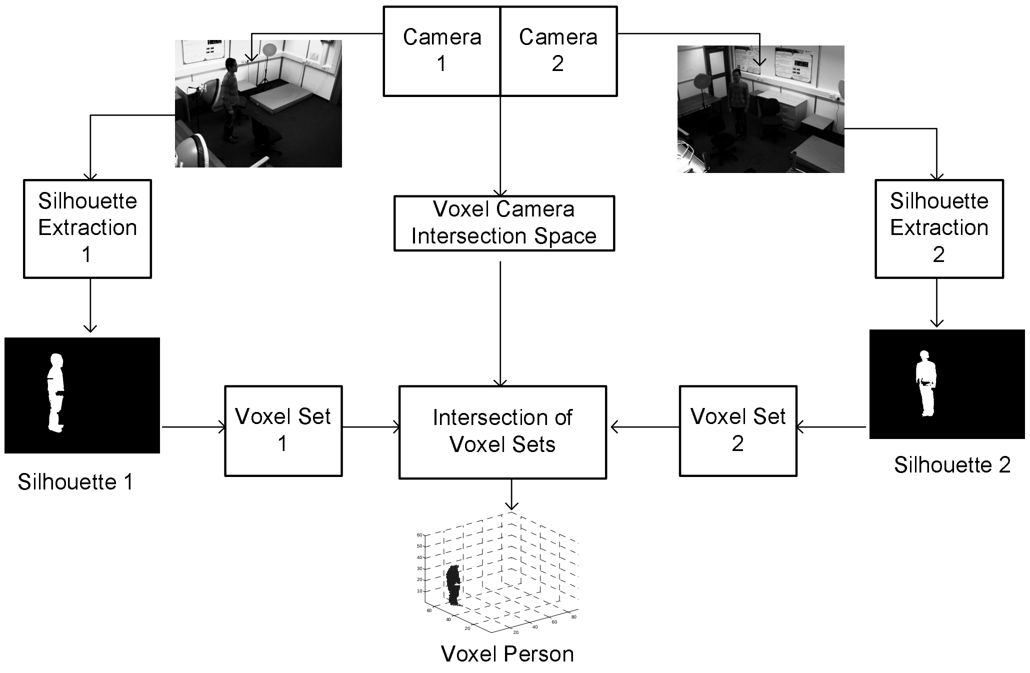


Figure 4.6. The procedure for 3D person reconstruction from two video camera measurements.

4.3.2 3D features extraction

After the voxel set $\mathbf{V}'_t$ is obtained, the position of the centroid and the orientation value (denoted as θ), which reflects the 3D angle between the constructed person and the ground floor plane, are then calculated.

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

$$\mathbf{u}_t = (\frac{1}{P})\sum_{j=1}^{P}\mathbf{V}'_{t,j} \tag{4.3.5}$$

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

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

The covariance matrix used to define the orientation information is:

$$(\frac{1}{P})\sum_{j=1}^{P}(\mathbf{V}'_{t,j} - \mathbf{u}_t)(\mathbf{V}'_{t,j} - \mathbf{u}_t)^T \tag{4.3.6}$$

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

The eigenvalues and eigenvectors of the covariance matrix are calculated according to [65] and the eigenvector corresponding to the largest eigenvalue at time t is denoted as $eigenvec_t$ and the orientation value θ_t is calculated as:

$$\theta_t = max(eigenvec_t \cdot \langle 0,0,1\rangle^T, -eigenvec_t \cdot \langle 0,0,1\rangle^T) \tag{4.3.7}$$

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

This value and its difference during Δ_t ($|\theta_{t+\Delta_t} - \theta_t|$) are chosen as the remaining elements of the feature vector. Finally, a 5-dimensional feature vector is obtained, which consists of the following elements:

The centroid's horizontal position change over Δ_t

The centroid's vertical position change over Δ_t

The centroid's horizontal position at the particular time

The θ value change over Δ_t

The θ value at the particular time

In this work, Δ_t is chosen to be 1s. The 3D video features are obtained and the ones which corresponding to fall activities are used to construct models representing falling, which are then used to distinguish fall and non-fall activities. Different forms of models are adopted and presented in the next section.

4.4 Model methods used for representing fall activities

As mentioned in [66], for a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ drawn from a distribution $\mathbf{P}$, different types of models can be constructed to fit the dataset, which obtains the supporting region (where the distribution value is larger than a particular threshold and the dataset $\mathbf{X}$ is most likely to be sampled) corresponding to the dataset $\mathbf{X}$. In this section, three popular models are introduced, they are the single Gaussian model, mixture of Gaussians model and one class support vector machine (OCSVM) model.

4.4.1 Single Gaussian model

For a single Gaussian model, it follows a distribution $P(\mathbf{x})$ with the form:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(\mathbf{x}-\mathbf{u})^T\Sigma^{-1}(\mathbf{x}-\mathbf{u})) \tag{4.4.1}$$

where d is the dimension of variable $\mathbf{x}$, $\mathbf{u}$ and Σ represent the mean and covariance matrix respectively.

If N dataset samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ are provided, these samples can be used to fit a single Gaussian model. The corresponding model parameters–$\mathbf{u}$ and Σ can be obtained from the maximum likelihood (ML) function as:

$$\begin{aligned} \ln((P(\mathbf{X})) &= \sum_{i=1}^{N} \ln(P(\mathbf{x}_i)) \\ &= \sum_{i=1}^{N} \ln(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(\mathbf{x}_i-\mathbf{u})^T\Sigma^{-1}(\mathbf{x}_i-\mathbf{u}))) \\ &= -\frac{Nd}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i-\mathbf{u})^T\Sigma^{-1}(\mathbf{x}_i-\mathbf{u}) \end{aligned} \tag{4.4.2}$$

To estimate the value of $\mathbf{u}$ which maximizes equation (4.4.2) (denoted as $\mathbf{u}_{ML}$), the derivative of equation (4.4.2) with respect to $\mathbf{u}$ is set to $\mathbf{0}$, which yields:

$$\sum_{i=1}^{N} \Sigma^{-1}(\mathbf{x}_i - \mathbf{u}) = \mathbf{0} \tag{4.4.3}$$

and $\mathbf{u}_{ML}$ is then calculated by solving equation (4.4.3) as:

$$\mathbf{u}_{ML} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \tag{4.4.4}$$

Another value Σ_{ML} is obtained by imposing the symmetry and positive definiteness constraints explicitly, which can be found in [67]. The corresponding Σ_{ML} which maximizes equation (4.4.2) is then estimated as:

$$\Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{u}_{ML})(\mathbf{x}_i - \mathbf{u}_{ML})^T \tag{4.4.5}$$

After $\mathbf{u}_{ML}$ and Σ_{ML} are calculated, the single Gaussian model with the distribution $P(\mathbf{x})$ is determined, which is used to fit the sample set $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$. A large value of $P(\mathbf{x})$ means that the sample $\mathbf{x}$ is likely to come from the supporting region from which the data set $\mathbf{X}$ is sampled.

4.4.2 Mixture of Gaussians model

For a dataset from a non-Gaussian distribution, a single Gaussian distribution can not fit the dataset accurately. Instead, a mixture of Gaussians model as shown in [31] is exploited, which is a linear combination of Gaussian distributions and represented as:

$$P(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mathbf{u}_k, \Sigma_k) \tag{4.4.6}$$

Each Gaussian density $N(\mathbf{x}|\mathbf{u}_k, \Sigma_k)$ is called a component of the mixture and has its own mean $\mathbf{u}_k$ and covariance matrix Σ_k. And $\pi_k,\quad k = 1, ..., K$ are the mixing coefficients which satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$.

Given N samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$, as in the single Gaussian case, the corresponding parameters $\mathbf{u}_k$, Σ_k and π_k of the mixture of Gaussians model which fits the sample set can be estimated by maximizing the log of the likelihood function as:

$$\ln(P(\mathbf{X})) = \sum_{i=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k N(\mathbf{x}_i|\mathbf{u}_k, \Sigma_k)\} \tag{4.4.7}$$

As presented in [31], the derivatives of equation (4.4.7) with respect to $\mathbf{u}_k$, Σ_k and π_k are calculated (for the π_k case a Lagrangian term is introduced due to the constraint $\sum_{k=1}^{K} \pi_k = 1$) and set to zero, which can then yield:

$$\begin{aligned}
\mathbf{u}_k &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \mathbf{x}_i \\
\Sigma_k &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(\mathbf{x}_i - \mathbf{u}_k)(\mathbf{x}_i - \mathbf{u}_k)^T \\
\pi_k &= \frac{N_k}{N}
\end{aligned} \tag{4.4.8}$$

where $\gamma(z_{ik}) = \frac{\pi_k N(\mathbf{x}_i|\mathbf{u}_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}_i|\mathbf{u}_j,\Sigma_j)}$, which represents the posterior probability that component k was responsible for generating $\mathbf{x}_i$ and $N_k = \sum_{i=1}^{N} \gamma(z_{ik})$.

From equation (4.4.8), it can be observed that the solutions of $\mathbf{u}_k$, Σ_k and π_k are not in a closed form–the solutions are related to $\gamma(z_{ik})$ which needs to be calculated from $\mathbf{u}_k$, Σ_k and π_k. So, no direct solutions for $\mathbf{u}_k$, Σ_k and π_k can be obtained and instead, an expectation-maximization (EM) algorithm [31] is applied to obtain these parameters, which is divided into an E-step and M-step as follows:

1. Initialize the means $\mathbf{u}_k$, covariances Σ_k and mixing coefficients π_k (normally be the K-means algorithm as mentioned in [31]), and evaluate the initial value of the log likelihood.

2. E step. Evaluate the responsibilities $\gamma(z_{ik})$, using the current $\mathbf{u}_k$, Σ_k and π_k values as

$$\gamma(z_{ik}) = \frac{\pi_k N(\mathbf{x}_i|\mathbf{u}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}_i|\mathbf{u}_j, \Sigma_j)} \tag{4.4.9}$$

3. M step. Re-estimate the parameters by equation (4.4.8) using the responsibilities calculated from equation (4.4.9).

4. Evaluate the log likelihood

$$\ln(P(\mathbf{X})) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mathbf{u}_k, \Sigma_k)\} \tag{4.4.10}$$

and check for convergence of either the parameters or the log likelihood (whether the parameter values or the log likelihood value remain almost constant or not). If the convergence criterion is not satisfied return to Step 2.

By using the EM algorithm, the parameters $\mathbf{u}_k$, Σ_k and π_k are estimated for fitting a mixture of Gaussians distribution for N samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$.

Another important parameter which must be determined for the mixture of Gaussians model is the number of components–K, in this work the most popularly used Bayes Information Criterion (BIC) as mentioned in [68] is applied for its determination, which is:

$$BIC(k) = \ln(P_k(\mathbf{X})) - \lambda d \ln(N) \tag{4.4.11}$$

where $BIC(k)$ represents the BIC value obtained for the k-component model, $\ln(P_k(\mathbf{X}))$ represents the log likelihood value for the fitted k-component mixture of Gaussians model, d is the number of free parameters in the mixture model, N is the number of samples and λ is a control parameter to balance the fitting of the data and the complexity of the model. For determining K, different values are tested and the value which maximizes equation (4.4.11) is taken as the number of components for the mixture of Gaussians model. Overall, BIC ensures the constructed model is a reasonable fit to the data whilst the complexity is not too high so that good generalization performance can be guaranteed. Discussions of other model selection criteria can be found in [69].

Compared with the single Gaussian model, the mixture of Gaussians model can fit a non-Gaussian distributed dataset more precisely; however, the mixture of Gaussians model still assumes the data needed to be fit follows a mixture of Gaussians distribution, which is unrealistic in many practical cases. In the next sub-section, a more elegant OCSVM model is proposed for which there is no assumption for a particular distribution that the data should follow.

4.4.3 OCSVM model

The OCSVM is a more elegant data fitting method which is also described in [70], the basic idea behind OCSVM model is that given a data set $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ drawn from an underlying probability distribution $\mathbf{P}$, the corresponding supporting region can be obtained by estimating a function f. If a sample is obtained from the supporting region, both the distribution probability value and the function f value are large; otherwise, small values of distribution probability and f are obtained. Compared with the single Gaussian model or mixture of Gaussians model, OCSVM is more flexible because there is no assumption in OCSVM that the data needed to be fitted should follow particular types of distributions (single Gaussian or mixture of Gaussians).

As proposed in [70], the function f is in a linear form as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - \rho \tag{4.4.12}$$

for the data sample $\mathbf{x}$. And in most cases, the data $\mathbf{x}$ is mapped into a feature space with $\mathbf{x} \rightarrow \Phi(\mathbf{x})$ in order to obtain a better non-linear result for data fitting and equation (4.4.12) becomes:

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) - \rho \tag{4.4.13}$$

In order to obtain the parameters $\mathbf{w}$ and ρ, the following quadratic problem needs to be solved based on the dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$:

$$\begin{aligned} \min_{\mathbf{w},\xi,\rho} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_i \xi_i - \rho \\ \text{subject to} \quad & (w \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{4.4.14}$$

where $\nu \in (0, 1]$ and the nonzero slack variables $\xi = [\xi_1, ..., \xi_N]$ are introduced to allow for the possibility of outliers (the data points which are not drawn from the supporting region), the range of i is from 1 to N.

Using multipliers $\alpha_i, \beta_i \geq 0$, where $i = 1, ..., N$, a Lagrangian function is introduced as:

$$\begin{aligned} L(\mathbf{w}, \xi, \rho, \alpha, \beta) = & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + \frac{1}{\nu N}\sum_i \xi_i - \rho \\ & - \sum_i \alpha_i((\mathbf{w} \cdot \Phi(\mathbf{x})) - \rho + \xi_i) - \sum_i \beta_i \xi_i \end{aligned} \tag{4.4.15}$$

where $\alpha = [\alpha_1, ..., \alpha_N]$ and $\beta = [\beta_1, ..., \beta_N]$; the derivatives of the above Lagrangian function with respect to $\mathbf{w}$, ξ and ρ are set to zeros, which yields:

$$\begin{aligned} \mathbf{w} &= \sum_i \alpha_i \Phi(\mathbf{x}_i) \\ \alpha_i &= \frac{1}{\nu N} - \beta_i \leq \frac{1}{\nu N} \\ \sum_i \alpha_i &= 1 \end{aligned} \tag{4.4.16}$$

The results of (4.4.16) are substituted into (4.4.15), which are:

$$\begin{aligned}
L(\mathbf{w}, \xi, \rho, \alpha, \beta) &= \frac{1}{2} \parallel \mathbf{w} \parallel^2 + \frac{1}{\nu N} \sum_i \xi_i - \rho - \sum_i \alpha_i ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - \rho + \xi_i) - \sum_i \beta_i \xi_i \\
&= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) + \frac{1}{\nu N} \sum_i \xi_i - \rho - \sum_{ij} \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) + \rho \\
&- \frac{1}{\nu N} \sum_i \xi_i + \sum_i \beta_i \xi_i - \sum_i \beta_i \xi_i \\
&= -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \qquad (4.4.17)
\end{aligned}$$

As mentioned in [50], a dual form of problem (4.4.14) is obtained by maximizing (4.4.17) with respect to α considering the constraints of α in (4.4.16), which is:

$$\begin{aligned}
&\min_{\alpha} \qquad \frac{1}{2} \sum_{ij} \alpha_i \alpha_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \\
&\text{subject to } 0 \leq \alpha_i \leq \frac{1}{\nu N}, \quad \sum_i \alpha_i = 1 \qquad (4.4.18)
\end{aligned}$$

The problem (4.4.18) is a convex problem and can be solved by the standard algorithm for solving the convex problem as mentioned in [50]. Instead of representing $\Phi(\mathbf{x})$ explicitly, the kernel technique [31] is applied and a kernel function $k(\mathbf{x}, \mathbf{y})$ is used to represent the dot product of samples in the feature space as:

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) \qquad (4.4.19)$$

and (4.4.18) is rewritten as:

$$\begin{aligned}
&\min_{\alpha} \qquad \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\
&\text{subject to } 0 \leq \alpha_i \leq \frac{1}{\nu N}, \quad \sum_i \alpha_i = 1 \qquad (4.4.20)
\end{aligned}$$

By introducing the concept of kernel, it is convenient to obtain the dot product of mapped samples without knowing their exact forms explicitly and in this work, the popular Gaussian kernel is applied with:

$$k(\mathbf{x},\mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \tag{4.4.21}$$

where σ is the Gaussian kernel parameter.

As mentioned in [50], the solution of α in (4.4.20) (denoted as α^*) is related to the solution of $\mathbf{w}$ in (4.4.14) (denoted as $\mathbf{w}^*$) with:

$$\mathbf{w}^* = \sum_i \alpha_i^* \Phi(\mathbf{x}_i) \tag{4.4.22}$$

The solution of parameter ρ in (4.4.14) (denoted as ρ^*) can be estimated from the Karush-Kuhn-Tucker (KKT) conditions as mentioned in [50], from which the following equations hold:

$$\begin{aligned} \alpha_i^*((\mathbf{w}^* \cdot \Phi(\mathbf{x})) - \rho^* + \xi_i^*) = 0 \\ \beta_i^* \xi_i^* = 0 \end{aligned} \tag{4.4.23}$$

where ξ_i^* and β_i^* denote the i-th solutions of ξ and β for minimizing (4.4.14).

It can be observed from equation (4.4.23) that for a particular index i, if α_i^* and β_i^* are non-zero, the corresponding data sample $\mathbf{x}_i$ satisfies:

$$\rho^* = (\mathbf{w}^* \cdot \Phi(\mathbf{x}_i)) \tag{4.4.24}$$

and from equation (4.4.16), $\mathbf{w}^*$ is replaced with $\sum_i \alpha_i^* \Phi(\mathbf{x}_i)$, then

$$\begin{aligned} \rho^* &= \sum_i \alpha_i^* (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i)) \\ &= \sum_j \alpha_j^* k(\mathbf{x}_j, \mathbf{x}_i) \end{aligned} \tag{4.4.25}$$

Finally, after α^* and ρ^* are obtained, the decision function (4.4.13) is determined as:

$$f(\mathbf{x}) = \sum_j \alpha_j^* k(\mathbf{x}_j, \mathbf{x}) - \rho^* \tag{4.4.26}$$

which is then used to model the dataset $\mathbf{X}$, if the value of $f(\mathbf{x})$ for a data sample $\mathbf{x}$ is large, then the sample $\mathbf{x}$ is likely to come from the supporting region where most samples of $\mathbf{X}$ come from.

Fuzzy OCSVM model

In the real world application, it is evident that a perfect dataset $\mathbf{X}$ can not always be obtained, some samples would be outliers and less important than other 'good' samples in the design of the OCSVM model. In order to reflect the importance of different samples, each training data point is assigned with an associated fuzzy membership and the training dataset becomes: $(\mathbf{x}_1, u_1), ..., (\mathbf{x}_N, u_N)$, where N is the number of samples in the dataset. The fuzzy membership u_i which represents the likelihood of the corresponding point $\mathbf{x}_i$ belonging to the target class is calculated as proposed in [71]:

$$u_i = 1 - \frac{||\mathbf{x}_i - \mathbf{x}_{mean}||}{r_{target}} \tag{4.4.27}$$

where $\mathbf{x}_{mean}$ represents the mean of the data sample from $\mathbf{x}_1$ to $\mathbf{x}_N$ and $r_{target} = max_i||\mathbf{x}_i - \mathbf{x}_{mean}||$.

As proposed in [72], the constrained optimization problem of the fuzzy OCSVM is formulated as:

$$\begin{aligned} \min_{\mathbf{w},\ \mathbf{h},\ \rho} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + \frac{1}{\nu N} \sum_i u_i h_i - \rho \\ \text{subject to} \quad & (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - h_i, \quad h_i \geq 0 \end{aligned} \tag{4.4.28}$$

using a similar procedure as when solving the traditional OCSVM problem, the dual problem is obtained as:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{ij}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le u_i\frac{1}{\nu N}, \quad \sum_i \alpha_i = 1 \end{aligned} \tag{4.4.29}$$

from which the optimal value of α for the fuzzy OCSVM can be obtained and the way of estimating ρ^* is similar to the OCSVM case. In the end, the decision function of fuzzy OCSVM can be obtained with the exact same form as the OCSVM case in (4.4.26). Compared with the traditional OCSVM, the fuzzy OCSVM assigns 'good' training samples with high weights and outliers with low weights, which is more robust to noise and obtains a better result under a noisy dataset as presented in the experimental section.

4.4.4 Model-based fall detection

For the particular fall detection problem, initially multiple fall activities are simulated and 3D video features are then extracted from the video sequences of fall activities to form the dataset $\mathbf{X}$. Different models are constructed from the dataset $\mathbf{X}$ and the constructed models are then used to distinguish fall and non-fall activities. For a 3D video feature vector (denoted as $\mathbf{x}$) of a certain type of activity, if $P(\mathbf{x})$ (the probability value for the single Gaussian model or mixture of Gaussians model) or $f(\mathbf{x})$ (the value of the estimated function f of the OCSVM model) is high, then $\mathbf{x}$ is from the supporting region corresponding to $\mathbf{X}$ and the corresponding activity is regarded as a fall; otherwise, the activity is regarded as a non-fall activity. A threshold is set and the following formula shows the criteria of detecting falls:

$$Activity = \begin{cases} Fall & Model\ Value \ge threshold \\ NonFall & Model\ Value < threshold \end{cases} \tag{4.4.30}$$

where a *threshold* value is set and if the corresponding $Model\ Value$ ($P(\mathbf{x})$ or $f(\mathbf{x})$) is no less than the threshold for an extracted feature vector $\mathbf{x}$, then the corresponding activity from which the feature vector is extracted belongs to fall activity; otherwise, the activity belongs to non-fall activity.

In this work, different threshold values are chosen and tested for all the three models mentioned in this section for distinguishing fall and non-fall activities; the corresponding results are compared and presented below.

4.5 Experimental Analysis

The experiments are performed in the Intelligent lab, Loughborough University. Two Basler A312fc cameras and the Streampix 5 software were applied to capture the video sequences. And the recorded sequence was then processed by VC++ 6.0 (with OpenCV 1.0) and Matlab R2010. Figure 4.7 shows the room scene captured by two cameras.

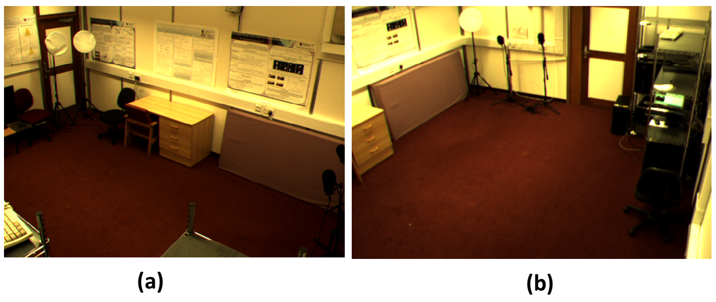


Figure 4.7. The room's scenes captured by two cameras. (a) the room scene captured by camera 1; (b) the room scene captured by camera 2

4.5.1 Camera calibration and 3D person construction

For the calibration of each camera, a large chessboard is printed and adhered to a plate, which is placed at a particular position in the room and captured by the camera needed to be calibrated. The corner points of the chessboard blocks are used for camera calibration, whose 3D world coordinates are convenient to calculate (the length and width of the chessboard plate are parallel to the world coordinate axes x_w and y_w) and the 2D image pixel coordinates are obtained manually from the image. The chessboard images captured by two cameras are presented in Figure 4.8 whose block corner points used for camera calibration are marked as red stars. Both cameras' parameters are estimated from the corresponding points set and the estimated parameters are shown in Tables 4.1 and 4.2.

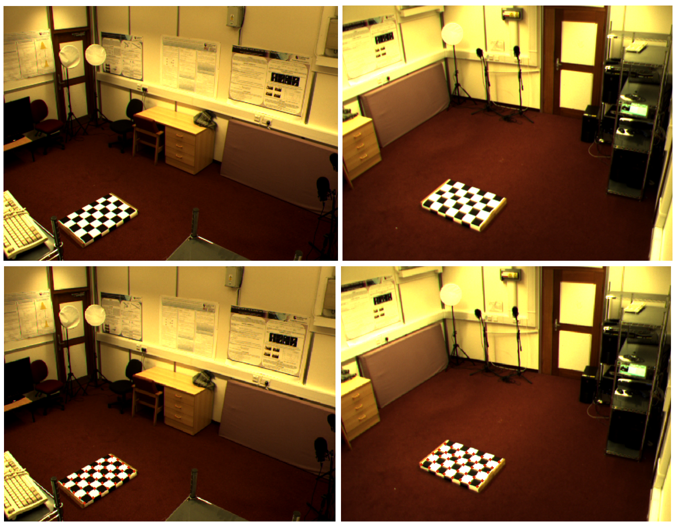


Figure 4.8. The chessboard plate used for camera calibration with the block corner points marked as red stars.

Table 4.1. Calibrated camera parameters for camera 1.

Parameters needed to be estimated	Calibration results
$\begin{bmatrix} r1 & r2 & r3 \\ r4 & r5 & r6 \\ r7 & r8 & r9 \end{bmatrix}$	$\begin{bmatrix} -0.7001 & -0.7105 & -0.0717 \\ -0.1772 & 0.2563 & -0.9502 \\ 0.6935 & -0.6525 & -0.3053 \end{bmatrix}$
$[T_x, T_y, T_z]$ (mm)	$[760.4500, 1122.9000, 5620.2000]$
$f(mm)$	7.5100
k	0.0033
s_x	0.9719

Table 4.2. Calibrated camera parameters for camera 2.

Parameters needed to be estimated	Calibration results
$\begin{bmatrix} r1 & r2 & r3 \\ r4 & r5 & r6 \\ r7 & r8 & r9 \end{bmatrix}$	$\begin{bmatrix} -0.4405 & 0.8959 & 0.0569 \\ 0.3853 & 0.1964 & -0.9016 \\ -0.8190 & -0.3753 & -0.4318 \end{bmatrix}$
$[T_x, T_y, T_z]$ (mm)	$[-1867.9000, 588.9082, 4325.1000]$
f (mm)	6.2373
k	0.0028
s_x	0.9462

The 3D person is constructed by dividing the room space into $94 \times 74 \times 50$ blocks along the x_w, y_w and z_w axes of the 3D world coordinate and each block has the size of $60mm \times 60mm \times 60mm$. Figure 4.9 shows the 3D person construction results for two different activities. Initially, codebook background subtraction is applied on both cameras to obtain the moving object, and the 3D person is then constructed from the background subtraction results and the estimated camera parameters. After 3D person construction, the 3D features are then extracted and the features corresponding to fall activities are applied to build the models for fall detection.

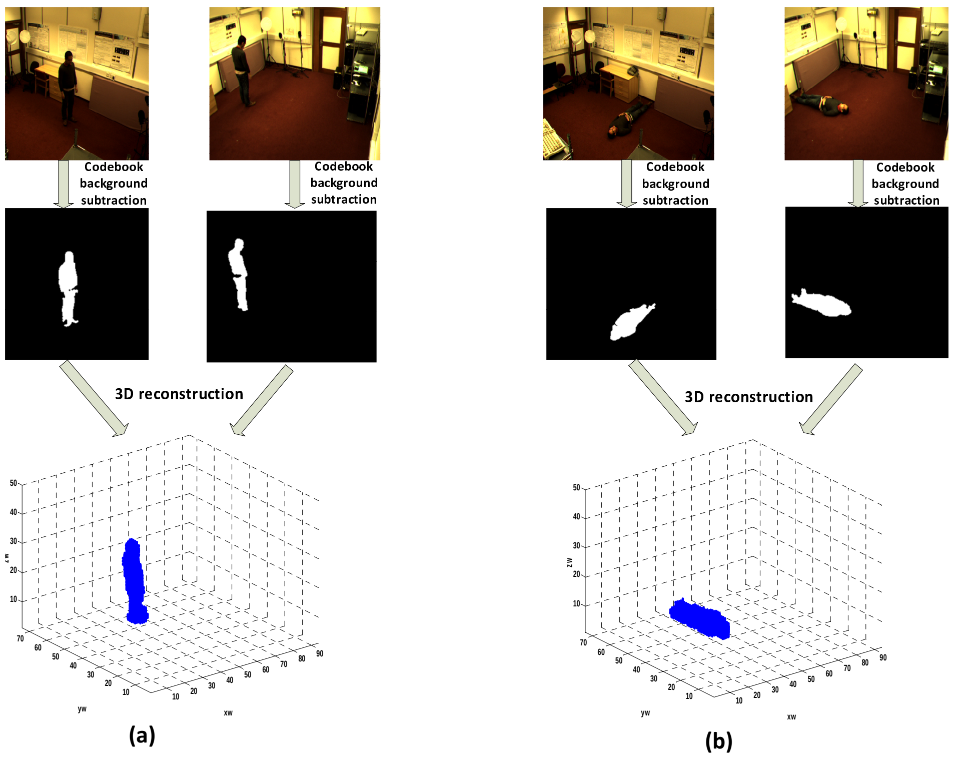

Codebook background subtraction
Codebook background subtraction
3D reconstruction
(a)
Codebook background subtraction
Codebook background subtraction
3D reconstruction
(b)
zw
yw
xw

Figure 4.9. 3D person construction by using background subtraction results from two cameras. (a) 3D construction result for standing (b) 3D construction result for lying

4.5.2 Model comparison results

In this part, the performances of three models–single Gaussian model, mixture of Gaussians model and OCSVM model are compared on a synthetic dataset and the real dataset of extracted 3D features.

Data fitting for a synthetic dataset

An 'X' shape dataset is generated as presented in Figure 4.10 (a) and three models are used to fit the dataset, the component number of the mixture of Gaussians model is set to 2 (which is obtained by maximizing the BIC criteria) and the parameters of the OCSVM model–ν and Gaussian kernel parameter σ are set to 0.01 and 0.2 respectively. The fitting results are presented in Figure 4.10 with the supporting region corresponding to this 'X' dataset is enclosed by black lines, which are obtained by the data description toolbox [73]. From the fitting results of (b), (c) and (d) in Figure 4.10, it can be observed that the estimated supporting region of OCSVM fits the 'X' dataset best.

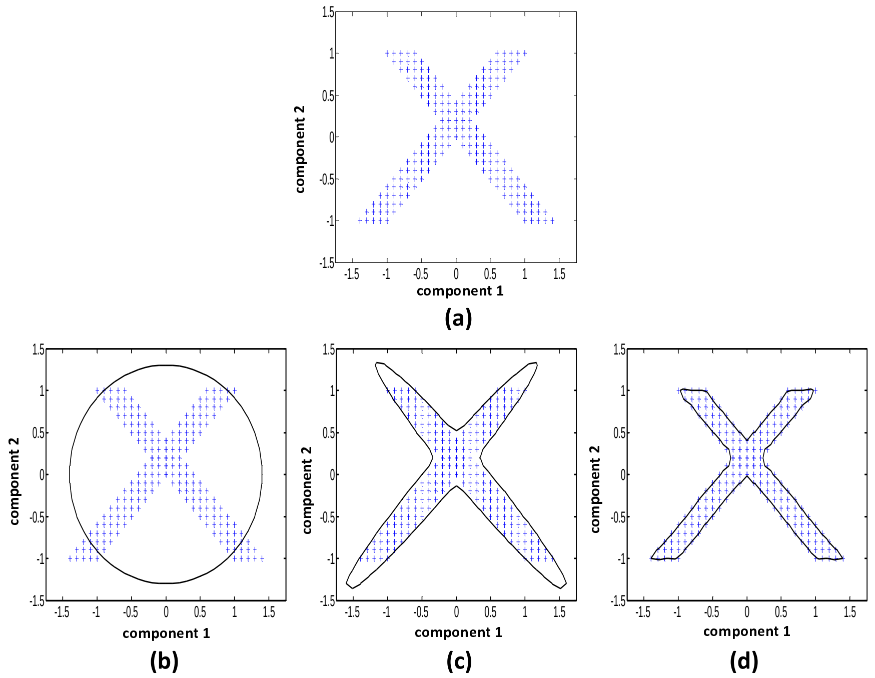


Figure 4.10. The comparisons of data fitting results for three models on an 'X' shape synthetic dataset: (a) fitting result of single Gaussian model (b) fitting result of mixture of Gaussians model (c) fitting result of OCSVM model

Fall detection results for a real dataset

To evaluate the performances of the three models for fall detection, two datasets are recorded. Eight people were invited to participate in the experiment and divided into two groups, each containing four people. One group simulates 40 fall activities (including 10 frontal falls, 10 backward falls and 20 side falls and each fall activity lasts for 1-2 seconds), which are used to compose a training dataset for model constructing. The other group simulates 40 different fall activities and 40 non-fall activities (including 8 walking actions, 8 rapid moving actions, 8 bending actions, 8 sitting actions and 8 lying actions), which are used for testing purpose. In this work, receiver operating characteristic (ROC) analysis [66] is applied for different models. Different thresholds are chosen and the true positive rate (TPR, which represents the percentage of falls which are correctly detected) and false negative rate (FNR, which represents the percentage of non-falls which are wrongly detected as falls) are calculated for these thresholds, and the results are plotted as a ROC curve. Ideally, for a perfect fall detection system, TPR should be 1 and FNR should be 0.

As proposed in [66], two criteria are used to evaluate the performance of each model, they are:

1. AUC value—which denotes the area under the ROC curve, a larger AUC value means a better performance of the corresponding model used for detecting falls.

2. optimal TPR and FNR pair under a particular threshold, which maximizes the geometric mean–$\sqrt{TPR * (1 - FPR)}$ whose range is $[0, 1]$, for a perfect system with unity TPR and FNR zero the corresponding value of geometric mean is unity.

Three models are constructed from the 3D feature vectors extracted from the training dataset (the component number of the mixture of Gaussians model is chosen as 2 from the BIC criteria, the parameters ν and σ of the

OCSVM are both set to 0.1), and the ROC curves results on the testing dataset are presented in Figure 4.11. The AUC value, optimal TPR and FNR values obtained from corresponding ROC curves are shown in Table 4.3. From Table 4.3, it can be observed that the single Gaussian model achieves the lowest AUC value indicating the worst performance among the three tested models for distinguishing fall and non-fall activities; and its optimal FNR is very large compared with that of the mixture of Gaussians model and OCSVM model, which means many non-fall activities are mistaken as fall activities. The mixture of Gaussian model achieves a higher AUC value than the single Gaussian model and the optimal FNR value is much lower than that of the single Gaussian model; however, its optimal TPR value is the lowest, which means the number of detected fall activities is the least. The OCSVM model achieves the best performance with a unity AUC value, and the optimal TPR and FNR values of OCSVM model are unity and zero respectively, all the fall activities are detected with no false detection.

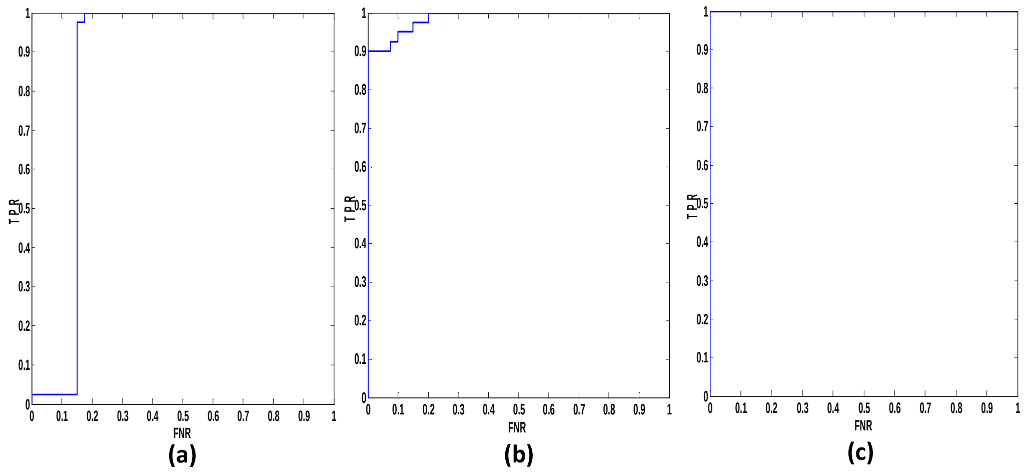


Figure 4.11. ROC curves for three models on the recorded dataset. (a) ROC curve for single Gaussian model (b) ROC curve for mixture of Gaussians model (c) ROC curve for OCSVM model

Table 4.3. Comparison results for three different models

	AUC value	optimal TPR	optimal FNR
single Gaussian model	0.8531	96.3%	15%
mixture of Gaussians model	0.9896	90%	1.3%
OCSVM model	1	100%	0%

4.5.3 Comparison of OCSVM and Fuzzy OCSVM

In the real case, a perfect training dataset can not be always obtained and there will inevitably be some outliers. In the particular fall detection problem, the outliers mainly come from:

1. Poor background subtraction results due to sudden light change and the similarity between the person's clothes and background, which will generate wrong feature vector values.

2. Wrong labelling, sometimes extracted features from non-fall activity will be labelled wrongly as fall activity features and used in the training dataset for model construction.

In this work, 10% outliers are introduced to make an imperfect training dataset. The corresponding OCSVM and fuzzy OCSVM models are then compared on the imperfect dataset (the parameters ν and σ are both set to 0.1). The comparison results are presented in Figure 4.12 and Table 4.4, from which it can be observed that compared with the OCSVM model, the AUC value of fuzzy OCSVM model is higher and its optimal FNR value is lower, which means that fuzzy OCSVM can achieve better performance with less non-fall activities being mistaken as fall activities under an imperfect training dataset.

Table 4.4. Comparison results for OCSVM and fuzzy OCSVM models based on a noisy dataset

	AUC value	optimal TPR	optimal FNR
OCSVM model	0.9825	100%	10%
fuzzy OCSVM model	0.9944	100%	5%

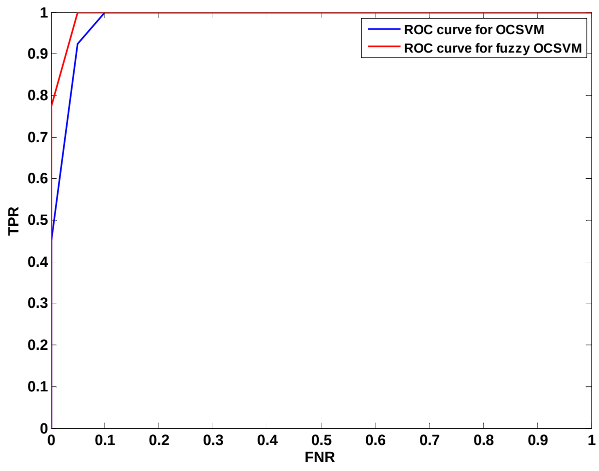


Figure 4.12. ROC curves for OCSVM and fuzzy OCSVM

4.6 Summary

This chapter proposed a fall detection scheme based on 3D features and a data fitting model scheme. Cameras were calibrated by the popular Tsai camera calibration method, a 3D person was then constructed from the obtained codebook background subtraction results from two cameras. 3D features, including the 3D position, velocity and orientation information corresponding to fall activities were extracted to build the model for distinguishing fall activities and non-fall activities. Three types of models–single Gaussian model, mixture of Gaussians model and OCSVM model were constructed and comparison results were presented in the experimental part, which showed that the OCSVM model achieved the best performance; besides, by introducing different weights to samples in the training dataset, a fuzzy OCSVM was obtained which achieved a better performance if the training dataset is contaminated by some outliers.

However, in order to obtain the 3D person construction results and extract the corresponding features more than one camera is used, which increases the economical costs; besides, each camera needs to be calibrated and the room dimension also needs to be measured beforehand, which causes inconvenience in the real application. In order to obtain a simple but effective fall detection system, a 2D postures recognition based fall detection scheme is next proposed, which uses only one camera and as compared with the scheme proposed in Chapter 3, this scheme can effectively distinguish fall activities and non-fall activities performed in different directions.

Chapter 5

TOWARDS A REAL HOME APPLICATION: SUPERVISED AND UNSUPERVISED FALL DETECTION SYSTEMS BASED ON POSTURE FEATURES

5.1 Introduction

In this chapter, effective fall detection systems based on posture features are proposed; more feature information and rules are utilized for fall confirmation and improved versions of corresponding classifiers are constructed for better classification purpose. The codebook background subtraction method is used to segment the human body posture, with some new post-processing techniques to obtain improved background subtraction results in a real home environment. Some types of features which can be used to describe the segmented human body posture are then extracted. Based on the obtained posture features, two types of fall detection systems are proposed. One is to use a supervised multi-class classifier, for which the features from various types of postures simulated by different persons are used to build a corresponding supervised classifier for posture classification. The other is to use an unsupervised classifier, for which the obtained features from one single person's normal postures are use to build a model and this model can then be used to distinguish normal from abnormal postures. The results of the multi-class classifier or model, together with some rules set according to the characteristics of fall activities, can then be used for detecting whether a fall happens or not. Experimental studies are used to assess the fall detection performance in a real home environment.

5.2 Background subtraction revisit

In Chapter 3, an efficient codebook background subtraction algorithm was applied to extract the moving object, and some post-processing techniques (including a blob operation and a morphological technique) were applied to obtain an improved background subtraction result. However, in a real home environment, there are three problems:

1. Sometimes the furniture (chair or table) in the home environment will be moved.

2. In some situations the person is static in the room for a long period of time (such as sitting on a chair) and the static human body region will be absorbed into the background.

3. The light condition will change dramatically when the light is turned on/off.

These three problems will definitely generate large background subtraction errors, which can not be solved by the basic post-processing technique as mentioned in Chapter 3. In order to obtain good background subtraction results when these problems occur, some other post-processing techniques are applied.

5.2.1 Post-processing technique for selective updating

In this work, a three step blob operation strategy is adopted after the codebook background subtraction procedure in order to remove the errors caused by the movement of furniture and the long static period for an elderly person, which is divided into:

Step1. Blob merging: firstly, a blob merging operation is applied on the original background subtraction result. If the distance between two blobs is less than a threshold, these two blobs will be merged (as shown in Figure 5.1 (d), the blobs B2 and B3 contain several separate blobs which are near to each other respectively). The distance between two blobs is defined as the minimum 4-distance [59] between two rectangles which enclose the blobs as given by:

$$Distance(B1, B2) = min_{p1 \in R1, p2 \in R2} d_4(p1, p2) \qquad (5.2.1)$$

where $B1$ and $B2$ are two blobs, $R1$ and $R2$ are two rectangles which enclose them, and $p1$ and $p2$ are points belonging to $R1$ and $R2$. Figure 5.2 shows examples of the distance between two blobs with respect to their positions.

Step2. Human body blob determination: Small blobs after blob merging are removed by the post-processing technique as mentioned in Chapter 3. According to the number of remaining blobs and assuming that the elderly person lives alone, so that normally there should be only one human moving object, giving three cases:

case 1: the number of remaining blobs is zero, which means that no large foreground object (human body) is in the scene and the room is empty.

case 2: the number of remaining blobs is one, which indicates that the blob represents the human body region.

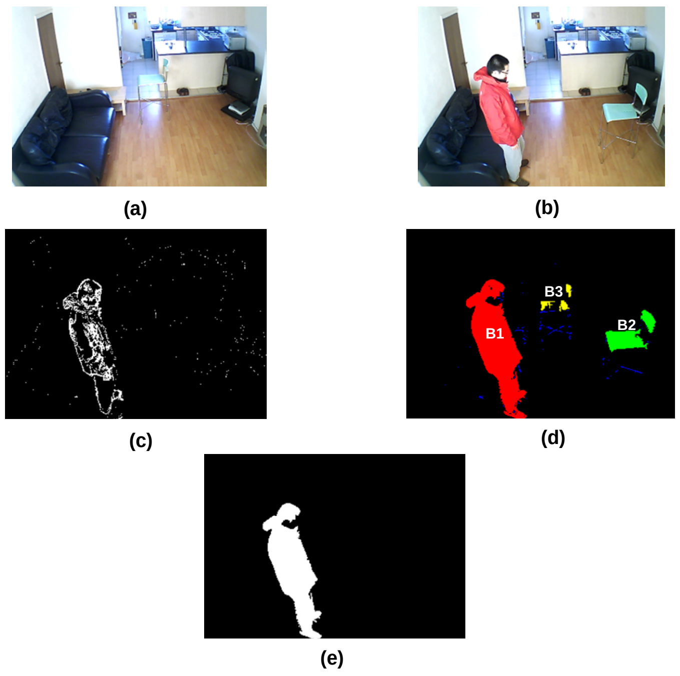


Figure 5.1. The background subtraction and the human body blob determination. a) Background image; b) Image with object; c) Frame difference result obtained from two consecutive frames; d) Original background subtraction result, there are three large blobs (B1, B2 and B3) after the blob merging operation and they are marked red, green and yellow, and the blue colour represents the small noise-like blobs; e) The final obtained human body blob.

case 3: the number of blobs after blob merging is more than one, which suggests some other regions (such as the chair regions at the new and previous positions as shown in Figure 5.1 (d)) are mistaken as foreground. In this case, the human body blob is determined by using the frame difference technique as:

$$D_t(x, y) = |I_t(x, y) - I_{t-1}(x, y)| \tag{5.2.2}$$

where $I_t(x, y)$ and $I_{t-1}(x, y)$ are the gray level pixel values of consecutive frames I_t and I_{t-1} at position (x, y), $|\cdot|$ denotes the operation to obtain the absolute value and $D_t(x, y)$ is the frame differencing result for the position (x, y) at time t.

Frame differencing is operated to obtain the moving pixels (shown in Figure 5.1 (c)) and the blob with the greatest number of moving pixels is taken as the human body blob. From Figure 5.1, it can be seen that the blob B1 contains the most moving pixels and so B1 is finally taken as the human body blobs.

Step3. Selective updating: The non-human body blobs are removed (as shown in Figure 5.1 (e), B2 and B3 are removed from the final background subtraction result) and their pixel values form new codewords to be added to the background codebook immediately for background model updating. And no updating is performed for pixels in the human body blob.

In this way, the errors generated by the movement of furniture are absorbed into the background model immediately, which obtains a better background subtraction result. Besides, the foreground human body object is not absorbed into the background even though he/she is static for a long time.

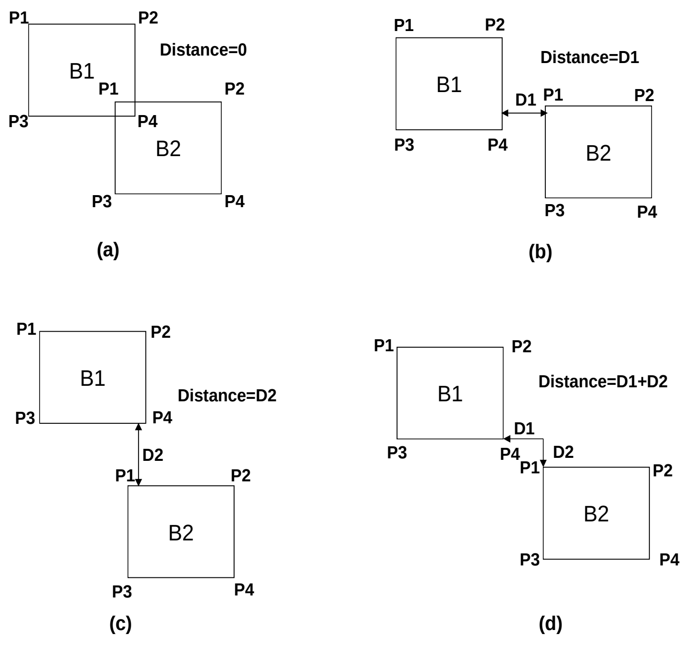


Figure 5.2. Four cases of the distance between two blobs with respect to their relative positions

5.2.2 Background model retraining for the sudden illumination change

The trained background codebook model can be affected in various of ways, such as dramatic global illumination change due to suddenly turning on/off the light. In this situation, the codebook needs to be re-trained because the previous codebook is no longer available. The dramatic global illumination change can be detected by frame differencing results, if the percent of the active pixels in an image is larger than a threshold (50% is set), then the dramatic global illumination change is regarded to occur and the background model is retrained.

By using selective updating and background model retraining, the practical problems (movement of the furniture, long time static for the elderly person and sudden light change) existing in the real home environment can be solved to obtain a better human body region extraction result, which is used for the next step in the posture feature extraction.

5.3 Features used for posture description

The extracted human body postures can be described in details by certain types of features. These can then be fed into some supervised or unsupervised classifier for classification. In this work, two types of such features are tested, they are projection histogram features and shape-structure features.

5.3.1 Projection histogram features

A widely used feature to describe the posture's detail information is the projection histogram, as mentioned in [32], [33] and [42]. This feature is computationally efficient so that it can be applied in a real time application, while achieving a good performance for posture classification purpose of the supervised classifier. In this work, the projection directions of the corresponding projection histogram are along the major and minor axes of the fitted ellipse as presented in Chapter 3. One example is as in Figure 5.3 where the projection histograms of the long and short axes of the fitted ellipses are obtained for different types of postures. From the results, it can be observed that there are differences in the patterns within the histograms between different postures, which are helpful for posture classification.

The numbers of bins of the major axis projection and minor axis projection histograms are all set to 30 in this work, a value found empirically to provide suitable detail whilst not introducing undue complexity. For particular bins of the projection histograms along the ellipse's major and minor axes, their values are calculated as in (5.3.1) and (5.3.2).

$$bin_{major}(i) = \frac{NoFP_{major}(i)}{L_{major}} \tag{5.3.1}$$

$$bin_{minor}(i) = \frac{NoFP_{minor}(i)}{L_{minor}} \tag{5.3.2}$$

where i is the index of bins, $NoFP_{major}(i)$ and $NoFP_{minor}(i)$ denote the number of foreground pixels along the ith-projection line in the directions of the major and minor axes respectively. The results are normalized by L_{major} and L_{minor}, which represent the length of the major and minor axes. The purpose of the normalization is to make sure this feature is invariant to the size of the foreground human body region (which will vary according to a person's distance to the camera).

The projection histogram features, together with two features obtained from the fitted ellipse result: the ratio between the major and minor axes rho and the orientation angle $theta$ can be applied for posture classification by a supervised classifier and the results are presented in section 5.7.

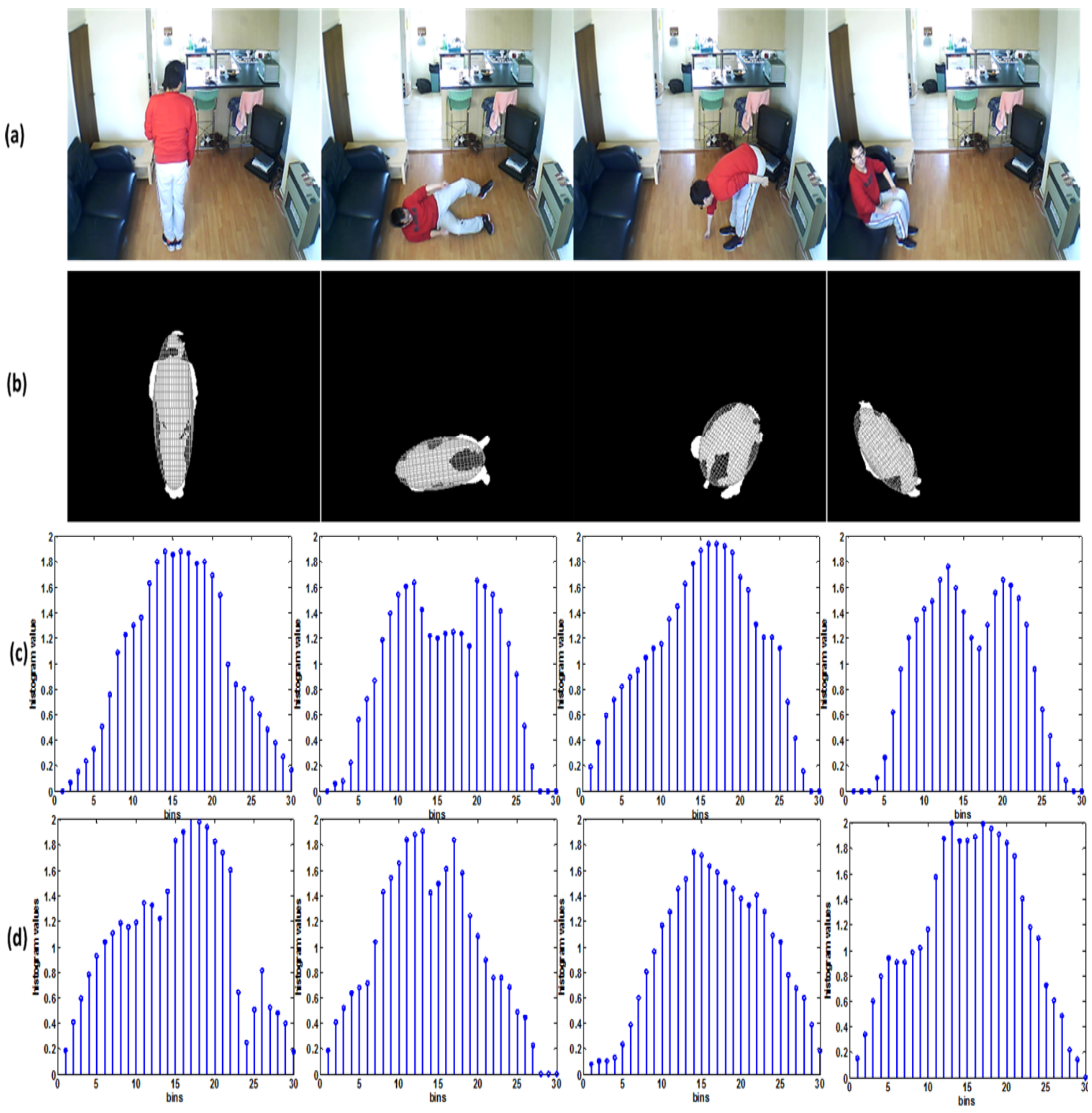


Figure 5.3. Projection histograms of four different types of postures. (a) original frames (b) Background subtraction results with fitted ellipses and projection lines (c) Projection histograms along the major axis of the ellipse (d). Projection histograms along the minor axis of the ellipse. The horizontal axis of the projection histogram represents the index of bins and the vertical axis represents the value of the projection histogram.

5.3.2 Shape-structure features

Another type of feature describes the posture's detail information by analyzing the posture's shape and structure, which can be extracted with the method as proposed in [74]. Initially, the perimeter contour of a human body posture is extracted by some contour detection method [59], which is represented as a $contourlist = [point_1, ..., point_N]$ where $point_i$ is a particular point on the perimeter contour. One example of an extracted perimeter contour is shown in Figure 5.4 (c).

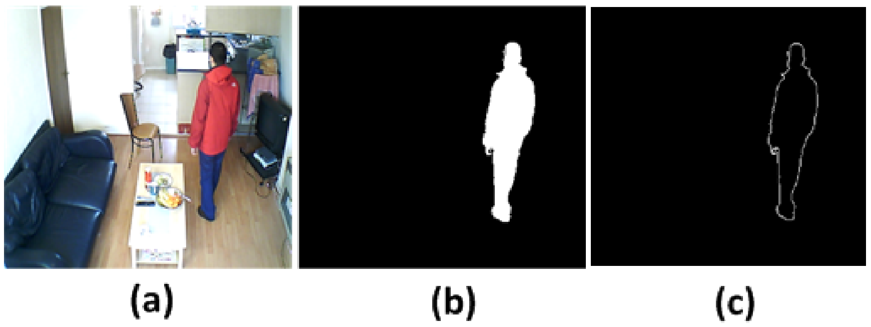


Figure 5.4. The result of the perimeter contour detection, (a) original image (b) background subtraction result (c) perimeter contour detection result.

Not all the points in the *contourlist* are necessary to represent a boundary, only the points with high curvature are the 'key points' which determine a boundary shape and these points form a more concise perimeter contour representation. As presented in Figure 5.5, the shape of the star can be totally determined by the high curvature points.

For a two-dimensional point p, its angle can be calculated by:

$$angle(p) = \arccos \frac{\| p - p^+ \|^2 + \| p - p^- \|^2 - \| p^+ - p^- \|^2}{2 \| p - p^- \| \times \| p - p^+ \|} \tag{5.3.3}$$

where p^+ and p^- are selected from both sides of p along the boundary and satisfy $d_{min} \leq \| p - p^+ \| \leq d_{max}$ and $d_{min} \leq \| p - p^- \| \leq d_{max}$, where d_{max} and d_{min} are chosen thresholds (which are taken as 1/20 and 1/30 of the boundary perimeter respectively from the empirical study). If $angle(p)$ is

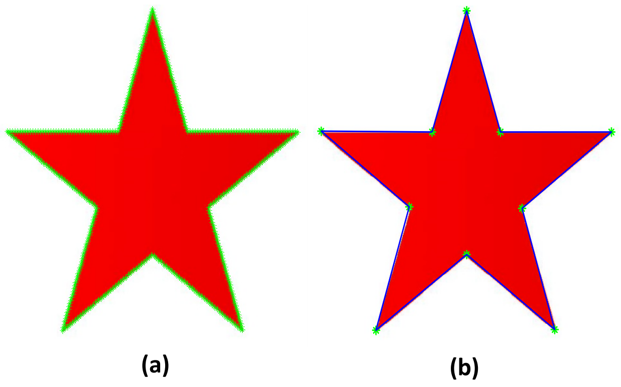


Figure 5.5. The representation of key points: (a) shows the contour points which are marked in green. Only the points with high curvature are retained in (b), from which it shows these high curvature points ('key points') are enough to determine the perimeter contour (the connection of these 'key points' forms the boundary, which is marked in blue.)

low then it means the curative of this point is high and this point is taken as a 'key point'. For two 'key points', their distance should be larger than a proper threshold (chosen the same as d_{min} in this work) so that if several points with low $angle(p)$ are near to each other, only one representation point is chosen as the 'key point'.

The result of the extracted 'key points' set (denoted as V) can be further applied to extract the skeleton structure of a posture from the constrained Delaunay triangulation technique. Assuming two adjacent 'key points' in the set V are connected by an edge, a polygon is then formed with the vertices being the 'key points' in V. If the set of interior points of V is denoted as Φ, for a triangulation $T \subset \Phi$ with three vertices v_i, v_j and v_k from the set V, T is said to be a constrained Delaunay triangle if:

(i) $v_k \in U_{ij}$, where $U_{ij} = \{v \in V | e(v_i, v) \subset \Phi, e(v_j, v) \subset \Phi\}$ and $e(a, b)$ denotes the line segment by connecting points a and b.

(ii) $C(v_i, v_j, v_k) \cap U_{ij} = \emptyset$, where C is a circum-circle of v_i, v_j and v_k and $\emptyset$ denotes the empty set. That is, the interior of $C(v_i, v_j, v_k)$ does not have a vertex $v \in U_{ij}$.

Based on the two conditions, an algorithm as proposed in [75] is applied for obtaining the constrained Delaunay triangulation result for the set V, the procedure of this algorithm is presented in Table 5.1.

Table 5.1. The procedure for obtaining the constrained Delaunay triangulation result.

1. Choose a starting edge $e(v_i, v_j)$ from the polygon formed by V.
2. Find the third vertex v_k of V that satisfies properties (i) and (ii).
3. Subdivide V into two sub-polygons: $V_a = \{v_i, v_k, v_{k+1}, ..., v_{i-1}, v_i\}$ and $V_b = \{v_j, v_{j+1}, ..., v_k, v_j\}$
4. Repeat Steps 1–3 on V_a and V_b until the processed polygon consists of only one triangle.

The skeleton can then be extracted from the constrained Delaunay triangulation result by connecting the centroid of neighboring triangulations. An example of the skeleton structure extraction is presented in Figure 5.6. The constrained Delaunay triangulation result for a standing posture is obtained and the corresponding skeleton is then extracted by connecting the neighboring triangulations' centroid.

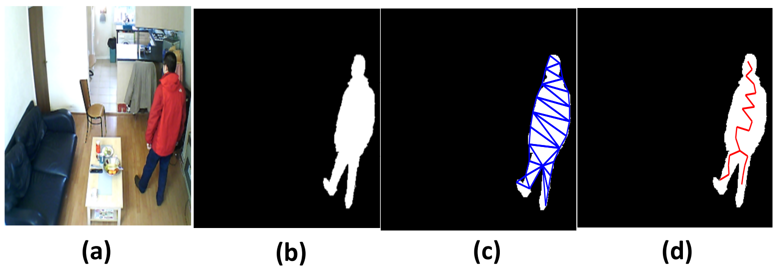


Figure 5.6. The extraction of the posture of a human body skeleton, (a) the original image (b) background subtraction result (c) the result of the constrained Delaunay triangulation and (d) the extracted skeleton by connecting the triangular centroid.

The centroid of the constrained Delaunay triangles which determine a posture's skeleton, together with the 'key points' set on the boundary can be used as information to describe the posture. As in [74], an accurate and efficient single centroid context shape descriptor is applied to 'summarize' the information to obtain the corresponding shape-structure features. Initially, the center of gravity of the whole human body posture is calculated; then fixing this centroid point as the origin, a polar coordinate system is constructed, which is equally divided into m shells and n sectors to form $m \times n$ bins (m and n were chosen as 8 and 30 respectively on the basis of empirical study), as seen in Figure 5.7 (d). An $m \times n$ histogram is then constructed to obtain the spatial distribution of points. Different from [74], in this work two modifications on the histogram construction procedure are made:

(1) Not only are the triangular centroid points used for histogram construction, but the 'key points' are also applied so that the constructed histogram can reflect both the skeleton structure and perimeter contour information.

(2) Instead of simply calculating the number of points in each bin for histogram construction, an improved strategy is applied to get a more accurate histogram result: for a point (either the 'key point' on the perimeter contour or the triangle's centroid), if it is in the kth bin, then histogram values are updated as:

$$h_{new}(k) = h_{old}(k) + 1 \tag{5.3.4}$$

$$h_{new}(k_{neighbors}) = h_{old}(k_{neighbors}) + 0.5 \tag{5.3.5}$$

where $h_{new}(k)$ and $h_{new}(k_{neighbors})$ represent respectively new histogram values of the kth bin and its neighbors after updating, and $h_{old}(k)$ and $h_{old}(k_{neighbors})$ represent old values; initially the histogram values are set to zero.

The obtained histogram is finally normalized to make sure the summation of the values of all bins is unity and the normalized histogram is obtained as the shape-structure feature which gives a detailed posture description. Figure 5.7 (e) shows the corresponding histograms for four postures of standing, sitting, bending and lying. The shape-structure feature, the ratio between the major and minor axes *rho* and the orientation angle *theta* obtained from the ellipse fitting results as shown in Chapter 3 and the centroid of the human body posture (which provides the position information) are together used for building an unsupervised model used for distinguishing fall activities and non-fall activities.

5.4 Supervised and unsupervised learning methods

The obtained features for describing postures can then be fed into the supervised classifier for the detection of different postures or a unsupervised model to distinguish normal and abnormal postures. In this section, some supervised and unsupervised learning methods are introduced.

5.4.1 Support vector machine based supervised classifier

Two class support vector machine

A support vector machine (SVM) is a recently emerging classification technique. As proposed in [76], an SVM is based on statistical learning theory and it has good generalization performance compared with the traditional classification methods, such as the nearest neighbour method and neural network based techniques. For a two class classification problem with the training dataset:$\{\mathbf{x}_i, y_i\}, y_i \in \{-1, 1\}, \mathbf{x}_i \in \mathbf{R}^d$, a hyperplane $h(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$ in a particular feature space is estimated to separate these two classes while making the margin (the smallest distance between the hyperplane and any

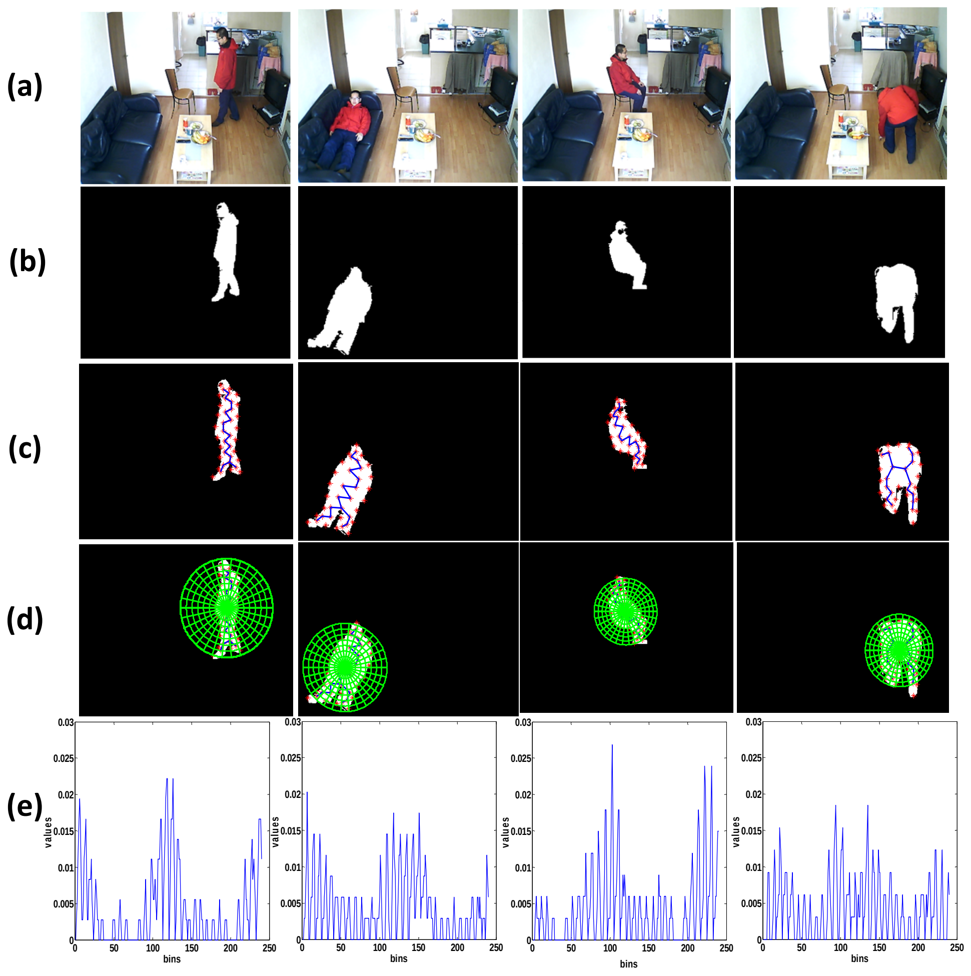


Figure 5.7. Histograms for four postures obtained from centroid context descriptor. (a) the original images for four postures (stand, lie, sit and bend) (b) the background subtraction results with postprocessing (c) the extracted skeleton (marked as blue) and points (including the 'key points' on the contour and triangular centroid, marked as red) (d) the polar coordinate system, which is composed of 8 shells and 30 sectors, total 240 bins (e) the finally obtained histograms, the horizontal axis represents the indices of bins and the vertical axis represents the values of the bins.

of the samples) maximum, as presented in Figure 5.8.

In order to obtain the hyperplane, the following quadratic problem needs

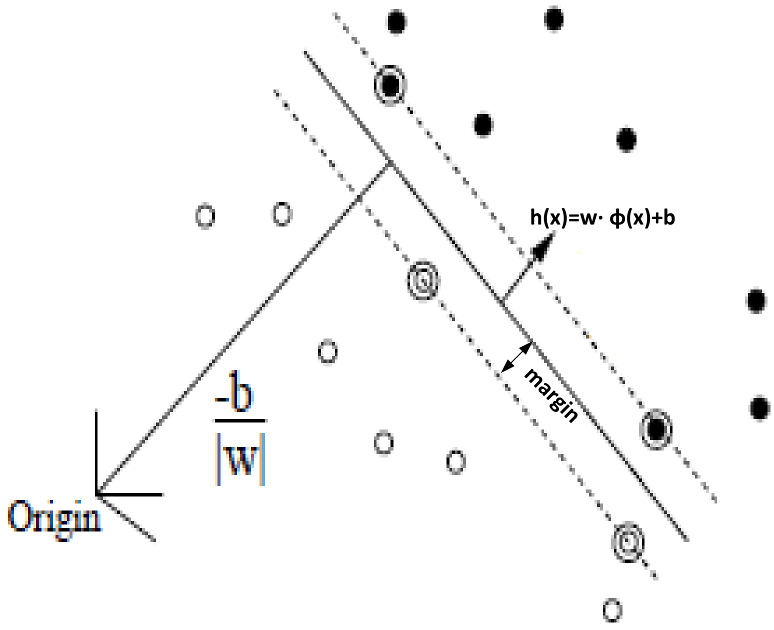


Figure 5.8. The illustration of a hyperplane to separate samples from two classes (white and black) in a particular feature space.

to be solved to obtain the hyperplane's parameters **w** and b:

$$\begin{aligned} &\min_{\mathbf{w},\xi,b} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ &\text{subject to} && y_i(\mathbf{w}\cdot\Phi(\mathbf{x}_i) - b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{5.4.1}$$

where **w** and b are the parameters determining a hyperplane, $\xi = [\xi_1, ..., \xi_N]$ are slack variables to cope with noise, C is called a penalized parameter which balances the noises and the margin and $\Phi(\cdot)$ is a mapping operation which maps the original sample into a feature space for better separation purpose as mentioned in [76] and [31].

Similar to the case of OCSVM as mentioned in Chapter 4, a Lagrangian function L is obtained by introducing multipliers $\alpha = [\alpha_1, ..., \alpha_N], \beta =$

$[\beta_1, ..., \beta_N]$ as:

$$L(\mathbf{w}, \xi, b, \alpha, \beta) = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_i \beta_i \xi_i \tag{5.4.2}$$

The derivatives of the above Lagrangian function with respect to $\mathbf{w}$, ξ and b are set to zeros, which yields:

$$\begin{aligned} \mathbf{w} &= \sum_i \alpha_i y_i \Phi(\mathbf{x}_i) \\ \alpha_i &= C - \beta_i \\ \sum_i \alpha_i y_i &= 0 \end{aligned} \tag{5.4.3}$$

From the results of (5.4.3), the Lagrangian function L in (5.4.2) then becomes:

$$\begin{aligned} L(\mathbf{w}, \xi, b, \alpha, \beta) &= \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_i \beta_i \xi_i \\ &= \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) + C \sum_i \xi_i - \sum_{ij} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \\ &- b \sum_i \alpha_i y_i + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \beta_i \xi_i \\ &= -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) + \sum_i \alpha_i \end{aligned} \tag{5.4.4}$$

According to [50], a dual form of the problem (5.4.1) is obtained by maximizing (5.4.4) with respect to α considering the constraints of α in (5.4.3), which is:

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) + \sum_i \alpha_i \\ & \text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0 \end{aligned} \tag{5.4.5}$$

which corresponds to the OCSVM case in Chapter 4; a Gaussian kernel function $k(\mathbf{x},\mathbf{y})$ is applied to replace the dot product of samples in the feature space $(\Phi(\mathbf{x})\cdot\Phi(\mathbf{y}))$, thus (5.4.5) is rewritten as:

$$\begin{aligned} \max_{\alpha} \qquad & -\frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)+\sum_i \alpha_i \\ & \text{subject to } \; 0\le \alpha_i \le \; C, \quad \sum_i y_i\alpha_i = 0 \end{aligned} \tag{5.4.6}$$

and the Gaussian kernel is applied in this work.

From the relationship of $\mathbf{w}$ and α as mentioned in (5.4.3), the solution of α in (5.4.6) (denoted as α^*) is related to the solution of $\mathbf{w}$ in (5.4.1) (denoted as $\mathbf{w}^*$) with:

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \Phi(\mathbf{x}_i) \tag{5.4.7}$$

The solution of parameter b in (5.4.1) (denoted as b^*) can be estimated from the Karush-Kuhn-Tucker (KKT) conditions as mentioned in [50], from which it follows that:

$$\begin{aligned} \alpha_i^*(y_i(\mathbf{w}^*\cdot\Phi(\mathbf{x}_i)+b^*)-1+\xi_i^*) &= 0 \\ \beta_i^*\xi_i^* &= 0 \end{aligned} \tag{5.4.8}$$

where ξ_i^* and β_i^* denote the i-th optimal solutions of ξ and β.

If $\alpha_i^* \neq 0$ and $\beta_i^* \neq 0$, then from the KKT conditions in (5.4.8):

$$b^* = y_i - \mathbf{w}^*\cdot\Phi(\mathbf{x}_i) \tag{5.4.9}$$

and if $\mathbf{w}^*$ is replaced with $\sum_i \alpha_i^* y_i \Phi(\mathbf{x}_i)$ by (5.4.7), then

$$\begin{aligned} b^* &= y_i - \sum_j \alpha_j^* y_j (\Phi(\mathbf{x}_j)\cdot\Phi(\mathbf{x}_i)) \\ &= y_i - \sum_j \alpha_j^* y_j k(\mathbf{x}_j,\mathbf{x}_i) \end{aligned} \tag{5.4.10}$$

Finally, after α^* and b^* are obtained, the hyperplane can be determined as:

$$h(\mathbf{x}) = \sum_i \alpha_i^* y_i k(\mathbf{x}_i, \mathbf{x}) + b^* \qquad (5.4.11)$$

the value of $h(\mathbf{x})$ is then used to determine to which class the sample $\mathbf{x}$ belongs. For an incoming test sample $\mathbf{x}$, if $h(\mathbf{x}) \geq 0$, then the corresponding symbol y is then determined as $+1$ and $\mathbf{x}$ belongs to the '+1' class; or else, the symbol value is -1 and $\mathbf{x}$ belongs to the '-1' class.

Directed acyclic graph support vector machine (DAGSVM) for multi-class classification

The traditional two-class SVM can only solve the two-class separation problem and several schemes have been applied to solve the multi-class classification problem, the representative ones are one-versus-one (1-v-1) [77] and one-versus-rest (1-v-r) [76]. Compared with these two methods, Platt et al. [1] proposed a concept of directed acyclic graph support vector machine (DAGSVM), which has a theoretically defined generalization error bound and it is more efficient than 1-v-1 and 1-v-r schemes with respect to the training and computation time.

As mentioned in [1], the DAGSVM is equivalent to operating on a list, which is initialized with all classes. For an incoming sample $\mathbf{x}$, it is evaluated by the two-class support vector machine corresponding to the first and last class element. After the evaluation by the two-class support vector machine, the sample $\mathbf{x}$ is determined to be one of the two classes and the class element that $\mathbf{x}$ does not belong to will be eliminated from the list. This procedure is repeated until only one class element remains in the list and this class element is taken as the class to which $\mathbf{x}$ belongs. In this way, it can be seen that for a problem with N classes, N-1 decisions will be evaluated in order to derive an answer.

An example of the decision procedure of the DAGSVM for a four class classification problem is shown in Figure 5.9, which presents a tree-like structure and each node in this structure corresponds to a two-class SVM. The decision process just follows the structure and is based on a sequence of two-class operations, a decision is made when a bottom node is reached.

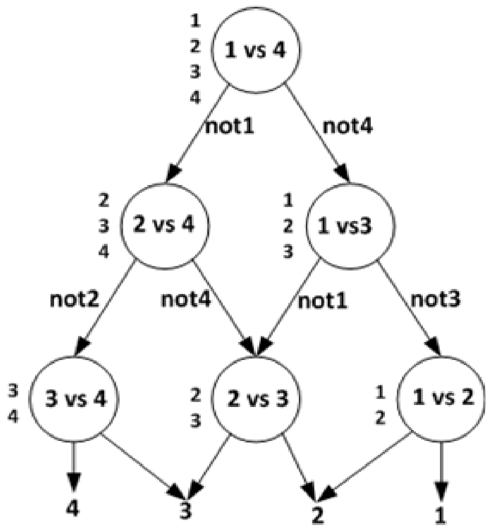


Figure 5.9. The decision process for the traditional DAGSVM for a four class problem [1]

As for all other classifiers, an indispensable step for training the DAGSVM classifier is to determine optimal parameters so that the classifier can achieve the optimal performance. There are two sets of parameters needed to be tuned for the DAGSVM classifier:

1) The list sequence for the DAGSVM, which is equivalent to the sequence of different two-class support vector machines for making a decision as shown in Figure 5.9; the list sequence is related to the performance of the DAGSVM and a proper list sequence is needed to guarantee good performance.

2) The parameters of each two-class SVM node, including the kernel parameters (Gaussian kernel is used here for non-linear classification) and the penalty parameter in the two-class SVM scheme.

Traditional cross-validation [31] can be used to find the optimal parameters, but it is time consuming when the number of parameters needed to be tuned is large. In this work, a new parameter optimization scheme is proposed to reduce the training time to a large extent.

For tuning the kernel parameters, the concept of distance between two classes (DBTC) is exploited, as shown in [78]. The calculation of DBTC in a feature space with n_1 and n_2 samples for two classes is defined as follows:

$$\begin{aligned}\mathrm{DBTC} &= \|\mathbf{m}_1^{\Phi} - \mathbf{m}_2^{\Phi}\|^2 \\ &= (\mathbf{m}_1^{\Phi} - \mathbf{m}_2^{\Phi})^T(\mathbf{m}_1^{\Phi} - \mathbf{m}_2^{\Phi}) \\ &= \frac{1}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1} k(\mathbf{x}_{1,i}, \mathbf{x}_{1,j}) + \frac{1}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2} k(\mathbf{x}_{2,i}, \mathbf{x}_{2,j}) \\ &- \frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k(\mathbf{x}_{1,i}, \mathbf{x}_{2,j})\end{aligned} \tag{5.4.12}$$

where $k(\mathbf{x}, \mathbf{y})$ is the kernel function mentioned previously and $\mathbf{m}_1^{\Phi}$, $\mathbf{m}_2^{\Phi}$ are the means of the two classes in the feature space: $\mathbf{m}_1^{\Phi} = \frac{1}{n_1}\sum_{i=1}^{n_1}\Phi(\mathbf{x}_{1,i})$ and $\mathbf{m}_2^{\Phi} = \frac{1}{n_2}\sum_{i=1}^{n_2}\Phi(\mathbf{x}_{2,i})$.

For the Gaussian kernel used in this work, the following equation holds:

$$\begin{aligned}\mathrm{DBTC} = (2 &- \frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k(\mathbf{x}_{1,i}, \mathbf{x}_{2,j})) \\ &- \frac{1}{2}(2 - \frac{2}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1} k(\mathbf{x}_{1,i}, \mathbf{x}_{1,j})) \\ &- \frac{1}{2}(2 - \frac{2}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2} k(\mathbf{x}_{2,i}, \mathbf{x}_{2,j})) \\ &= \frac{d(C_1, C_2)}{n_1 n_2} - \frac{d(C_1, C_1)}{2n_1^2} - \frac{d(C_2, C_2)}{2n_2^2}\end{aligned} \tag{5.4.13}$$

where $d(C_i, C_j)$ is calculated as: $d(C_i, C_j) = \sum_{\mathbf{x}\in C_i}\sum_{\mathbf{y}\in C_j}\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|^2$, which measure the distance between two classes.

From Eq. (5.4.13), it can be observed that a large DBTC value means a large inter-distance value $d(C_i, C_j)$ and small intra-distance values $d(C_i, C_i)$ and $d(C_j, C_j)$, which means the two classes have high separation level.

For each two class SVM, the optimal Gaussian kernel parameter can then be obtained by maximizing the DBTC value between the corresponding two classes, which reduces the training time because compared with the traditional cross-validation method [67] to tune this parameter, the time consuming two-class SVM training procedure is avoided. The cross-validation method is still applied to tune the penalty parameter of the two-class SVM.

Assuming that the parameters for every two-class SVM node have already been tuned, the two-class SVMs are trained and the DBTCs are calculated under the tuned parameters to obtain two lists:

$$\begin{aligned} DBTClist &= DBTC_{1,1}, .., DBTC_{1,n}, .., DBTC_{i,i+1}, .., DBTC_{i,n}, .., DBTC_{n-1,n} \\ SVMlist &= SVM_{1,1}, .., SVM_{1,n}, .., SVM_{i,i+1}, .., SVM_{i,n}, .., SVM_{n-1,n} \end{aligned} \tag{5.4.14}$$

For an incoming sample, the sequence of different two-class support vector machines for making a decision is determined by $DBTClist$, which is summarized in Table 5.2 and a four class example is presented in Figure 5.10.

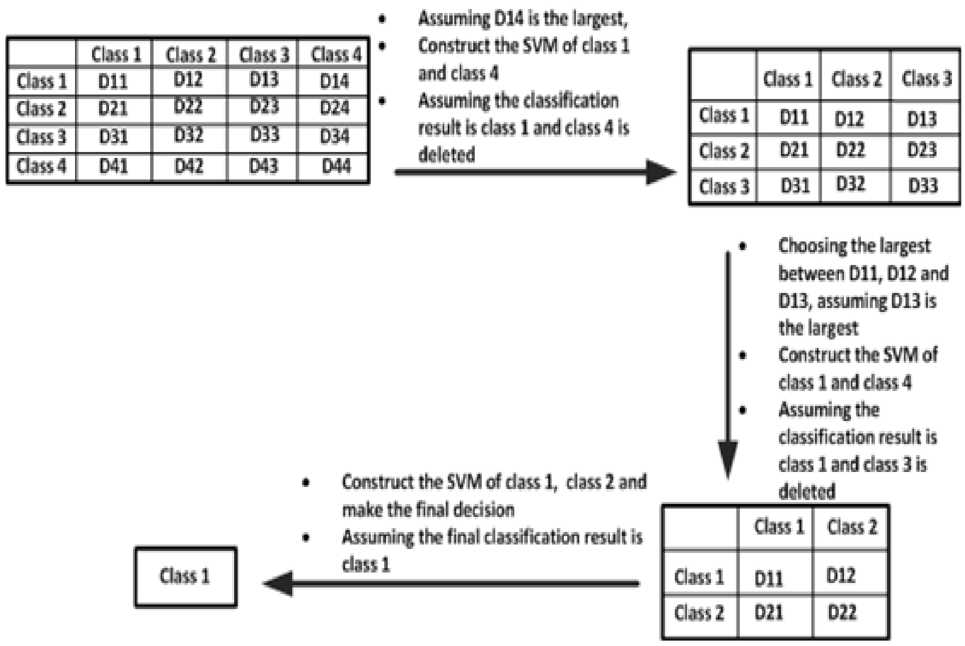
Table 5.2. Optimal sequence of two-class support vector machines for decision making

Optimal sequence for decision making
Step1. Initially, the largest value in the $DBTClist$ is chosen. Assuming the largest value is $DBTC_{x,y}$, $SVM_{x,y}$ is then used to make a decision.
Step2. After the decision, one class is eliminated. Assuming the eliminated class is x, all the $DBTCs$ and $SVMs$ whose indexes contain x will be eliminated from the $DBTClist$ and $SVMlist$.
Step3. Choosing the largest value among the $DBTC$ values whose index contain y for the remaining elements in the $DBTClist$. And the corresponding SVM in the $SVMlist$ will be applied for the second round classification.
Step4. Repeating step 2-3, until one element is left in $DBTClist$ and $SVMlist$, and the final classification is then made.

The procedure in Table 5.2 guarantees that at every step, the two classes used to build up the two-class SVM for classification are always the most separable (with the largest DBTC value in the current DBTClist), thus good generalization performance can be achieved. Another advantage of this decision making scheme is that it avoids the trial of every possible list sequence of DAGSVM for cross validation, thus greatly saving the parameter optimization time.

For the particular fall detection problem, the projection histogram features of a posture and features obtained from the posture's ellipse fitting result (the ratio between the major and minor axes *rho* and the orientation angle *theta*) are used for DAGSVM construction or posture classification.

This sub-section proposes a supervised classification method of classifying different types of postures for fall detection; in the next sub-section, another unsupervised method is proposed, which constructs a model from extracted features for distinguishing normal and abnormal postures.



	Class 1	Class 2	Class 3	Class 4
Class 1	D11	D12	D13	D14
Class 2	D21	D22	D23	D24
Class 3	D31	D32	D33	D34
Class 4	D41	D42	D43	D44

- Assuming D14 is the largest,
- Construct the SVM of class 1 and class 4
- Assuming the classification result is class 1 and class 4 is deleted

	Class 1	Class 2	Class 3
Class 1	D11	D12	D13
Class 2	D21	D22	D23
Class 3	D31	D32	D33

- Choosing the largest between D11, D12 and D13, assuming D13 is the largest
- Construct the SVM of class 1 and class 4
- Assuming the classification result is class 1 and class 3 is deleted

	Class 1	Class 2
Class 1	D11	D12
Class 2	D21	D22

- Construct the SVM of class 1, class 2 and make the final decision
- Assuming the final classification result is class 1

Class 1

Figure 5.10. The decision process for the DAGSVM based on the DBTC values for a four class example, the DBTC value between classes i and j is denoted as D_{ij}.

5.4.2 Unsupervised online OCSVM

A one class support vector machine (OCSVM) is applied in this work for building the model corresponding to normal postures after the features are extracted, and the obtained model is then used to distinguish normal and abnormal postures for further fall detection. The concept of one class support vector machine (OCSVM) was proposed in Chapter 4, which is an efficient data fitting method. Given a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ drawn from an underlying probability distribution $\mathbf{P}$, the OCSVM obtains the supporting region from which the samples in the dataset $\mathbf{X}$ are most likely to lie by estimating a function f. If a sample $\mathbf{x}$ is from the supporting region, the function f value is large; otherwise, a small value of f is obtained. In Chapter 4, a batch learning algorithm is applied to obtain the function f (which means estimating f from a particular training dataset); however, this batch learning scheme is not a good choice for the practical application of fall detection due to the following reasons: firstly, the computational time of the batch learning method is very long and it is inadequate to be used on a large training dataset, which is necessary for constructing the model of normal postures in the practical application; secondly, the batch learning method is a one-time solution, all the training samples need to be gathered during the training phase and the trained OCSVM is kept unchanged after training, which contradicts the situation when an infinite datastream with varying data distribution is encountered in the real-time application. So, instead of a batch learning method, an online learning method ([75] and [79]) is applied for OCSVM model construction.

Online learning algorithm

As it is mentioned in [75] and [79] for the online learning scheme, at time instance t, f can be explicitly represented as by the samples $\mathbf{x}_1, ..., \mathbf{x}_{t-1}$ as:

$$f_t = \sum_{i=1}^{t-1} \alpha_i k(\mathbf{x}_i, \cdot) \tag{5.4.15}$$

where $\mathbf{x}_i, i = 1, ..., t-1$ are the incoming samples before time t and $k(\mathbf{x}, \cdot)$ is a kernel function.

The online OCSVM algorithm is operated in a sample-by-sample way, when a new sample $\mathbf{x}_t$ arrives, the online OCSVM algorithm finds a new set of coefficients $\alpha_i, i = 1, ..., t$ to determine a new function $f_{t+1} = \sum_{i=1}^{t} \alpha_i k(\mathbf{x}_i, \cdot)$. This can be achieved by minimizing a modified regularized risk for the online scheme as proposed in [75]:

$$R(f) = \frac{1}{2} \parallel f - f_t \parallel_H^2 + \eta(\frac{\lambda}{2} \parallel f \parallel_H^2 + C \cdot (\gamma - f(\mathbf{x}_t))_+) \tag{5.4.16}$$

where $\parallel \cdot \parallel_H^2$ means the reproducing kernel Hilbert space (RKHS) distance [79] and $(\cdot)_+$ is used to represent $max\{\cdot, 0\}$, the first term measures the RKHS distance of f from the previous predicted function and the second term is the traditional regularized risk, which controls the complexity of f and the convex loss for the sample $\mathbf{x}_t$. Due to the convex property of $R(f)$, the optimal f_{t+1} can be found by setting the gradient of $R(f)$ to zero, by some algebraic operations as mentioned in [75], the optimal f_{t+1} is represented as:

$$f_{t+1} = \frac{1}{1+\eta\lambda} f_t - \frac{C}{1+\eta\lambda}\beta_t k(\mathbf{x}_t, \cdot) \tag{5.4.17}$$

where $\beta_t \in [-1, 0]$ and normally η is set to one. An auxiliary variable $\tau = \frac{\lambda}{1+\lambda}$ is introduced and f_{t+1} is rewritten as:

$$f_{t+1} = (1-\tau)f_t - (1-\tau)C\beta_t k(\mathbf{x}_t, \cdot) \tag{5.4.18}$$

Note previously, f_{t+1} is represented as the following form:

$$f_{t+1} = \sum_{i=1}^{t} \alpha_i k(\mathbf{x}_i, \cdot) \tag{5.4.19}$$

By comparing (5.4.18) and (5.4.19), it can be observed the coefficients of f_{t+1} can be updated as:

$$\begin{aligned} \alpha_i &= (1-\tau)\alpha_i, \quad i = 1, ..., t-1 \\ \alpha_t &= -(1-\tau)C\beta_t \end{aligned} \tag{5.4.20}$$

Moreover, as discussed in [75] and [79], for the optimal α_t, $\gamma - f_{t+1}(\mathbf{x}_t) = 0$ is needed to minimize the convex loss $(\gamma - f(\mathbf{x}_t))_+$ in (5.4.16), which leads to:

$$\gamma - ((1-\tau)f_t(\mathbf{x}_t) + \alpha_t k(\mathbf{x}_t, \mathbf{x}_t)) = 0 \tag{5.4.21}$$

and the optimal α_t is calculated as:

$$\alpha_t = \frac{\gamma - (1-\tau) f_t(\mathbf{x}_t)}{k(\mathbf{x}_t, \mathbf{x}_t)} \tag{5.4.22}$$

From the above derivations, the coefficients of f_{t+1} can be updated for a new incoming sample $\mathbf{x}_t$ using (5.4.20) and (5.4.22). Note, α_t must be truncated into the range of $[0, (1-\tau)C]$ if it lies outside this range after (5.4.22). As time increases, the number of samples determining the OCSVM function f will become very large, which will lead to memory overload. So in the real application, the previous samples whose coefficients become lower than a small threshold need to be deleted as f updates.

For the obtained f at a time instance t, whether the next incoming sample $\mathbf{x}_{t+1}$ comes from the supporting region corresponding to the previous samples can be determined by:

$$D(\mathbf{x}_{t+1}) = \begin{cases} 1 & \gamma - f(\mathbf{x}_{t+1}) \leq \textit{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{5.4.23}$$

where $D(\cdot)$ is an indication function, $D(\mathbf{x}_{t+1}) = 1$ implies the sample is from the supporting region; otherwise, the sample is not from the supporting region and is taken as an outlier. The value of $\gamma - f(\mathbf{x}_{t+1})$ is calculated and compared with a preset threshold to make a decision. In the experimental results, the receiver operation characteristic (ROC) analysis of the proposed fall detection system's performance is presented by choosing different *thresholds*.

Kernel selection for normal posture model construction: A proper kernel needs to be chosen for the implement of the OCSVM. In this work, the most popular RBF kernel [76] is applied to obtain a non-linear model boundary for a better distinguishing result of normal and abnormal samples. The form of RBF kernel can be represented as (5.4.24):

$$k(\mathbf{F}_1, \mathbf{F}_2) = exp(-\gamma \cdot d(\mathbf{F}_1, \mathbf{F}_2)) \tag{5.4.24}$$

where γ is defined as the kernel parameter and $d(\mathbf{F}_1, \mathbf{F}_2)$ measures the distance between two vectors $\mathbf{F}_1$ and $\mathbf{F}_2$, normally, Euclidean distance is applied. In this work for modelling postures, different types of features (ellipse features, shape-structure features and position features) are concatenated into one feature vector as $\mathbf{F} = [\mathbf{f}_e, \mathbf{f}_s, \mathbf{f}_p]$ for OCSVM model construction, where $\mathbf{F}$ is the finally concatenated feature vector, $\mathbf{f}_e$, $\mathbf{f}_s$ and $\mathbf{f}_p$ represent the ellipse features, shape-structure features and position features respectively.

For different types of features, the scales are different (for example, the shape-structure histogram values will always be less than one and for the position feature, the value will be much larger). In this case, it is not proper to apply the Euclidean distance because the small scale features will be omitted and therefore the distance between two feature vectors $\mathbf{F}_1$ and $\mathbf{F}_2$ is measured in the following way:

$$d(\mathbf{F}_1, \mathbf{F}_2) = w_1 \cdot D_{Euclidean}(\mathbf{f}_{e1}, \mathbf{f}_{e2}) + w_2 \cdot D_{Bhattacharyya}(\mathbf{f}_{s1}, \mathbf{f}_{s2}) + w_3 \cdot D_{Euclidean}(\mathbf{f}_{p1}, \mathbf{f}_{p2}) \tag{5.4.25}$$

where $D_{Euclidean}$ is the normal Euclidean distance and $D_{Bhattacharyya}$ is the Bhattacharyya distance [59] measuring the distance between histograms. Distances between different types of features are calculated and summed in a weighted form (weighted by the set w_1, w_2 and w_3) to compensate for different scales of different types of feature.

By substituting (5.4.25) into (5.4.24), the final kernel form is obtained as:

$$\begin{aligned} k(\mathbf{F}_1, \mathbf{F}_2) &= exp(-\gamma \cdot d(\mathbf{F}_1, \mathbf{F}_2)) \\ &= exp(-\gamma w_1 \cdot D_{Euclidean}(\mathbf{f}_{e1}, \mathbf{f}_{e2})) + exp(-\gamma w_2 \cdot D_{Bhattacharyya}(\mathbf{f}_{s1}, \mathbf{f}_{s2})) \\ &+ exp(-\gamma w_3 \cdot D_{Euclidean}(\mathbf{f}_{p1}, \mathbf{f}_{p2})) \\ &= exp(-c_1 \cdot D_{Euclidean}(\mathbf{f}_{e1}, \mathbf{f}_{e2})) + exp(-c_2 \cdot D_{Bhattacharyya}(\mathbf{f}_{s1}, \mathbf{f}_{s2})) \\ &+ exp(-c_3 \cdot D_{Euclidean}(\mathbf{f}_{p1}, \mathbf{f}_{p2})) \qquad (5.4.26) \end{aligned}$$

here c_1, c_2 and c_3 are used to replace γw_1, γw_2 and γw_3 and this RBF kernel based on the weighted summation of different distances is used in implementation of the OCSVM model for normal postures.

5.5 Rules used for fall detection

After classifying a particular posture by a supervised classifier or determining whether a posture is a normal one or not by an unsupervised classifier, some rules can be used to further confirm whether a fall happens or not, these rules and the results obtained from postures are used together to achieve a robust fall detection system.

5.5.1 Rules used for supervised fall detection

After obtaining the posture class by using a supervised classifier, a fall activity is determined if the following three rules are met:

1) The posture is classified as 'lie' or 'bend'.

2) The posture is inside the ground region.

3) The above two conditions are kept for a certain time, which exceeds a preset time threshold (20s is used).

The three rules are set according to the characteristic of fall activity, in most cases, fall activities end up with a 'lie' posture and this 'lie' posture usually remains for a certain time due to the period of immobility of an elderly person after the fall. Moreover, different from lying on the bed/sofa, the posture should be inside the ground region (or at least a large part is inside the ground region). And considering that an elderly person rarely 'bends' for a long time in the ground region (here the 'bend' class is defined as postures of bending to fasten a shoe lace or bending to pick up something, which is very common to occur in an elderly person's daily life), if a 'bend' posture is detected in the ground region for a certain time, it is also regarded as an abnormal activity (this can happen when an elderly person falls while ending up with a bend-like posture, an example will be given in the experimental section).

In order to detect falls by the three rules, the ground region needs to be determined initially. Before the fall detection phase, floor detection is carried out. In this phase, when the posture is classified as stand or sit, the region nearby the lower extreme point of the ellipse (an 8×8 block is used here) is marked as the ground region. Figure 5.11 shows the result of floor detection.

Note, as shown in (d) and (e), sometimes the furniture is moved and the floor region has to be updated accordingly. The detected floor region is extremely helpful to distinguish a fall on the floor from lying on a sofa, which is shown in the experimental part.

5.5.2 Rules used for unsupervised fall detection

Two rules are used together with the OCSVM model constructed for describing normal postures to achieve a robust fall detection system, they are:

1) A fall is only reported when a large movement is detected. In Chapter 3, a measurement of the amplitude of movement is proposed by using the motion history image (MHI); however, the frame difference results used to construct the MHI are easily affected by noise and illumination change in

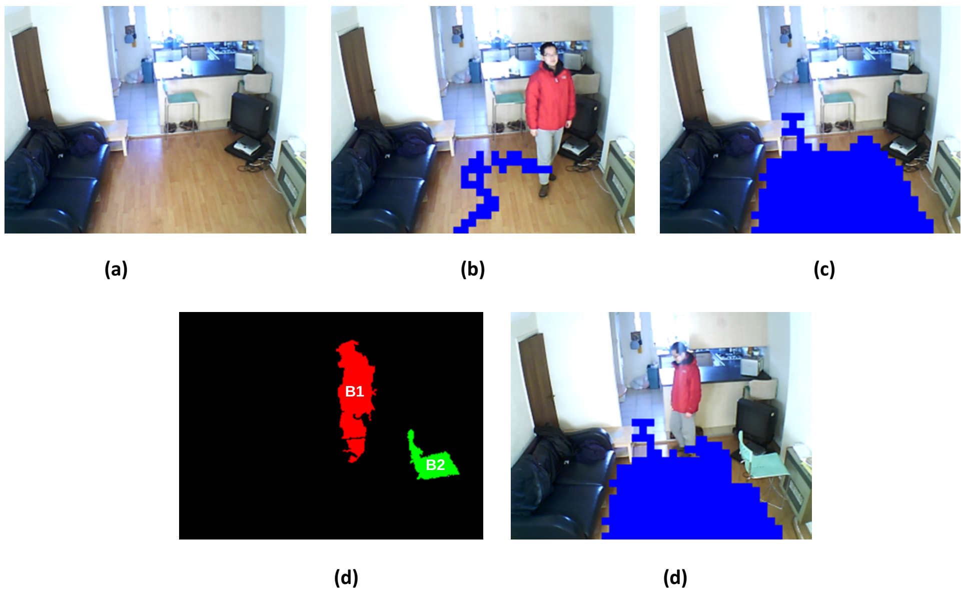


Figure 5.11. The floor detection results. a) Original image; b) Detected floor region while a person is walking; c) Floor detection result after some time; d) More than one blob after the furniture is moved, the moving blob (human body) is marked in red, the static blob (furniture) is marked green; e) The updated floor region result after moving furniture. The region nearby the new position of the furniture is unmarked and that nearby the person's feet is marked as the floor region.

the environment. In this work, a new measurement is proposed based on the motion energy image (MEI) [80]. The amplitude of movement is measured by the area ratio (denoted as AR) between the area of certain number of MEI frames and the area of the current frame's foreground region. One example is shown in Figure 5.12, from this figure, it can be observed for a large movement, fall activity has a larger AR value than the other three types of activities (walking, sitting and bending).

In a video sequence, a sliding window method is applied to estimate the AR value for each frame, which is illustrated in Figure 5.13. For a particular frame F_t, the AR value is calculated as the ratio between the area of the MEI of the frames in the sliding window and the area of the F_t's foreground region. For the next time, the sliding window moves forward over one frame

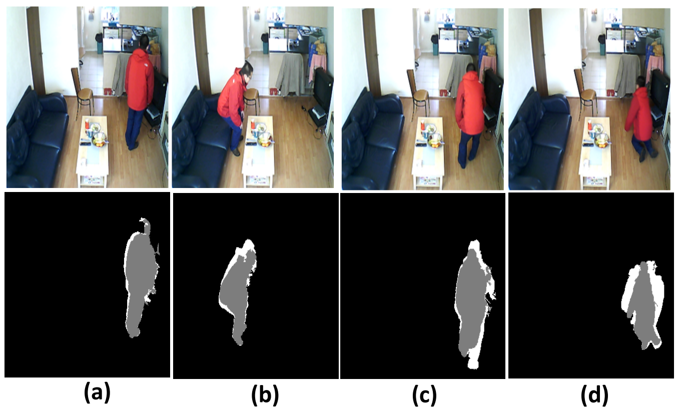


Figure 5.12. The movement amplitude measurement for four activities. The first line shows the original images and the second line shows the MEI results (non-black region with the current images' foreground regions marked as gray). The calculated AR values are: (a) walking, AR=1.13 (b) sitting, AR=1.28 (c) bending, AR=1.34 and (d) falling, AR=1.67.

and the new AR value is calculated, which is compared with the previous one and the larger value is then retained. The final AR value of F_t is obtained when the sliding window passes over this frame (as shown in Figure 5.13 (e)), which reflects the largest movement around this frame. In this work, a threshold of 1.5 (chosen by empirical study) is set on the AR value for distinguishing large and small movements.

2) Normally after an old person falls, he/she will be most likely to lie on the ground for a certain time interval. So, a fall is reported only if the abnormal posture lasts longer than a time interval (20s is used but this could be tuned in application), this will avoid occasional abnormal postures which do not last for a predifined threshold (such as bending to fasten the shoe ties).

By using these two rules, better performance can be achieved than when using only the OCSVM model for abnormal posture detection with the less number of non-fall activities being taken as fall activities, the corresponding experimental results are presented in the experimental section.

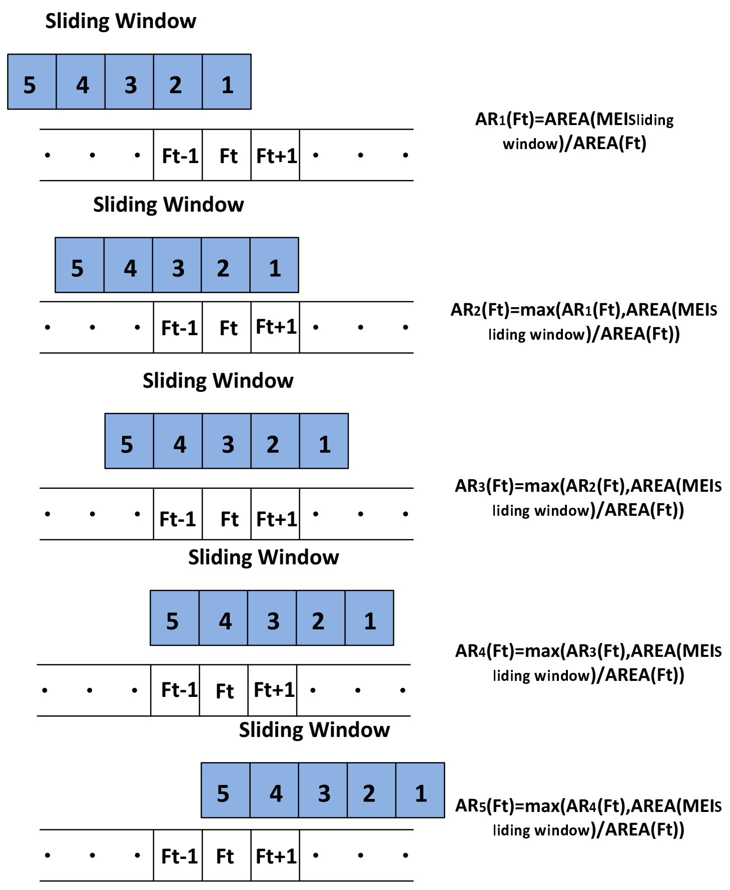

Sliding Window
5 4 3 2 1
• • • Ft-1 Ft Ft+1 • • •
AR1(Ft)=AREA(MEISliding window)/AREA(Ft)
AR2(Ft)=max(AR1(Ft),AREA(MEISliding window)/AREA(Ft))
AR3(Ft)=max(AR2(Ft),AREA(MEISliding window)/AREA(Ft))
AR4(Ft)=max(AR3(Ft),AREA(MEISliding window)/AREA(Ft))
AR5(Ft)=max(AR4(Ft),AREA(MEISliding window)/AREA(Ft))

Figure 5.13. Estimation of the AR value for a frame using a sliding window (a length of five). At each time, the sliding window moves forward over one frame and the AR value for frame F_t is calculated as the maximum value between the new calculated AR value and the previous one. The final AR value (AR_5) is obtained when the sliding window passes over the frame.

5.6 The supervised and unsupervised fall detection systems

The supervised and unsupervised classifiers for posture classification, together with the corresponding rules are used to construct robust fall detection systems for fall detection, which are presented in Figures 5.14 and 5.15.

Figure 5.14 shows the fall detection system based on supervised posture classification method. An efficient codebook background subtraction algo-

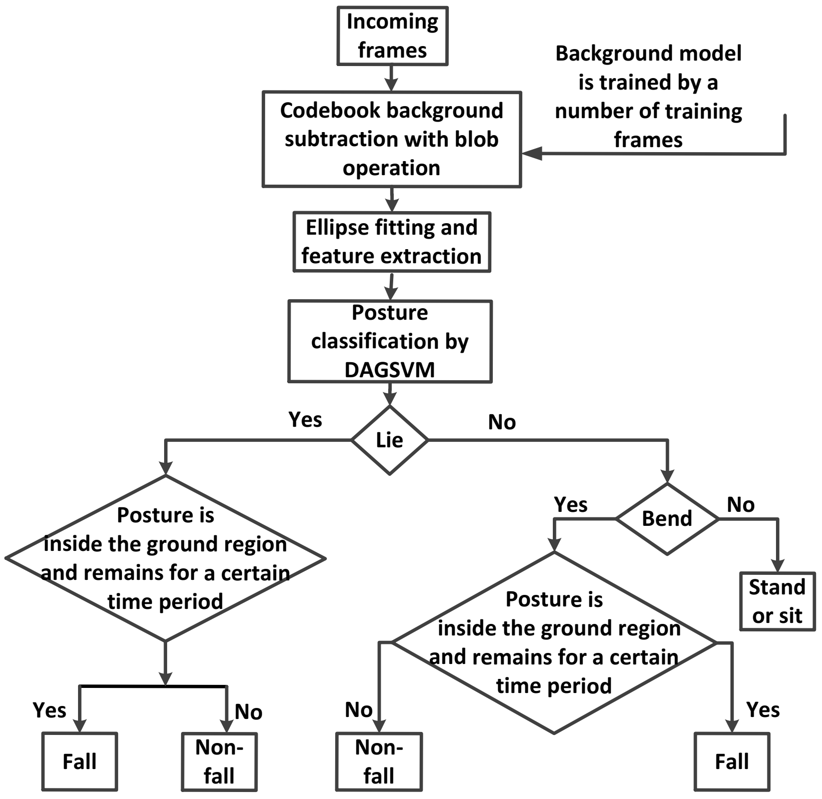

Incoming frames
Background model is trained by a number of training frames
Codebook background subtraction with blob operation
Ellipse fitting and feature extraction
Posture classification by DAGSVM
Yes
Lie
No
Posture is inside the ground region and remains for a certain time period
Yes
Bend
No
Stand or sit
Posture is inside the ground region and remains for a certain time period
Yes
No
Fall
Non-fall
No
Non-fall
Yes
Fall

Figure 5.14. The flow chart of the proposed supervised DAGSVM classifier based fall detection system.

rithm is initially applied to extract the human body foreground and some post-processing is applied to improve the results. From the extracted foreground silhouette, features are extracted from the fitted ellipse and projection histogram, which are used for classification purposes. These features are fed into the DAGSVM (which is trained from a dataset containing features extracted from different postures in different orientations) and the extracted foreground silhouette is classified as one of four different postures (bend, lie, sit and stand). The classification results, together with the detected floor information, are then used to determine fall or non-fall activities.

Figure 5.15 shows the fall detection system based on the online OCSVM model. Firstly, the codebook background subtraction method is applied to extract the human body silhouette and some post-processing is applied

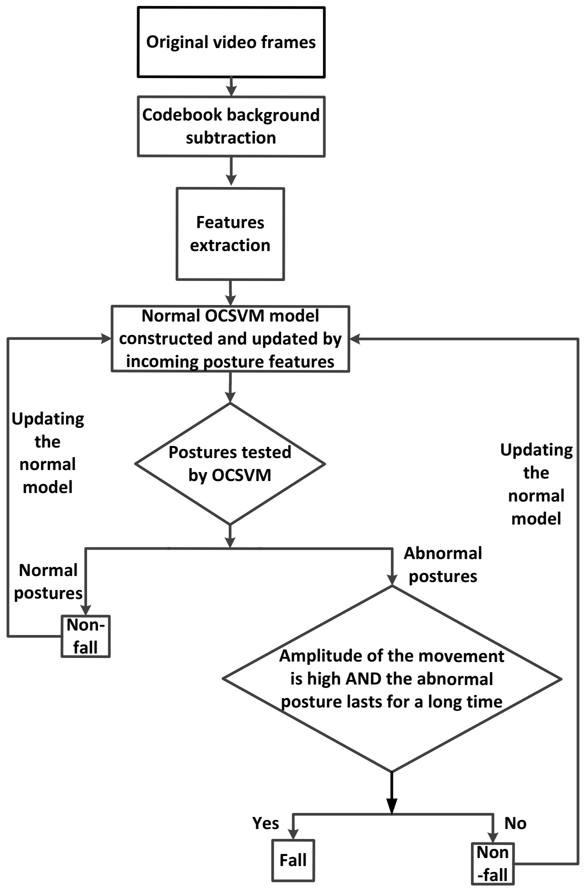

Original video frames
Codebook background subtraction
Features extraction
Normal OCSVM model constructed and updated by incoming posture features
Updating the normal model
Postures tested by OCSVM
Updating the normal model
Normal postures
Abnormal postures
Non-fall
Amplitude of the movement is high AND the abnormal posture lasts for a long time
Yes
No
Fall
Non -fall

Figure 5.15. The flow chart of the proposed unsupervised online OCSVM based fall detection system.

to improve the results. In order to describe fully the posture, three types of features, including ellipse features, shape structure features and position features are extracted. After the extraction of these features, an online OCSVM is then applied to describe the normal region described by these features, which can be updated to adapt to new postures. To further improve classification performance, two rules are added, one rule is to measure the amplitude of the movement, if there is not a large movement, a fall will not be reported even though abnormal postures are detected by the online OCSVM. The other is the duration of an abnormal posture; a fall is reported only if the duration of an abnormal posture is longer than a threshold, which will effectively avoid false alarms when the person occasionally bends quickly. By combination of the online OCSVM and these two rules, good performance can be achieved as will be shown in the later experimental analysis section. If no fall is detected, normal postures are then used to update the normal OCSVM model in order to adapt to new normal postures.

The evaluations of performance of these two fall detection systems are presented in the experimental part.

5.7 Experimental analysis

The performances of the proposed supervised and unsupervised fall detection systems are next presented. The experiments were carried out in a simulated home environment. A USB camera was used for recording the real video sequence with the image size of 320 × 240, the recorded video sequence is processed by using VC++ 6.0 (with OpenCv library 1.0) and MATLAB on an Intel(R) Core(TM)2 Duo CPU laptop with 1.00GB Memory. Figure 5.16 shows the camera used in the experiment and the background image it records.

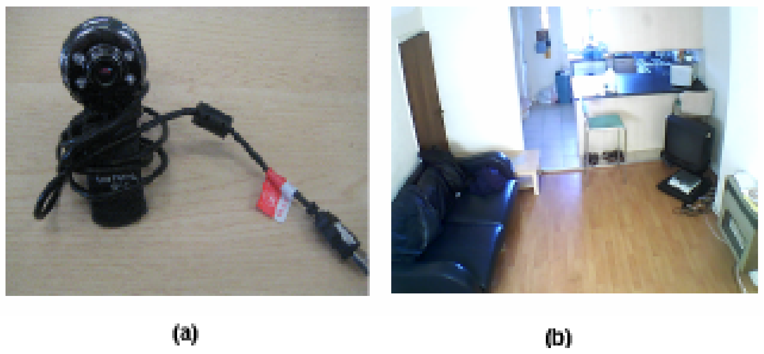


Figure 5.16. The USB camera used in the experimental room environment (a) and the experimental environment (b)

5.7.1 Background subtraction results

Some background subtraction results in this more realistic home environment are shown in Figures 5.17, 5.18 and 5.19, in which three challenging scenarios which occur commonly in a home environment are analyzed. Initially, a short video clip which contains 100 frames is applied for training the original background model, which will be updated with the evolution of time.

Figure 5.17 shows the background subtraction results at different times of day with gradual light change. Images (a) and (c) show a frame captured at noon time and the corresponding background subtraction result. The background model is updated to cope with the gradual light change and the results are shown in (b) and (d), where (b) is a frame captured later in the afternoon and (d) is the background subtraction result with the updated background model.

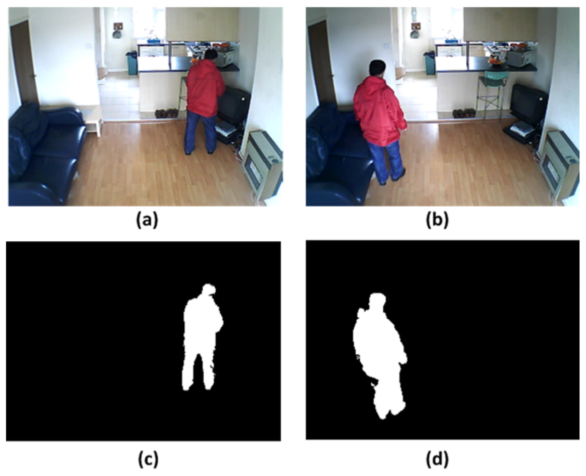


Figure 5.17. Background subtraction results at different times of a day; (a) and (c) show an original frame captured at noon time and the corresponding background subtraction result; (b) is a frame captured in the afternoon with the light condition changed and (d) is the background subtraction result of (b) with the updated background model.

Figure 5.18 shows the background subtraction results in the presence of moving objects. Line (a) shows four frames sampled in a short video sequence, which shows that a person moves the table and fruit plate. Line (b) shows the background subtraction results by directly applying the codebook background subtraction method. It can be seen that there are many segmentation errors due to the movement of the table and fruit plate. Frame differencing results are shown in line (c), which indicate active pixels and help to find the active blob representing the human body. By the post-processing technique as discussed in Section 5.2, improved background subtraction results are obtained in line (d).

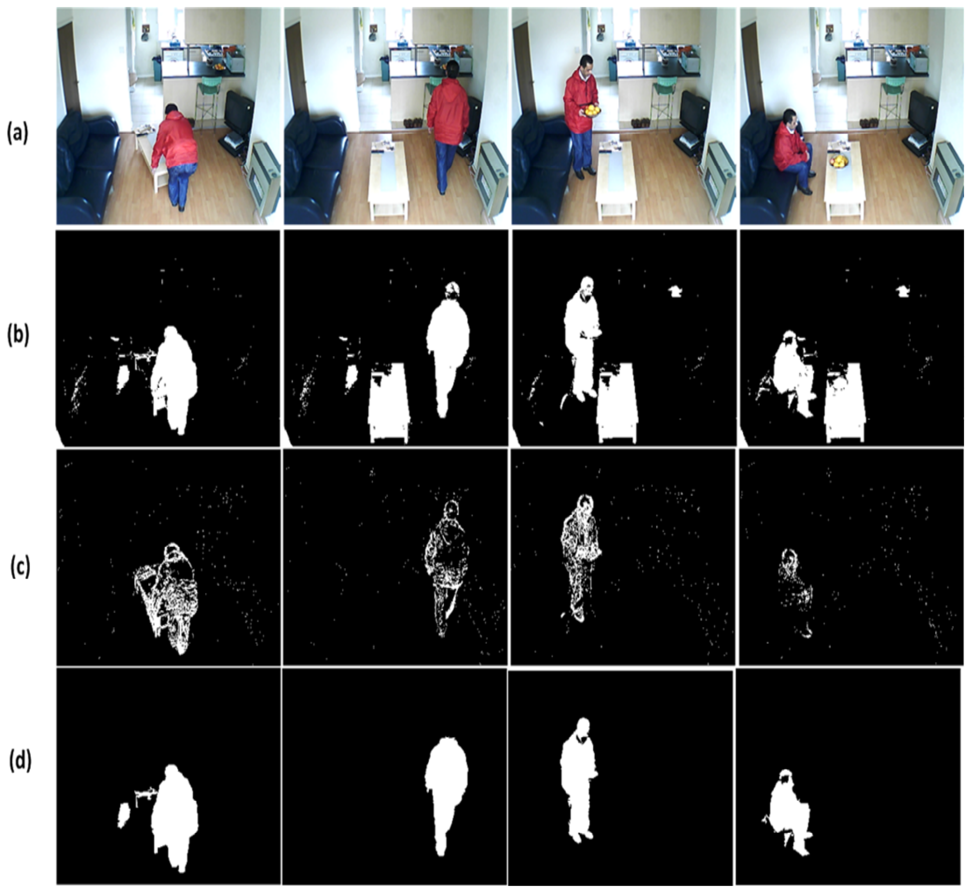


Figure 5.18. Background subtraction results for moving furniture. Line (a) shows original frames of a person moving the table and fruit plate. Codebook background subtraction results are shown in line (b). Line (c) shows the frame differencing results which indicate active pixels. From the frame differencing results and blob operations, improved background subtraction results are obtained in line (d).

Figure 5.19 shows a case of sudden illumination change. At frame (c), the light is turned on and a large illumination change can be observed. This sudden change of illumination can be detected by the frame differencing result as shown in (g), with more than 50% of pixels being marked as active ones (white). The background model is then retrained to cope with this sudden illumination change. As shown in (d) and (h), good segmentation is obtained by the retrained background model.

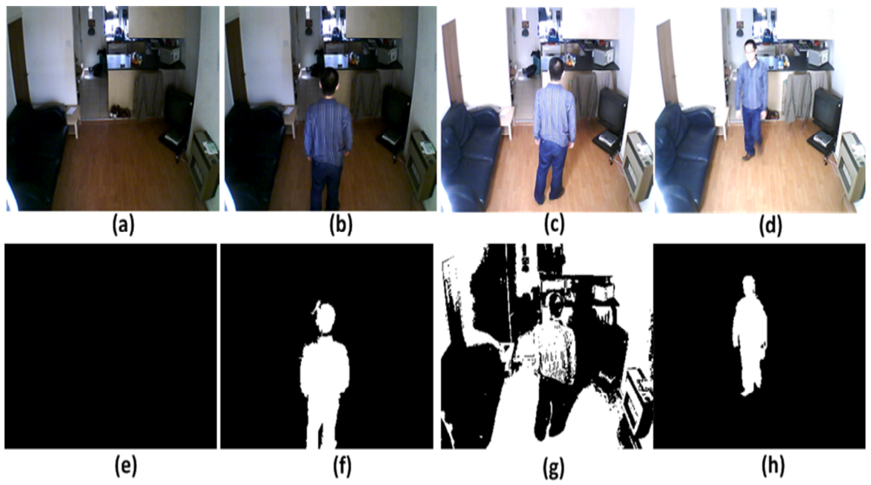


Figure 5.19. Background subtraction for sudden illumination change. Frames (a) and (b) are captured with the light off, at frame (c) the light is turned on and a drastic illumination change can be observed. And frame (d) is captured after the light is turned on for a certain time. Frames (e) and (f) are the background subtraction results of (a) and (b). Image (g) is the frame difference result for (c), sudden illumination is detected because more than 50% of the pixels are marked as active ones and the background model is retrained. Frame (h) is the subtraction result of (d) by the retrained background model.

5.7.2 Results for the supervised fall detection system

For the experiment of the fall detection based on supervised classifiers, 15 people (11 males and 4 females) were invited to attend the experiments for simulating different postures and activities. The characteristics of the

15 people are summarized in Table 5.3. It has to be noted that real elderly people were not invited to participate in the experiments because it is unsafe for an elderly person to simulate fall activities; instead, younger people were invited to mimic elderly people.

Table 5.3. The characteristics of 15 participators in the experiments

No. of people	**15**
Male/Female	**11/4**
Age	**25-49**
Weight	**57-94 (kg)**
Height	**158-187 (cm)**

Posture classification results

To form the posture dataset, 3200 postures (including 800 stands, 800 sits, 800 lies and 800 bends) from 15 people were recorded. As in [54], each person was asked to simulate postures in different directions so that the constructed classifier is robust to view angles. Some samples are shown in Figure 5.20.

Based on this recorded dataset, a commonly used 10-fold validation [31] was applied to evaluate the performance of the posture classification and three types of comparisons were made:

The first evaluation is the feature comparison, which is shown in Table 5.4; the classification results are compared when using the ellipse features or projection histogram features alone and using a combination of these two features. The DAGSVM is applied for classification.

Table 5.4. Classification result by different types of features.

	Global features	**Local features**	**Combined features**
Classification rate	**89.72%**	**76.70%**	**96.09%**

From Table 5.4, it can be seen that the classification result by the combined feature is clearly better than using either feature alone.

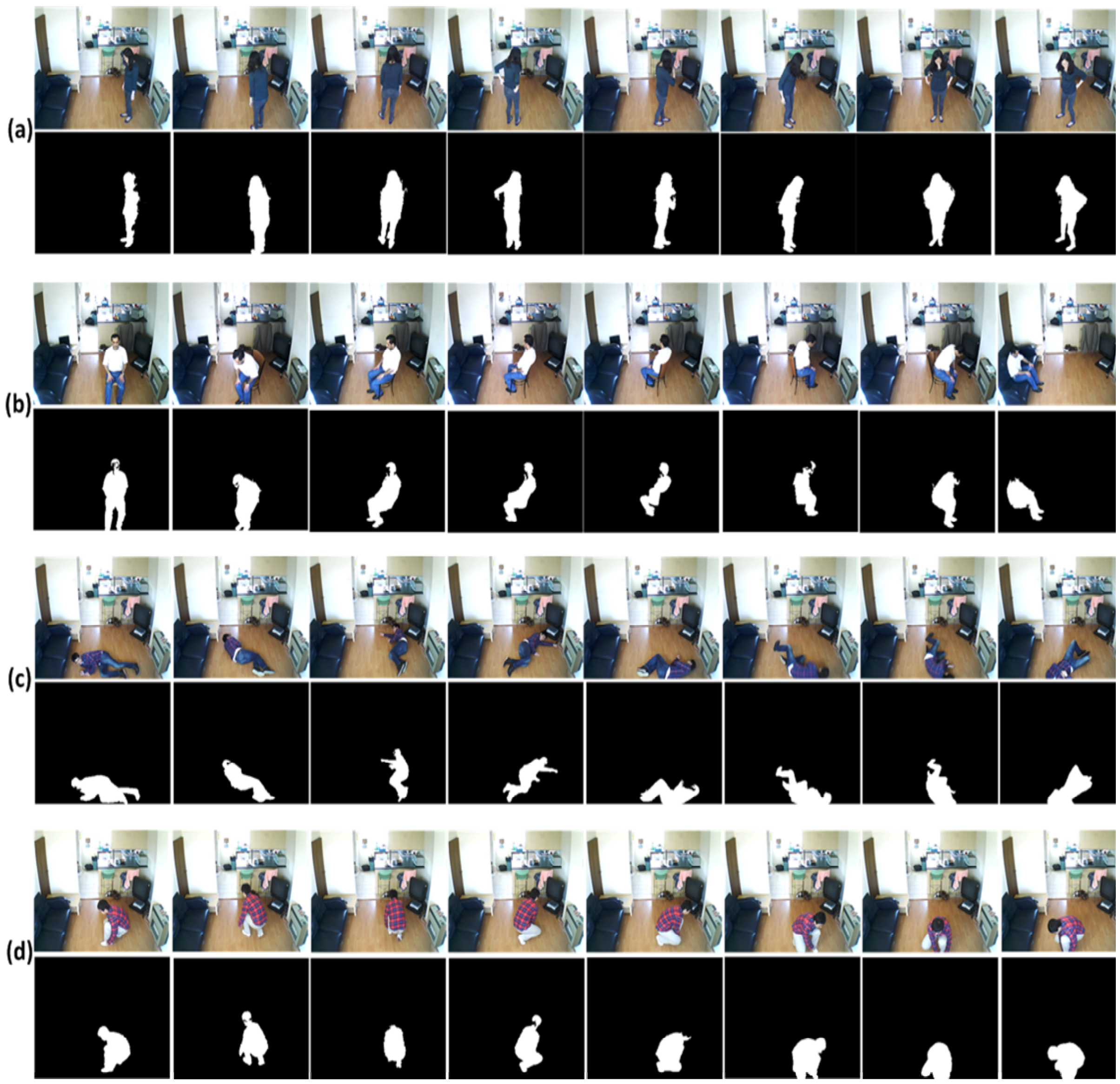


Figure 5.20. Posture samples simulated by different participates in different orientations: (a) stand (b) sit (c) lie and (d) bend.

For the second assessment, two types of parameter optimization methods are compared for DAGSVM: the 10-fold validation based method and the DBTC based method, which is shown in Table 5.5.

Table 5.5. Comparison of different parameter optimization methods.

	10-fold validation method	**DBTC based method**
Training time (s)	**2453.40**	**397.87**
Classification rate	**95.63%**	**96.09%**

From Table 5.5, it can be seen that the DBTC based parameters optimization method can greatly reduce the training time of DAGSVM while achieving an even slightly better performance, so that it is preferred in this work.

In the third assessment, the performance of DAGSVM is compared with other traditional classifiers (k-nearest neighbour (KNN), multi-layer perceptron neural network (MLPNN), naive Bayes classifier (NBC) and binary decision tree (BDT) implemented in PRtools–a well-known software package for pattern recognition [81]), the results of which are shown in Table 5.6. For a fair comparison, the parameters of the comparison classifiers are tuned to be optimal by using the cross validation function in PRtools.

Table 5.6. Comparison of different classifiers.

	DAGSVM	**KNN**	**MLPNN**	**NBC**	**BDT**
Classification rate	**96.09%**	**92.64%**	**92.53%**	**75.27%**	**83.72%**

It can be observed that DAGSVM achieves the best performance with a slightly better classification rate performance than k-nearest neighbour and the MLP neural network.

However, in the real situation, the training dataset is not perfect and it is common that some outliers exist in the training dataset. In posture classification, outliers are mainly caused by extremely bad background subtraction results and wrong labeling (the feature of one class is mislabeled as another class). In Table 5.7, the comparison results of the classifiers are presented on a dataset which includes 10% outliers.

Table 5.7. Comparison of different classifiers on a dataset which is corrupted by 10% outliers.

	DAGSVM	**KNN**	**MLPNN**	**NBC**	**BDT**
Classification rate	**95.51%**	**84.07%**	**85.93%**	**72.42%**	**58.72%**

Table 5.7 shows a more obvious advantage of the classification rate of the DAGSVM over other classifiers on this noisy dataset. And compared with other classifiers, DAGSVM is the most robust to such noise due to the reason that slack variables are introduced in the two-class SVMs of the DAGSVM classifier to cope with the noises as mentioned in Section 5.4 (with only a 0.58% drop in classification rate compared with the result of Table 5.6). So, according to Table 5.6 and Table 5.7, DAGSVM is better for posture classification than other traditional classifiers.

Fall detection by the supervised DAGSVM classifier

Posture classification results along with the detected floor information are used to detect falls according to the three conditions in Section 5.5.1. To illustrate, five cases are presented in Figure 5.21, (a) shows a person who has fallen on the floor, and a 'lie' posture is detected and most parts of the human body region are in the detected ground region; and this posture is kept for a certain time (longer than 20s), so a fall is detected. For (b), although a 'lie' posture is detected, the human body blob is not in the floor region, so the lying on the sofa case is correctly classified as non-fall. For (c), (d) and (e), the postures are all classified as 'bend'; however, for (c), a large portion of the human body is in the ground region and the 'bend' posture remains for longer than 20s. It is generally impossible for an elderly person to bend for a long time in the ground region, so this is abnormal activity and it is detected as a fall. For (d) and (e), either the detected 'bend' posture does not hold for a long time (for case (d), a person ties his shoe lace and the 'bend' posture recovers to 'stand' posture in a short time), or the posture is not in the ground region (only a small portion of the human body region is in the ground), so they are not detected as falls.

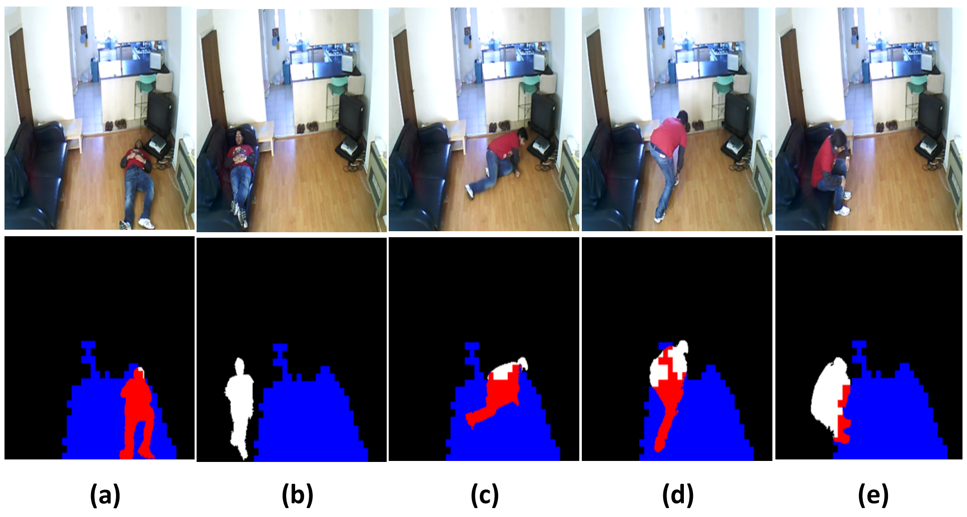


Figure 5.21. Different cases of fall and non-fall activities. a) Fall on the floor; b) Lie on the sofa; c) Fall on the floor; d) Bend to fasten the shoe tie; e) Sit on the sofa. For (a) and (b), the postures are classified as 'lie' and for (c), (d) and (e), the postures are classified as 'bend'. The blue region represents the detected floor, the red region represents the intersected part of the foreground human body with the detected floor region, the white region represents the foreground human body part which is not intersected with the detected floor region and the remaining background region is marked as black. The proposed system successfully classifies (a), (c) as falls and (b), (d) and (e) as non-falls.

For evaluating this fall detection system, each person is asked to simulate 16 fall activities and 16 non-fall activities in different directions. A total number of 240 fall activities and 240 non-fall activities are recorded, which are used to evaluate the proposed fall detection system. For classifying one person's activity, the postures from other people in the recorded posture dataset are used to construct the DAGSVM classifier. Final results are given in Table 5.8, which show that 233 out of 240 (97.08%) falls can be detected while only 2 out of 240 (0.8%) non-falls are mistaken as falls; and a high fall detection rate is obtained with a very low false detection rate.

Table 5.8. Evaluation of the proposed fall detection system.

	No. of activities	**Detected as falls**	**Detected as non-falls**
Falls	**240**	**233**	**7**
Walking around	**60**	**0**	**60**
Sitting on the sofa/chair	**60**	**2**	**58**
Bending	**60**	**0**	**60**
Lying on the sofa	**60**	**0**	**60**

5.7.3 Results for the unsupervised fall detection system

For the unsupervised fall detection system, one man simulates the normal activities of an old person in the hall and kitchen (such as normal walking, sitting on the sofa/chair and lying on the sofa) and a total number of 6882 postures was chosen to build up the normal model. Some samples are presented in Figure 5.22, which show a person's activities in a cluttered environment, for some frames the human body is even seriously occluded by the background (such as the third line in Figure 5.22).

Fall detection by the OCSVM model

The collected normal postures were used to construct a normal model by using the online OCSVM scheme. Initially, the parameters of the OCSVM model needed to be set properly, in the experiment, $\gamma = 1$, $C = 0.01$, kernel parameters $c_1 = 0.6$, $c_2 = 1.1$ and $c_3 = 0.06$ and auxiliary variable $\tau = 10^{-5}$ were set; 38 fall activities (falls were simulated in every possible place in the environment) and 34 non-fall activities were recorded for evaluating the proposed fall detection system. Two schemes were evaluated and compared, the first scheme was using only the OCSVM, falls were reported if abnormal postures were detected by the OCSVM; the second scheme was the combination of the OCSVM with two rules (in Section 5.5.2) for fall detection. Different *threshold* values defined in (5.4.23) for the OCSVM model are tested for

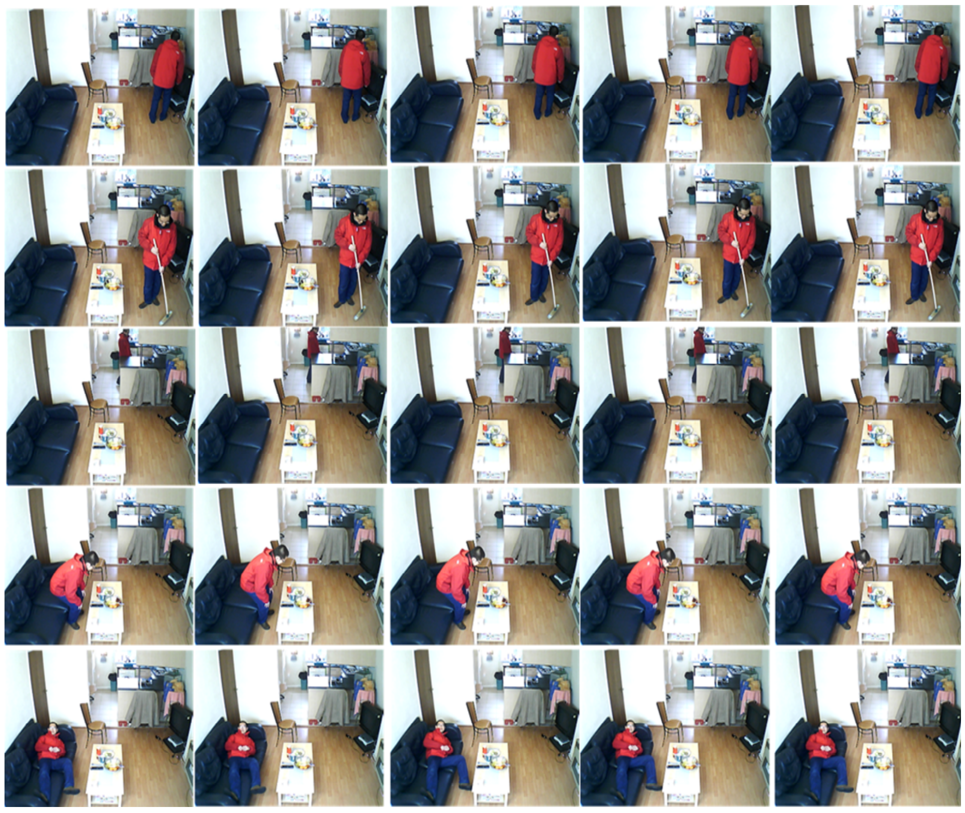


Figure 5.22. Sample frames of simulated normal activities.

the ROC analysis and the ROC curves for these two schemes are plotted in Figure 5.23; the AUC value, optimal TPR and FNR pair are presented in Table 5.9. From Table 5.9, it can be observed that by using rules, the AUC value can reach unity, the optimal TPR and FNR can be unity and zero respectively (under the optimal threshold value 0.5 according to the trial on a range between $[0, 1]$ with the interval 0.05), which means that all the fall activities are detected without mistaking any non-fall activities as falls.

Table 5.9. Comparison results for OCSVM model for fall detection with and without rules

	AUC value	optimal TPR	optimal FNR
OCSVM without rules	0.9781	95%	6%
OCSVM with rules	1	100%	0%

Two examples are presented to illustrate the effect of the proposed two

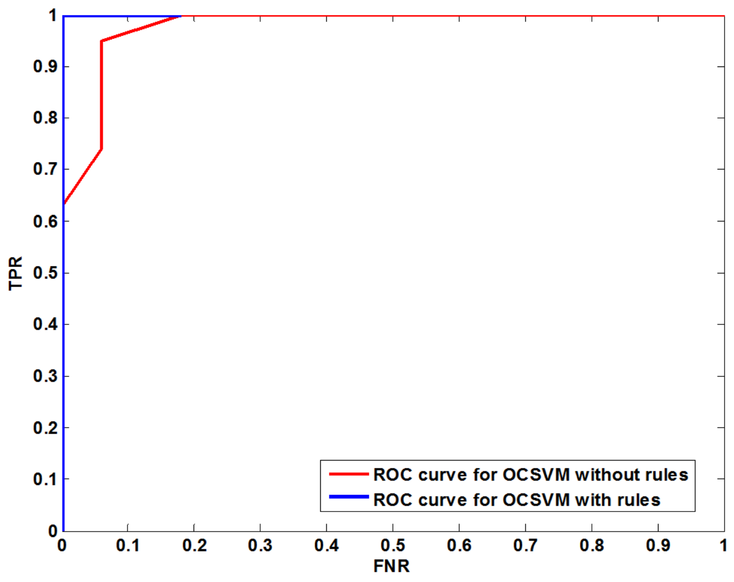


Figure 5.23. The ROC curves comparison for OCSVM with and without rules.

rules on reducing FNR, the first example shows a sequence of moving furniture, a chair, with the selective frame samples being shown in Figure 5.24. The corresponding AR values (calculated with a sliding window size of five) and results calculated by the trained OCSVM model are shown in Figure 5.25. From Figure 5.25 (b), it can be seen that the posture is taken as abnormal (the OCSVM value is less than the preset threshold 0.5, under which the best performance can be achieved according to Table I) after around the 150th frame. However, from (a), it shows that the AR values are always less than the threshold 1.5, so according to Rule 1 in Section 5.5.2, no falls are reported because large activity is not detected.

Another example shows a sequence of fast bending, the selective frame samples are shown in Figure 5.26. Figure 5.27 shows the results of AR and OCSVM for this video sequence and a large movement happens around the 60th frame (bending happens) and the bending posture is detected as the abnormal one. However, this abnormal state only lasts a very short time (the posture recovers to the normal standing posture after a short bending

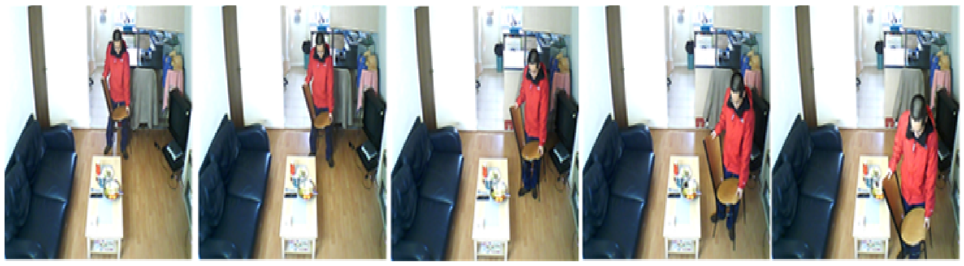


Figure 5.24. Selective frame samples for moving a piece of furniture, a chair.

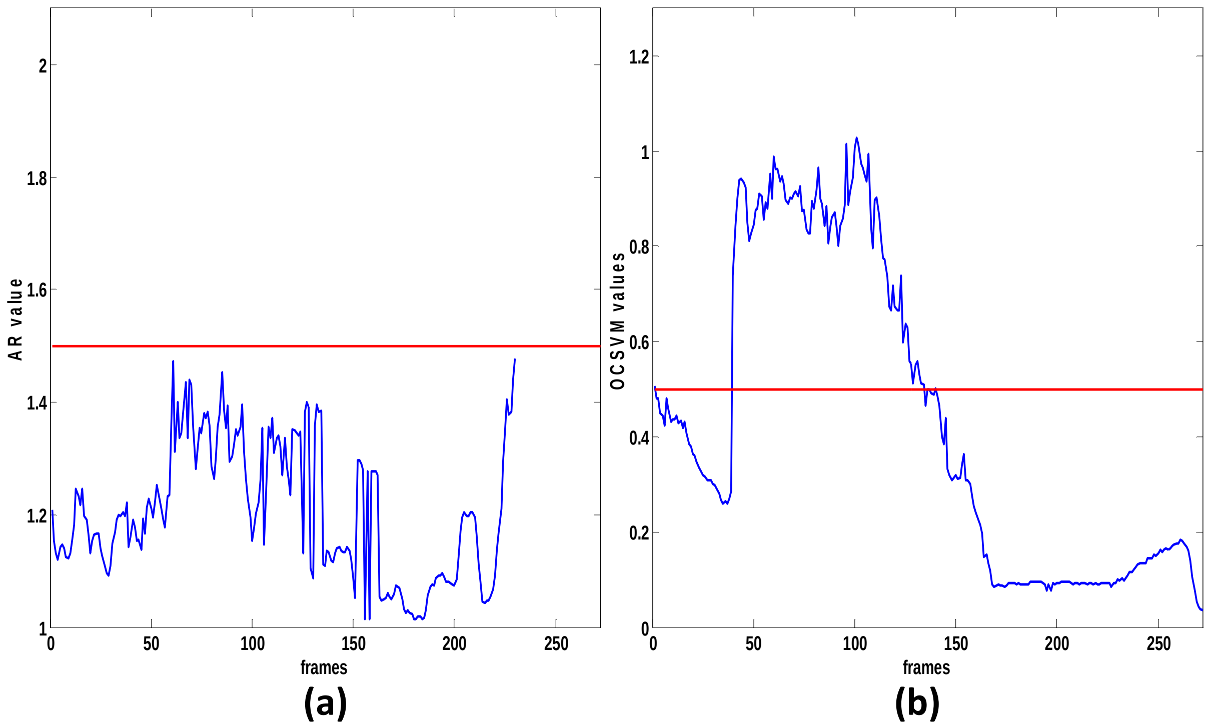


Figure 5.25. The AR and OCSVM values for a sequence of moving furniture. (a) AR results for moving furniture sequence (b) OCSVM results for moving furniture sequence

period). So, according to Rule 2, the duration of the abnormal posture does not exceed the preset threshold (20s), so no falls are reported.

The two examples show that some non-fall activities can be effectively avoided to be taken as fall activities by using the proposed two rules, which

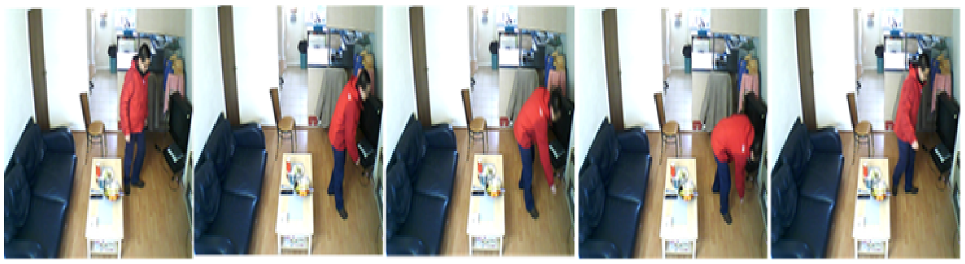


Figure 5.26. Selective frame samples for fast bending activity.

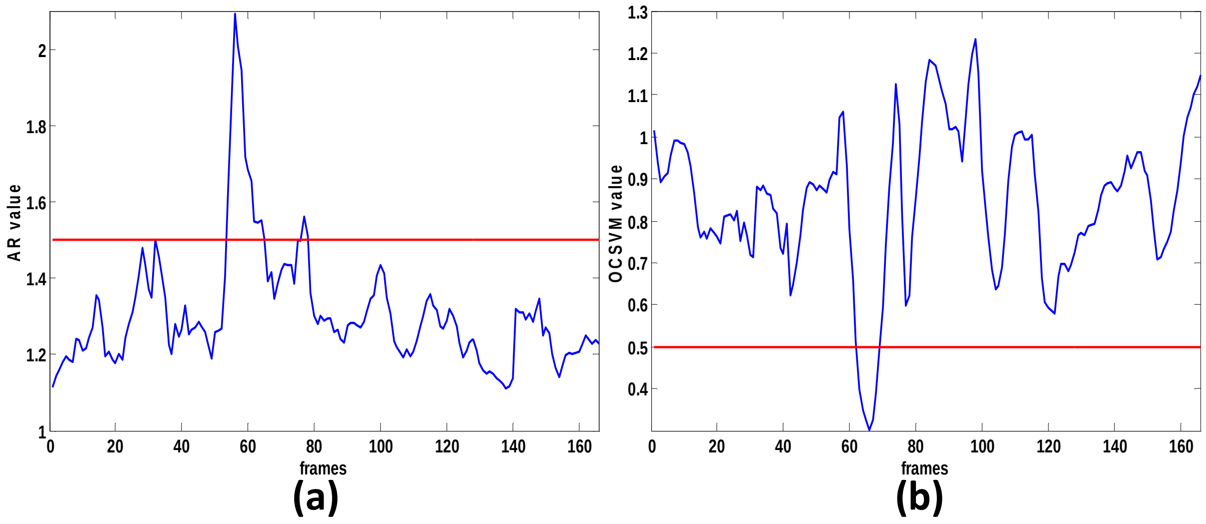


Figure 5.27. The AR and OCSVM values for a sequence of bending activity. (a) AR results for bending sequence (b) OCSVM results for bending sequence.

obtains a better fall detection system than using only the OCSVM model for abnormal posture detection.

New Postures Adaptation

An old person's behavior will not be unchanged over time and sometimes his behavior changes and new postures emerge, so a good fall detection system needs to be capable of adapting to the changes. In the following, an example of adapting to new postures with the online OCSVM is presented. A video sequence in which a person sits at a new position is recorded, selective frame samples are shown in Figure 5.28.

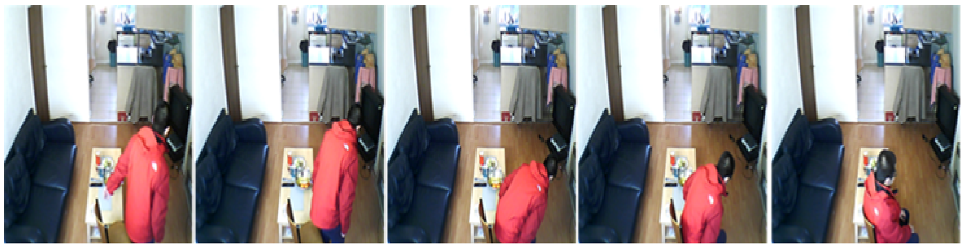


Figure 5.28. Selective frame samples of the person sitting at a new position.

The variations of AR values of the sequence are shown in Figure 5.29, from which it can be seen that no large movement is detected because the AR value does not exceed the threshold so no falls are reported. The online OCSVM scheme is applied to update the trained OCSVM model described in Section 5.4.2 using the posture features extracted from this video sequence, the evolutions of the OCSVM results for this sequence during the updating procedure are shown in Figure 5.30. Different parameters C are chosen for this online updating procedure and two phenomena can be observed: (1) the OCSVM value for this new sitting posture increases with time, initially the OCSVM value for this new posture is below the threshold but this value will increase over time and exceed the threshold with the aid of the online OCSVM scheme for updating; (2) a larger C value means a faster updating rate. In a real application, a proper value C needs to be chosen to control the updating time for new postures.

In order to illustrate the advantage of OCSVM model updating, an example that a person who sits at the new position bends quickly to pick something (selective frame samples are shown in Figure 5.31) is presented. And in Figure 5.32, the AR values and OCSVM results are calculated for this fast bending sequence example. The AR results in (a) show that large movement (bending) is detected during the initial frames; (b) shows the re-

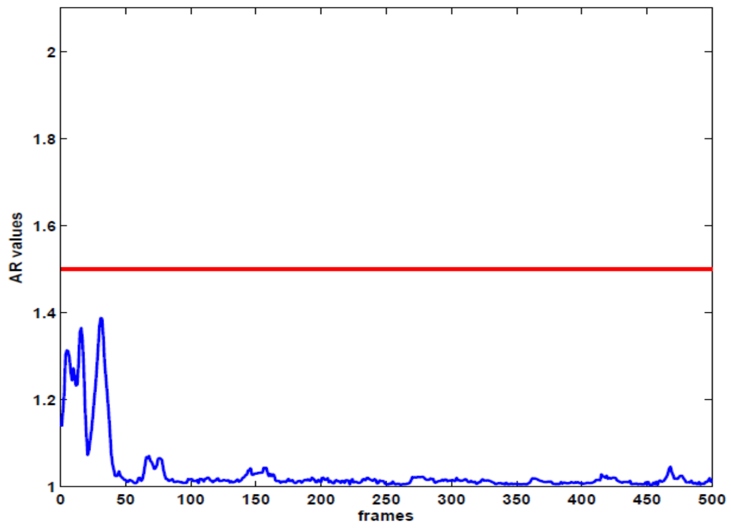


Figure 5.29. The variations of AR values for a sequence of sitting at a new position.

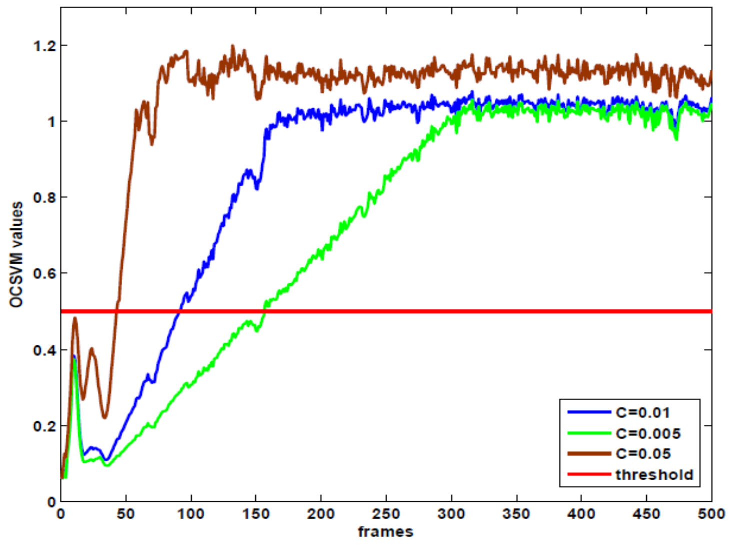


Figure 5.30. The evolutions of OCSVM values using the online scheme with different penalty parameters C.

sults of the OCSVM models with and without the updating procedure (for a fair comparison, same parameters are adopted for the two models, as shown in Section 4.2), from which it can be observed that results of the two models all fall below the threshold during the initial frames when bending occurs.

However, for the updated OCSVM model, the value returns to be above the threshold when the person recovers to sit after a very short interval, so according to Rule 2 no falls are reported. Additionally, for the OCSVM model without updating, the value is always less than the threshold and a fall is wrongly reported when the abnormal state lasts for longer than 20s (around the 100th frame).

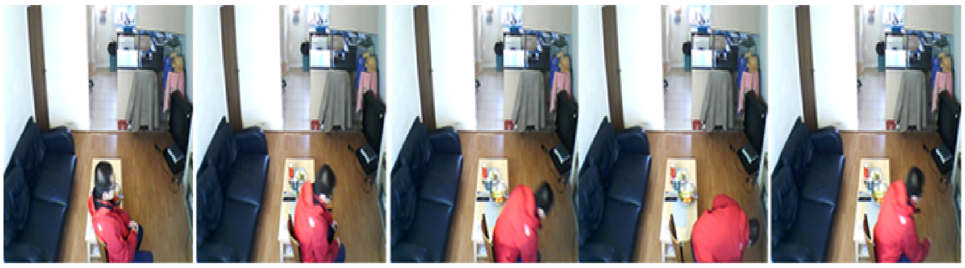


Figure 5.31. Selective frame samples of fast bending to pick something at a new position.

5.8 Summary

This chapter presented two fall detection systems based on supervised and unsupervised methods. The codebook background subtraction was used to extract the silhouettes and some post-processing technique was applied to solve the background subtraction errors caused by some environmental changes in a real home environment. Some features (projection histogram and shape-structure features) which can describe postures in details were extracted and used to construct the corresponding DAGSVM classifier or online OCSVM together with some simple features (ellipse features and position features). The classification results of DAGSVM or online OCSVM, together with some rules were used to distinguish falls and non-falls. Experimental results showed that acceptable fall detection results can be obtained for both supervised and unsupervised fall detection systems.

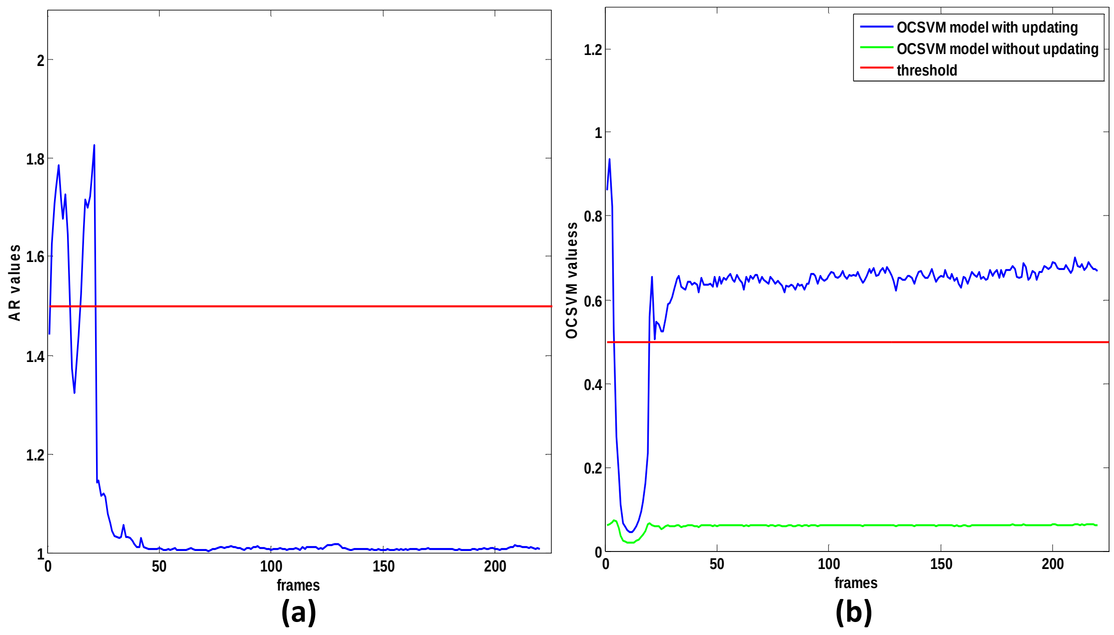


Figure 5.32. The variations of AR values (a) and OCSVM values (b) for a video sequence of fast bending while sitting at new position. For comparison, both the values of the OCSVM models with and without updating are presented.

Both the supervised fall detection system and unsupervised fall detection system have their own merits and drawbacks. For the fall detection based on the supervised method, initially a large posture dataset composed of postures from different persons is needed for training the classifier, which is inconvenient. But the trained classifier can be immediately applied for classifying postures for fall detection for a particular elderly person as long as the classifier is well trained. For the unsupervised classifier, it provides a person-specific solution, there is no need to collect different people's postures and the normal postures captured from the normal activities of that specific person (the person who is monitored) are used to construct the normal model. However, during the time period for collecting enough normal postures for model building, falls can not be detected.

Besides the information obtained from postures, other types of information can be applied to achieve a more robust fall detection system. One such type of information can be obtained from audio as proposed in [18] and [21], the floor sounds generated for fall activities and non-fall activities are different and this sound information is helpful to distinguish fall activities and non-fall activities when poor results of posture features are obtained (such as the elderly person wearing clothes whose colour is very similar to the background thus generating bad posture segmentation results). The sound and posture information together can be used to compose a more robust multimodal (audio and video) fall detection system, which is a possible next step for the research.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This thesis proposes different types of computer vision based fall detection technique for detecting fall activities of an elderly person living alone at home. 2D and 3D features are extracted from video sequences recorded by one or multiple cameras, and an analytical algorithm (Chapter 3) or machine learning algorithms (Chapter 4 and Chapter 5) were then exploited to process the extracted features for detecting fall activities. Evaluations were performed in both a lab and real home environments.

Chapter 3 proposed a simple and robust fall detection system by using three cues: head velocity, shape and movement amplitude cues. The head was tracked by a particle filter using gradient and colour information; the velocity of the head between consecutive frames was then estimated from the head tracking results. The codebook background subtraction method and moment-based ellipse fitting were applied to fit the human body region to an ellipse. Shape change was reflected by the velocities of the orientation angle and the ratio of the lengths of the two axes of an ellipse. Finally, the movement amplitude of a person was estimated by a C_{motion} value obtained from the MHI for a further confirmation of a fall when both the head velocity and shape change exceed certain threshold values. Experiments on a six person dataset in the lab environment showed that by combining these three cues–head velocity, shape change and movement amplitude, a simple but robust fall detection system can be achieved which can distinguish 12 falls performed in two directions from 24 different types of non-fall activities when the optimal threshold values, found empirically, were used.

Chapter 4 proposed a fall detection scheme based on 3D features and a data fitting model scheme. Cameras were firstly calibrated by the popular Tsai camera calibration method and a 3D person was then constructed from the obtained codebook background subtraction results from two calibrated cameras. 3D features, including the 3D position, velocity and orientation information corresponding to fall activities were extracted to build the model for distinguishing fall activities and non-fall activities. Three types of models: single Gaussian model, mixture of Gaussians model and OCSVM model were constructed from a training dataset including 40 short video clips of different fall activities simulated by four people, from which the 3D features features were extracted for model construction. The comparison results on a test dataset including 40 fall activities and 40 non-fall activities simulated by another four people showed that the OCSVM model achieves the best performance with 100% fall detection rate and 0% false detection rate with the optimal threshold; moreover, by introducing different weights to samples in the training dataset, a fuzzy OCSVM was obtained which achieved a better performance if the training dataset was contaminated by some outliers.

Chapter 5 presented two fall detection systems based on supervised and unsupervised learning methods. The codebook background subtraction approach was used to extract the postures and certain post-processing techniques were applied to solve the background subtraction errors caused by some environmental changes in a real home environment. Some features (projection histogram and shape-structure features) which can describe postures in detail were extracted and used to construct the corresponding DAGSVM classifier or online OCSVM with some simple features (ellipse features and position features). The classification results of DAGSVM or online OCSVM, together with certain rules were used to distinguish falls from non-falls. Experimental results in a real home environment showed that acceptable fall detection results can be obtained, with 97.08% fall detection rate and 0.8% false detection rate for a 15 person dataset using the supervised fall detection system, and 100% fall detection rate with no false detection for a single monitored person using the unsupervised method.

In summary, this thesis has proposed effective schemes for solving the problems listed at the end of Chapter 2: 1. Instead of using only one particular feature for distinguishing falls, different types of features were exploited and a decision of fall or non-fall was made by comparing the features with proper thresholds in an organized way. 2. Directional invariant 3D features were extracted from the reconstructed 3D person, which was obtained from the background subtraction results from video sequences recorded by calibrated cameras, and the 3D features were used to train a particular data fitting model to distinguish fall and non-fall activities by using only one threshold. 3. Supervised and unsupervised classifiers based on posture features, together with certain types of information (floor information and movement amplitude information) were applied to construct more robust fall detection systems, which were thoroughly evaluated on datasets recorded in a real home environment representative of an assisted living application.

6.2 Future work

The research can be extended from algorithm aspects, information aspects and hardware equipment aspects:

1. From algorithm aspects, elegant computer vision algorithms can be

applied as the components of the proposed fall detection system. A more efficient background subtraction method as proposed in [82] can be applied for better human body segmentation, with online self-adaptive mechanism to update model parameters. In this way, the change of the illumination, which is a common phenomenon in the indoor environment, could be compensated; for head tracking, an improvement of the SIR filter as proposed in [83] could be used to obtain a more accurate head tracking result even in a cluttered home environment. Besides, considering sometimes there will be more than one moving object at home (such as the elderly person with a pet), some object classification algorithms in [84] and [85] could be applied to determine the moving blob corresponding to the human region. These algorithms can be added into the current fall detection system as new modules, which is helpful in enhancing the performance of the proposed fall detection systems.

2. From information aspects, instead of using only the video information, more types of information (such as acoustic information) could be extracted to compensate for the limitations of video information (such as poor video information will be obtained when an elderly person wears clothes whose colour is similar to the background). A more robust multimodal fall detection system could be constructed by fusing different types of information (video information and audio information) for improving the performance. The limitation of the audio information is that it can easily be affected by background noises, especially TV; however, this problem can be ameliorated by using the modern blind source separation (BSS) technique [86] to reduce this type of interference.

3. Instead of using only ordinary cameras, more advanced hardware equipment could be used. As presented in [87], a new-emerging Kinect sensor could be applied, which can obtain additional depth information for better human body segmentation when an elderly person wears clothes whose colour is similar to the background.

References

[1] J. Platt, N. Cristianini, and J. Shawe-Taylor, “Large margin DAGs for multiclass classification,” *Advances in Neural Information Processing Systems, MIT Press*, pp. 547–553, 2000.

[2] “http://www.guardian.co.uk/world/2009/jul/20/census-population-ageing-global,” *accessed May 2012.*

[3] F. Kemp and R. Acheson, “Care in the community - elderly people living alone at home,” *Community Medicine*, vol. 11, no. 1, pp. 21–26, 1989.

[4] C. Lord and D. Colvin, “Falls in the elderly: detection and assessment,” *Annual international conference of the IEEE Engineering in Medicine and Biology Society*, vol. 13, no. 4, pp. 1938–1939, 1991.

[5] L. Lin, F. Chiou, and H. Cohen, “Slip and fall accident prevention: a review of research, practice and regulations,” *Journal of Safety Research*, vol. 26, no. 4, pp. 203–212, 1995.

[6] S. Sadigh, A. Reimers, R. Andersson, and L. Laflamme, “Falls and fall-related injuries among the elderly: a survey of residential-care facilities in a Swedish municipality,” *Journal of Community Health*, vol. 29, no. 2, pp. 129–140, 2004.

[7] X. Yu, “Approaches and principles of fall detection for elderly and patient,” *10th International Conference on e-Health Networking, Applications and Services-Healthcom, Singapore*, 2008.

[8] "http://www.eldertech.missouri.edu/docs/," *accessed December 2012.*

[9] M. Gibson, R. Andres, B. Issacs, T. Radebaugh, and J. Peterson, "The presention of falls in later life. A report of the Kellogg international work group on the prevention of falls by the elderly.," *Danish Medical Bulletin*, vol. 34, no. 4, pp. 1–24, 1987.

[10] S. Lord, C. Sherrington, and H. Menz, "Falls in older people: risk factors and strategies for prevention," *Wiley, New York*, 1998.

[11] D. Karantonis, M. Narayanan, M. Mathie, N. Lovell, and B. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, 2006.

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

[13] M. Estudillo-Valderrama, L. Roa, J. Reina-Tosina, and D. Naranjo-Hernandez, "A proposal of a fall detection algorithm for a multidevice personal intelligent platform," *8th IEEE International Conference on Bioinformatics and Bioengineering, Athens, Greece*, 2008.

[14] M. Estudillo-Valderrama, L. Roa, J. Reina-Tosina, and D. Naranjo-Hernandez, "Design and implementation of a distributed fall detection system-personal server," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 874–881, 2009.

[15] D. Naranjo-Hernandez, L. Roa, J. Reina-Tosina, and M. Estudillo-Valderrama, "Personalization and adaptation to the medium and context in a fall detection system," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 2, pp. 264–271, 2012.

[16] Y. Li, Z. Zeng, M. Popescu, and K. Ho, "Acoustic fall detection using a circular microphone array," *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Buenos Aires, Argentina*, 2010.

[17] B. Mungamuru and P. Aarabi, "Enhanced sound localization," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 34, no. 3, pp. 1526–1540, 2004.

[18] Y. Li, K. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 2, pp. 1291–1301, 2012.

[19] M. Popescu and A. Mahnot, "Acoustic fall detection using one-class classifiers," *31st Annual International Conference of the IEEE EMBS, Minneapolis, Minnesota, USA*, 2009.

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

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

[22] S. Miaou, F. Shih, and C. Huang, "A smart vision-based human fall detection system for telehealth applications," *The Third International Conference on Telehealth (IASTED), Anaheim, CA, USA*, 2007.

[23] S. Miaou, P. Sung, and C. Huang, "A customized human fall detection system using omni-camera images and personal information," *1st Transdis-*

ciplinary Conference on Distributed Diagnosis and Home Healthcare, Arlington, VA, USA, 2006.

[24] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Fall detection from human shape and motion history using video surveillance," *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW), Niagara Falls, Ont., Canada*, 2007.

[25] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Procrustes shape analysis for fall detection," *ECCV 8th International Workshop on Visual Surveillance (VS 2008), Marseille, France*, 2008.

[26] M. Shoaib, R. Dragon, and J. Ostermann, "View-invariant fall detection for elderly in real home environment," *Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT), Singapore*, 2010.

[27] A. Leone, G. Diraco, and P. Siciliano, "An automated active vision system for fall detection and posture analysis in ambient assisted living applications," *2010 IEEE International Symposium on Industrial Electronics (ISIE), Bari, Italy*, 2010.

[28] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Monocular 3d head tracking to detect falls of elderly people," *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), New York City, New York, USA*, 2006.

[29] A. Nghiem, E. Auvinet, and J. Meunier, "Head detection using Kinect camera and its application to fall detection," *The 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), Montreal, Canada*, 2012.

[30] E. Auvinet, F. Multon, A. Saint-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based

on 3-d silhouette vertical distribution," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 2, pp. 290–300, 2011.

[31] C. Bishop, "Pattern recognition and machine learning," *Springer*, 2006.

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

[33] C. Liu, C. Lee, and P. Lin, "A fall detection system using k-nearest neighbor classifier," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7174–7181, 2010.

[34] R. Cucchiara, A. Prati, and R. Vezzani, "An intelligent surveillance system for dangerous situation detection in home environments," *Intelligenza Artificiale*, vol. 1, no. 1, pp. 11–15, 2004.

[35] M. Yu, A. Rhuma, S. Naqvi, J. Chambers, and L. Wang, "Posture recognition based fall detection system for monitoring an elderly person in a smart home environment," vol. 16, no. 6, pp. 1274–1286, 2012.

[36] L. Hazelhoff, J. Han, and P. H. With, "Video-based fall detection in the home using principal component analysis," *In Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems, Juan-les-Pins, France*, 2008.

[37] B. Ni, N. Dat, and P. Moulin, "RGBD-camera based get-up event detection for hospital fall prevention," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Kyoto, Japan*, 2012.

[38] B. Mirmahboub, S. Samavi, N. Karimi, and S. Shirani, "View-invariant fall detection system based on silhouette area and orientation," *IEEE Inter-*

national Conference on Multimedia and Expo (ICME), Melbourne, Australia, 2012.

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

[40] N. Thome, S. Miguet, and S. Ambellouis, "A real-time, multiview fall detection system: A LHMM-based approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1522–1532, 2008.

[41] Z. Htike, S. Egerton, and K. Chow, "A monocular view-invariant fall detection system for the elderly in assisted home environments," *7th International Conference on Intelligent Environments (ICIE), Nottingham, UK*, 2011.

[42] R. Cucchiara, A. Prati, and R. Vezzani, "Vezzani: A multi-camera vision system for fall detection and alarm generation," *Expert Systems Journal*, vol. 24, no. 5, pp. 334–345, 2007.

[43] M. Belshaw, B. Taati, J. Snoek, and A. Mihailidis, "Towards a single sensor passive solution for automated fall detection," *33rd Annual International Conference of the IEEE EMBS, Boston, Massachusetts USA*, 2011.

[44] H. Foroughi, A. Rezvanian, and A. Paziraee, "Robust fall detection using human shape and multi-class support vector machine," *Sixth Indian Conference on Computer Vision, Graphics Image Processing (ICVGIP), Bhubaneswar, India*, 2008.

[45] H. Nait-Charif and S. McKenna, "Activity summarisation and fall detection in a supportive home environment," *Proceedings of the 17th International Conference on Pattern Recognition (ICPR), Cambridge, UK*, 2004.

[46] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Robust video surveillance for fall detection based on human shape deformation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 5, pp. 611–622, 2011.

[47] B. Ristic, S. Arulampalam, and N. Gordon, “Beyond the Kalman filter: particle filters for tracking applications,” *Artech House*, 2004.

[48] A. Kong, J. Liu, and W. Wong, “Sequtential imputations and bayesian missing data problems,” *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[49] X. Xu and B. Li, “Head tracking using particle filter with intensity gradient and colour histogram,” *In IEEE International Conference on Multimedia and Exploration, Amsterdam, The Netherlands*, 2005.

[50] S. Birchfield, “Elliptical head tracking using intensity gradients and color histograms,” *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California*, 1998.

[51] S. Birchfield, “An elliptical head tracker,” *In Proceedings of the 31st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California*, 1997.

[52] K. Nummiaro, E. Koller-Meier, L. V. Gool, and L. V. Gaal, “Object tracking with an adaptive color-based particle filter,” *In Symposium for Pattern Recognition of the DAGM*, 2002.

[53] M. Yu, S. Naqvi, and J. Chambers, “A robust fall detection system for the elderly in a smart room,” *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Dallas, TX, USA*, 2010.

[54] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfnder: Real-

time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[55] T. Horprasert, D. Harwood, and L. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," *In Proceedings of the International Conference on Computer Vision, Corfu, Greece*, 1999.

[56] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Int. Conf. Computer Vision and Pattern Recognition, Fort Collins, CO, USA*, 1999.

[57] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," *European Conf. Computer Vision*, 2000.

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

[59] R. Gonzalez, "Digital image processing," *Third Edition, Pearson Education*, 2008.

[60] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[61] Y. Abdel-Aziz and H. Karara, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry," *Papers from the 1971 ASP Symposium on Close-Range Photogrammetry, Falls Church, VA: American Society of Photogrammetry*, 1971.

[62] Z. Zhang, "A flexible new technique for camera calibration," *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

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

[64] K. Madsen, H. Nielsen, and O. Tingleff, “Methods for non-linear least squares problems,” *Lecture note, Informatics and Mathematical Modelling, Technical University of Denmark, DTU*, 2004.

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

[66] D. Tax, “One-class classification,” *PhD Thesis, TU Delft, NL*, 2001.

[67] J. Magnus and H. Neudecker, “Matrix differential calculus with applications in statistics and econometrics,” *Wiley*, 1999.

[68] J. Russell and E. Adrian, “Performance of Bayesian model selection criteria for Gaussian mixture models,” *Technical Report, University of Washington, USA*, 2009.

[69] J. Kadane and N. Lazar, “Methods and criteria for model selection,” *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 279–290, 2004.

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

[71] P. Hao, “Fuzzy one-class support vector machines,” *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2317 – 2336, 2008.

[72] M. Yu, S. Naqvi, R. Adel, and J. Chambers, “Fall detection in a smart room by using a fuzzy one class support vector machine and imperfect training data,” *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Prague , Czech*, 2011.

[73] D. Tax, “Ddtools, the data description toolbox for Matlab,” May 2012. version 1.9.1.

[74] J. Hsieh, Y. Hsu, H. Liao, and C. Chen, “Video-based human movement analysis and its application to surveillance systems,” *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 372–384, 2008.

[75] L. Chew, “Constrained Delaunay Triangulations,” *Algorithmica*, vol. 4, no. 1, pp. 97–108, 1989.

[76] V. Vapnik, “Statistical learning theory,” *Wiley, New York*, 1998.

[77] J. Friedman, “Another approach to polychotomous classification,” *Technical report, Stanford Department of Statistics*, 1996.

[78] J. Sun, C. Zheng, X. Li, and Y. Zhou, “Analysis of the distance between two classes for tuning SVM hyperparameters,” *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 305–318, 2010.

[79] J. Kivinen, A. Smola, and R. Williamson, “Online learning with kernels,” *IEEE Transactions on Signal Processing*, vol. 100, no. 10, pp. 1–12, 2010.

[80] A. Bobick and J. Davis, “The recognition of human movement using temporal templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[81] F. Heijden, R. Duin, D. Ridder, and D. Tax, “Classification, parameter estimation and state estimation: An engineering approach using MATLAB,” *Wiley*, 2004.

[82] H. Bhaskar, L. Mihaylova, and A. Achim, “Video foreground detection based on symmetric alpha-stable mixture models,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 8, pp. 1133–1138, 2010.

[83] N. Bouaynaya, "An online motion-based particle filter for head tracking applications," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Philadelphia, PA, USA*, 2005.

[84] Y. Chen, L. Zhu, A. Yuille, and H. Zhang, "Unsupervised learning of probabilistic object models (POMs) for object classification, segmentation, and recognition using knowledge propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1747–1761, 2009.

[85] F. Lecumberry, A. Pardo, and G. Sapiro, "Simultaneous object classification and segmentation with high-order multiple shape models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 625–635, 2010.

[86] S. Naqvi, M. Yu, and J. Chambers, "A multimodal approach to blind source separation of moving sources," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 895–910, 2010.

[87] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos, "A viewpoint-independent statistical method for fall detection," *21st International Conference on Pattern Recognition (ICPR), Tsukuba Science City, Japan*, 2012.