Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Ανίχνευση πτώσης μέσω ανάλυσης βίντεο (Video-based Fall Detection)

Ελένη Πιπέλη (ΑΜ: 2090)
E-mail: epipeli@yahoo.com

Σταυρούλα-Νίκη Γιάννη (ΑΜ: 2047)
E-mail: niki-g_@hotmail.com

Ηράκλειο – 01/10/2015

Επιβλέπων καθηγητής: Αν. Καθηγητής Τσικνάκης Εμμανουήλ

Επιτροπή Αξιολόγησης: Καθηγητής Δημοσθένης Ακουμιανάκης,
Αν. Καθηγητής Αθανάσιος Μαλάμος

Υπεύθυνη Δήλωση: *Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχουμε αναφέρει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνουμε ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής του Τ.Ε.Ι. Κρήτης.*

Acknowledgments

First and foremost, we would like to offer our deepest gratitude to our supervisor Ass. Professor Tsiknakis Manolis for his help and support throughout this thesis.

Besides our supervisor, we would like to thank Dr Matthew Pediaditis for his continuous help and guidance in the completion of this thesis, as well as for his immeasurable patience.

Also, we would like to thank Giorgos Vavoulas and all our fellow and lab mates in the Biomedical Informatics & eHealth (BMI) lab at the Technological Educational Institute of Crete.

Last but not least, a big thank you goes to our families and friends for their love and support all these years.

Abstract

Falls are one of the greatest causes of accidents for elderly people that can result in limited activity or even death. The purpose of this thesis is the development of a video-based fall detection system. For this purpose, videos of simulated falls from a freely available video dataset with everyday living activities and different kinds of falls were used. For the purpose of using effectively the features extracted, four of the eight cameras were selected from the dataset.

A first step in fall detection concerns the detection of moving person within the video sequence, the so called background-foreground subtraction. For this a simple method is to assume that the first frame of the video is in fact the background and then perform a subtraction between the pixel values of that first frame with the following ones, but this approach has poor results in dynamic scenes, so in our approach the mixture of Gaussians subtraction algorithm, which is a more advanced technique, was used. Then, the moving object (i.e. the person) tracking occurs by using two different methods that result in different outputs. The first method used was the Kalman filter in order to compute the center of mass and the second method used was the dense optical flow velocity vectors.

For the evaluation of our system the 10-Fold Cross Validation technique was used, as well as a percentage split method. The percentage split scenario consists of a 33% train and 66% test case and a 66% train and 33% test case.

The results obtained showed that the Optical Flow method performs better than the Kalman filtering approach, as well as that the percentage split scenario of the 66% train and 33% test case presents better results than both the Cross Validation technique and the 33% train and 66% test case respectively. The classification algorithm that produces those best results is the Sequential Minimal Optimization (SMO) algorithm.

Σύνοψη

Οι πτώσεις είναι μία από τις μεγαλύτερες αιτίες ατυχημάτων για τους ηλικιωμένους ανθρώπους που μπορεί να οδηγήσουν σε περιορισμένη κινητική δραστηριότητα ή ακόμη και θάνατο. Ο σκοπός αυτής της πτυχιακής είναι η ανάπτυξη ενός συστήματος ανίχνευσης πτώσης μέσω βίντεο. Για το σκοπό αυτό, χρησιμοποιήθηκε ένα ελεύθερα διαθέσιμο σύνολο δεδομένων βίντεο, το οποίο περιέχει καθημερινές δραστηριότητες διαβίωσης και προσομοιώσεις διαφορετικών ειδών πτώσεων (π.χ. πλάγια πτώση, εμπρόσθια πτώση, κλπ). Για το σκοπό της αποτελεσματικότερης χρήσης των χαρακτηριστικών γνωρισμάτων που εξάγαμε, τέσσερις από τις οκτώ κάμερες επιλέχθηκαν από το ελεύθερο σύνολο δεδομένων των βίντεο.

Το πρώτο βήμα για την ανίχνευση πτώσεων αποτελεί την ανίχνευση του κινούμενου ατόμου μέσα στην ακολουθία του κάθε βίντεο. Έτσι θα πρέπει να γίνει διαχωρισμός του στατικού φόντου από το μη-στατικό. Για την επίτευξη αυτού του βήματος, μπορεί να χρησιμοποιηθεί μία απλή μέθοδος, η οποία θεωρεί ότι η πρώτη εικόνα του βίντεο είναι και το στατικό φόντο και έπειτα να γίνει μία απλή αφαίρεση των τιμών των εικονοστοιχείων των επόμενων εικόνων της ακολουθίας του βίντεο με την πρώτη εικόνα, όπου αν το αποτέλεσμα της αφαίρεσης μεταξύ των τιμών αυτών δεν είναι ίσο με το 0, τότε θεωρεί ότι το κινούμενο άτομο έχει ανιχνευθεί. Η συγκεκριμένη προσέγγιση έχει πολύ φτωχά αποτελέσματα στις δυναμικές σκηνές και για αυτό το λόγο στην πτυχιακή αυτή χρησιμοποιήθηκε μία πιο προηγμένη τεχνική που είναι βασισμένη σε μίγμα από Gaussians. Έπειτα, το επόμενο βήμα είναι η παρακολούθηση του εντοπισμένου πλέον κινούμενου αντικειμένου (δηλαδή του ατόμου), η οποία επιτυγχάνεται με την χρήση δύο διαφορετικών μεθόδων, με διαφορετικά αποτελέσματα η καθεμία. Η πρώτη μέθοδος που χρησιμοποιήθηκε ήταν το φίλτρο Kalman, για την παρακολούθηση του κέντρου μάζας του κινούμενου αντικειμένου και την εξαγωγή των διανυσμάτων ταχύτητας του. Η δεύτερη μέθοδος ήταν η dense optical flow, από την οποία υπολογίστηκαν τα διανύσματα ταχύτητας ολόκληρης της μάζας του κινούμενου αντικειμένου.

Έπειτα από την επεξεργασία των επιλεγμένων βίντεο, ώστε να ταιριάζουν στις απαιτήσεις μας, χρησιμοποιήθηκε ένα εργαλείο επισήμανσης για την εποπτευόμενη, μη-αυτοματοποιημένη επισήμανση των καρέ του κάθε βίντεο, το οποίο αντιπροσωπεύει την πραγματικότητα, ώστε να μπορέσει να πραγματοποιηθεί η σωστή αξιολόγηση του συστήματος της πτυχιακής.

Για την αξιολόγηση του συστήματος αυτού, χρησιμοποιήθηκε η τεχνική 10-Fold Cross Validation, καθώς και μία μέθοδος διάσπασης ποσοστού (percentage split). Η μέθοδος διάσπασης ποσοστού αποτελείται με την σειρά της από δύο σενάρια, το 33% σετ εκπαίδευσης και 66% σετ δοκιμής και το 66% σετ εκπαίδευσης και 33% σετ δοκιμής.

Τα αποτελέσματα της πτυχιακής έδειξαν ότι η μέθοδος με Optical Flow αποδίδει καλύτερα αποτελέσματα από την μέθοδο με το φίλτρο Kalman. Επίσης, το σενάριο της διάσπασης ποσοστού με 66% σετ εκπαίδευσης και 33% σετ δοκιμής, παρουσιάζει πιο βέλτιστα αποτελέσματα και από την 10-Fold Cross Validation μέθοδο, καθώς και από την περίπτωση διάσπασης ποσοστού με 33% σετ εκπαίδευσης και 66% σετ δοκιμής. Ο αλγόριθμος κατηγοριοποίησης που παράγει αυτά τα βέλτιστα αποτελέσματα είναι ο Sequential Minimal Optimization (SMO).

Table of Contents

List of Figures

List of Tables

List of Acronyms

ARFF	Attribute-Relation File Format
CCA	Cross Correlation Analysis
CCL	Connected Component Labeling
CDT	C Development Tools
CUDA	Computer Unified Device Architecture
EER	Equal Error Rate
FN	False Negatives
FP	False Positives
FSM	Finite State Machine
GMM	Gaussian Mixture Model
GNU	General Public License
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
HSV	Hue Saturation Value
IBk	Instance-Based k (k-NN)
ID3	Iterative Dichotomizer 3
IDE	Integrated Development Environment
JDT	Java Development Tools
K-NN	K-Neatest Neighbor
MAP	Maximum A Posteriori
MHI	Motion History Image
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MML	Minimum Message Length
MoG	Mixture of Gaussians
MVFI	Motion Vector Flow Instance
NCC	Normalized Cross Correlation
NN	Neural Network
OpenCV	Open Source Computer Vision
POSIT	Pose from Orthography and Scaling with Iterations
RANSAC	Random Sample Consensus
SAKBOT	Statistic and Knowledge-Based Object Tracker
SDK	Software Development Kit
SGD	Stochastic Gradient Decent
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
TN	True Negatives
TOF	Time Of Flight
TP	True Positives
WEKA	Waikato Environment for Knowledge Analysis

1. Introduction

1.1 Problem Overview

In around 35 years from now, i.e. by 2050, it's estimated that more than one in each group of five people will be aged 65 or over. In this age group, falling is one of the most serious life-threatening events that can occur, as approximately one-third to one-half of the population aged 65 and over experience falls on a yearly basis and half of these elderly do fall repeatedly [**1**]. So, the automatic detection of falls would help reducing the time of arrival of medical caregiver, and accordingly reducing the mortality rate [**2**]. Falls are the leading cause of injury in elderly people and the leading cause of accidental death in those 75 years of age and older [**3**]. Also, more than 90% of hip fractures occur as a result of falls in persons aged 70 years and over [**4**].

Therefore, falls are one of the greatest causes of accidents for elderly people that can result in limited activity or even death. More than one third of people 65 years old and older fall every year and in many cases the falls happen more than once, resulting to visits to the emergency department [**5**]. Most of the times a heart attack or a deterioration of a patient with chronic obstructive pulmonary disease leads to a fall. Other factors that lead to a fall are diabetes, alterations of balance, and previous falls. Falls not only cause physical injury such as many disabling fractures [**4**], they also have dramatic psychological, medical, and social consequences. The emerging picture is that falls are not a rare occurrence among older persons.

The high occurrence of falls results to a high health care cost, for example in the U.S., falls cost overall 10 billion dollars per year [**6**]. In case of an emergency, in a telecare home environment, the person can call for help by pressing a button connected to a wireless wearable device or by the use of smart phones. If the device has an accelerometer, like the smart phones, it can automatically detect falls and send an emergency signal. After the fall, some of the people may have lost their consciousness and may not be able to use their devices or as it often happens, elderly people may forget their devices or even forget to charge the battery of the device. The solution to all these problems is computer vision systems with the use of cameras, which record the human activity. From this recording some features are extracted that describe real time human activities with image and video processing methods.

According to [**7**] a fall can be described in four phases:

- The **Pre-fall phase** includes daily activities with sudden movements towards the ground, e.g. sitting on a chair.
- The **Critical phase** includes the actual fall that is very short and is detected by a sudden movement towards the ground or an impact with the floor.
- The **Post-fall phase** is the period of time where the person is motionless on the ground after the fall.
- The **Recovery phase** occurs if the person is able to stand back up after the fall.

The following figure (Fig. 1) presents the four phases of a fall.

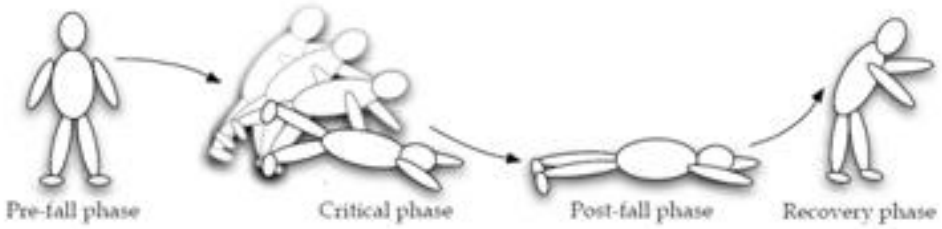

Figure 1: The four phases of a fall, as shown in [7]

1.2 Scope and Objectives

The above information and the overview of the problem to be addressed as well as its importance, we believe, clarifies the motivation of the work in the present thesis.

The overall scope of this thesis is two fold: (i) to execute an in depth review of available approaches, methods and techniques for vision-based (human) fall detection and (ii) to implement a specific solution to the video-based fall detection problem, using publicly available datasets and state-of-the-art methods and techniques.

In the context of the present work the fall detection process is completed in three phases. In the first phase the images are separated in background and foreground. The foreground includes the objects of interest, e.g. the moving person, and the background represents the static environment. In the second phase the moving objects are detected and their motion tracking occurs. The extracted features that measure the motion or the posture of each moving object are subsequently computed. Finally, these features (e.g. the silhouette, the motion vectors, the histograms) are used to perform the necessary classification task, i.e. to train an appropriate classifier.

1.3 Outline

In the first chapter we describe the problem and the purpose of this thesis. We also present the structure of the thesis. In the second chapter we present the State of the Art of the video based fall detection systems, while we select thirty three out of fifty papers and focusing our research on only vision based fall detection systems, the date a paper was published, marker less methods and indoor activities. We review and analyze the various approaches related to the issues of background subtraction, feature extraction, the actual fall detection method and report their results.

In the third chapter we describe the software used for our method, the dataset preparation, and the background subtraction and feature extraction methods that we apply on the dataset. We also describe the classification methods and we present a summary of the classification algorithms we used in order to obtain the results.

In the fourth chapter, we present our results and the classification algorithms that achieved the best results. Finally in the fifth chapter, we present our conclusion and in the sixth chapter, we talk about a continuation on this thesis for future work.

2. Review of the State of the Art

There has been significant interest in falls both from a research and commercial perspective for many years. A variety of approaches have been taken technologically towards the automated detection of human falls with varying degrees of accuracy. A number of attempts have been made to monitor not only falls, but also to generally monitor daily activities unobtrusively, i.e. without attaching devices to the body, and to subsequently detect and ultimately prevent falls accordingly.

Many approaches that are using accelerometers to detect falls have been proposed [**8**]. In those approaches, a change in body orientation from upright to lying that occurs immediately after a large negative acceleration indicates a fall. However, generally despite all the research dedicated to fall detection, there still isn't a 100% reliable algorithm that catches all falls with no false alarms. Furthermore, a limited number of research studies have been conducted concerning the issue of fall prediction via monitoring and modeling patients' behavior in order to take protective actions that prevent the occurrence of falls.

This section presents a short review of research work performed in order to highlight various solutions proposed for tackling the problem of fall detection and prevention from different perspectives. Our focus of this review activity is on published research efforts using vision-based systems for the detection and, more importantly, the prediction of falls.

In [**9**] human motion in video is modeled using Hidden Markov Models (HMM) in addition to using the audio track of the video to distinguish a person simply sitting on a floor from a person stumbling and falling or in other words to use the impact sound of a falling person as an additional clue of a fall and to avoid false positive classification of a falling event. The video analysis algorithm starts with the detection of a moving region in the current image. A bounding box of the moving region is determined and parameters describing the bounding box are estimated. In this way, a time-series signal describing the motion of a person in video is extracted. The wavelet transform of this signal is computed and used in HMMs, which were trained according to possible human being motions. The reported results indicate that the wavelet transform domain signal provides better results than the time-domain signal. The reason for this is the fact that wavelets capture sudden changes in the signal and ignore stationary parts of the signal [**10**]. The proposed approach has been proven to be computationally efficient and can be implemented in real-time. However, due to using a low cost standard camera instead of an omnidirectional camera it is hard to estimate moving object trajectories in a room. So, authors concluded that the proposed fall detection method can achieve a better performance, if an omnidirectional camera is available.

In [**11**] the use of 'unusual inactivity' detection as a clue for fall detection is demonstrated. Motion trajectories extracted from an omnidirectional video are used to determine falling persons, however without considering audio information to understand video events. In [**12**], the person was tracked using an ellipse and inferring falling incident when target person is detected as inactive outside normal zones of inactivity like chairs or sofas. The tracker uses a coarse ellipse model and a particle filter to cope with cluttered scenes with multiple sources of illumination. Summarization in terms of semantic regions is demonstrated using acted scenes through automatic recovery of the instructions given to the subject.

Authors in [**13**] propose a vision-based fall detection system for the elderly and patients at home or in health-care centers. The system proposed uses an omnidirectional camera to avoid blind spot (where no light rays captured). The recognition features proposed for the system

include angle and length variation associated with the body line and motion history images. Given these features, a simple thresholding and decision tree technique is adopted for fall detection. Experimental results show that the proposed system can solve the problems of light source glimmer and static abandoned objects. The system successfully recognized most fall events, however it disregard the type of falling as recognition errors occur when a normal walking person is classified as being falling.

Also, in [**14**] authors used the normalized vertical and horizontal projection of segmented object as feature vectors. So, in the first of the study of Foroughi et al, a method was proposed to detect various posture-based daily life and unusual events in a typical elderly monitoring application in a video surveillance scenario with a particular interest to the problem of fall detection. The proposed approach provided a useful clue for detection of different behaviors via applying a combination of best-fit approximated ellipse around the human body, projection histograms of the segmented silhouette, and temporal changes of head position. Then extracted feature vectors are fed to a Multilayer Perceptron (MLP) Neural Network for precise classification of motions and determination of fall event. The main advantage of the proposed system is that it is able to detect type of fall incident (forward, backward or sideway), while most existing fall detection systems are only able to detect occurrence of fall behavior. The approach proposed in this research has been applied to a dataset of videos in a simulation environment and nothing has been mentioned regarding time or computations cost/complexity for the system to be applied in real life environments.

Finally, authors in [**15**] used a similar approach in addition to considering the k-Nearest Neighbor (k-NN) algorithm and evidence accumulation technique to infer human postures for fall detection. Furthermore, they used the speed of fall to differentiate real fall incident and an event where the person is simply lying without falling. Authors concluded that due to evidence accumulation technique, an event will not be instantaneously detected; however it takes an average of 8 frames to accumulate enough evidence of fall detection.

For the purpose of a more structured categorizing the existing methods for vision-based fall detection systems, the review in [**16**] is taken as reference, which describes the architecture of such systems that is separated into four steps. The first step is the Video Acquisition, where the determination of the way that data is collected occurs. There are different approaches that can be used in order to build a camera system, and in [**16**] three of these approaches are described:

- The use of one Omni-directional camera
- The use of one infrared camera, and
- The use of many wide-angle cameras

The second step is the Background Subtraction and Moving Object Tracking, where the search of the appropriate method in order to estimate the background and detect the moving object occurs. According to [**16**], the most popular background subtraction methods are:

- Frame differencing
- Median filtering
- Mixture of Gaussians, and
- Other methods

A good method of estimating the moving object is the use of a Kalman filter on the data obtained by the foreground model from the background subtraction step, as well as the Optical flow method and/or the Particle filtering method respectively.

The third step in the architecture of a video-based fall detection system is the Video Processing and Feature Extraction step. After the moving object detection is completed, the system focuses on the moving objects. Measurements of useful parameters of the objects are essential for the fall recognition, so the following features are important for the completion of this step:

a) Human shape, which consists of:
   - Histograms
   - Main axes
   - Human Centroid
   - Bounding Box and/or Fitting Ellipse
b) Fall Angle
c) Height/Width ratio, also referred to as Aspect ratio
d) Inactivity
e) Variation of motion
f) Head detection
g) Posture

The fourth and final step of the system's architecture is the Fall Detection, in which the performance of the fall recognition algorithms that are used is evaluated.

Table 1 below shows an overview of the **first two steps** mentioned above. Table 2 presents an overview of **the third step** mentioned above, and Table 3 shows an overview of **the results of the final step**. Chapters 2.1, 2.2, 2.3 and 2.4 describe these steps in depth for all the existing video-based fall detection systems reviewed.

AUTHOR	VIDEO ACQUISI-TION		BACKGROUND SUBTRACTION & MOVING OBJECT TRACKING			
	Single View	Multi-View	Frame Differencing	Median Filtering	Mixture of Gaussians	Other
[37]Anderson et al.(2006)	✓				✓	
[43]Anderson et al.(2008)		✓	✓			
[17]Belshaw et al.(2011)	✓ (Omni-Directional camera)		✓			✓ (Single Gaus-sian)
[23]Cañas et al.(2007)		✓ (Calibra-ted)	✓			
[40]Chen et al	✓					✓ (Bayesian Approach)
[39]Cucchiara et al.(2007)	✓					✓ (SAKBOT)

[36]Debard et al.(2011)		✓	✓	✓		✓ (Cross Correla-tion Analysis)
[28]Diraco et al.(2010)	✓ (ToF-Infrared Calibrated)				✓	✓ (Kalman, Bayesian, RANSAC)
[14]Foroughi et al_a(2008)		✓	✓			
[34]Hazelhoff et al.(2008)		✓	✓		✓	
[29]Jansen et al.(2007)	✓ (ToF-Infrared Calibrated,3D)		✓			✓ (Morphological Filter)
[44]Khan & Habib(2009)	✓		✓			
[32]Kroputaponchai & Suvonvorn (2009)		✓	✓			✓ (Morphological Filter)
[18]Lee & Mihailidis (2005)	✓ (Omni-Directional camera)		✓ (Connected Component Labeling)			
[27]Lin et al.(2007)	✓ (Pan-tilt-zoom camera)		✓ (Motion Vectors & Connected Component Labeling)	✓		
[46]Liu & Zuo(2012)	✓		✓			
[30]Liu et al.(2010)	✓		✓ (Connected Component Labeling)			✓ (Mean Filtering)
[33]Meunier et al. a (2008)	✓		✓			✓ (Canny, Gaussian)
[35]Meunier et al. b		✓ (Calibrated)		✓		
[13]Miaou et al.(2006)	✓ (Omni-directional camera, MapCam)		✓ (Connected Component Labeling)			✓ (Morphological Filter)

[11]Nait-Charif and McKenna	✓ (Omni-Directional) Uncalibrated					✓ (Particle Filter)
[15]Nasution and Emmanuel	✓				✓ (Gaussian Stauffer)	
[38]Olivieri et al.(2012)					✓	✓ (Morphological Filter)
[41]Rhuma et al	✓		✓			
[45]Rougier et al.(2006)	✓ (Calibrated)		✓			
[35]Rougier et al_a(2011)	✓	✓	✓			✓ (Canny edge detector)
[7]Rougier et al_b(2011)	✓	✓ (3D)	✓			✓ (Canny, Particle Filter)
[19]Schulze et al.(2009)	✓ (Omni-directional camera, Fish-eye lens)		✓			
[31]Shieh & Huang (2012)		✓	✓			✓ (Mean Filtering, Morphological Filter, Sobel Filter)
[20]Spehr et al.(2008)	✓ (Omni-directional camera)	✓	✓ (YUV)			✓ (Gaussian)
[25]Stone & Skubic (2011)		✓ (Calibra-ted)	✓ (Connected Component Algorithm)			✓ (3D-Voxel Space)
[26]Thome et al.(2008)		✓ (Calibra-ted)			✓ (Gaussian Stauffer)	
[9]Toreyin et al.	✓		✓			
[21]Vishwakarma et al.(2007)	✓ (Omni-directional camera)				✓ (YCbCr, Connected Component Algorithm)	
[42]Wang et		✓	✓			

al.(2011)						
[11]Willems et al.(2009)	✓ (Omni-directional camera & Side/Front view)			✓		✓ (NCC)

Table 1: An overview of the *Video Acquisition* and the *Background Subtraction and Moving Object Tracking* steps, which are the first two steps of the architecture of the existing video-based fall detection systems as mentioned in [16].

AUTHOR	VIDEO PROCESSING & FEATURE EXTRACTION									
	Human Shape				Fall Angle	Height /Width Ratio	Inacti-vity	Variation of Motion	Head Detection	Posture
	Histograms	Main Axes	Human Centroid	Bounding Box & Best Fitting Ellipse						
[37]Anderson et al.(2006)				✓		✓				✓
[43]Anderson et al.(2008)			✓				✓	✓		✓
[17]Belshaw et al.(2011)						✓	✓			
[23]Cañas et al.(2007)	✓ HSV			✓				✓		
[40]Chen et al		✓	✓	✓			✓	✓		✓ (Distance map of human skeleton)
[39]Cucchiara et al.(2007)	✓									✓
[36]Debard et al.(2011)		✓		✓	✓	✓			✓	
[28]Diraco et al.(2010)			✓			✓	✓			✓ (Discret Reeb Graph)
[14]Foroughi et al.(2008)	✓			✓					✓	✓
[34]Hazelhoff et al.(2008)		✓		✓	✓	✓	✓		✓ (Skin color	✓

									model)	
[29]Jansen et al.(2007)			✓	✓		✓				✓
[44]Khan & Habib(2009)	✓			✓		✓		✓ (Motion History Image)		✓
[32]Kroputaponchai & Suvonvorn (2009)		✓		✓	✓	✓		✓		
[18]Lee & Mihailidis (2005)		✓	✓	✓			✓	✓		✓
[27]Lin et al.(2007)	✓		✓	✓				✓		
[46]Liu & Zuo (2012)			✓	✓		✓				
[30]Liu et al.(2010)	✓			✓		✓		✓		✓
[33]Meunier et al. a (2008)	✓			✓			✓			✓
[35]Meunier et al.b										✓
[13]Miaou et al.(2006)				✓		✓				✓
[11]Nait-Charif and McKenna			✓	✓			✓	✓		
[15]Nasution and Emmanuel	✓			✓	✓			✓		✓ (k-NN)
[38]Olivieri et al.(2012)	✓			✓				✓ (Motion History Image & MVFI)		✓
[41]Rhuma et al	✓ (Logpolar)	✓	✓	✓						✓
[45]Rougier et al.(2006)				✓	✓	✓			✓ (POSIT, Particle Filter)	
[35]Rougier et al_a(2011)	✓ Logpolar			✓			✓			✓
[7]Rougier et al_b(2011)	✓		✓	✓ (head ellipse)		✓	✓	✓ (Motion History Image)	✓ (3D)	✓
[19]Schulze							✓	✓		✓

et al.(2009)								(Moore automaton)		
[31]Shieh & Huang (2012)										✓
[20]Spehr et al.(2008)			✓		✓		✓	✓		✓
[25]Stone & Skubic (2011)										✓
[26]Thome et al.(2008)				✓ (MBR)	✓	✓				✓ (3D)
[9]Toreyin et al				✓				✓		✓ (HMM)
[21]Vishwakarma et al.(2007)			✓	✓	✓	✓				✓
[42]Wang et al.(2011)				✓ (GrabCut)			✓			✓ (HoG)
[22]Willems et al.(2009)	✓VPH	✓		✓	✓	✓				

Table 2: An overview of the *Video Processing and Feature Extraction* step, which is the third one of the architecture of the existing video-based fall detection systems as mentioned in [16**].**

2.1 Video Acquisition

The first step in a fall detection method is the collection of data. Vision-based systems use video sequences, captured by either a **single view calibrated o**r **un-calibrated** camera or a **multi-view** surveillance system. Moreover, it is mentioned where the camera or cameras are mounted in the room and their type.

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. It is the process of finding the intrinsic parameters of the camera that produced a given photograph or video.

There are two categories of the methods analyzed here, that relate to the dimension of the calibration objects:

- **3D reference object based calibration:**

  Camera calibration is performed by observing a calibration object whose geometry in 3D space is known with very good precision. Calibration can be computed very efficiently. The calibration object usually consists of two or three planes orthogonal to each other. This approach requires an expensive calibration system and a complicated setup.

- **2D plane based calibration:**

Techniques in this category require observing a planar pattern shown at a few different orientations. Because almost anyone can make such a calibration pattern by themselves, the setup is easier for camera calibration.

The highest accuracy is obtained by using a 3D system, so it should be used when accuracy is needed most and when the cost of using 3D system is justified. In general, in computer vision, researchers are mainly using calibration with a 2D system which seems to be the best choice in most situations because of its ease of use and relatively high accuracy.

Furthermore, there are three approaches to monitoring a room:

- One **Omni-directional** camera, which is a camera with a 360-degree field of view in the horizontal plane, or with a visual field that covers the entire sphere. It is usually mounted on the ceiling. An Omni-directional camera is used in [**17**], [**18**], [**13**](where an Omni-directional camera called "MapCam" is used), [**19**], [**20**], [**21**] and [**22**].
- One or many **wide-angle** cameras. These types of cameras allow more of the scene to be included in the field of view and are usually wall mounted. This approach is used in the majority of the methods described in the literature. The cameras are calibrated in [**23**], [**7**] and [**24**], in conjunction with a 3D reconstruction of the scene; in [**25**], [**26**] and [**27**], where a wide-angle Pan-tilt-zoom camera is used.
- One **infrared** camera, which is a device that captures an image using infrared radiation, similar to a common camera that forms an image using visible light. In this category of cameras, the **Time-of-Flight** (**TOF**) camera is a range imaging camera system that resolves distance based on the known speed of light, measuring the time-of-flight of a light signal between the camera and the subject for each point of the image.
- The **Illumination unit,** which illuminates the scene and the illumination normally, uses infrared light to make the illumination unobtrusive. This approach is used combined with calibration and is found in [**28**] and [**29**].

2.2 Background Subtraction & Moving Object Tracking

The next step after collecting the data is to use background subtraction methods to detect any moving object in the image in order to result to the detected person. The most popular techniques are Frame Differencing, Median Filtering and Mixture of Gaussians.

2.2.1 Frame Differencing

Frame differencing is a technique where the computer calculates the difference between two video frames. If the pixels have changed, there was apparently something changing in the image (moving for example). Most techniques use image smoothing (blur) and thresholding, to distinct real movement from noise. The background is to be the previous frame.

This technique is sensitive to noise and variations in illumination. Furthermore, another disadvantage is that the detected objects must be continuously moving. If an object stays still for more than a frame period (1/fps), it becomes part of the background.

On the other hand, this method does have two major advantages. The first one is the modest computational load and the second one is that the background model is highly adaptive. Since

the background is based solely on the previous frame, it can adapt to changes in the background faster than any other method (at precisely 1/fps).

This technique is used in most of the methods presented here, combined with connected component labeling (CCL) in [**18**], [**27**] , [**30**], [**13**], [**25**], with motion vectors in [**27**], with mean filtering to eliminate noise in [**30**] and [**31**], with morphological filtering (dilation and erosion) to remove noise in [**29**], [**32**] and [**13**]; also with 3D voxel space to improve human segmentation in [**25**].

Some other methods combined with the frame differencing technique are the single Gaussian in [**17**], [**33**] and [**20**], the mixture of Gaussians in [**34**], the Canny edge detector in [**33**], [**35**] and [**7**] also with the use of particle filters in [**7**].

2.2.2 Median Filtering

Median filtering is a non-linear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is widely used in image processing, because under certain conditions, it preserves edges while removing noise. When it is used for background subtraction, the median of the previous n frames is used as the background model, contrary to the frame differencing technique, where the previous frame is used as the background model.

This technique is used in [**36**], [**27**], [**24**], [**22**], with cross correlation analysis (CCA) in [**36**] and normalized cross correlation (NCC) in [**22**] for shadow detection.

2.2.3 Mixture of Gaussians

Mixture of Gaussians (MoG) is an algorithm, where the values of a particular pixel are modeled as a mixture of adaptive Gaussians. It is a mixture model because multiple surfaces appear in a pixel and it is adaptive because lighting conditions change. At every iteration, Gaussians are evaluated using a simple heuristic to determine which ones are most likely to correspond to the background, which means that the Gaussians with the most supporting evidence and least variance should correspond to the background. Pixels that do not match with the "background Gaussians" are classified as foreground. Foreground pixels are grouped using 2D connected component analysis. This method utilizes Bayesian frameworks, color and ingredient information and the detected objects are allowed to become part of the existing background model, without destroying it. It also provides fast recovery of the scene.

This technique is used in [**37**], [**28**], [**34**], [**38**], [**26**] and [**21**]. In addition, Kalman Filtering and RANSAC is used in [**28**], a morphological filter is used in [**38**], Stauffer MoG is used in [**26**] and YCbCr with Connected Component Algorithm is used in [**21**].

In [**39**] the SAKBOT system, which is a system for detecting and tracking moving objects in traffic monitoring and video surveillance applications, is used; in [**40**], a Bayesian approach to obtain the foreground mask is used.

2.3 Video Processing & Feature Extraction

This step focuses on the detected moving objects. The most important features are Human Shape, Height/Width Ratio, Fall Angle, Inactivity, Variation of Motion, Head Detection and Posture.

2.3.1 Human Shape

Human Shape and the characteristics from its analysis, which are:

- **Bounding Box** or **Fitting Ellipse**
- **Main Axes (size)**
- (Vertical) **Histograms**
- **Human Centroid**

Most methods make use of the Human Shape feature and its characteristics, where in [**23**] an HSV histogram is used, in [**35**] and [**41**] a log-polar histogram, in [**7**] a head ellipse, in [**26**] a Minimum Rectangle Box as the person's bounding box and in [**42**] a bounding box via GrabCut, which is an image segmentation method based on graph cuts.

Graph cut is an optimization technique that achieves robust segmentation when foreground and background color distributions are not well separated.

2.3.2 Height/Width Ratio

Height/Width ratio is the height/width ratio of the **bounding box** mentioned above, also known as **aspect ratio**, or the height/width ratio of a person's **main axes**, also mentioned above. This feature is commonly used in the methods presented.

2.3.3 Fall Angle

Fall angle is the angle between the person's main axes and the ground. If the angle is less than 45 degrees or more than 135 degrees it is presumed that a fall occurred. Fall angle is also used by the majority of the methods presented.

2.3.4 Inactivity

Inactivity is a small period of time without motion (about 5 seconds). This feature is used in [**43**], [**17**], [**28**], [**34**], [**18**], [**33**], [**35**], **[7],** [**19**], [**20**], [**42**], [**41**] and [**40**].

2.3.5 Variation of Motion

The occurrence of a fall may be detected through a large movement and a large variation of the **main axes** and the **fall angle**. In general, if there is a large variation in the velocity or the angle of the detected person. In [**44**], [**38**] and **[7]** a Motion History Image (MHI) is used for the human tracking in order to do the extraction. Also in [**38**] a spatio-temporal motion representation is proposed, called Motion Vector Flow Instance (MVFI) template. In [**19**] a Moore Automaton is used. The Moore Automaton is a self-operating finite-state machine, whose output values are determined solely by its current state. This is also used in [**40**].

2.3.6 Head Detection

Head detection is the tracking of the detected person's head, because it is usually visible in the scene and has a large movement variation during a fall. This feature is used in [**36**], [**14**], [**34**], [**45**] and [**7**], where in [**34**] a skin color model is used and in [**45**] the POSIT is used for head localization, with particle filtering.

2.3.7 Posture

Posture is the state of the extracted silhouettes. They are labeled as "standing", "walking", "lying", etc, in order to train a classifier to detect whether there is a fall incident or not.

This feature is also used very often in the literature, where in [**28**] a Discreet Reeb Graph is used. A Discreet Reeb Graph describes the evolution of the level sets of a real-valued function of an object on a topological space. It is used in computational geometry and computer graphics, including surface segmentation. In [**42**] a HOG (Histogram of Oriented Gradients) is used. In [**40**], the extracted human skeleton is used in order to categorize the postures.

2.4 Performance of reported fall detection algorithms

The **fall detection** is accomplished with combining different techniques or their results from the "Background Subtraction & Moving Object Tracking" and the "Video Processing & Feature Extraction" sections, and is the final step of a video-based fall detection system. The results in each paper are referred on a percentage of True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Sensitivity, Specificity, Accuracy and Error Rate. In some papers the results are described with a sentence or a paragraph.

An overview of the results of this final step is presented in the following table (Table 3).

AUTHOR	FALL DETECTION	RESULTS
[37]Anderson et al.(2006)	A Hidden Markov Model (HMM)	"The train models are able to correctly recognize the corresponding activities"
[43]Anderson et al.(2008)	A Threshold Based fall detection and fuzzy inference	"The system correctly classified all of the falls"
[17]Belshaw et al.(2011)	Neural Networks (a multi-layered Perception NN)	TP 97% and FP 5%
[23]Cañas et al.(2007)	An Evolutive Algorithm for 3D people tracking with 3Dpoints and with 3d prisms	"the 3D prism is more computer intensive than 3Dpoint which is more successful"
[40]Chen et al.(2010)	A Threshold on the motion of the human and on the orientation of the ellipse.	Fall detection rate 90.91% and false alarm rate 9.09%
[39]Cucchiara et al.(2007)	a Hidden Markov Model (HMM)	"in case of bottom occlusions the classifier fails, in case of middle occlusions the tracking and posture are misled and when occluded part is too large, posture classifier fails after some frames".
[36]Debard et al.(2011)	A Support Vector Machine (SVM)	a recall of 0.896 (± 0.194) and precision of 0.257 (± 0.073)

[28]Diraco et al.(2010)	A Threshold Based fall detection	80% Efficiency and 97.3% Reliability at 0, 4 m
[14]Foroughi et al.(2008)	A 4-layered MLP network with back propagation learning schema	Sensitivity 92.80% and specificity at 97.60% and Reliable average recognition rate at 91.12%
[34]Hazelhoff et al.(2008)	A multi-frame Gaussian classifier	the algorithm can operate at real time speed with more than 85% fall detection rate and 100% without large occlusions
[29]Jansen et al.(2007)	A linear calibration method of height above the ground of the silhouette classifying the pose	10 sequences sitting subject, 10 sequences lying down subject and 10 sequences walking subject all correctly categorized
[44]Khan & Habib(2009)	A Threshold Based fall detection and inactivity	Front Fall 19 /20 events, Backward Fall 20/20 and Lying down 18 / 20 events
[32]Kroputaponchai & Suvonvorn (2011)	A Threshold Based fall detection	Best Result 80% Fall Detection, 1FP Worst Result 60% Fall Detection, 2FP
[18]Lee & Mihailidis (2005)	Simple signal detection theory	77% TP, 23% FN, 5% FP, 95% TN
[27]Lin et al.(2007)	A Threshold Based fall detection	correctness ratio ≈ 93%, miss ratio 13%, real time high accuracy and FP ratio 0%
[46]Liu & Zuo(2012)	A Threshold Based fall detection	“Effectively prevent misjudgments increase of accuracy of detection results and good robustness.”
[30]Liu et al.(2010)	K-NN classifier and threshold based fall detection	total accuracy rate is ≈ 84.4% where lying down accuracy rate is ≈ 86.62% fall incident detection is ≈ 82.22% the system leads to false alarm events due to similarity of lying and falling
[33]Meunier et al.(2008)	Full Procrustes Distance between 2 consecutive human shapes	Sensitivity 95.5%, Robust Specificity 96.4% Realistic Accuracy 98.89%

[35]Meunier et al. b (2011)	A Threshold on the time of the person lying on the floor in 5sec.	"with 4 cameras the system achieved almost 100% sensitivity & specificity "
[13]Miaou et al.(2006)	A simple decision threshold algorithm with and without personal information (e.g. height and weight) based on Width-Height Ratio	with personal information : Accuracy 81% ,Sensitivity 90% and Specificity 86% and without personal information : Accuracy 70% ,Sensitivity 78% and Specificity 60%
[11]Nait-Charif and McKenna	A Threshold on the Inactivity of the detected object using a MAP estimation of a Gaussian mixture model	All Falls were correctly detected
[15]Nasution and Emmanuel	A k-Nearest Neighbor (k-NN) algorithm and evidence accumulation technique to infer human postures for fall detection. Also, the speed of fall to differentiate real fall incident and an event where the person is simply lying without falling was used	Robust Recognition rate > 90%
[38]Olivieri et al.(2012)	N-fold cross validation training (known sequences) & K-NN classifier (unknown sequences)	Fall : MVFI 99% , MHI 85% , Silhouette 92.0 %
[41]Rhuma et al.()	A threshold on the posture of the human (lie or bend) and another threshold inside the grand region. A third threshold on the time the conditions above are kept.	31 out of 32 (96.88%) falls can be detected while only 3 out of 64 (4.7%) non falls were mistaken as falls.
[45]Rougier et al.(2006)	A Threshold Based fall detection	" a majority of normal activities are not detected as falls and the detection rate is 2 out of 3 falls"
[35]Rougier et al_a(2011)	A Gaussian Mixture Model (GMM) classifier with full Procrustes distance or mean – matching cost and Inactivity	Human Shape Deformation with full Procrustes Distance Equal Error Rate (EER) 3.8% and Mean-Matching Cost 4.6%

[7]Rougier et al_b(2011)	I. A Threshold Based fall detection with Head Tracking II. And with Human-Eva dataset	I.10 /10 falls with Vertical Velocity and 9 / 10 falls with Head Height II. mean error ≈ 5% in 5m
[19]Schulze et al.(2009)	Variation of Motion & Inactivity	"Under artificial lighting from ceiling excellent results were achieved. If the main light entered through the windows, shadows led to tracking errors and undetected falls"
[31]Shieh & Huang (2012)	A Software Pipelining Mechanism	"the algorithm can precisely detect falling postures with sensitivity and specificity ≈ 90% 2.1 times through and improving"
[20]Spehr et al.(2008)	Lying Pose Estimation & Orientation of the human body which is calculated by a function	FP 81% and FN 22%.
[25]Stone & Skubic (2011)	Human vs. Nonhuman (a set of heuristic rules is used to classify each voxel object) and Background Updating	"quite good at reducing artifacts due to lighting and moving non human objects"
[26]Thome et al.(2008)	Layered Hidden Markov Model (HMM)	an 82% rate of real falling cases and a 18% FN rate"
[9]Toreyin et al.	An HMM of the wavelet domain data	100% rate of correctly detected falls when audio is also taken under consideration with the video computations
[21]Vishwakarma et al.(2007)	A Threshold Based fall detection and State Transition with 2-State Finite State Machine (FSM) are used for Fall Confirmation	"in Omni-single Accuracy 94%, Specificity 96% and Sensitivity 90% and in Indoor- single Accuracy, Specificity and Sensitivity 100%"
[42]Wang et al.(2011)	Lying Pose Estimation	"it outperforms other pose estimation approaches"
[22]Willems et al.(2009)	A Threshold Based fall detection	Correct 85%, 0% FP and 15% FN

Table 3: An overview of the results of the *Fall detection* step of the existing video-based fall detection systems as mentioned in [16].

2.4.1 Classification schemes

Threshold-based fall detection is when a threshold is applied on an extracted feature, such as fall angle, height/width ratio and variation of motion to determine a fall. The best results with this technique are presented in the following papers. The authors in [**46**] report that "*Effectively preventing misjudgments increases of accuracy of detection results and good robustness*" and also in [**21**] a State Transition with 2-State Finite State Machine (FSM) is used for fall confirmation. With this approach promising results have been achieved with one person only in the scene. Specifically in the Omni-single case results with Accuracy of 94%, Specificity of 96% and Sensitivity of 90% have been achieved and for the Indoor – single subject case, the results obtained indicate an Accuracy, Specificity and Sensitivity of 100%.

A ***threshold-based fall detection and fuzzy inference*** is used in [**43**], where the system correctly classified all of the falls and in [**24**] a threshold is used at the time of the lying person on the floor which is 5 seconds. The authors state that "*with 4 cameras the system achieved almost 100% sensitivity and specificity*". Threshold-based fall detection with (i) Head Tracking and (ii) the Human-Eva dataset is used in [**7**]. The results reported indicate correct classification of 10 out of 10 falls with vertical velocity and 9 out of 10 falls with head height. Also head localization with Human-Eva dataset again in [7] resulting at a mean error of approximately 5% in 5m. A further threshold-based technique is ***Variation of Motion & Inactivity***, which is used in [**19**]. The authors report that under artificial lighting from ceiling excellent results were achieved, whilst if the main light entered through the windows, the resulting shadows led to tracking errors and undetected falls.

A ***Hidden Markov Model (HMM) (or Layered HMM)*** classification is used in [**37**], stating that "The train models are able to correctly recognize the corresponding activities", which are "walking", "kneeling" and "falling". In [**26**] the results show an 82% rate of correct fall detections and an 18% FN-rate. In [**39**] it is mentioned that in case of bottom occlusions the HMM posture classifier fails, while in case of middle occlusions the tracking and posture are misled, and when occluded part is too large the classifier fails after some frames.

A ***k-NN classifier*** and a threshold based fall detection are used in [**30**] where it is reported that the total accuracy rate is ≈ 84.4%, where lying down accuracy rate is ≈ 86.62% and fall incident detection is ≈ 82.22%.A k-NN classifier is a supervised machine learning technique for learning a function from training data which classifies objects based on closest training data.

Lying Pose Estimation is used in [**42**], where the authors report that it outperforms other pose estimation approaches. This technique provides a detailed representation of human bodies and uses information from segmentation. Orientation of the human body which is calculated by a function, whose result is either standing or lying, is used in [**20**] with a FP rate of 81% and a FN rate of 22%.

A ***linear calibration method*** of the height above the ground of the silhouette is used for classifying the pose in [**29**] with 10 sequences of a sitting subject, 10 sequences of a lying down subject and 10 sequences of a walking subject all correctly categorized.

Another reported technique uses ***Full Procrustes Distance*** between 2 consecutive human shapes, used in [**33**] with Sensitivity 95.5%, Robust Specificity 96.4% Realistic Accuracy 98.89%. A Gaussian Mixture Model (GMM) classifier with full Procrustes distance or mean – matching cost and Inactivity is used in [**35**], where Human Shape Deformation with full Procrustes Distance Equal Error Rate (EER) 3.8% and Mean-Matching Cost 4.6%. Procrustes analysis is a statistical shape analysis with object adjustment.

Human vs. Non-human (a set of ***heuristic rules*** is used to classify each voxel object) and Background Updating is used in [**25**] where the results are "quite good at reducing artifacts due to lighting and moving non-human objects".

A ***Software Pipelining Mechanism*** is used in [**31**] where they report that "the algorithm can precisely detect falling postures with sensitivity and specificity ≈ 90% 2.1 times through and improving".

A ***multi-frame Gaussian classifier*** is used in [**34**], which reports that the algorithm can operate at real time speed with more than 85% fall detection rate and 100% without large occlusions.

A ***Support Vector Machine*** (SVM) is a fall detector which classifies a timeslot (by its features) as normal or abnormal event. This technique is used in [**36**] with a recall of 0.896 (± 0.194) and precision of 0.257 (± 0.073).

A fall detection with ***Neural Networks (NN)***, a ***MLP***, is done in [**17**] with TP 97% rate and FP 5% rate. A 4-layered MLP network with back propagation learning schema is used in [**14**] the results of which are Sensitivity 92.80% and specificity at 97.60% and Reliable average recognition rate at 91.12%. This measure is essentially based on correctly detected events. An MLP is a supervised NN that can have multiple inputs and outputs and multiple hidden layers with an arbitrary number of neurons. Simple signal detection theory is used in [**18**] where their results were 77% TP rate, 23% FN rate, 5% FP rate and 95% TN rate.

An ***Evolutive Algorithm for 3D people tracking*** with 3D points and with 3D prisms used in [**23**] concludes that the 3D prism is more computer intensive than 3D point which is more successful. In this paper the authors did not publish evaluation metrics but focused more on computational cost.

Finally, a ***N-fold cross validation training*** (for known sequences) & a k-NN classifier (for unknown sequences) is used in [**38**] where their results are MVFI 99%, MHI 85%, (cf. Chapter 2.3.5) Silhouette 92.0 %.Cross Validation used to examine how accurately a predictive model will perform.

3. Methods

This thesis has two different approaches on the matter. The first approach is after computing the foreground model in the Background-Foreground Subtraction step, a Kalman filter is used on it, in order to track the moving object, and its output is used in the Classification step. The second approach is using a dense optical flow technique applied on the foreground model, in order to track the moving object and then using its results to detect if a fall had occurred.

The following figure (Fig. 2) presents the flow chart of our two different approaches.

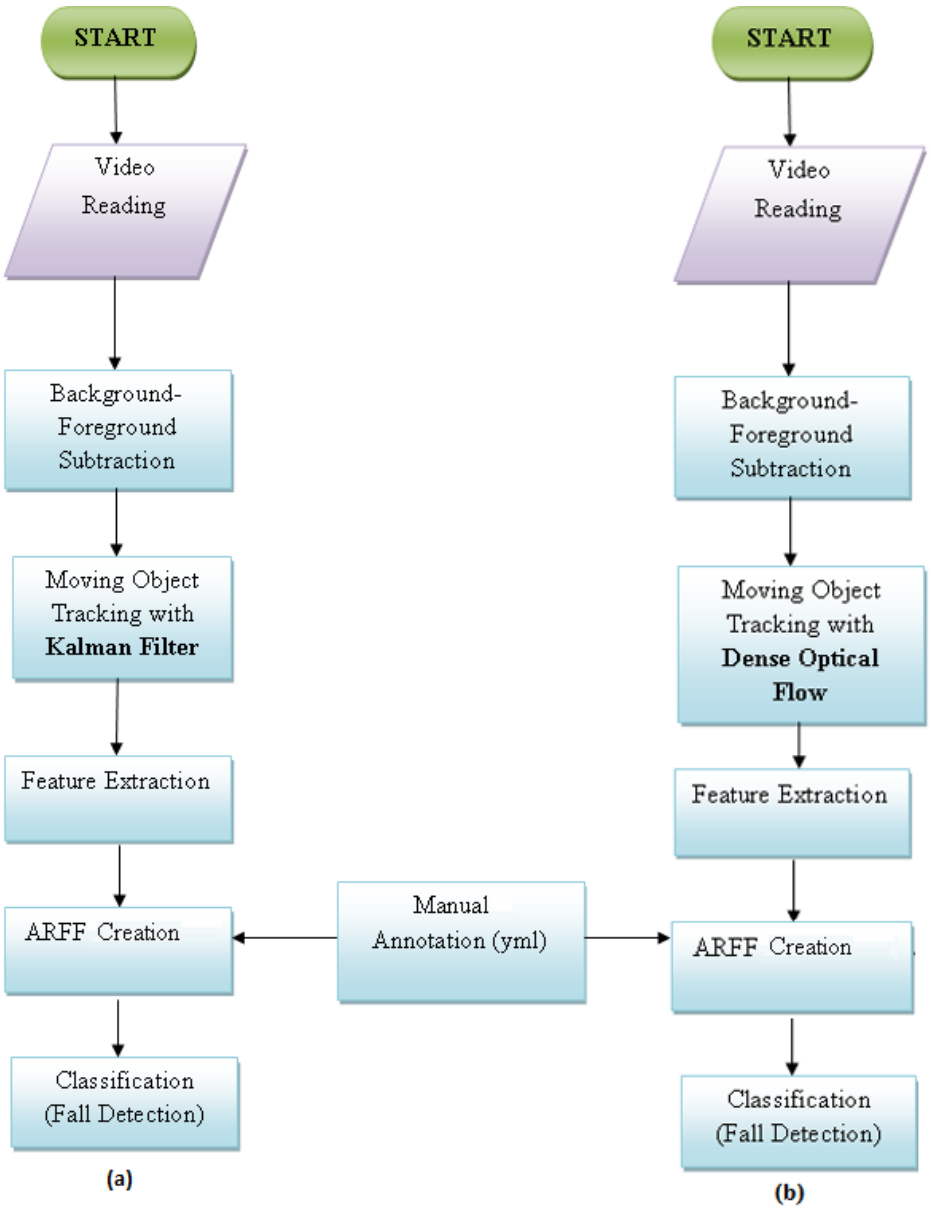

Figure 2: The schematic representation of our system's two approaches; first is the approach with the Kalman Filter for the moving object tracking (a), and second is the Dense Optical Flow technique for the same reason (b)

3.1 Computational Tools

3.1.1 OpenCV

OpenCV (Open Source Computer Vision) is, as its name indicates, an open source computer vision library, developed by Intel in 2000, and available from http://opencv.org/.

It is written in C and C++, although its primary interface is in C++. It also provides of interfaces in Java, Python, Ruby, Matlab, Octave, and other programming languages and since 2010 it also includes an interface based in CUDA GPU.

OpenCV runs on multiple computer platforms, such as Windows, Linux, Android, iOS, and others, because its libraries are cross-platform. It uses the CMake, a software that is cross-platform itself.

Some of the applications that the OpenCV includes are: a Face Recognition System, a Gesture Recognition System, Motion Tracking, Augmented Reality, and others in the Computer Vision and Image Processing areas. In order to support some of these areas OpenCV includes a statistical machine learning library than consists of the k-NN algorithm, the Naïve Bayes Classifier, Artificial Neural Networks, Support Vector Machine, and other classification algorithms [**47**].

3.1.2 Eclipse IDE

Eclipse is an Integrated Development Environment (IDE), first released in 2004 by the Eclipse Foundation (https://eclipse.org/), which contains a workspace and a plug-in system for customization. It is an open source software and through the use of various plug-ins it can be used to develop applications in programming languages such as C, C++, Ruby, Python and others, despite that it is written in Java.

It also includes the Software Development Kit (SDK), which contains the Java Development Tools (JDT) and is for the Java developers. For the C/C++ developers Eclipse includes the C Development Tools (CDT), as well as other development toolkits for various programming languages [**48**].

3.1.3Weka 3 .7

Weka (Waikato Environment for Knowledge Analysis) [**49**] is a Java-implemented machine learning tool. This tool is open source software issued under the GNU General Public License and is used for research, education, and applications.

Weka has a collection of machine learning algorithms for data mining tasks which could be applied to a dataset. Weka contains tools for data pre-processing, classification, regression, association rules and clustering. Weka is used for the classification task in the context of the work of the present thesis.

3.1.4 Microsoft Movie Maker

The Microsoft Movie Maker is a video processing tool used for the modification of the selected videos from the dataset in order to show only one person entering the room and falling. The waiting time after the fall was shortened for that purpose.

3.2 Dataset Preparation

For the experimental work in the content of this thesis, the publically available dataset «Multiple cameras fall dataset» [**50**] was used. The dataset consists of eight inexpensive IP cameras with wide angle, to cover the entire room, as shown in Figure 3.

The dataset is composed of several simulated normal daily activities and falls viewed from all the cameras and performed by one subject. Some examples are shown in Figure 4 and Figure 5. More specifically, it consists of:

- **Normal Daily activities**, such as walking, housekeeping, sitting down/standing up, and
- **Simulated falls**, such as forward falls, backward falls, loss of balance, sideway falls. The falls were done in different directions with respect to the camera point of view. A mattress was used in order to protect the person during the simulated falls.

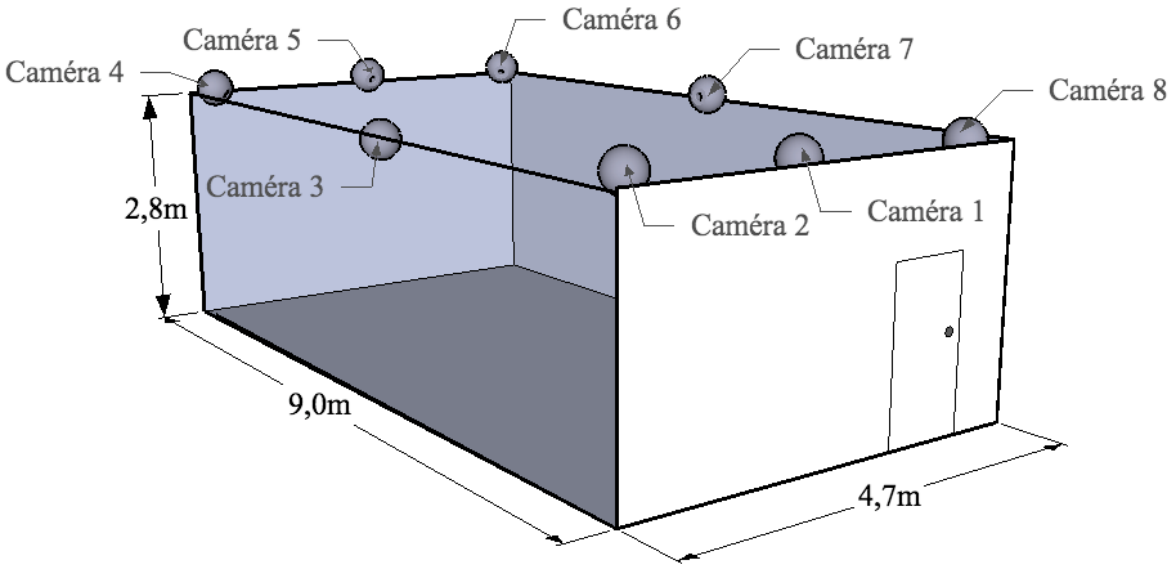

Figure 3: The eight mounted IP cameras in the room [50**]**

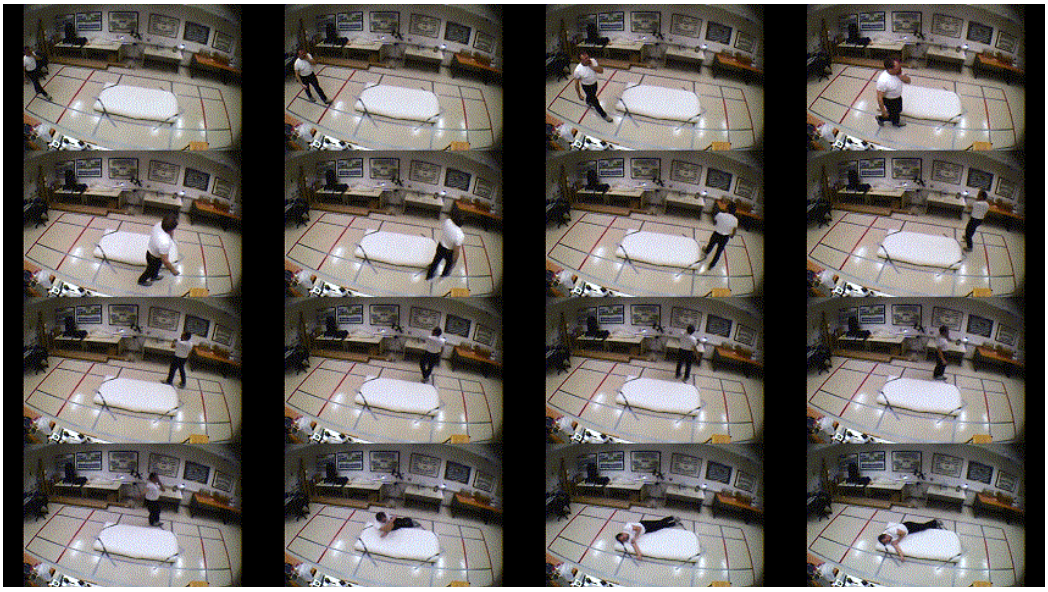

Figure 4: Camera 2-chute 02, showing a side fall

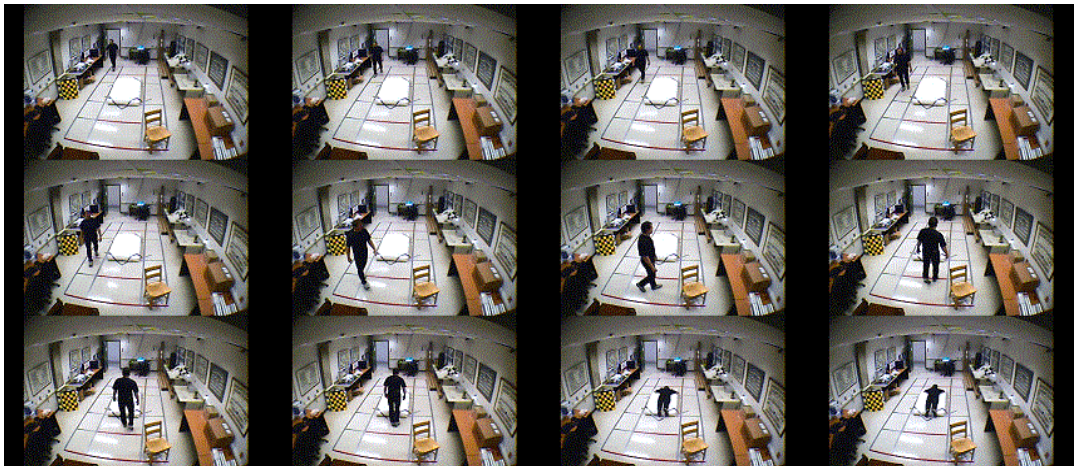

Figure 5: Camera 5-chute 03, showing a front fall

In our work only the simulated falls from the dataset were considered. More specifically, the selected and edited videos used in this approach present one person walking and falling without occlusions. For the purpose of using effectively the features extracted, four of the eight cameras were selected. The four cameras are: Camera 2, Camera 4, Camera 5, and Camera 7 that were mounted in the room as shown in Figure 3. The other cameras of the room were skipped. Nine takes from each camera were used:

- a back fall (ch06).
- a back fall while putting on a jacket (ch01).
- a double fall (ch04) where the person stands up and falls again.
- a front fall (ch03) from the back side of the room.
- a front fall (ch05) from the front side of the room.
- a side fall (ch02) from the left side of the room.
- a side fall (ch08) from the left side of the room.
- a side fall (ch11) from the right side of the room.
- a side fall (ch12) from the left side of the room.

3.3 Feature Extraction

3.3.1 Background Subtraction

The *Background Subtraction* is a highly important filtering step in the video preprocessing stage; therefore, it must be estimated optimally. A simple method is to assume that the first frame of the video is in fact the background and then perform a subtraction between the pixel values of that first frame with the following ones. Then, the pixels with zero values will represent the static background, while the ones with non-zero values will represent the moving objects in the foreground. This method shows poor results, especially in threshold based approaches, because it detects all the movements however small those may be, so it is prone to lighting variation, or furniture rearrangement.

A better approach is to acknowledge that the background is not constant and that the background frame could be maintained dynamically. Therefore, the moving average (e.g. the mean or the median) of each pixel of the last N frames is computed. After the N time period, an update of every pixel will occur, allowing the lighting changes more smoothly, although a

moving object that slowly changes its motion might be considered as one with the background [**51**] [**52**]**.**

In this approach, the second method is used for the background subtraction, with the help of an adaptive Mixture of Gaussians.

3.3.1.1 Mixture of Gaussians

The background modeling consists of two steps. The first step is the Background Initialization and the second step is the Background Update, for adaptation in possible changes in the scene.

In our approach the OpenCV's ***BackgroundSubtractorMOG2()*** function was used. It is based on the Mixture of Gaussians technique, which is among the most fundamental and widely used statistical models. The method constantly adapts the parameters as well as the number of the Gaussian components of the mixture for each pixel. Also, the algorithm can automatically, fully adapt to the scene, by choosing the number of components for each pixel in an on-line procedure. The lighting in the scene could change gradually, e.g. daytime to nighttime, or suddenly, e.g. turning on or off a light in an indoor scene. Also, a new object could enter the scene or a present object could be removed from it. Therefore, the algorithm's training set is updated in order to adapt to those changes. For this reason, the *Mahalanobis distance*[1] is computed, to decide if a sample is close to existing components. If the sample is not close to any of the components, a new component is generated.

This approach has also a shadow detection function for the shadow pixels in the foreground mask. If a pixel is more than twice darker than the background, then it is not a shadow; it becomes background. It consists of multinomial distribution with prior knowledge computed by the Maximum a Posteriori (MAP) and returns a likelihood function, which gives the Maximum Likelihood (ML) estimation. The mixing weights are constraint to sum up to one. In addition, the Minimum Message Length (MML) criterion is used for the selection of proper models for given data; that results in the processing time being very little [**53**].

The MoG gives a representation that is suitable for further processing and is also a better model for the static scenes [**54**].

In the following figure (Fig. 6) the resulting foreground mask of this method is shown.

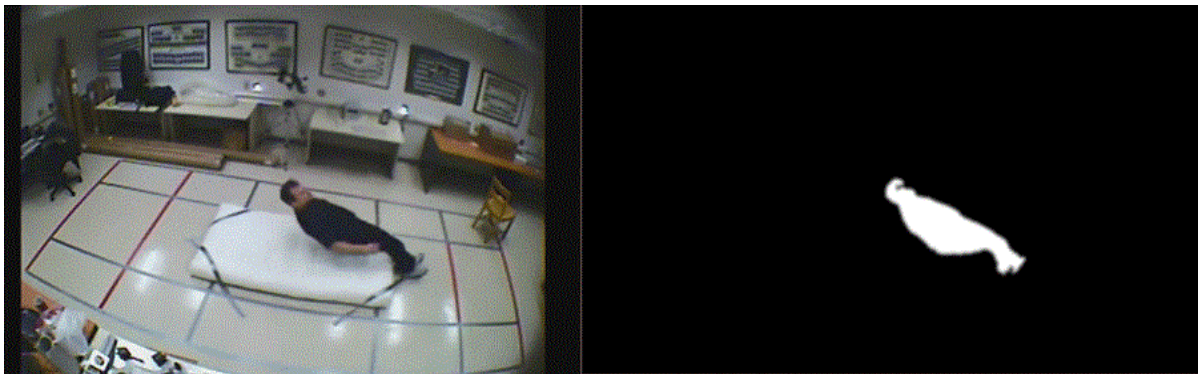

Figure 6: The foreground mask produced by the Mixture of Gaussians background subtraction

[1] The **Mahalanobis distance** is a measure of the distance between a point P and a distribution D, introduced by P. C. Mahalanobis in 1936. It is a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D.

3.3.1.2 Contours

Contours are a list of points that represent line segments either curved or straight, that separate one region from the other, in an image. In order to obtain the contours, an edge detection algorithm must be used for segmentation and because the edges are also features of the image; this method is also known as feature-based segmentation.

In OpenCV, contours are sequences of points that hold information about the location of the next point on the curve. They are computed from the output of the background subtraction (i.e. the foreground), which are binary images. [**55**] [**56**] The contours need to be identified or manipulated for better detection. Some of their properties are:

- Perimeter
- Area
- Rectangle of minimum area(bounding box)
- Convex hull

In our work the OpenCV function ***findContours()*** is used, with the mode ***CV_RETR_CCOMP***; it retrieves all the contours and organizes them into a two-level hierarchy. The top (first) level has the external boundaries whereas the second level has the boundaries of the holes (i.e. the zero regions). It also uses the connected components algorithm to the contour points. The method ***CV_CHAIN_APPROX_SIMPLE*** was also used; it leaves only the end points of the contours, by compressing the horizontal, vertical and diagonal segments. For example, a rectangular contour consists of only 4 points. This method's outputs are sequences of vertices (i.e. polygons) [**55**].

3.3.1.3 Bounding Box & Height/Width Ratio

Every detected object can be enclosed by a bounding box, which is no more than the minimum rectangle that surrounds it. Usually, the bounding box is oriented along the x and y axes of the Cartesian coordination system and consists of four values: the top-left corner with (x, y) coordinates, the top-right corner with (x + w, y) coordinates, the bottom-left corner with (x, y + h) coordinates, and the bottom-right corner with (x + w, y + h) coordinates.

Figure 7: Example of the Bounding box of our approach (i.e. the blue rectangle surrounding the person in the image) and the contours (i.e. the red shape surrounding the person in the image)

It can be used to enclose the contour of the moving object as well, which is also the approach that we take. The features obtained by the bounding box include the origin, the dimensions, and the aspect ratio. The Aspect Ratio, or Height/Width Ratio as mentioned in chapter 2.3.2, is the ratio of the longer side to the shorter one, and it is a parameter often used to describe the shape of an object [**57**].

The following figure (Fig. 7) presents the computed contours as well as the bounding box of this approach.

3.3.1.4 Moments and Center of Mass

The moments are a tool for measurement, analysis, and design. In computer vision they represent a specific weighted average of the intensity of the frame's pixels. They are used after the segmentation of the object is completed, thus only the foreground pixel values are taken under consideration, for the motion analysis of the detected object. Therefore they can be used for binary or gray scale area description. The moments of connected grey leveled areas can also be computed entirely from the contour points of that area, with the use of Green's theorem, thus taking under consideration the positive as well as the negative pixel values of the grey levels. This approach is also faster than computing the spatial moments of an area [**58**]**.** [**59**].

A moment of order (p + q) in digital images is given by the following equation:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p * j^q * f(i,j)$$

where p, q, i, j are coordinates from the point region.

Moments can be used for various purposes such as edge detection, feature extraction, and feature measurement, because they are considered to be features of the detected object. Some of the properties that the moments have are: information about the orientation, the area of the blob, the eccentricity (i.e. how round the object is), and the centroid [**60**] [**51**] [**61**] [**62**] [**59**].

The centroid can be visualized, if we imagine holding a pen and trying to balance it horizontally on the tip of one finger. The region where the finger must be placed in order to achieve that is the centroid of the pen. In general, the center of mass is the mean of all the weighted points by the local density, whereas, the centroid is the mean position of all the points in an object (shape). If an object has homogenous density, then its center of mass is the same as the centroid. The moments that compute the centroid are called central moments and achieve translation invariance by using the following equation:

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p * (j - y_c)^q * f(i,j)$$

in which the x, y, i, j, are coordinates from the point region, and the x_c, y_c are the coordinates of the area's centroid that can be computed by the following equation:

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}}$$

where the M_{00} equals the sum of the grey levels. Therefore the area (A) of the object [**61**] [**62**] [**59**] is given by:

$$A = M_{00}$$

3.3.2 Moving Object Tracking

3.3.2.1 Kalman Filtering

The Kalman filter is first introduced in 1961 [**63**], and remains one of the main methods of *tracking* till the present days. Its basic idea is that under a set of assumptions that are considered reasonable, meaning that these assumptions are not restrictive, the method is helpful for various actual problems that exist in the world. Therefore, it is considered that with a given history of a system's measurements, a model of the state of that system can be build. This state model maximizes the a posteriori probability[2] of the previous measurements, meaning that the distribution, which explains "what really happened", is the most likely one, based on the observed data.

The a posteriori probability can be maximized by keeping a history (not very long) of the previous measurements. In order to achieve that, an iterative update is made to the state model of the system, and only the state model for the next iteration is kept. As a result, the computations are simpler. Figure 8 shows the Kalman filter cycle.

In theory, the most important assumptions that are required for the construction of a Kalman filter are [**55**]**:**

1) The modeled system is linear
2) The noise of the measurements is "white"
3) The noise of the measurements is Gaussian

Therefore the following concepts exist:

1) The Kalman filter is discrete, because it relies on measurements taken between repeated but constant periods of time.
2) The Kalman filter is recursive, because its prediction relies on the state of the present measurements (e.g. position, velocity, acceleration, etc) as well as a guess about what any parts tried to do to affect the situation.
3) If the state model is totally consistent with what is actually happening then the Kalman filter's estimate will eventually assemble what is actually happening.

The Kalman filter during its initialization expects to know the following:

a. The system's mathematical model, that is represented by the matrices A, B , and H
b. The initial estimate about the state of the system, computed in a vector x
c. The initial estimate about the error, given by the matrix P
d. The process in general estimates, as well as the measurement error of the system, computed in the matrices Q and R consequently.

In every time step, the following information must be given:

---

[2] The posterior probability is the probability of the parameters θgiven the evidence X: $p(\theta|X)$. It contrasts with the likelihood function, which is the probability of the evidence given the parameters: $p(X|\theta)$.

a. A vector u that contains the most current state, and is the system's guess considering the situation and how it was affected.
b. A vector z that contains the current measurements used to compute the state of the system.

Finally, after all the above calculations, the following information is obtained: the most current estimate of the actual state of the system and the most current estimate of all the errors in the system.

In general the Kalman filter has as inputs the control vector Un that is the magnitude of the control system in the problem to be solved, and the measurement vector Zn, which contains the real measurement received in this given time.

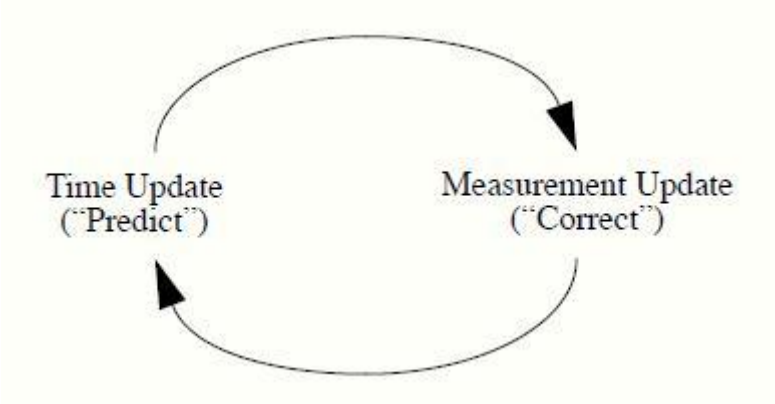

Figure 8: The Kalman filter cycle. As shown, the filter recursively predicts and corrects the state model of the system [45]

Its outputs are the Xn, which is the newest estimate of the current true state, and the Pn, which is the newest estimate of the average error of the state. Its constants are the state transition matrix A, which helps predict the state of the next time step; the control matrix B, which defines the linear equations for any control factors; the observation matrix H, where if the state vector is multiplied by H, it translates into a measurements vector; the estimated process error covariance Q, and the estimated measurement error covariance.

The equations that combine all the above and make the Kalman filter work, are:

State Prediction = $x_{PREDICTED} = Ax_{n-1} + Bu_n$

Covariance Prediction = $P_{PREDICTED} = AP_{n-1}A^T + Q$

Innovation = $\tilde{y} = z_n - Hx_{PREDICTED}$

Innovation Covariance = $S = HP_{PREDICTED}H^T + R$

Kalman Gain = $K = P_{PREDICTED}H^T S^{-1}$

State Update = $x_n = x_{PREDICTED} + K\tilde{y}$

Covariance Update = $P_n = (I - KH)P_{PREDICTED}$

Since a Kalman filter produces an estimate of the system's next state, he State Prediction computes where the detected object is going to be placed in the next frame, the Covariance Prediction predicts the error, the Innovation compares the reality against the prediction, the

Innovation Covariance compares the real error against the predicted one, the Kalman Gain moderates the prediction, the State Update is a new estimate of where we are, and the Covariance Update is a new estimate of the error [**64**].

The detailed example of the Kalman filter at a certain time step is shown in Figure 9, and the Kalman filter in our approach is presented in Figure 10.

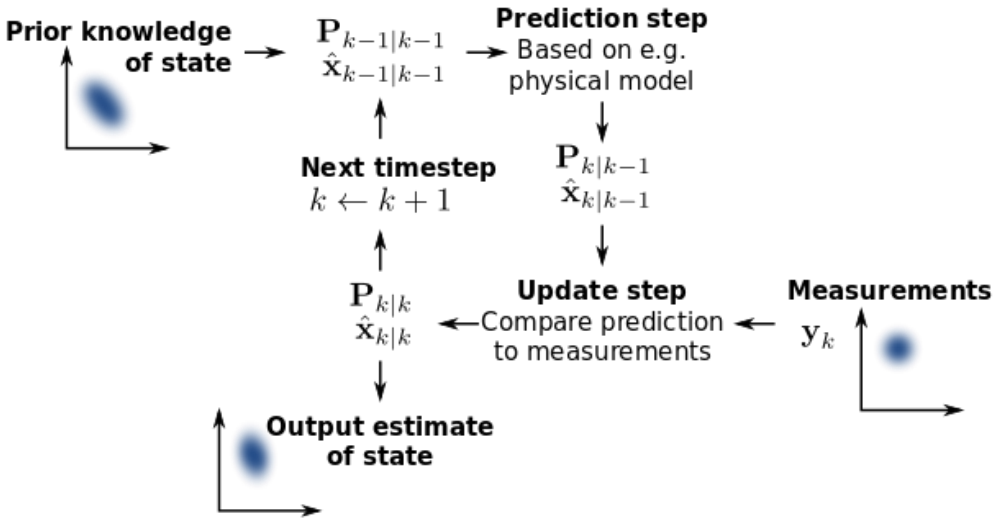

Figure 9: A more detailed example of the Kalman filter, at a certain time step of the system [65**]**

Figure 10: The Kalman filter as shown in our method. The blue "x" is the computed center of mass (centroid of contours), whereas the white "x" is the estimated center of mass by the Kalman filter

3.3.2.2 Dense Optical Flow

In general, if we consider a sequence, there is the idea of motion. Displacements in the physical world caused by the relative motion between an observer (e.g. eye or camera), and the scene, are known as optical flow. The optical flow and the 2-D motion have the same

qualitative properties, and thus, the optical flow holds information about the 3-D motion behavior of the object or the geometric structure of the world [**66**].

An Optical flow example is shown in Figure 11 and the Dense Optical flow of our method is presented in Figure 12.

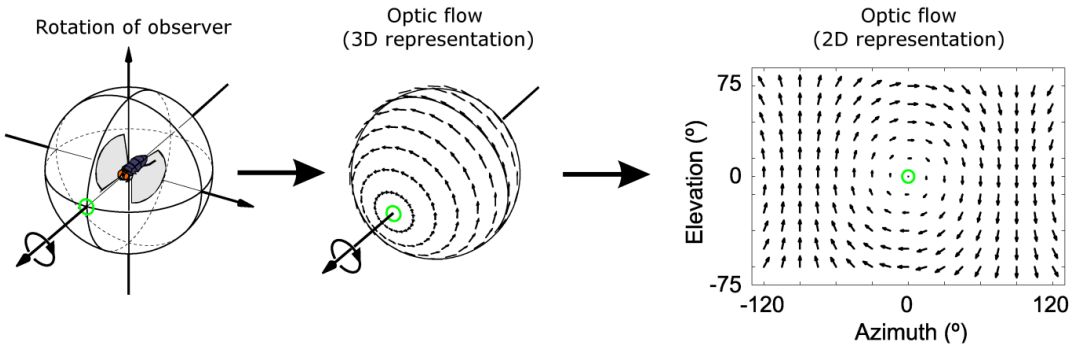

Figure 11: Optical flow example [67**]**

For example, assume that we have two frames, and a pixel I(x, y, t) that is in the first frame, and assume that it moves by distance (dx, dy) in the following frame that is taken after dt time. Since we are analyzing the same pixels through time, and the intensity does not change, we can assume that:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Optical flow can be used in motion estimation, robotics, and video compression. In robotics it is used for object detection, motion detection, robot navigation, and even object tracking [**67**].

In our approach in order to achieve object tracking, we employed the OpenCV dense optical flow function, ***calcOpticalFlowFarneback ()***.

This dense optical flow function computes the optical flow for all the points in a frame and returns as output a 2-channel array with the optical flow vectors (u, v), where u is the computed magnitude and v is the computed direction. For example:

$$prev(y, x) \sim next(y + flow(y, x)[1], x + flow(y, x)[0])$$

This method is based on Gunnar Farneback's algorithm, [**68**] which is a two-frame motion estimation algorithm that uses quadratic polynomial expansion transform to both frames in the first step, in order to approximate their neighborhood, and then a method to estimate the displacement fields from the polynomial expansion coefficients is acquired.

For example, when we have two frames where we used polynomial expansion to both of them, we had as output the expansion coefficients for each one of the frames; consider $A_1(x)$, $b_1(x)$, and $c_1(x)$ as the coefficients of the first frame, and $A_2(x)$, $b_2(x)$, and $c_2(x)$ as the coefficients of the second frame respectively.

Ideally, this should result in $A_1 = A_2$, but the following approximation is more realistic:

$$A(x) = \frac{A_1(x) + A_2(x)}{2}$$

and the following equation:

$$\Delta b(x) = -\frac{1}{2} * (b_2(x) - b_1(x))$$

is considered as well, in order to obtain A(x)*d(x) = Δb(x), where d(x) is the spatially varying displacement field.

Because the results can be too noisy, the information over a neighborhood of every pixel is integrated by minimizing the following:

$$\sum_{\Delta x \in I} w(\Delta x)||A(x+\Delta x)d(x) - \Delta b(x+\Delta x)||^2$$

where w(Δx) is the weight function of the neighborhood points. Afterwards indexing is used to make the expression more reliable and the minimum is given by:

$$e(x) = \left(\sum w\Delta b^T * \Delta b\right) - d(x)^T * \sum wA^T\Delta b$$

This equation practically means that the A^TA, $A^T\Delta b$, and $\Delta b^T\Delta b$ were computed point wise and averaged with a w before they are solved for the displacement. The solution is unique and exists for the neighborhood, unless the entire neighborhood is exposed to the aperture problem.

The aperture problem, as shown in Figure 13, is when the observer watches the movement through a window or aperture and cannot know the exact direction of the motion or even the speed, because the moving contours could have different directions or speed but the observer realizes them as the same. [**69**]

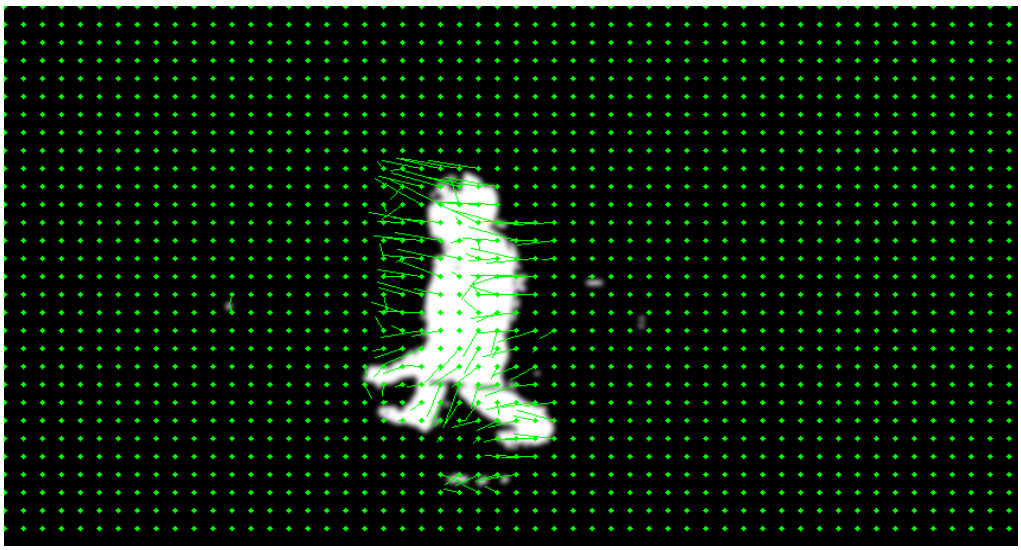

Figure 12: The dense optical flow as shown in our approach

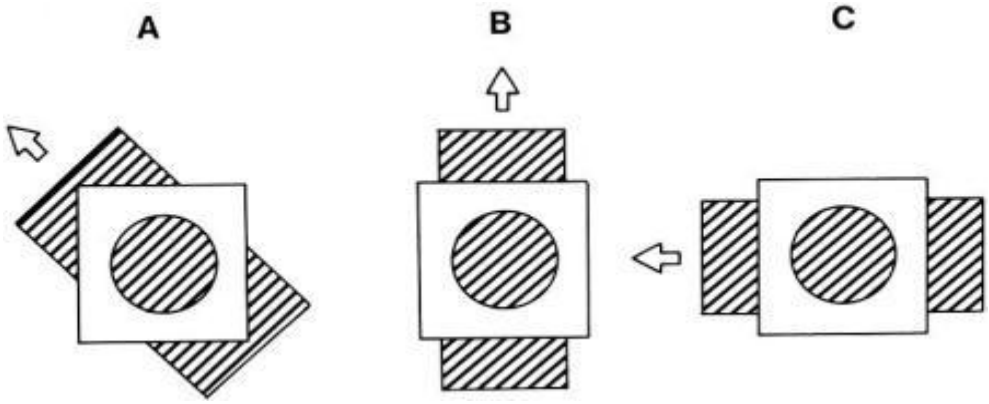

Figure 13: The aperture problem [70**]**

3.3.2.3 Magnitude & Angle

The magnitude and the angle features are the values describing the velocity vectors. In order to be computed, the ***cartToPolar()*** function of OpenCV was used. This function converts the Cartesian Coordinates into Polar Coordinates [**71**].

In general, the 2-D Cartesian coordinate system informs us of the height (i.e. how far up) and the width (i.e. how far along) a point is. Whereas, the 2-D Polar coordinate system informs us of the distance (i.e. how far away) a point is and the angle that it has [**72**]. Figure 14 shows and indicative example of this.

Therefore, this method can inform us of the position, the acceleration and the velocity of a set of points in time (e.g. the detected moving object).

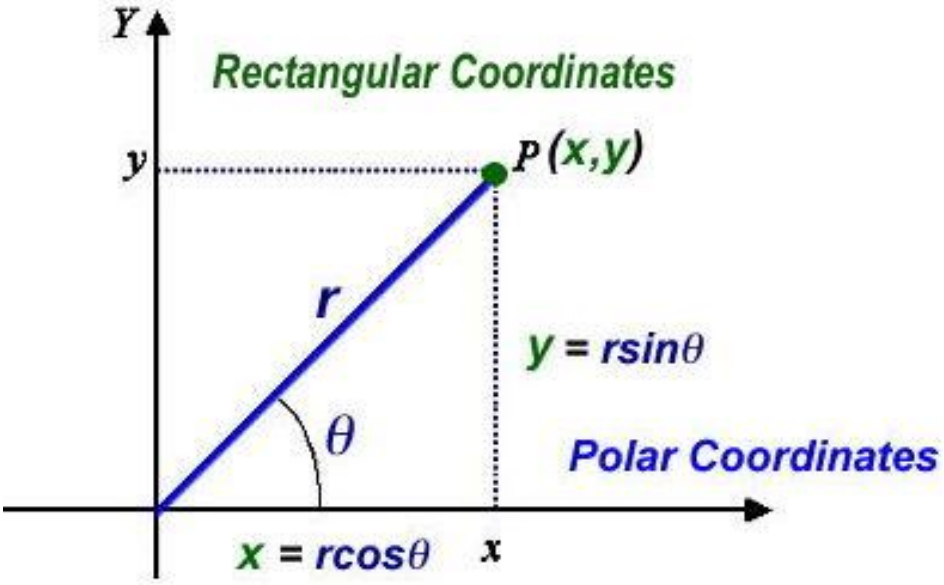

Figure 14: The Cartesian and the Polar 2-D Coordinates [73**]**

3.3.2.4 Final Selected Features

From all the techniques mentioned above the final features were extracted and written in the output *yml* files in order to be later classified. The following table presents an overview of the selected features of each method used in our approach.

Approach	Feature 1	Feature 2	Feature 3
Kalman Filtering	**Aspect Ratio** (of Bounding Box)	**Magnitude** (of the velocity vectors computed by the tracked moving object's **centroid**)	**Angle** (of the velocity vectors computed by the tracked moving object's **centroid**)
Optical Flow	**Aspect Ratio** (of Bounding Box)	**Magnitude** (of the velocity vectors computed by the **Dense Opt. Flow** technique)	**Angle** (of the velocity vectors computed by the **Dense Opt. Flow** technique)

Table 4: Overview of the selected features in every method of our approach

In the Kalman filtering approach, as shown in Table 4, the features consist of the aspect ratio of the bounding box, and the magnitude and the angle of the velocity vectors resulting from the Kalman filtering of the moving object's centroid (central moments of contours) tracking and the *cartToPolar* function.

In the Optical Flow approach, the final features extracted are again the aspect ratio of the bounding box, the magnitude and the angle, but this time of the velocity vectors produced by the dense Optical Flow technique and the *cartToPolar* function.

3.4 Annotation

3.4.1 Supervised Frame Annotation

After the editing of the videos, as explained in chapter 3.2, a video frame annotation tool has been used for the supervised annotation on the frames of each video. The input of the tool was the video and the output was a *yml* file.

When the moving object, that is the person in our case, enters the room**,** a rectangle is placed manually around the person as well as a label on each frame**.** When the person was standing or walking**,** the label was "No Fall", as shown in Figure 15.

When the person's position turns from vertical to horizontal and touching the mattress, the label was "Has Fallen", as shown in Figure 16.

Figure 15: Frame Annotation example "No Fall"

Figure 16: Frame Annotation example "Has Fallen"

The supervised *yml* files include the number of the frames in each video, the object label, in our case "Person". They also include the object, the x and y position of the object and the height and width of the rectangle around it, although these data have not been used in the classification. Finally, the *yml* includes the attributes, "No Fall" and "Has Fallen".

An example of an *yml* file is shown in Figure 17.

```
%YAML:1.0
total_frames: 52
Object_labels:
   - [ "", "", "", "", "" ]
   - [ "", "", "", "", "" ]
   - [ "", "", "", "", "" ]
   - [ "", "", "", "", "" ]
   - [ "", "", "", "", "" ]
   - [ "", "", "", "", "" ]
   - [ Person, "", "", "", "" ]
   - [ Person, "", "", "", "" ]
   - [ Person, "", "", "", "" ]
Objects:
   - [ { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0,
       h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 } ]
   - [ { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0,
       h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 } ]
   - [ { x:229, y:280, w:229, h:80 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0,
       w:0, h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 } ]
   - [ { x:229, y:280, w:229, h:80 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0,
       w:0, h:0 }, { x:0, y:0, w:0, h:0 }, { x:0, y:0, w:0, h:0 } ]
Attributes:
   - [ [ "", "", "", "" ], [ "", "", "", "" ], [ "", "", "", "" ], [ "",
       "", "", "" ], [ "", "", "", "" ] ]
   - [ [ "", "", "", "" ], [ "", "", "", "" ], [ "", "", "", "" ], [ "",
       "", "", "" ], [ "", "", "", "" ] ]
   - [ [ "", NoFall, "", "" ], [ "", "", "", "" ], [ "", "", "", "" ], [
       "", "", "", "" ], [ "", "", "", "" ] ]
   - [ [ "", NoFall, "", "" ], [ "", "", "", "" ], [ "", "", "", "" ], [
       "", "", "", "" ], [ "", "", "", "" ] ]
   - [ [ HasFallen, "", "", "" ], [ "", "", "", "" ], [ "", "", "", "" ],
       [ "", "", "", "" ], [ "", "", "", "" ] ]
   - [ [ HasFallen, "", "", "" ], [ "", "", "", "" ], [ "", "", "", "" ],
       [ "", "", "", "" ], [ "", "", "", "" ] ]
```

Figure 17: Yml example image

3.4.2 Final Data

3.4.2.1 Attribute-Relation File Format (ARFF)

After the *yml* extraction as explained in 3.4.1 and 3.3.2.4 section, a program has been created in C++ language in OpenCV. This program has as input the 2 yml files from the same video and as output the ARFF file. The ARFF file has in the @relation field the name of the video and whether the features used are from the Kalman or the Optical Flow method.

The @attributes field contains the frame number which is a numeric type, the file name which is a string type, the Aspect ratio, the Magnitude and the Angle which all are numeric types as well as the classes "Has Fallen" and "No Fall". An example image of an ARFF file is presented in Figure 18.

```
@relation cam2kal

@attribute FrameNo numeric
@attribute FileName string
@attribute AspectRatio numeric
@attribute Angle numeric
@attribute Magnitude numeric
@attribute class {HasFallen,NoFall}

@data

1,cam2backFallch01mod.avi,0.988188,283.521,16.3923,?
2,cam2backFallch01mod.avi,0.988188,106.042,15.8737,?
3,cam2backFallch01mod.avi,0.988188,32.28,16.1919,?
4,cam2backFallch01mod.avi,5.3514,6.32456,18.4387,?
5,cam2backFallch01mod.avi,1.09091,344.681,174.34,?
6,cam2backFallch01mod.avi,0.972973,218.586,175.803,NoFall
7,cam2backFallch01mod.avi,1,145.344,176.056,NoFall
8,cam2backFallch01mod.avi,1.97143,99.1262,182.891,NoFall
9,cam2backFallch01mod.avi,0.914634,61.2944,185.617,NoFall
10,cam2backFallch01mod.avi,0.91,42.2966,186.788,NoFall
11,cam2backFallch01mod.avi,0.931373,26.6833,192.996,NoFall
12,cam2backFallch01mod.avi,1.37179,15.2315,203.203,NoFall
13,cam2backFallch01mod.avi,1,8.06226,262.876,NoFall
14,cam2backFallch01mod.avi,1.5042,13.4164,296.567,NoFall
15,cam2backFallch01mod.avi,1.9899,17.0294,310.239,NoFall
16,cam2backFallch01mod.avi,2.37179,21.9317,316.839,NoFall
17,cam2backFallch01mod.avi,1.16216,23.7697,337.747,NoFall
18,cam2backFallch01mod.avi,1.1453,27.6586,347.47,NoFall
19,cam2backFallch01mod.avi,2.08081,30.0832,338.548,NoFall
```

Figure 18: ARFF example image

3.5 Classification Algorithms

The algorithms that were selected were the most frequently used from the state of the art, and they are the following: J48 and Random Forest from the Decision Trees, Naïve Bayes, Logistic, MLP from Neural Networks, SGD and SMO from SVM and IBk from KNN.

3.5.1 Decision Trees

Decision trees are a decision system that uses a tree-like graph decisions and their possible after-effect. A Decision Tree, or a classification tree, is also used to learn a classification function which concludes the value of a dependent attribute (variable) given the values of the independent (input) attributes (variables) [**74**].

Decision trees are easy to interpret, they select attributes automatically and they are able to handle numeric as well as nominal values so they are the most powerful approach in knowledge discovery and data mining [**75**],[**74**].They are easy to understand by the end user and they are able to process incorrect datasets or missing values. They also have high performance with small number of efforts and they can implement data mining packages over a variety of platforms [**74**].

3.5.1.1 J48

J48 uses the C4.5 algorithm to generate a decision tree. This algorithm is an extension of ID3 algorithm and creates a small tree. First, it checks whether all cases belong to the same class, then the tree that is a leaf is labelled with that class. Then, for each attribute, it calculates the information and the information gain. It finds the best splitting attribute (depending upon current selection criterion). Then it calculates the information gain, and the Entropy that is used in this process [**74**].

3.5.1.2 Random Forest

As mentioned in [**76**], the definition of the algorithm is: "*A Random Forest is a classifier consisting of a collection of tree structured classifiers {h (**x**, ∪k) , k=1, ...} where the {∪k} are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input **x**."*

In general, after a large number of trees are generated, they vote for the most popular class. The accuracy of a random forest depends on the strength of the individual tree classifiers and the measure of the dependence between them. There are two ways of selecting a node of the tree, one that uses random selection from the original inputs and the other uses random linear combinations of inputs. Usually, selecting one or two features gives near optimum results [**76**]. Generalization error is a function that measures how well a learning machine generalizes the unseen data. It is measured as the distance between the error on the training set and the test set and is averaged over the entire set of possible training data that can be generated after each iteration of the learning process [**77**].

Some random forest algorithms reported in the literature have consistently lower generalization error than others [**76**].

3.5.2 Naive Bayes

Naive Bayes is one of the simplest density estimation methods from which we can form one of the standard classification techniques in machine learning. *Naive Bayes* is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. *Naive Bayes* is very easy to program, is fast to train and to use as a classifier and is very easy to deal with missing attributes [**78**].
All *Naive Bayes* classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable [**79**]. A *Naive Bayes* classifier assumes

that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature [**80**].

3.5.3 Logistic Regression

Logistic Regression is one of the most commonly used tools for applied statistics and discrete data analysis [**81**]. It estimates the probability of an event occurring. What we want to predict from a knowledge of relevant independent variables is not a precise numerical value of a dependent variable, but rather the probability (*p*) that it is 1 (event occurring) rather than 0 (event not occurring) [**82**].

Two models of logistic regression are appropriate for inclusion in our work, namely *binomial/binary logistic regression* and *multinomial logistic regression*. When the dependent variable is dichotomous and the independent variables are either continuous or categorical variables, binary logistic regression is used. Logistic regression is best used when there are two classes. When the dependent variable is composed of more than two classes, a multinomial logistic regression can be used [**83**].

3.5.4 Neural Networks

Artificial Neural Networks (ANN) and *neurocomputers* are models inspired by brain functions and are defined as mathematical models of mind and brain activity. The objective is to understand by these models how the brain learns and how a person has reasoning and how such "computations" are arranged and carried out in the brain.

Neural Networks are being developed as a technological discipline that can automatically develop operational capabilities to adaptively respond to an information environment [**84**].

3.5.4.1 Multilayer Perceptron

The *Multilayer Perceptron* is the earliest of the Neural Network paradigms. A Perceptron of the simplest form is used for linear classification problems only. So for more complicated problems, perceptrons with one or multiple hidden layers in between input and output of the system are used. The reasons that enable a *Multilayer Perceptron* to learn complex tasks by extracting more meaningful features from the input patterns are three distinct characteristics. They are one or more layers of hidden neurons used that are not part of the input or output of the network. A smooth nonlinearity employed at the output end of each neuron and that there is a high connectivity in the network [**84**].

3.5.5 Support Vector Machines

The general idea of the *Support Vector Machine* is to linearly separate feature spaces. The idea is to detect the pair of parallel hyper planes that lead to the maximum separation between two classes of feature in order to minimize the errors. The pair of parallel hyper planes that have specific sets of feature points are called "support vectors". The number of the vectors is one number more than the number of the dimensions. For example, if the dimensions are two the support vectors needed are three.

One of the advantages of the SVM is the use of the smallest possible number of defining example patterns (the support vectors). The disadvantage is that the basic method only works when the dataset is linearly separable. So in order to overcome this problem, it is possible to change the training and test data to a feature space of higher dimension where the data does become linearly separable [**85**].

3.5.5.1 SGD

The Stochastic Gradient Decent (SGD) algorithm implements stochastic gradient decent for learning various linear models (binary class SVM, binary class logistic and linear regression).It globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes, so the coefficients in the output are based on the normalized data. This implementation can be trained incrementally on (potentially) infinite data streams [**86**].

3.5.5.2 SMO

Support Vector Machine (linear, polynomial and RBF kernel) with Sequential Minimal Optimization Algorithm. SMO implements a sequential minimal optimization algorithm for training a Support Vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. Multi-class problems are solved using pair wise classification [**86**].

3.5.6 k-Nearest Neighbour classifiers

The nearest neighbour algorithm is comparing an input image pattern against a number of paradigms and classifies it according to the class of the one with the closest match. The Nearest Neighbour algorithm has two disadvantages. The first one is that when the different patterns are close to each other there is difficulty in distinguishing them. The second one is that minor translations, rotations or noise prevent the algorithm to recognize the patterns. For that reason all the possible patterns of each class are needed [**85**].

3.5.6.1 IBk

IBk is a *K-nearest neighbor* classifier. It can select an appropriate value of *k*, based on cross-validation and it can also compute distance weighting. It is an Instance-Based learner with fixed neighbourhood; *K* sets the number of neighbours to use [**86**].

3.6 Evaluation

For the evaluation of our system we use two scenarios: a) 10-Fold Cross Validation and b) a train and test dataset separation scenario.

3.6.1 10-Fold Cross Validation scenario

In 10-Fold Cross Validation the samples are randomly divided into 10 subsets. Each time one subset is used for testing and the other nine are used for training of the algorithm.

This method is used for each classification algorithm mentioned in 3.3.4 on cameras 2, 4, 5, 7 individually and on the total of them, with the features Aspect Ratio, Magnitude and Angle from the Kalman filter. The same procedure is done for the features Aspect Ratio, Magnitude and Angle from the Optical Flow vectors.

3.6.2 Percentage split scenario

The dataset was split into “training set” and “test set” for a more realistic evaluation. The system is trained with the “training set” and in the “test set” it tries to find if the class is “No Fall” or “Has Fallen”. The video sequences (clips) for the train and the test set were divided in a way that in each set, train and test, includes all types of falls; side, front and back ones.

The 66.6% Train – 33.3% Test split provides a more realistic evaluation approach, since the test set video sequences are completely unknown to the classifier.

3.6.2.1 Scenario 1: 33.3% Train- 66.6% Test

The “test set” consists of:

- cam2sideFallch02
- cam2frontFallch03
- cam2backFallch01

Plus the same video sequences from camera 4, camera 5 and camera 7. In Total 12 files.

The “training set” consists of:

- cam2sideFallch12
- cam2sideFallch11
- cam2sideFallch8
- cam2frontFallch05
- cam2doubleFallch04
- cam2backFallch06

Plus the same video sequences from camera 4, camera 5 and camera 7. In Total 24 files.

3.6.2.2 Scenario 2: 66.6% Train- 33.3%Test

The “training set” consists of:

- cam2sideFallch02
- cam2frontFallch03
- cam2backFallch01

as well as the same video sequences from camera 4, camera 5 and camera 7. In Total 12 files.

The “test set” consists of:

- cam2sideFallch12
- cam2sideFallch11
- cam2sideFallch8
- cam2frontFallch05
- cam2doubleFallch04
- cam2backFallch06

As well as the same video sequences from camera 4, camera 5 and camera 7. In Total 24 files.

4. Results

4.1 Evaluation measures

The evaluation is measured with the help of counting *True Positives (TP)*, *False Positives (FP)*, *True Negatives (TN)*, and *False Negatives (FN)*. A TP is when a fall occurs and the system correctly detects it; a FP is when the system recognizes a fall when no fall occurs; a TN is when a fall does not occur and the system correctly does not detect it; a FN is when a fall occurs but the system does not detect it.

Precision is a percentage of frames labelled as "Has Fallen" that belong indeed in that class. Although, it does not take into account the number of frames incorrectly labelled as "Has Fallen".

$$\textbf{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall (or Sensitivity) is the percentage of frames from the "Has Fallen" class that have been labelled as frames that actually belong in that class. Although, it does not hold any information about the number of items that were incorrectly labelled belonging in the "Has Fallen" class.

$$\textbf{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

In general, as shown in chapter 4.2, when the precision value is high the recall value is low and vice versa.

F-measure is a combined metric of recall and precision so it effectively references the True Positives to the Arithmetic Mean of Predicted Positives and Real Positives, being a constructed rate normalized to an idealized value.

$$\textbf{Fmeasure} = \frac{2 * \mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

According to David M W Powers [**87**] the application of Recall, Precision and F-Measure are argued to be flawed as they ignore the True Negatives. This system focuses in positive example and predictions so these values are taken under consideration.

Specificity relates to the system's ability to identify negative results.

$$\textbf{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Accuracy describes the efficiency of the system.

$$\textbf{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Finally, the Error Rate describes the deficiency of the system.

$$\textbf{Error Rate} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

4.2 Results

For the efficiency of the algorithms this project took under consideration the F-Measure percentage because it represents the mean value of the classification of the algorithm. The Precision value is also important because it is the percentage of the correctly predicted falls among all the detected falls, whereas the Recall value shows how many of the correctly detected falls (TP) were found but does not take under consideration the False Positives, thus making it unreliable.

a) 10-Fold Cross Validation

In the following tables (Table 5 and 6 respectively), the classification results with cross validation from the features Aspect Ratio, Magnitude and Angle that were computed with the help of the Kalman Filter, are presented. As it is shown, the efficiency of the algorithms is tested on each camera separately and then on all the cameras together in the column "Total". The **Random Forest** algorithm has the best results in general and the SMO algorithm has the worst.

KALMAN cameras & total Aspect Ratio-Magnitude-Angle						
Classification Algorithms	**Total**			**cam2**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
1.J48	**0,78**	**0,76**	**0,76**	**0,85**	**0,79**	**0,81**
2.Random Forest	**0,77**	**0,8**	**0,79**	**0,83**	**0,85**	**0,84**
3.Naïve Bayes	**0,64**	**0,73**	**0,68**	**0,65**	**0,88**	**0,75**
4.Logistic	**0,77**	**0,75**	**0,76**	**0,77**	**0,76**	**0,76**
5.MLP	**0,74**	**0,78**	**0,76**	**0,77**	**0,76**	**0,76**
6.SGD	**0,75**	**0,79**	**0,77**	**0,76**	**0,78**	**0,77**
7.SMO	**0**	**0**	**0**	**0,73**	**0,77**	**0,75**
8.IBk	**0,75**	**0,75**	**0,75**	**0,81**	**0,81**	**0,81**

Table 5: Overview of the camera 2 results and of the total for the Kalman Aspect Ratio, Magnitude and Angle approach

KALMAN cameras & total Aspect Ratio-Magnitude-Angle									
Classification Algorithms	**cam4**			**cam5**			**cam7**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
1.J48	**0,85**	**0,78**	**0,81**	**0,76**	**0,81**	**0,78**	**0,88**	**0,73**	**0,8**
2.Random Forest	**0,84**	**0,86**	**0,85**	**0,8**	**0,83**	**0,82**	**0,86**	**0,86**	**0,86**
3.Naïve Bayes	**0,7**	**0,87**	**0,78**	**0,67**	**0,78**	**0,72**	**0,7**	**0,74**	**0,72**
4.Logistic	**0,81**	**0,81**	**0,81**	**0,73**	**0,54**	**0,62**	**0,76**	**0,66**	**0,7**
5.MLP	**0,84**	**0,79**	**0,81**	**0,73**	**0,74**	**0,71**	**0,75**	**0,71**	**0,72**
6.SGD	**0,8**	**0,82**	**0,81**	**0**	**0**	**0**	**0,8**	**0,64**	**0,7**
7.SMO	**0,78**	**0,83**	**0,8**	**0**	**0**	**0**	**0,77**	**0,63**	**0,69**
8.Ibk	**0,86**	**0,88**	**0,87**	**0,78**	**0,79**	**0,78**	**0,8**	**0,87**	**0,83**

Table 6: Overview of cameras 4, 5 and 6 results for the Kalman Aspect Ratio, Magnitude and Angle approach

In more detail, the best and worst results presented in Tables 5 and 6 are:

- **Total:**

  Best algorithm is **Random Forest** with Precision 0.77, Recall 0.8 and F-Measure 0.79

  Worst algorithm is SMO with Precision 0, Recall 0 and F-Measure 0.

- **Cam2:**

  Best algorithms are **Random Forest** with Precision 0.83, Recall 0.85 and F-Measure 0.84 and also **J48** with Precision 0.85, Recall 0.79 and F-Measure 0.81 because the precision value is higher.

  Worst algorithm is SMO with Precision 0.73, Recall 0.77 and F-Measure 0.75

- **Cam4:**

  Best algorithms are **IBk** with Precision 0.86, Recall 0.88 and F-Measure 0.87 and also **Random Forest** with Precision 0.84, Recall 0.86 and F-Measure 0.85.

  Worst algorithm is Naïve Bayes with Precision 0.7, Recall 0.87 and F-Measure 0.78.

- **Cam5:**

  Best algorithm is **Random Forest** with Precision 0.8, Recall 0.83 and F-Measure 0.82

  Worst algorithms are SMO and SGD with Precision 0, Recall 0 and F-Measure 0 in both of them.

- **Cam7:**

  Best algorithms are **Random Forest** with Precision 0.86, Recall 0.86 and F-Measure 0.86 and also **J48** with Precision 0.88, Recall 0.73 and F-Measure 0.80 because the precision percentage is higher.

  Worst algorithms are SMO with Precision 0.77, Recall 0.63 and F-Measure 0.69 and also MLP with Precision 0.75, Recall 0.71 and F-Measure 0.72.

In the following Tables 7 and 8 the classification results with cross validation from the features Aspect Ratio, Magnitude and Angle that were computed with the help of the Optical Flow method, are presented. The **J48**, **Random Forest** and **MLP** algorithms have the best results in general and MLP algorithm has also the worst results in camera 5.

OPTICAL FLOW cameras & total Aspect Ratio-Magnitude-Angle						
Classification Algorithms	**Total**			**cam2**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
1.J48	**0,85**	**0,72**	**0,78**	**0,89**	**0,79**	**0,83**
2.Random Forest	**0,79**	**0,81**	**0,8**	**0,85**	**0,85**	**0,85**
3.Naïve Bayes	**0,77**	**0,82**	**0,79**	**0,84**	**0,73**	**0,77**
4.Logistic	**0,79**	**0,81**	**0,8**	**0,81**	**0,78**	**0,79**
5.MLP	**0,8**	**0,8**	**0,8**	**0,82**	**0,8**	**0,81**
6.SGD	**0,79**	**0,82**	**0,8**	**0,79**	**0,79**	**0,79**
7.SMO	**0,79**	**0,75**	**0,77**	**0,77**	**0,79**	**0,78**
8.IBk	**0,76**	**0,75**	**0,75**	**0,81**	**0,79**	**0,8**

Table 7: Overview of camera 2 results and of the total for the Optical Flow's Aspect Ratio, Magnitude and Angle approach

OPTICAL FLOW cameras & total Aspect Ratio-Magnitude-Angle									
Classification Algorithms	**cam4**			**cam5**			**cam7**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
1.J48	**0,82**	**0,87**	**0,84**	**0,85**	**0,65**	**0,73**	**0,86**	**0,83**	**0,84**
2.Random Forest	**0,83**	**0,85**	**0,84**	**0,76**	**0,78**	**0,76**	**0,85**	**0,86**	**0,86**
3.Naïve Bayes	**0,71**	**0,93**	**0,8**	**0,68**	**0,85**	**0,75**	**0,77**	**0,88**	**0,82**
4.Logistic	**0,83**	**0,85**	**0,84**	**0,69**	**0,76**	**0,72**	**0,79**	**0,83**	**0,81**
5.MLP	**0,83**	**0,87**	**0,85**	**0,72**	**0,73**	**0,7**	**0,81**	**0,79**	**0,79**
6.SGD	**0,83**	**0,87**	**0,85**	**0,7**	**0,83**	**0,76**	**0,78**	**0,84**	**0,81**
7.SMO	**0,82**	**0,88**	**0,85**	**0,71**	**0,82**	**0,76**	**0,77**	**0,85**	**0,81**
8.Ibk	**0,82**	**0,82**	**0,82**	**0,74**	**0,75**	**0,74**	**0,87**	**0,87**	**0,87**

Table 8: Overview of cameras 4, 5 and 6 results for the Optical Flow's Aspect Ratio, Magnitude and Angle approach

In more detail, the best and worst results, as shown in the tables above, are:

- **Total:**
  Best algorithms are **MLP** with Precision 0.8, Recall 0.8 and F-Measure 0.8 and also **J48** with Precision 0.85, Recall 0.72 and F-Measure 0.78.
  Worst algorithm is IBk with Precision 0.76, Recall 0.75 and F-Measure 0.75

- **Cam2:**
  Best algorithms are **Random Forest** with Precision 0.85, Recall 0.85 and F-Measure 0.85 and also **J48** with Precision 0.89, Recall 0.79 and F-Measure 0.83 because it has better precision..
  Worst algorithm is SMO with Precision 0.77, Recall 0.79 and F-Measure 0.79.

- **Cam4:**
  Best algorithms are both **MLP and SGD** with Precision 0.83, Recall 0.87 and F-Measure 0.85.
  Worst algorithm is Naïve Bayes with Precision 0.71, Recall 0.93 and F-Measure 0.80.

- **Cam5:**
  Best algorithm is **Random Forest** with Precision 0.76, Recall 0.78 and F-Measure 0.76.
  Worst algorithm is Logistic with Precision 0.69, Recall 0.76 and F-Measure 0.72.

- **Cam7:**
  Best algorithm is **IBk** with Precision 0.87, Recall 0.87 and F-Measure 0.87.
  Worst algorithm is MLP with Precision 0.81, Recall 0.79 and F-Measure 0.79.

Comparing the results obtained using the Kalman Filtering and the results when employing the Optical Flow method, using the 10-fold cross validation scenario, we observe that both approaches show approximately the same F-Measure percentages which are 0.79 for the Random Forest algorithm (Table 5, Kalman filter) and 0.80 for the MLP (Table 7 Optical

Flow) algorithm in the "Total" column respectively. Although the Random Forest, Logistic and SGD algorithms show the same F-measure (0.8), they exhibit a slightly greater imbalance between Precision and Recall. Another observation is that Camera 7 shows better results in both approaches.

The following figures (Fig.19, Fig.20, and Fig.21) present graphically the results of the comparison of the Kalman method's Precision, Recall and F-Measure of the total results to the Optical Flow method's Precision, Recall and F-Measure in this scenario.

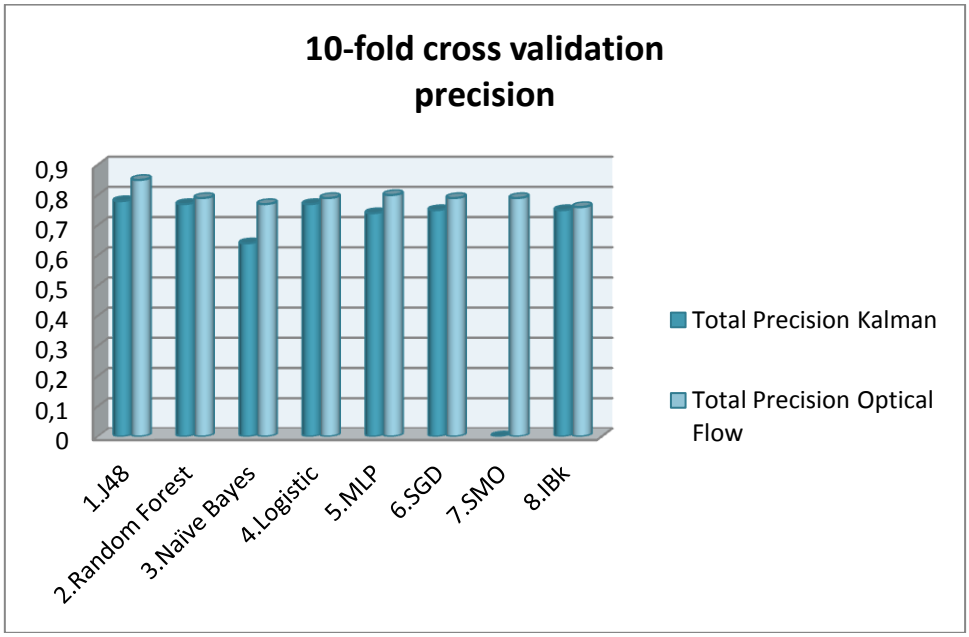

Figure 19: The comparison between the Precision of the Kalman and the Optical Flow methods in the 10-Fold Cross Validation scenario

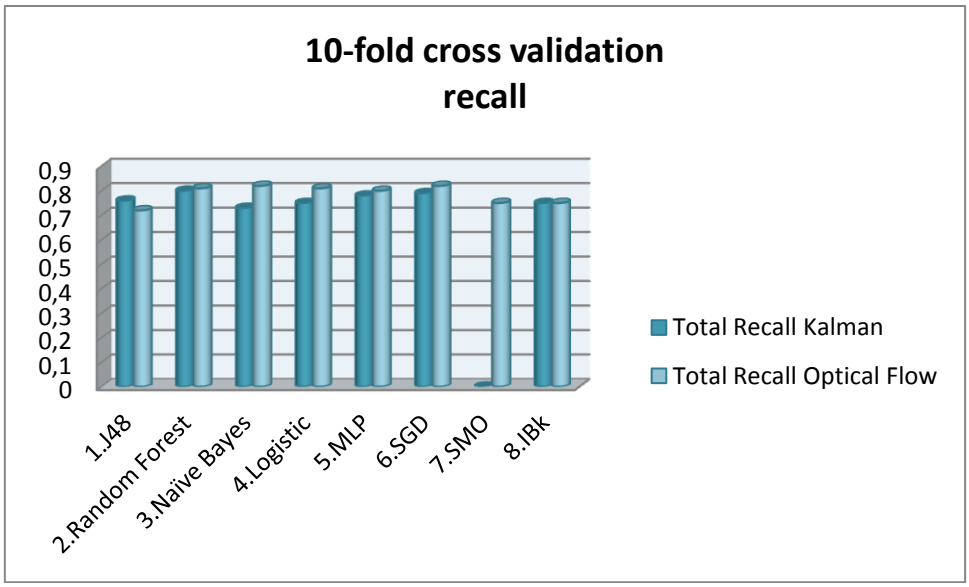

Figure 20: The comparison between the Recall of the Kalman and the Optical Flow methods in the 10-Fold Cross Validation scenario

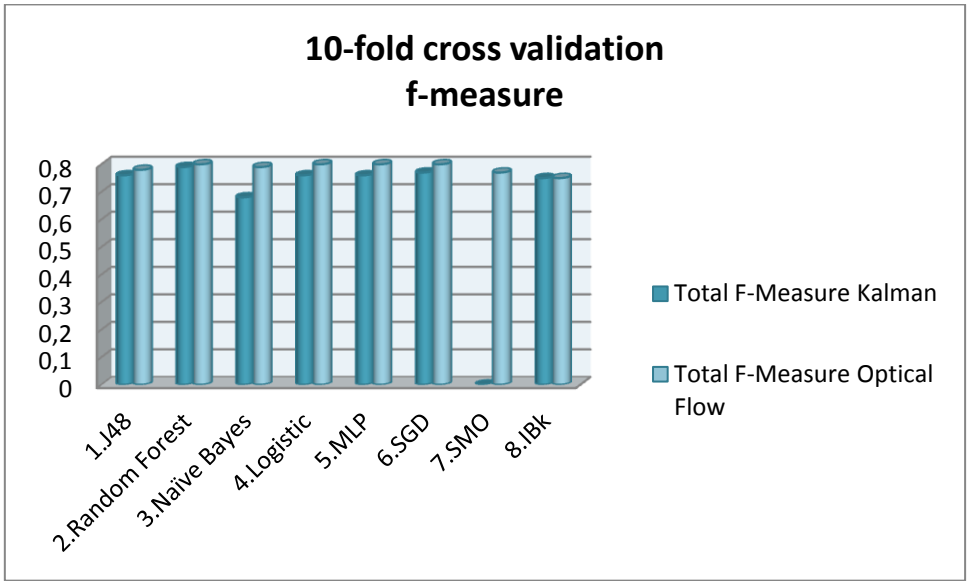

Figure 21: The comparison between the F-Measure of the Kalman and the Optical Flow methods in the 10-Fold Cross Validation scenario

b) Percentage split scenario

KALMAN 33% train 66%test Aspect Ratio-Magnitude-Angle			
Classification Algorithms	**Precision**	**Recall**	**F-Measure**
1.J48	**0,641**	**0,802**	**0,713**
2.Random Forest	**0,614**	**0,804**	**0,696**
3.Naïve Bayes	**0,486**	**0,729**	**0,583**
4.Logistic	**0,639**	**0,766**	**0,696**
5.MLP	**0,672**	**0,754**	**0,71**
6.SGD	**0,647**	**0,776**	**0,705**
7.SMO	**0**	**0**	**0**
8.IBk	**0,592**	**0,751**	**0,662**

Table 9: Overview of the Kalman results of the 33.3% train and 66.6% test scenario

In the above table (Table 9), the "33.3% Train"-"66.6% Test" method with the features extracted with the help of the Kalman filter is presented and the following results were observed:

Best algorithms are both **J48** with Precision 0.641, Recall 0.802 and F-Measure 0.713 and also **MLP** with Precision 0.672, Recall 0.754 and F-Measure 0.710.

Worst algorithm is SMO with Precision 0, Recall 0 and F-Measure 0

In Table 10 the "66.6% Train"-"33.3% Test" method of the features extracted with the use of the Kalman filter is shown.

KALMAN 66% train 33%test Aspect Ratio-Magnitude-Angle			
Classification Algorithms	**Precision**	**Recall**	**F-Measure**
1.J48	**0,945**	**0,673**	**0,786**
2.Random Forest	**0,822**	**0,641**	**0,72**
3.Naïve Bayes	**0,778**	**0,903**	**0,836**
4.Logistic	**0,972**	**0,645**	**0,776**
5.MLP	**0,958**	**0,53**	**0,682**
6.SGD	**0,869**	**0,843**	**0,856**
7.SMO	**0,889**	**0,645**	**0,748**
8.IBk	**0,773**	**0,627**	**0,692**

Table 10: Overview of the Kalman results of the 66.6% train and 33.3% test scenario

The results presented are:

Best algorithms are both **SGD** with Precision 0.869, Recall 0.843 and F-Measure 0.856 and also **Naïve Bayes** with Precision 0.778, Recall 0.903 and F-Measure 0.836.

It is also noticeable that the Precision in some algorithms is very high although the Recall and consequently the F-Measure are very low. For example in **Logistic** algorithm the precision is **0.972**. This happens because the training set has enough values for the classification process.

Worst algorithm is IBk with Precision 0.773, Recall 0.627 and F-Measure 0.692.

In Table 11 the "33.3% Train"-"66.6% Test" method with the features extracted with the Optical Flow method is presented.

OPTICAL FLOW 33% train 66%test Aspect Ratio-Magnitude-Angle			
Classification Algorithms	**Precision**	**Recall**	**F-Measure**
1.J48	**0,655**	**0,798**	**0,72**
2.Random Forest	**0,645**	**0,816**	**0,72**
3.Naïve Bayes	**0,605**	**0,852**	**0,708**
4.Logistic	**0,678**	**0,859**	**0,758**
5.MLP	**0,705**	**0,819**	**0,758**
6.SGD	**0,707**	**0,836**	**0,766**
7.SMO	**0,688**	**0,753**	**0,719**
8.IBk	**0,649**	**0,789**	**0,713**

Table 11: Overview of the Optical Flow results of the 33.3% train and 66.6% test scenario

The results shown are:

Best algorithm is **SGD** with Precision 0.707, Recall 0.836 and F-Measure 0.766. It is also noticeable that the Precision in general is very low and the Recall is high due to the lack of enough values in the training set.

Worst algorithm is Naïve Bayes with Precision 0.605, Recall 0.852 and F-Measure 0.708.

In Table 12 the “66.6% Train”-“33.3% Test” method with the features extracted with the Optical Flow method is shown.

OPTICAL FLOW 66% train 33%test Aspect Ratio-Magnitude-Angle			
Classification Algorithms	**Precision**	**Recall**	**F-Measure**
1.J48	**0,909**	**0,555**	**0,69**
2.Random Forest	**0,55**	**0,668**	**0,75**
3.Naïve Bayes	**0,882**	**0,795**	**0,836**
4.Logistic	**0,914**	**0,733**	**0,813**
5.MLP	**0,898**	**0,832**	**0,864**
6.SGD	**0,907**	**0,811**	**0,856**
7.SMO	**0,916**	**0,802**	**0,855**
8.IBk	**0,809**	**0,636**	**0,712**

Table 12: Overview of the Optical Flow results of the 66.6% train and 33.3% test scenario

The results are:

Best algorithms are **MLP** with Precision 0.898, Recall 0.832 and F-Measure 0.864 and also **SMO** with Precision 0.916, Recall 0.802 and F-Measure 0.855.

Worst algorithm is Random Forest with Precision 0.550, Recall 0.668 and F-Measure 0.750.

In conclusion, as it is shown from the tables above, the 66.6% train and 33.3% test technique of the Optical Flow approach has the best results out of all the other approaches. Therefore, we consider these values as the most important out of all the other values because as we explained in 3.6.2 chapter this method is more realistic.

The following figures (Fig.22, Fig.23, Fig.24) show an in depth comparison of the Kalman method Precision, Recall and F-Measure of the total results to the Optical Flow method Precision, Recall and F-Measure in the 33% train and 66% test case, whereas figures 25, 26 and 27 present the same comparison for the 66% train and 33% test case of this scenario respectively.

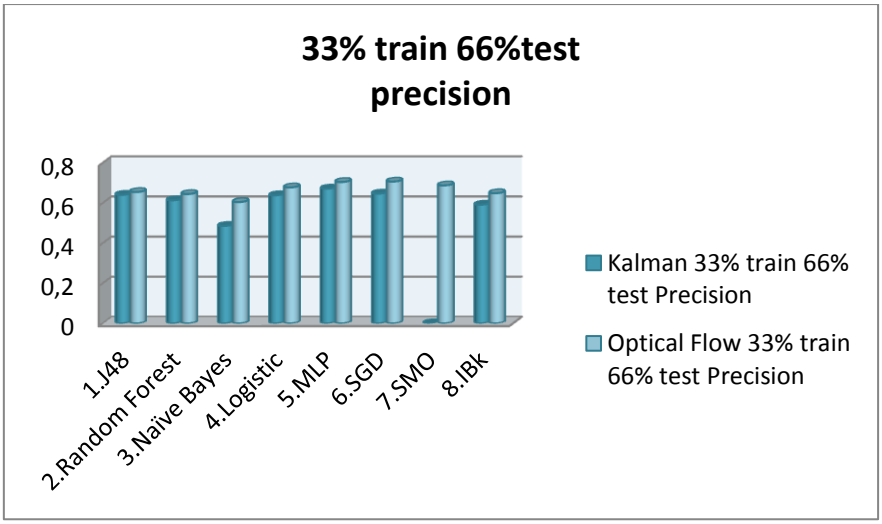

Figure 22: The comparison between the Precision of the Kalman and the Optical Flow methods in the percentage split scenario and the 33% train and 66% test case.

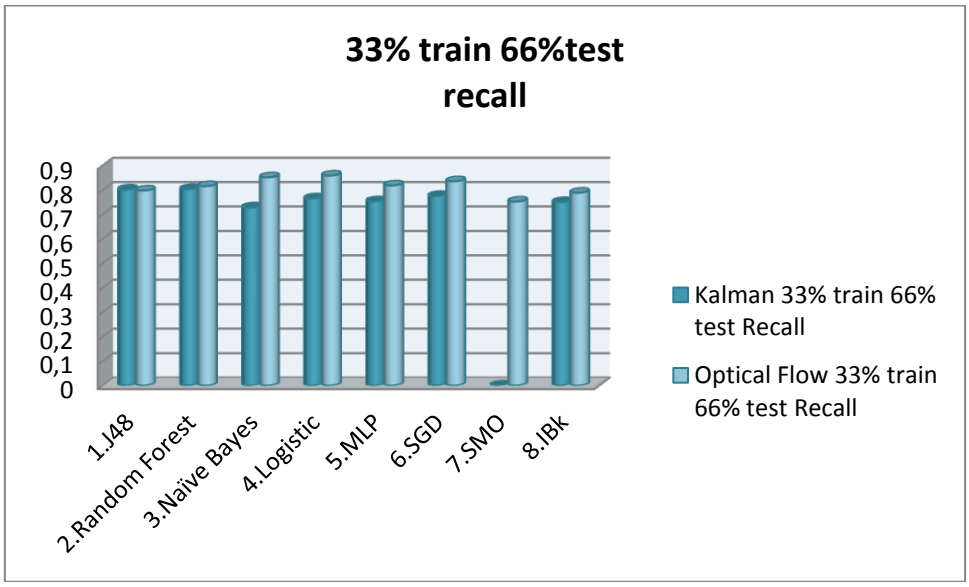

Figure 23: The comparison between the Recall of the Kalman and the Optical Flow methods in the 33% train and 66% test case.

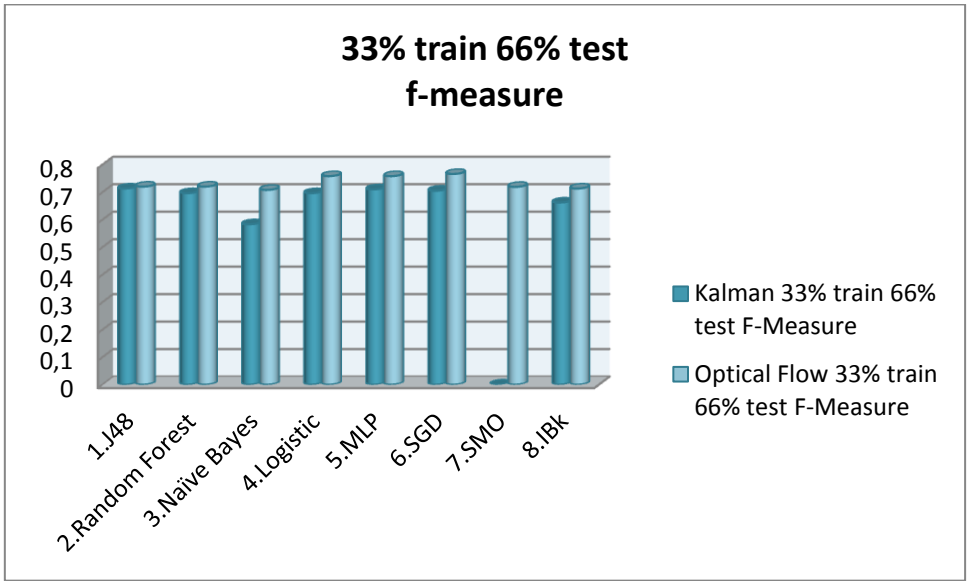

Figure 24: The comparison between the F-Measure of the Kalman and the Optical Flow methods in the 33% train and 66% test case.

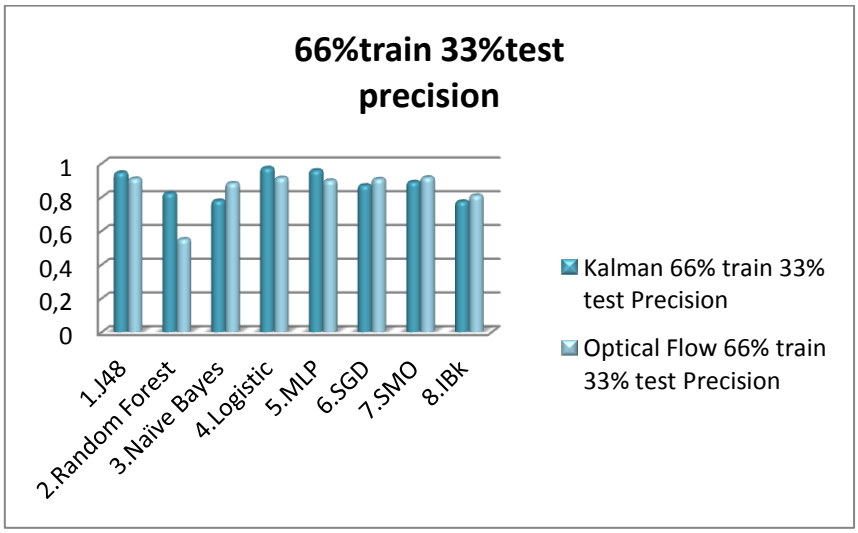

Figure 25: The comparison between the Precision of the Kalman and the Optical Flow methods in the percentage split scenario and the 66% train and 33% test case.

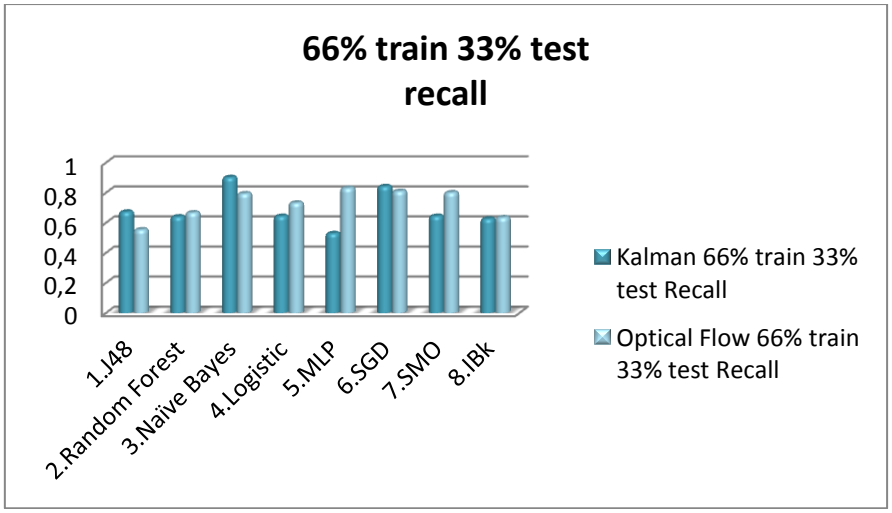

Figure 26: The comparison between the Recall of the Kalman and the Optical Flow methods in the 66% train and 33% test case.

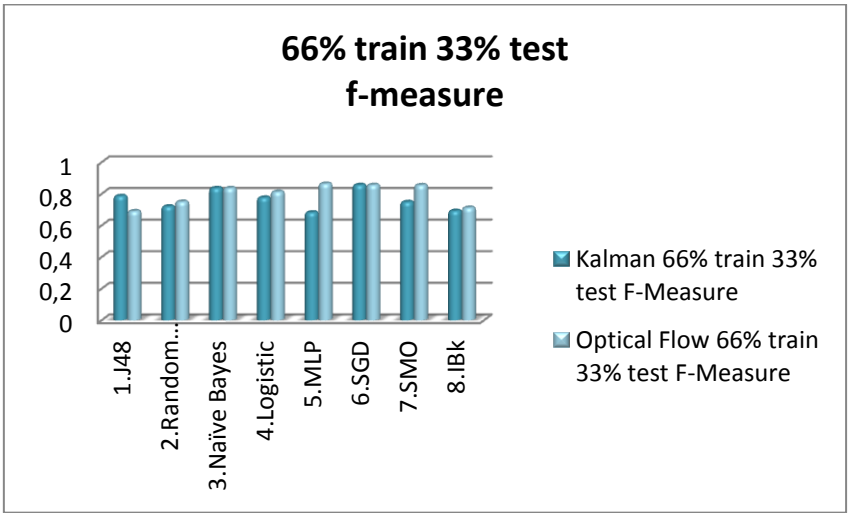

Figure 27: The comparison between the F-Measure of the Kalman and the Optical Flow methods in the 66% train and 33% test case.

5. Conclusions

The work described in this thesis has been concerned with the development of a Video-Based Fall Detection System.

The video dataset that was used in this thesis was incorporated with some difficulties. These were the high video compression (MPEG-4) that introduce artifacts in the image; the shadows and reflections that could be detected as moving objects during the background/foreground subtraction method, as well as the cluttered and textured background; the varying illumination that must be taken into account during the process of updating the background; and lastly the different clothes with different color and texture, as well as putting on and off a coat [**50**].

Taking into consideration the technical difficulties mentioned above two different approaches were applied with the purpose of accumulating experimental evidence regarding the appropriateness of each method to support our ultimate objective, i.e. the design of a reliable video-based fall detection system.

The first approach employs the Mixture of Gaussians technique for the Background/Foreground subtraction, then the use of a Kalman filter for the tracking of a moving object and finally includes a Data Annotation and Classification (Fall Detection) stage applied to video Recordings from the video dataset that shows simulations of falls.

In the second approach, the video reading, the Background/Foreground subtraction, as well as the Data Annotation and the Classification (Fall Detection) remain the same, but the tracking of a moving object is done by using the *Dense Optical Flow* technique rather than Kalman filtering.

The Classification (Fall Detection) task, in both approaches, is computed with two different scenarios. The first scenario is the 10-fold Cross Validation and the second one is the percentage split, which consists of two different cases. The separation of the dataset is done as follows: 66% of the data is used for forming the training set and 33% of the dataset available is used for testing for the first case, and for the second case the separation of the dataset is 33% for training and 66% for testing.

In all the above scenarios and cases eight different algorithms were used. These algorithms are the J48, the Random Forest, the Naïve Bayes, the Logistic, the MLP, the SGD, the SMO and the IBk.

The worst results were obtained by the 10-Fold Cross Validation scenario with the Kalman filter used for the tracking of a moving object and the SMO algorithm who showed 0 values for the Precision, Recall and F-Measure respectively. The best results were obtained when using the 66.6% of the available data for training and 33.3% for testing and the Optical Flow approach for tracking. In this later case the MLP and the SMO algorithms delivered the best results out of all the other algorithms. The MLP algorithm had a Precision of 0.898, Recall equal to 0.832 and F-Measure equal to 0.864, whereas the SMO algorithm had Precision equal to 0.916, Recall equal to 0.802 and F-Measure equal to 0.855.

As mentioned above, some of the results were far from satisfactory while others were more robust. This may be due to the quality of the extracted features using both the previously mentioned techniques, where the velocity vectors of the center of mass that were obtained from the Kalman filter approach were less reliable (weak features) than the velocity vectors

that were obtained by the dense optical flow approach (strong features). Also, the aspect ratio feature in both methods had better results for the two selected cameras that were mounted on the center of the walls of the room than the other two selected cameras that were mounted on the corners of the room. This is a result, in our view, of the fact that the height/width ratio was greater and therefore had better variation whenever the moving object was walking and then falling perpendicularly from the selected camera.

Also, another known problem was the fact that a relatively low number of falls exist in the dataset. There were only nine simulations of a specific fall (person walking and then falling), so the features extracted and the results that were computed have a low power for generalization.

Nevertheless, taking into consideration these problems of the dataset available, we feel that a robust method for the automated detection of human falls based on the analysis of video data has been constructed. In this process we have experimented with numerous different algorithmic approaches for the different steps of the computational pipeline and performed comparative assessment of their performance. The application of this computational pipeline to a high quality video dataset including human falls, we are sure will produce much better results, to the degree that would enable the realization of such systems for practical use.

6. Future Work

Although the results presented in previous chapters, have shown the effectiveness of our fall classification and detection approach, the system could be further developed in a number of ways. These are described below:

Extending the algorithm to work by combining the multiple cameras outputs for each frame

As mentioned in previous chapters, in our approach we use four different cameras that are mounted on four different places on the walls (as obtained by the public dataset used [**50**]), and we examine every camera separately and compute the features that feed the classifiers.

It is mentioned by the authors of [**24**], who employ the same public dataset, that the results computed by the combination of the cameras outputs have far more realistic information about the object's true position in the scene, thus being more useful, should an occlusion or a fall occur.

Working in the 3D space for Motion Tracking

Also, in [**24**], as well as in other papers we cited, the tracking is computed in the 3D space, rather than the 2D, and the results presented are more realistic than all the other approaches, because the information that is obtained in the 3D space is more accurate than the one obtained from the 2D; this method is computationally more expensive for that reason.

It also requires the cameras to be calibrated.

Utilization of videos that include, apart from falls, several everyday living activities, as well as occlusions for the design of a more complete fall detection system.

In our system we used only videos of falls (side falls, front falls, back falls) and another way of improving the system could be the use of all the videos in the public dataset, which include falls with occlusions and everyday living activities (e.g. sitting on couch, cleaning, etc).

There is clearly much work to be done in the area of Video-based Fall Detection and Classification. Perhaps the most direct extension of this work is by using a supervised training of the dataset to express the properties of all the situations included in it, thus helping the system to classify more effectively every new video entry presenting some sort of fall.

References

[1] M.E. Tinetti and M. Speechley, "Prevention of Falls Among the Elderly," *The New England Journal of Medicine*, vol. 320, no. 16, pp. 1055-1059, 1989.

[2] P.V.D. Ven, M.Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. Olaighin and J. Nelson A.K. Bourke, "Evaluation of Waist-mounted Tri-axial Accelerometer Based Fall-detection Algorithms During Scripted and Continuous Unscripted Activities," *Journal of Biomechanics*, vol. 43, no. 15, pp. 3051-3057, 2010.

[3] C.E. Cougler, "Falls and Imbalance," *Rehab Management*, vol. 5, pp. 53-117, 1992.

[4] A. Reimers, R. Andersson and L. Laflamme S. Sadigh, "Falls and Fall-related Injuries Among the Elderly: a Survey of Residential-care Facilities in Swedish Municipality," *Journal of Community Health*, vol. 29, no. 2, pp. 129-140, 2004.

[5] M.D. Mary E. Tinetti, "Preventing Falls in Elderly Persons," *The New England journal of Medicine*, vol. 348, January 2003.

[6] F. Padilla-Ruiz, J.J. Jimenez-Moleon ,C.A. Peinado-Alonso & R. Ga lvez-Vargas A. Bueno-Cavanillas, "Risk factors in falls among the elderly according to extrinsic and intrinsic precipitating causes," *European Journal of Epidemiology*, vol. 16, pp. 849-859, December 2000.

[7] Alain St-Arnaud, Jacqueline Rousseau and Jean Meunier Caroline Rougier, "Video Surveillance for Fall Detection," Department of Computer Science and Operations Research, University of Montreal and Research Center of the Geriatric Institute, University of Montreal, Montreal, 2011.

[8] M. Pediaditis, C. Chatzaki, E. Spanakis, M. Tsiknakis G. Vavoulas, "The MobiFall Dataset: Fall Detection and Classification with a Smartphone," *International Journal of Monitoring and Surveillance Technologies Research*, vol. 2, no. 1, pp. 44-56, January-March 2014.

[9] Y. Dedeoglu, and A.E. Cetin B.U. Toreyin, "HMM Based Falling Person Detection Using Both Audio and Video," in *Proc. IEEE 14th Signal Processing and Communications Applications*, Antalya, Turkey, 2006, pp. 1-4.

[10] J.B. Weaver, D.M. Healy, J. Lu Y. Xu, "Wavelet Transform Domain Filters: A Spatially Selective Noise Filtration Technique," *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 747-758, November 1994.

[11] H. Nait-Charif and S.J. McKenna, "Activity Summarisation and Fall Detection in a Supportive Home Environment," in *Proc. 17th International Conference on Pattern Recognition (ICPR 2004)*, Cambridge, UK, 2004, pp. 323-326.

[12] Y.M. Huang, J.H. Park, and H.C. Chao C.F. Lai, "Adaptive Body Posture Analysis for Elderly-Falling Detection with Multisensors," *IEEE Intelligent Systems*, vol. 25, no. 2, pp. 20-30, 2010.

[13] P.H. Sung, and C.Y. Huang S.G. Miaou, "A Customized Human Fall Detection System Using Omni-Camera Images and Personal Information," in *Proc. 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, Virginia, USA, 2006, pp. 39-42.

[14] B.S. Aski, and H. Pourreza H. Foroughi, "Intelligent Video Surveillance for Monitoring Fall Detection of Elderly in Home Environmemts," in *Proc. 11th International Conference on Computer and Information Technology (ICCIT 2008)* , Khulna, Bangladesh, 2008, pp. 219-224.

[15] A.H. Nasution and S. Emmanuel, "Intelligent Video Surveillance for Monitoring Elderly in Home Environments," in *Proc. IEEE 9th Workshop on Multimedia Signal Processing (MMSP 2007)*, Greece, 2007, pp. 203-206.

[16] C. Doukas, D. Vergados, I. Anagnostopoulos, I. Maglogiannis X. Kolovou, "*Open Research Issues in Vision-Based Fall Detection Systems (FDS)*," 12TH Mediterranean Conference on Medical and Biological Engineering and Computing, Chalkidiki, Greece, 2010.

[17] B. Taati, D. Giesbrecht, A. Mihailidis M. Belshaw, "Intelligent vision-based fall detection system: preliminary results from a real-world deployment," Intelligent Assistive Technology and Systems Lab (IATSL), University of Toronto, Toronto, 2011.

[18] A. Mihailidis T. Lee, "An intelligent emergency response system:preliminary development and

testing of automated fall detection," *Journal of Telemedicine and Telecare*, vol. 11, no. 4, 2005.

[19] Martin Floeck, and Lothar Litz Bernd Schulze, "Concept and Design of a Video Monitoring System for Activity Recognition and Fall Detection," Institute of Automatic Control, University of Kaiserslautern, Kaiserslautern, Germany, 2009.

[20] M. Gövercin, S. Winkelbach, E. Steinhagen-Thiessen, and F. M. Wahl J. Spehr, "Visual Fall Detection in Home Environments," Institute for Robotics and Process Control of the Technische Universität Braunschweig and Geriatrie of the Charité Universitätsmedizin Berlin, Germany, 2008.

[21] Chittaranjan Mandal, and Shamik Sural Vinay Vishwakarma, "Automatic Detection of Human Fall in Video," in *PReMI 2007, LNCS 4815*, 2007, pp. 616–623.

[22] Glen Debard, Bart Vanrumste2 and Toon Goedem Jared Willems, "A Video-based Algorithm for Elderly Fall Detection," in *IFMBE Proceedings*, 2009, pp. 312–315.

[23] S. Marugán,M. Marrón, J. C. García J. M. Cañas, "Evolutive Algorithms for Visual Fall Detection in," Univ. Rey Juan Carlos,Univ. Alcalá, Móstoles and Alcalá de Henares, 2007.

[24] Franck Multon, Alain Saint-Arnaud, Jacqueline Rousseau, and Jean Meunier Edouard Auvinet, "Fall Detection With Multiple Cameras: An Occlusion-Resistant Method Based on 3-D Silhouette Vertical Distribution," *IEEE Trans. Information Technology in Biomedicine*, vol. 15, no. 2, MARCH 2011.

[25] Marjorie Skubic Erik E. Stone, "Silhouette Classification Using Pixel and Voxel Features for Improved Elder Monitoring in Dynamic Environments," in *Smart Environments to Enhance Health Care*, 2011.

[26] Serge Miguet, and Sébastien Ambellouis Nicolas Thome, "A Real-Time, Multiview Fall Detection System:A LHMM-Based Approach," *IEEE Trans. on Circuits and Systems for video technology*, vol. 18, no. 11, Nov. 2008.

[27] Zhi-Hong Ling,Yeng-Cheng Chang, Chia-Wen Lin, "Compressed-domain Fall Incident Detection for Intelligent Homecare," *Journal of VLSI Signal Processing*, vol. 49, pp. 393–408, 2007.

[28] A. Leone, P. Siciliano G. Diraco, "An Active Vision System for Fall Detection and Posture Recognition in Elderly Healthcare," CNR-IMM, Lecce, Italy, 2010.

[29] F. Temmermans, R. Deklerck B. Jansen, "3D human pose recognition for home monitoring of elderly," Department of Electrical Engineering and Informatics,Vrije Universiteit Brussel, Brussels, Belgium, 2007.

[30] Chia-Hoang Lee, Ping-Min Lin Chien-Liang Liu, "A fall detection system using k-nearest neighbor classifier," *Expert Systems with Applications*, vol. 37, pp. 7174–7181, 2010.

[31] Ju-Chin Huang Wann-Yun Shieh, "Falling-incident detection and throughput enhancement in a multi-camera video-surveillance system," *Medical Engineering & Physics*, vol. 34, pp. 954– 963, 2012.

[32] N. Suvonvorn T. Kroputaponchai, "Vision-based Fall Detection and Alert System Suitable for the Elderly and Disabled Peoples," Department of Computer Engineering, Faculty of Engineering, Thailand,.

[33] Alain St-Arnaud,Jacqueline Rousseau Caroline Rougier and Jean Meunier, "Procrustes Shape Analysis for Fall Detection," in *The Eighth International Workshop on Visual Surveillance - VS2008*, Marseille : France, 2008.

[34] J. Han, P. H.N. de With L. Hazelhoff, "Video-Based Fall Detection in the Home Using Principal Component Analysis," University of Technology Eindhoven and CycloMedia Technology B.V., Eindhoven and Waardenburg, the Netherlands, 2008.

[35] Jean Meunier, Alain St-Arnaud, and Jacqueline Rousseau Caroline Rougier, "Robust Video Surveillance for Fall Detection Based on Human Shape Deformation," *IEEE Trans. on Curcuits and Systems for Video Technology*, vol. 21, no. 5, May 2011.

[36] P. Karsmakers, et al., "Camera based fall detection using multiple features validated with real life

video," MOBILAB: Biosciences and Technology Department,Center for Health Services and Nursing Research,Lessius Campus De Nayer,Department of Internal Medicine, Division of Geriatric Medicine, University Hospitals Leuven,K.U.Leuven, Belgium, 2011.

[37] J.M.Keller,M.Skubic,X.Chen,Z.He D.Anderson, "Recognizing Falls from Silhouettes," Department of Electrical and Computer Engineering, University of Missouri, Columbia, 2006.

[38] Iván Gómez Conde, Xosé Antón Vila Sobrino David Nicholas Olivieri, "Eigenspace-based fall detection and activity recognition from motion templates and machine learning," *Expert Systems with Applications*, vol. 39, pp. 5935–5945, 2012.

[39] A. Prati ,R. Vezzani R. Cucchiara, "A multi-camera vision system for fall detection," *Expert Systems*, vol. 24, no. 5, November 2007.

[40] Yu-Ching Lin, and Wen-Hsien Fang Yie-Tarng Chen, "A Video-based Human Fall Detection System for Smart Homes," *Journal of the Chinese Institute of Engineers*, vol. 33, no. 5, pp. 681-690, 2010.

[41] Miao Yu, and Jonathon Chambers Adel Rhuma, "Posture Recognition Based Fall Detection System," *Lecture Notes on Software Engineering*, vol. 1, no. 4, November 2013.

[42] Salim Zabir,Bastian Leibe Simin Wang, "Lying Pose Recognition for Elderly Fall Detection," Proceedings of Robotics: Science and Systems, Los Angeles, USA, 2011.

[43] R. H. Luke, M. Skubic, J. M. Keller, M. Rantz, M. Aud D. Anderson, "Evaluation of a Video-Based Fall Recognition System for Elders Using Voxel Space," University of Missouri, Columbia, 2008.

[44] H. A. Habib M. J. Khan, "Video Analytic for Fall Detection from Shape Features and Motion Gradients," in *Proceedings of the World Congress on Engineering and Computer Science,WCECS 2009*, vol. Vol II,WCECS 2009, San Francisco, USA, October 2009.

[45] Jean Meunier, Alain St-Arnaud and Jacqueline Rousseau Caroline Rougier, "Monocular 3D Head Tracking to Detect Falls of Elderly People," in *Proceedings of the 28th IEEE EMBS Annual International Conference*, New York City, USA, 2006.

[46] Changling Zuo Hong Liu, "An Improved Algorithm of Automatic Fall Detection," in *2012 AASRI Conference on Computational Intelligence and Bioinformatics*, 2012, pp. 353 – 358.

[47] Wikipedia. [Online]. http://en.wikipedia.org/wiki/OpenCV

[48] Wikipedia. Eclipse. [Online]. http://en.wikipedia.org/wiki/Eclipse_%28software%29

[49] Weka. [Online]. http://www.cs.waikato.ac.nz/ml/weka/

[50] C. Rougier, J.Meunier, A. St-Arnaud, J. Rousseau E. Auvinet, "Multiple cameras fall dataset," Université de Montréal, Montréal, Technical report 1350, DIRO 2010.

[51] Vaclav Hlavac, Roger Boyle Milan Sonka, *Image Processing. Analysis. and Machine Vision. International Student Edition*, Hilda Gowans, Ed. United States of America: Thomson Learning, part of the Thomson Corporation, 2008.

[52] Jean Ponce David A. Forsyth, Computer Vision: A Modern Approach, 2nd ed., Michael Hirsch, Ed. New Jersey, United States of America: Pearson Education, 2012.

[53] Zoran Zivkovic, "Improved Adaptive Gaussian Mixture Model for Background Subtraction," in *Proc. ICPR*, 2004.

[54] Ferdinand van der Heijden Zoran Zivkovic, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, November 2006.

[55] Gary Bradski and Adrian Kaehler, *Learning OpenCV*, 1st ed., Mike Loukides, Ed. United States of America: O'Reilly Media, 2008.

[56] Brian C. Lovell Sambhunath Biswas, *Bezier and Splines in Image Processing and Machine Vision*. London: Springer-Verlag, 2008.

[57] Bernd Streicher-Abel,Carsten Garnica Christian Demant, *Industrial Image Processing*, Second Revised ed. Stuttgart, Germany: Springer, 2013.

[58] Wikipedia. Green's Theorem. [Online]. http://en.wikipedia.org/wiki/Green%27s_theorem

[59] Johannes Kilian, "Simple Image Analysis By Moments," 2001.

[60] Daisheng Luo, *Pattern Recognition and Image Processing*, 1st ed.: Woodhead Publishing Limited, 1998.

[61] Mark J. Burge Wilhelm Burger, *Principles of Digital Image Processing*, Ian Mackie, Ed. London: Springer-Verlag, 2009.

[62] Wikipedia. Moments. [Online]. http://en.wikipedia.org/wiki/Image_moment

[63] R.S. Busy R.E. Kalman, "New Results in Linear Filtering and Prediction Theory," *Transactions of the ASME, Journal of Basic Engineering*, pp. 95-108, March 1961.

[64] Greg Czerniak. Kalman Filter. [Online]. http://greg.czerniak.info/guides/kalman1/

[65] Wikipedia. Kalman Filter. [Online]. http://en.wikipedia.org/wiki/Kalman_filter

[66] Pierre Kornprobst Gilles Aubert, *Mathematical Problems in Image Processing*, 2nd ed., J.E. Marsden,L.Sirovic S.S. Antman, Ed.: Springer, 2000.

[67] Wikipedia. Optical Flow. [Online]. http://en.wikipedia.org/wiki/Optical_flow

[68] Gunnar Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion," Computer Vision Laboratory, Link□oping University, Link□oping, Sweden, 2003.

[69] Wikipedia. The aperture problem. [Online]. http://en.wikipedia.org/wiki/Motion_perception#The_aperture_problem

[70] Stoomey. Aperture Problem. [Online]. https://stoomey.wordpress.com/2008/04/18/20/

[71] OpenCV. Operations of Matrices. [Online]. http://docs.opencv.org/modules/gpu/doc/operations_on_matrices.html

[72] Polar-Cartecian Coordinates. [Online]. http://www.mathsisfun.com/polar-cartesian-coordinates.html

[73] kb6nu. Cartesian and Polar Coordinations. [Online]. http://www.kb6nu.com/extra-class-question-of-the-day-impedance-plots-and-coordinate-systems

[74] Neeraj et al, "Decision Tree analysis on J48 algorithm for data mining," *International Journal of Advanced Research in Computer Science and Software Engineering*, pp. 1114-1119, June 2013.

[75] Patrick, "Artificial Intelligence," Radboud University Nijmegen, Nijmegen, BSc Thesis 2008.

[76] Leo Breiman, "Random Forests," Statistics Department, University of Berkeley, Berkeley, California, 2001.

[77] Wikipedia. Generalization Error. [Online]. http://en.wikipedia.org/wiki/Generalization_error

[78] (2005) Naive Bayes. [Online]. http://www.inf.ed.ac.uk/teaching/courses/lfd/lectures/lfd_2005_naive.pdf

[79] Wikipedia. Naive Bayes Classifier. [Online]. http://en.wikipedia.org/wiki/Naive_Bayes_classifier

[80] Princeton University. Naive Bayes Classifier. [Online]. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Naive_Bayes_classifier.html

[81] Lectures. [Online]. http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf

[82] Logistic Regression. [Online]. http://www.strath.ac.uk/aer/materials/5furtherquantitativeresearchdesignandanalysis/unit6/whatislogisticregression/

[83] Shakesha Anderson, "Logistic Regression," [Online] http://schatz.sju.edu/multivar/guide/logistic.pdf

[84] Sing-Tze Bow, *Pattern Recognition and Image Pre-processing*, Second Revised and Expanded, Ed. New York: Marcel Dekker, 2002.

[85] E.R Davies, *Computer & Vision: Theory, Algorithms, Practicalities* , Fourth ed.: Elsevier, 2012.

[86] Theofilis George-Nektarios, "Weka Classifiers Summary," Athens University of Economics and Bussiness Intracom-Telecom, Athens , 2013. [Online]

https://www.academia.edu/5167325/Weka_Classifiers_Summary

[87] David M W Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies 2*, pp. 37–63, 2007.