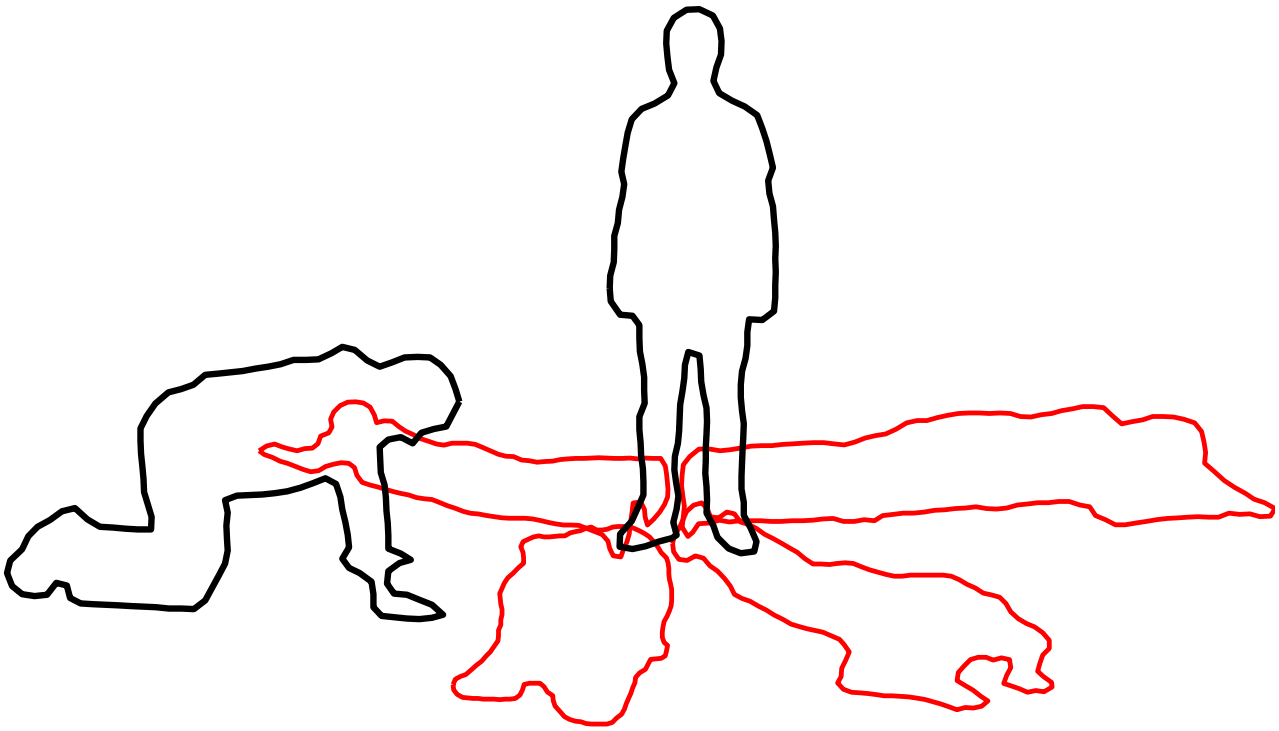
CHALMERS



Detecting falls and poses in image silhouettes

Master's thesis in Complex Adaptive Systems

NIKLAS SCHRÄDER

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011
Master's thesis

MASTER'S THESIS IN COMPLEX ADAPTIVE SYSTEMS

Detecting falls and poses in image silhouettes

NIKLAS SCHRÄDER

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011

Detecting falls and poses in image silhouettes
NIKLAS SCHRÄDER

Master's thesis
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Some of the silhouettes that are used in developing the algorithm of this thesis. Falls are drawn in red and non-falls in black.

Chalmers Reproservice
Gothenburg, Sweden 2011

Detecting falls and poses in image silhouettes
Master's thesis in Complex Adaptive Systems
NIKLAS SCHRÄDER
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

ABSTRACT

About one third of all people aged 65 and above will accidentally fall during one year. A fall can have severe consequences, such as fractures, and a fallen person might need assistance at getting up again. A lot of research has been dedicated into the development of automatic fall detection methods during the recent years. These automatic methods are created to detect falls so an alarm can be raised and help can come.

In this thesis, a part of a fall detection system for a household robot aimed at helping the elderly is developed. The system is able to classify human pose from a silhouette in an image. By associating the pose "lying down" with a fallen person, the system can be used for fall detection. The algorithm is based on an image analysis feature called shape contexts. These shape contexts describe distributions of edge points by binning them into polar histograms. Altough the dataset used for training contains falls in many difficult angles, the algorithm classifies falls correctly for 97 % of a set of unseen images.

Keywords: Fall detection, pose recognition, image analysis, shape context

ACKNOWLEDGEMENTS

I would like to thank Dr. Krister Wolff for supervising me in this thesis, helping me find my way forward and sharing his thoughts and valuable comments with me. And thank you Dr. David Sandberg for our interesting discussion about the implementation in C#. I would also like to thank Dr. Alessandro Romeo for introducing me to the amazing field of digital image processing, and Prof. Bernhard Mehlig and Prof. Mattias Wahde for inspiring me to learn more about neural networks and optimization methods. Finally, I would like to thank my models Olov, Per, Viktor, Fadi, Suvash, Simon and Anders, not just for allowing me to make a big dataset, but also for all coffee and sushi breaks and your valuable comments. It's been a great pleasure working with you. Thank you all!

CONTENTS

Chapter 1

Introduction

As humans become old, their bodies weaken and the risk of accidental falls increases. There are many kinds of falls that can occur, e.g. an elderly person might fall while walking down a staircase or trying to sit down in a chair that turns out not to be there [7]. Statistically, about one third of those aged 65 and above experience a fall during one year [17, 20] and falls are also the reason for more than 90 % of hip fractures among elderly in the United States [17]. Moreover, an elderly and possibly injured person might not be able to get back up again after a fall, due to either fractures or declining muscular activity [14, 40]. The dangers of remaining on the floor for a long time after a fall include pressure sores and dehydration, even death [14]. In addition, further consequences of falls can also be psychological, such that the fear of another fall limits the activities a person chooses to perform [17, 40].

The dangers associated with not being able to get up are for obvious reasons greater for elderly people living alone, than for those living with their spouse or if their home is frequently attended by elder care nurses. As the population of elderly is increasing dramatically in almost all countries of the world [46], having very frequent nurse visits is not a feasible option for monitoring of falls. A better solution that is used widely today is to provide wearable call alarms for the elderly. Unfortunately, many elderly forget to wear the alarms or are hesitant to use them, even if they have fallen. The reasons include a will to be independent and fear of being taken into hospital [14]. Hence, there is a need to develop automatic fall detection systems that are able to either raise the alarm or help the fallen person to get back up safely.

This thesis describes methods for detecting if a person in a digital camera image has fallen. It also covers results of a promising detection method. The result will contribute to the development of an automatic system that uses cameras for providing safety and thereby increased well-being and independence for the elderly.

1.1 Motivation

During the recent years, a lot of work has been done on detecting falls using a wide range of sensor types, including pressure [42], acceleration [7, 9, 31] and sound [24, 41] sensors as well as depth field cameras [23], infrared [42] and visible spectrum cameras [3, 5, 8, 11, 16, 22, 28, 30, 32, 33, 36, 47, 50]. The vision-based systems that use cameras have important benefits over the other ones. For instance, the wearable accelerometers that associate a high vertical acceleration with a fall are good detectors, but occasionally elderly people forget to wear them [5, 11, 15, 36]. In contrast, a camera can monitor the elderly at all times even if they forget about the existence of the security system.

Another advantage of a vision-based system is that it provides a lot of data for the detection algorithm to analyze. By using a combination of the images from the actual fall and from the following events, it can be possible to determine not just that a fall has occurred but also to classify how serious the fall was. As an example, camera images clearly show if the person is able to get up again after the fall or not. Other sensors such as accelerometers cannot differentiate between the severity of these two cases as easily. In addition, if an accelerometer based system fails at detecting a fall, it has no second chance at detecting a fallen person.

Automatic systems that make use of cameras are also important from a privacy aspect [3]. The captured images can be processed locally and then be removed immediately after analysis, unless some abnormal activity is detected [11, 32]. Abnormal activities include falls, and in case one is detected, that image can be sent to a human operator who will choose whether to raise the alarm or not. Such a combination of human and machine knowledge leads to a system that yields a good compromise between providing security for the elderly and still preserving their privacy.

1.2 Objective

The goal of this thesis is to investigate methods that can differentiate between various types of human poses, to be used in a future automatic system for fall detection. Specifically, the system should be able to decide whether a person in the image has fallen and might need assistance or not. This system is constructed to be a part of a household robot that will aid an elderly person in his or her home. An important feature of such a machine is to detect emergencies, such as falls in the home environment, and raise the alarm if these unusual and possibly dangerous events happen.

1.3 Limitations

This project covers methods for detecting a fallen (i.e. lying) person in still images, where it should be possible to use the resulting method on a future household service robot. Other features of such a robot are not investigated, such as the design of the robot or how it will interact with humans.

The fact that the system will eventually be mounted on a moving robot complicates the problem in many ways. One important consequence is that segmenting the taken images into human foreground and non-human background cannot be done in the same way as with stationary surveillance cameras [12]. Therefore, the segmentation problem will be outside of the scope of this project. Instead, the methods developed will assume that the input images have already been segmented automatically using another algorithm.

A set of training images will be used to optimize the algorithm and set its parameters. Because the average probability of fall for an elderly person is around 30% per year [17, 20], the generation of small datasets of real fall data require long periods of time [7]. In this project, images used for training and validating the algorithm are taken in a lab with university students as models. These are asked to assume a set of poses, among them to lie down as if they had fallen. The training images are then segmented manually using photo editing software.

As the product of this project is designed to fit a robot that will interact with humans, a certain amount of false positives can be allowed. A false positive means that the system detects an action, such as a fall, even if none has happen in reality. When the robot detects a person lying on the floor, it can then use its interaction skills to determine if it needs to help or call for assistance. Hence, a larger ratio of false positives can be accepted than the number of false negatives, i.e. a fall has occurred but the algorithm does not detect it.

1.4 Related work

A wide range of methods have been proposed for detecting a fallen person in images. The existing work covers two areas: pose recognition and fall detection. Pose recognition is about detecting human postures in images. These methods are designed to differentiate between poses such as standing up and lying down (after a fall). Fall recognition methods use temporal sequences of images to capture and detect the actual fall event. By combining the two techniques, a fall alarm can be raised whenever a fall event is detected and followed by longer times of lying down on the floor.

A common pose detection method uses properties of the bounding box, i.e. the box that surrounds the human in the image. Its width to height ratio is used by many for its simplicity [3, 28, 41]. Others also use the height and direction of this box as features in their classification [22, 33, 50]. A more advanced version of the bounding box that gives more information about the pose are the so called vertical projection histograms [24, 30, 42]. These histograms are created by counting the number of pixels (or voxels in the 3-D case [5, 8]) on various levels above the floor.

There are also proposed methods for finding humans in images. Several of these use a histogram of gradient (HOG) method [2, 37, 39], where the gradient vectors in images are calculated and binned into histogram. As large image gradients are created from strong edges, these histograms can be used for finding key points such as a pair of shoulders or elbows [2]. Some other methods use the fact that a human body has a particular shape. For instance, the human torso can be found and an ellipsis that covers it yields information on where the person is and his or her pose [15, 47]. Other methods detect the limbs of a person, either from a silhouette or by starting from the head or body trunk [13]. Such methods have been proven well for detecting pedestrians walking across cluttered backgrounds [44].

The detection of fall events is typically performed by comparing a sequence of images [15, 23, 30, 48]. A fall is typically detected if the vertical fall speed is higher than a certain threshold value [22]. The speed can be estimated from the bounding box or body trunk ellipse. It has also been estimated by calculating optical flow [11], i.e. how pixels move from image to image [38]. Several other features have been proposed for increasing the fall detection accuracy, such as microphones capturing the fall sounds [24, 41]. Another proposed method is to divide the living area into zones, where long times of inactivity will only raise an alarm if the person is not in a low activity zone, such as a bed, an armchair or a sofa [23, 24, 36].

In addition to using a wide range of features, a wide range of camera settings as well as classification methods have been employed. These include standard image analysis tools such as linear regressors [1], support vector machines (SVMs) [10, 33], hidden Markov models (HMMs) [3, 41, 49], and feed forward neural networks (FFNNs) [15, 35].

1.5 Reading guidance

This thesis report is organized as follows. Chapter 2 explains some basic theory in image processing and analysis. The important image features called shape contexts are also introduced here. Chapter 3 describes the developed algorithm for pose detection. Chapter 4 lists the results as well as explains the effect of the parameters of the model. Finally, the report is concluded by chapter 5 where some suggestions for further research are also made.

Chapter 2

Theory

In this chapter some fundamental concepts of digital imaging are presented. In addition, the shape descriptors called shape contexts are introduced as well as some general information on classification methods.

2.1 Digital images and computer vision

Human beings use many senses for the perception of their surrounding world. It is possible for us to conclude many things from what we observe, such as where we are, what weather it is or if another person looks happy or sad. The reproduction of similar functions using electronic images is commonly referred to as *computer vision* or *machine vision*, and requires that images are digitalized. Digital images are stored as numbers, and from these numeric values it is possible to calculate many features and devise classification methods, out of which a subset are described in the following sections.

In general, computer vision tasks can be divided into three categories: image processing, image analysis and machine learning. Image processing is usually the first step in an algorithm, as it involves methods for enhancing images, e.g. increasing contrast, brightness and sharpness as well as deblurring and compressing images. The next step is image analysis, where the images are analyzed by calculating features from their content. Finally, the computer learns from and decides what to do with the features it just calculated, such as a robot deciding to change its direction of movement.

2.1.1 Image representations

A basic digital image describes the amount of light at different positions. It can be seen as a function $f(x,y)$ of some finite physical property in the 2-dimensional space that is defined by coordinates x and y [18]. Because digital images are stored as numeric values, the function of intensity has to be discretized in space, which is referred to as *sampling*. This corresponds to only using the intensity values at a limited set of points, called *pixels*. Obviously, the intensity values also have to be described by a limited set of numbers, and this process is called *quantization*. As an example, a typical digital camera today produces images sampled to more than 10 megapixels and quantizes the intensity of incoming light to an integer from 0 (black) to 255 (white).

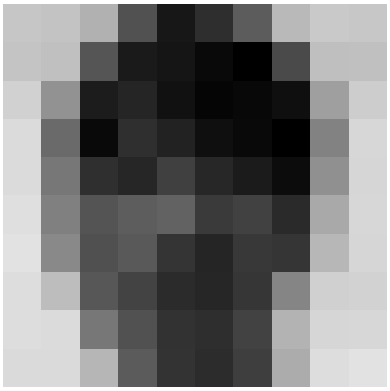
With digital images described as a matrix of sampled and quantized intensity, it is possible to describe the light intensity in a grayscale image as in figure 2.1. To make color images, three matrices are typically used where the triplet at a particular index describe the color of that pixel. What is actually stored in each layer depends on the color model. Some common models in digital imaging include RGB where the three layers describe the amount of red, green and blue for the pixel and HSI where the layers describe the hue (the pure color such as red, yellow or green), the saturation (if it is gray or colorful) and intensity (brightness from dark to bright) [18]. In some applications, it is beneficial to use a specific color model, and easily implemented transition functions exist between the most common ones.

2.1.2 Segmentation

One important task in image processing and analysis is to divide an image into different areas, where the pixels in each area have some property in common [38]. This *segmentation* can for instance divide an image into a foreground (e.g. a person) and background (e.g. grass, walls, trees). The output of the segmentation can then be used for calculating features for each interesting part of the image or to apply other image processing methods to them.

198	195	177	81	23	46	92	185	201	199
196	192	85	26	21	9	0	74	191	192
209	146	28	37	17	5	8	15	159	205
219	106	9	47	34	15	9	0	130	215
219	119	47	38	64	39	27	12	144	214
223	128	84	93	98	58	65	42	169	215
226	137	80	89	52	37	56	53	183	214
221	189	87	67	43	38	54	133	208	210
221	219	119	81	50	46	66	179	214	215
218	218	182	91	51	44	62	172	221	226

(a) *Numeric values.*



(b) *Visual display.*

Figure 2.1: *A 10-by-10 pixel image of a human head in two different representations.*

2.2 Object recognition

A common computer vision task is to recognize objects in images. This process is performed in several steps. The first is to segment the image into regions or to find the pixels that might correspond to an object. For each region of interest, a certain set of features are calculated. Then, the actual recognition is performed by comparing these sample image features to a database of features from known shapes. A classifier is eventually employed to make the decision based on the comparison results.

The image region features used in this project are all shape descriptors. They describe only the border of the segmented regions. Other features can be calculated from the actual pixels within the regions and be used for tasks such as texture matching [38]. A simple feature is to count the number of pixels in a region, thereby calculating its area. As an example application, it has been used in monitoring the sizes of plants during their early growth in laboratory experiments [25].

2.2.1 Shape descriptors

Shape descriptors are features that capture and describe the shape of a segmented area. A property shared among many of them is their ability to recognize shapes even if the shapes are moved, resized or rotated in the image. This is usually referred to as invariance to translation, scaling and rotation, respectively. In addition, some invariant features are also able to handle perspective projections, illumination changes and various levels of occlusion in the image [26]. The following sections describe the shape descriptors used in this project and their invariant properties.

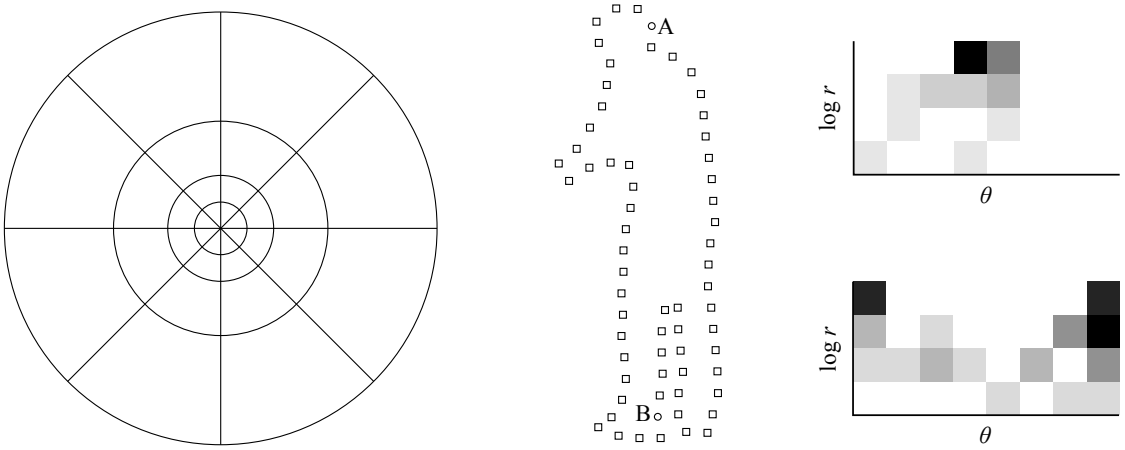
Bounding rectangle

The bounding rectangle or bounding box is the rectangle of minimum area that bounds the shape. This rectangle can be constrained to be aligned with the coordinate axes, which is typical in programming [29], or rotated into the direction that minimizes its area [38]. The aspect ratio, that is the width of the rectangle divided by its length, is an example of a scale invariant feature that tells whether the major direction of the object is horizontal or vertical. If the box has no restriction on its direction, the information about major spread will instead be given by the tilting angle of the box.

Shape context

The *shape context* is a sophisticated shape descriptor that describes the relationships between border points of a shape. For every border point described by position vector $\vec{p}_i = \{x_i, y_i\}$, the shape context describe the distribution of other border points, relative to the first. Hence, for point k, the set of vectors $\{\vec{p}_i - \vec{p}_k, i \neq k\}$ is used and the shape contexts are translation invariant. These vectors are binned into a two-dimensional log-polar histogram, which defines the actual shape context [6].

The polar histogram contains bins for vector length and direction. A log-polar histogram is similar, but using the logarithmic vector length instead. Typical number of bins in these histograms are 12 for the number of angles and 5 for logarithmic length [6]. In addition, the logarithmic radial scale requires a minimum and a maximum. This enables for making the shape contexts invariant also to scaling by selecting the limits as ratios of the overall shape size. The sizes of bins for a histogram that covers the height of a shape is shown in figure 2.2a.



(a) *Log-polar histogram with 4 radial and 8 angular bins.*

(b) *A human shape with two marked points.*

(c) *Histogram values for point A (top) and B (bottom).*

Figure 2.2: *Demonstration of how shape histograms are calculated for the border points of a human silhouette.*

Since there can be many border points on the shape, corresponding to equally many shape histograms, a very large number of data values are required for describing a shape. It is more feasible to first sample the border points by picking only a limited subset of them, and to only use these for calculating the histogram. Figure 2.2b demonstrates a shape border downsampled to 64 points and the values of two shape context histograms are displayed in figure 2.2c.

Another useful property of shape histograms is that they can be used for comparing shapes and finding correspondences. This can be performed by associating the border points on one shape with those on another. By giving each association a certain cost, the best correspondences can be found using minimization methods [6]. This might prove useful for finding how a shape has been distorted, or identifying body parts in human silhouettes.

2.2.2 Set divisions

An important aspect of algorithms that classify data is their performance on unseen data. One method of evaluating this is to divide the data set into subsets, typically a *training set* and a *validation set*. Obviously, the training set is used for setting the parameters of the model or to "train" it. Thereafter, the general performance of the algorithm is evaluated by applying the validation set to it, as this is unseen data for the algorithm.

Almost any data set will contain a certain degree of noise. Consequently, an algorithm may adapt to this noise if it is optimized using training data only. This phenomenon is known as *overfitting*, and can be identified by monitoring the validation set performance during training. Initially, validation set performance will increase and reach a maximum, followed by a decrease. Therefore, one method of solving the overfitting problem is to use the parameters as they were when validation set performance started to decrease, a method called *holdout validation*. If an algorithm is trained using holdout validation, it has been optimized for both the training and validation sets. Hence, a third data set is typically tested with the algorithm for evaluating its performance on unseen data. This is referred to as the *test set* [43].

2.2.3 Classification methods

A *classifier* is the part of an algorithm that is able to tell which class a specific set of feature values belong to. These feature values can be regarded as an input vector $\vec{x}$, and the classification output as another vector $\vec{y}$. One interpretation of $\vec{y}$ is that each component y_i provides a measure on how close the feature vector is to the class with index i. Hence, the index of the largest value in $\vec{y}$ can be selected as the class which the feature vector $\vec{x}$ belongs to. Another method is to associate the feature vector with all classes where the output values are larger than a certain threshold.

One method of generating a classifier is to find the matrix $\mathbf{M}$ such that the product $\mathbf{M}\vec{x}$ is as close to the desired output vector $\vec{y}$ as possible. This matrix can be found by assembling all feature vectors $\vec{x}_i$ into a matrix $\mathbf{X} = [\vec{x}_1 \dots \vec{x}_N]$ and classification vectors $\vec{y}_i$ into $\mathbf{Y} = [\vec{y}_1 \dots \vec{y}_N]$, where N is the number of samples in the training data set. Then, the desired matrix is obtained by solving $\mathbf{MX} = \mathbf{Y}$. This system is usually heavily overdetermined, so a least-squares regression estimate $\tilde{\mathbf{M}}$ can be found by solving the transposed problem $\mathbf{X}^T\tilde{\mathbf{M}}^T = \mathbf{Y}^T$ using QR decomposition through Householder transformations [21], or by using the so-called normal equations [1].

An alternative method is based on artificial neural networks. These networks rely on mathematical models of simplified biological neural cells (neurons) for computation. The basic idea is that a neuron fires, meaning it turns itself on, if it receives enough input from other firing neurons. A feed-forward neural network (FFNN) uses layers of neurons that only send signals to the next layer, thus feeding the signal forward. Each layer is calculated from the previous layer by multiplying a weight matrix to a vector of the neuron states. This raw input is then passed through an activation function which limits the range of output values, and is typically the sigmoid $\frac{1}{1+e^{-x}}$ or $\tanh(x)$. The weights of the feed-forward neural networks can be optimized using several different algorithms. A straightforward method for iteratively updating the network is the *backtracking* algorithm, which can be derived by applying the gradient descent method on the network [19].

2.2.4 Evaluators

The performance of a classification algorithm can be evaluated in many ways. A very common method is to only consider two possible outcomes, such as "belongs to class A" and "does not belong to class A". Since a training database contains both images from which features are calculated as well as their classification, a "ground truth" exists. Depending on the ground truth and the calculated output from the classifier, there are four different cases. These are known as true positive (TP), false positive (FP), true negative (TN) and false negative (FN). Simply explained using the confusion matrix in figure 2.3, a positive means that the object is classified as a certain object and a negative that it is not classified as a such. The prefixes true and false refer to whether this classification was correct or not. For instance, a true positive means that the object belongs to a certain class and was classified as a such. On the contrary, a false negative means that it was wrongly classified not to belong to the class it ideally should. Hence, the diagonal elements correspond to correct classifications, and all others to errors.

	Classified as A	Classified as not A
Is A	TP	FN
Is not A	FP	TN

Figure 2.3: *Confusion matrix.*

A confusion matrix can also be extended to show several categories. In this case, the diagonal elements will still show the correctly classified ratios. The off-diagonal entries will not just show the ratio of mis-classifications, but also how these are distributed among the other possible categories.

Two common performance measures are commonly calculated from the confusion matrix. The *sensitivity* contains the fraction of correctly classified objects that are of a certain class. Similarly, the *specificity* contains the fraction of correctly classified objects that do not belong to the certain class [7]. Both of these measures are equal to 1 for a perfect classifier, and they are calculated as follows.

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2.1}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}} \tag{2.2}$$

Chapter 3

Method and implementation

This section describes the algorithm for fall detection and how it can be trained. Instead of capturing the actual fall event, the method should be able to detect human poses, with focus on differentiating a person lying down from a person standing, sitting or crouching. This section also describes the dataset of images used in training the algorithm.

3.1 Pose dataset

Since the project is about detecting a fallen person in an image, the algorithm has to be able to differentiate between a fallen person and a non-fallen person. Hence, a data set consisting of various poses from everyday life has to be created. These poses can be grouped together into four classes, since they describe the same pose but a different action. All poses with their class are listed in table 3.1 together with the number of photos of each pose.

Table 3.1: The nine poses used in the dataset.

Pose ID	Description	Class	Number of images
1	Standing, arms resting	Standing	37
2	Standing, working with arms out	Standing	37
3	Walking slowly	Standing	36
4	Walking fast	Standing	38
5	Crouching	Crouching	39
6	Lying on front	Lying	33
7	Lying on back	Lying	33
8	Lying on side	Lying	30
9	Sitting on chair	Sitting	30

Photos are taken in a bright room using a Nikon D40 camera mounted on a tripod. A group of 7 university students are used as models and were asked to assume all poses in various directions. To make segmentation into human and non-human in the images easier, the images are first segmented automatically by comparing them to a background image without a person. The difference is thresholded, meaning that all pixels where the difference is larger than a certain threshold are marked as foreground. As this simple method of subtracting the background can give rise to a very noisy segmentation, morphological opening and closing [18] are used, as well as manual refinement. The segmentation is stored in the images by coloring all background pixels black. In addition, the images are cropped to remove unnecessary background pixels.

All training samples are divided into a training and validation set. The division is based on grouping all combinations of (model, pose) and picking two from each group. This yields a training set consisting of 197 images and a validation set consisting of 116 images. For increasing the algorithm robustness, the training set is doubled by adding a mirrored copy of each image in it. Similarly, the test set is constructed by mirroring all images in the validation set.

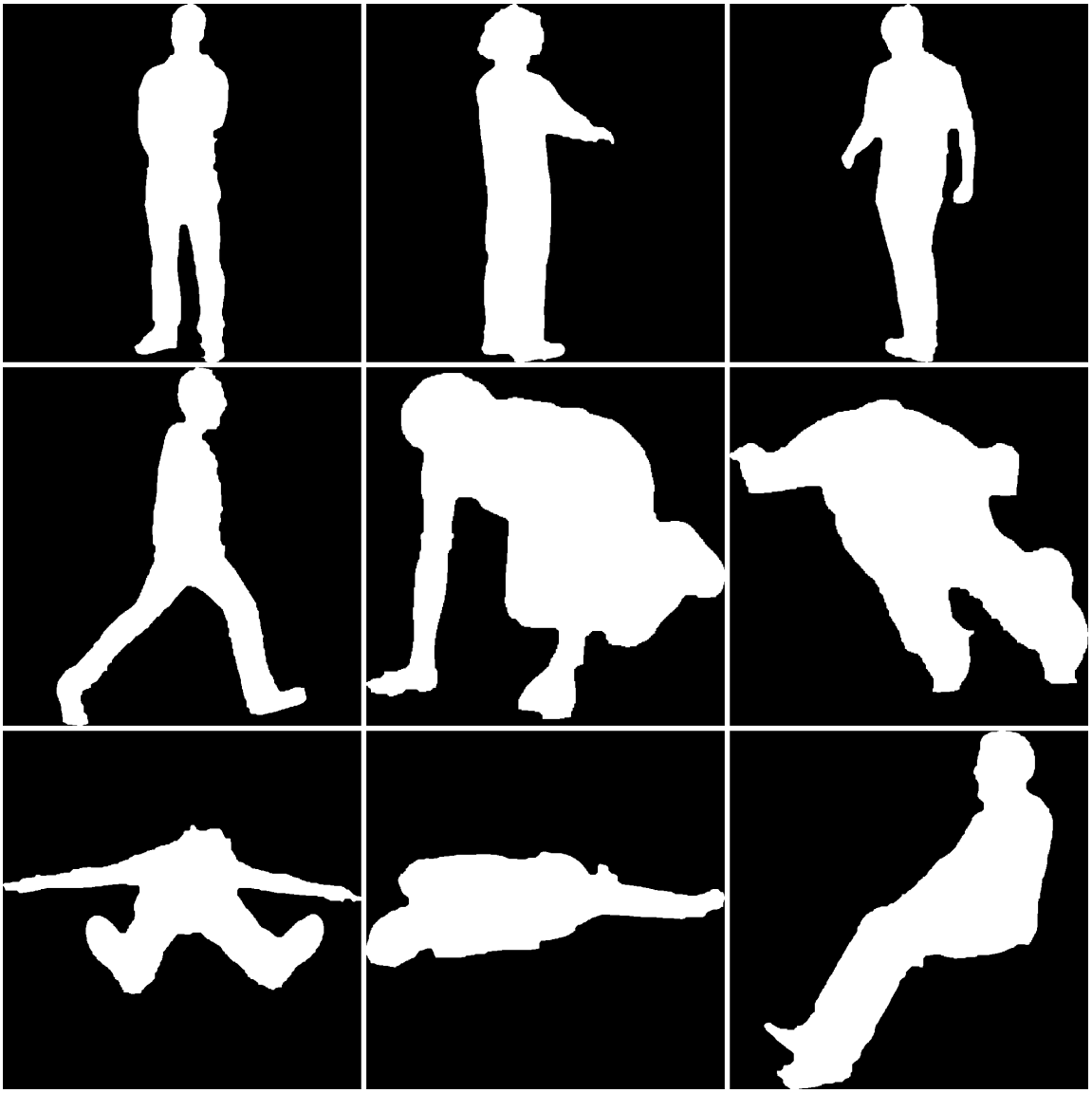


Figure 3.1: *Shapes from the dataset, ordered according to pose ID row by row.*

3.2 Algorithm description

The algorithm that is devised in this thesis takes an image with corresponding segmentation information as input, inspired by the method in [1]. After processing as depicted in figure 3.2, the algorithm is able to classify the image as one of the nine poses presented in table 3.1. As the problem of segmenting images is very complex and depends on the conditions at which the images are taken [12, 34], it is assumed that some segmentation method has already been employed.

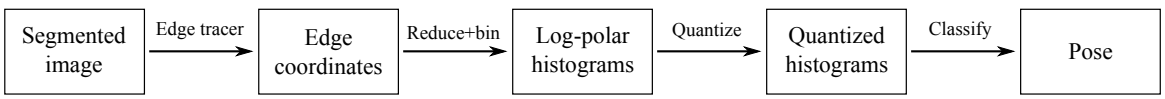
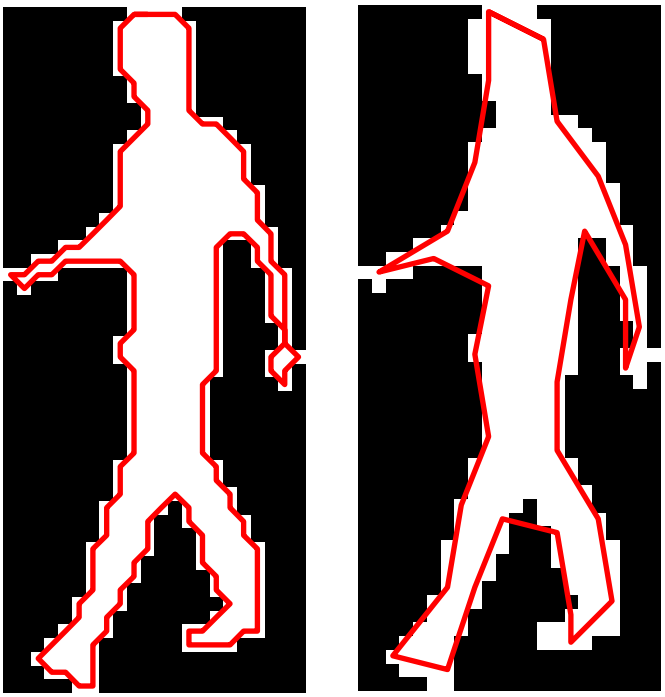


Figure 3.2: *The workflow of the algorithm.*

The steps included are to trace the edges, calculate histograms, quantize these histograms, and then to classify using these quantized values. All steps can be performed in many different ways. Therefore, the following sections describe these steps in detail and mention other possible methods than can be employed for the same task.

3.2.1 Edge tracing

The segmentation that comes with the image can be represented as a matrix of the same size as the image. It is then possible to use common image processing operations and methods for finding the edge points in this image. An edge point is here considered to be a foreground pixel with at least one background pixel in its 4-neighborhood (the nearest pixel above, below, to the right and left). A description of an edge corresponds to the coordinates of these edge points, but ordered around the shape, as shown in figure 3.3a. All edges in the dataset images are traced using a common edge tracing algorithm outlined in [38]. If the edge tracing algorithm finds several edges, due to noise in the segmentation, only the one with most points is kept and the others are discarded.



(a) *Full edge* (b) *Reduced to 32 points*

Figure 3.3: *A silhouette from the dataset with its traced edge in red. The image has been downsampled to make individual pixels distinguishable.*

The traced edges may contain several thousand edge points for high-resolution images, such as those in the training database. To allow for comparison and classification of images of varying size, as well as reducing the computational load, all edges are sampled to a limited number of points. Figure 3.3b shows what a downsampled edge may look like, and it is from these edges that the shape context histograms are calculated. The number of points in the reduced edge are taken from the powers of 2 (32, 64, 128, 256), since some other shape descriptors require the use of the Fast Fourier Transform (FFT) which is optimized for these numbers of points.

3.2.2 Shape context calculation and quantization

Shape context histograms (see section 2.2.1) are calculated for each point in the reduced edge. Because these histograms describe the relative position of points by angle and logarithmic distance, the histogram bins have to be set accordingly. As the logarithmic scale requires a minimum value > 0, both the minimum and maximum radial bin size has to be specified. In addition, these limits can be set to be a fraction of the overall shape size, defined as the maximum value of the bounding box width and height, to make the shape contexts scale invariant. The minimum is selected depending on the number of edge points, so that the smallest bins are slightly larger than the average spacing between points in the reduced edge, typically 3–5 % of the shape size. The maximum distance in the histogram should be large enough to include the major part of the shape, typically 50–80 % of shape size. The number of angular bins is chosen to be 12 as in the original article [6], but the radial bins to 3–4 instead of 5.

With a shape context histogram containing 36 or 48 values for each edge point, the shape is described by many data values. Similar to [1], the solution here is to quantize these histograms. The quantization process groups edge points

together if their shape context histograms are similar, by finding a set of typical shape context histograms, called *cluster centers*. Each real histogram from a shape is then associated with the nearest cluster center, where distance is calculated as sum of squared differences (L_2 norm).

By considering each histogram as a vector of values, any vector quantization (VQ) method can be utilized for finding a set of cluster centers. Here, the k-means clustering algorithm [38] is used on all shape contexts from the training set. With typical values, such as 200 images in the training set, edges reduced to 128 points and 64 cluster centers, each iteration of the clustering algorithm requires the computation of 1.6 million distances, each in 36 or 48 dimensions. Hence, the method is computationally expensive and requires a large part of the overall processing time of the algorithm, and is limited to perform a maximum of 40 iterations.

After clustering, each point of the reduced edge has been associated with the nearest cluster center. The shape can then be described by a vector of the same dimension as the number of cluster centers. In this vector, entry *i* is the number of edge points in the current shape nearest to cluster center *i*. Hence, it can be regarded as second level of histogramming, where this vector describes the distribution of shape contexts of that particular shape, and the cluster centers define the bins. In addition, the number of cluster centers should be small enough to group several points on the same shape together. Yet the number of cluster centers has to be large enough to cover the varieties of the dataset. Therefore, it is desirable to keep the number of cluster centers to around 0.5–0.8 times the number of edge points.

As an illustrative example, a result of shape context clustering from images of humans can yield a cluster center that is characteristic for a left foot. If an image of a standing person is supplied to the algorithm, several of the edge points from the left foot should be associated with that particular cluster, giving it a high value in the vector of quantized values. On the other hand, if the image is that of a crouching person, the same counter might be zero. Therefore, this type of descriptor can be regarded as a descriptor that identifies body parts from silhouettes.

3.2.3 Classification

The purpose of the classifier is to transform any input vector consisting of quantized histograms into an output vector that describes the detected pose. In this algorithm, the output vector contains 9 elements, each one corresponding to each of the poses. For instance, entry 2 in this vector gives a measure on the similarity between the classified shape and the database shapes of a standing person with arms working (pose ID = 2). For training, all entries in the desired output vector are set to zero except for the correct pose, as this value is set to unity.

Two approaches that both use matrix multiplications are tested for their ability to classify the sample data set. The first one involves a single multiplication by a regression matrix. This matrix is obtained by linear least squares regression calculated through QR decomposition [4]. The second approach is a feed-forward neural network (FFNN) with a sigmoid function as activation function. It is trained in epochs using backpropagation. An epoch is defined as a pass through all training data [19], which here are taken in random order. Between each epoch, the validation set performance is evaluated by calculating the squared sum of differences between network output and desired output vector. Training goes on for a maximum of 100 epochs until the error has increased to more than 5 % of its lowest value. The final network weights are then taken where the moving average error has its minimum, in accordance with the holdout validation method.

The final classification pose is taken as the index with highest value in the output vector from the regression matrix or the neural network. In addition, this result is grouped into one of the four pose classes from table 3.1, where lying down should raise a fall alert in the system.

Chapter 4

Results and discussion

The shape contexts presented in this thesis are very good at classifying poses, even though the dataset contains many difficult angles. For a wide range of parameters, the algorithm detects a fall properly in more than 30 out of 34 unseen images of a lying person. A screenshot from the implementation in C# is available in Appendix A.

4.1 Bounding rectangle

A comparison of the aspect ratio of the bounding box between poses shows that there is no clear threshold at which a fall can be distinguished from a non-fall. This is demonstrated in figure 4.1 where there is a large overlap between poses 5 (crouching) and 6–8 (lying down). On the other hand, these results give rise to two other limits: there are no falls with very small (< 0.5) ratios, and large (> 3) ratios always correspond to a fall. Hence, if the computational power is limited and shared with other tasks, large ratios can always be classified as "fallen" and low with "non-fallen" without further calculations of more advanced features, such as shape contexts.

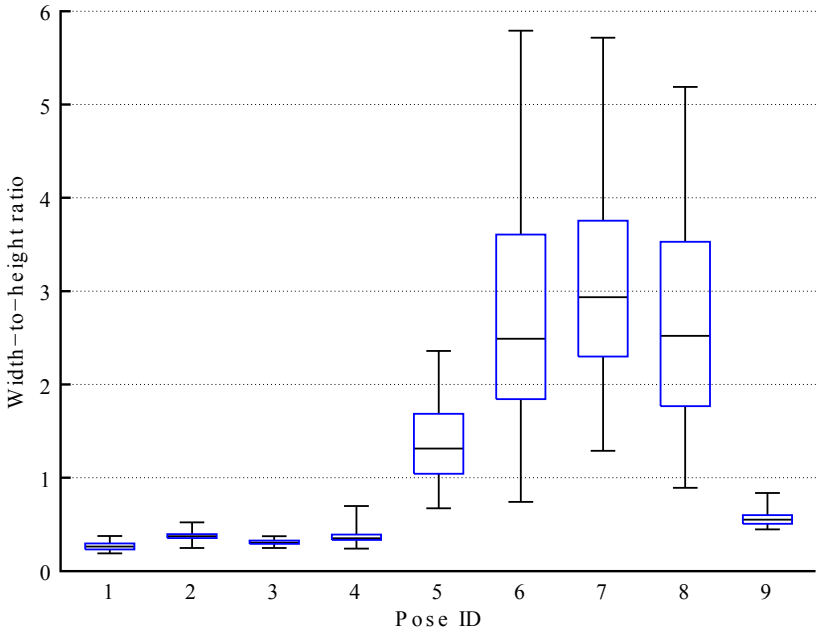


Figure 4.1: *Box charts showing median, quartiles and spread of the width to height ratios for the nine poses in the dataset. Poses 6–8 correspond to falls and have the highest ratios.*

The large variation in aspect ratio may appear strange. It might seem obvious that a standing person has a higher aspect ratio than one lying down. This is also true, but only if seen from a particular angle. A person lying down in a direction perpendicular to the camera direction will have a much larger aspect ratio than a person lying in the direction

(a) *Classified as lying on side.* (b) *Classified as crouching.*

Figure 4.2: *Two images from validation set, showing a person lying on the side from two different angles.*

of the camera. The silhouettes in figure 4.2 show two images of the same pose but from different angles. In addition, persons with their arms stretched out can be wider than their total height [45]. Hence, a robust fall detection mechanism that only uses width to height ratios for classification needs to use several images from different angles. On the contrary, a non-stationary camera mounted on a moving household robot can move around a person to obtain better viewing angles if a suspected fall is detected.

4.2 Shape contexts

The confusion matrices from one run of the algorithm is given for both training and validation set in table 4.1, corresponding to a fall sensitivity of 94.1 % and specificity of 97.6 %. Parameters were set to 128 edge points per shape; 12 angular bins and 3 radial bins with radial maximum 40 % of the silhouette size; and 100 cluster centers with a linear regressor as classifier. These results show that the algorithm is very good at separating a standing silhouette from one lying down. In addition, it is able to recognize the difference between lying and crouching in most cases, but still fails for a few. A comparison of the confusion matrices for training and validation set show that no particular overfitting has occurred in this case.

Table 4.1: Confusion matrices for a particular run of the algorithm.

True pose	Classified pose			
	Standing	Crouching	Lying	Sitting
Training set				
Standing	92	-	-	-
Crouching	-	24	1	-
Lying	-	1	61	-
Sitting	-	-	-	18
Validation set				
Standing	56	-	-	-
Crouching	-	13	1	-
Lying	-	2	32	-
Sitting	-	-	-	12

Just as for the case of using a bounding box, the reason for misclassifications are the difficulties induced by the perspectives in the images. Figure 4.2 shows this effect for two images of the same person lying down on the back, where one is correctly classified as lying but the other one as crouching. For a moving robot, this problem can be circumvented by moving to robot to a position where it obtains a better viewing angle if it suspects that a fall has occurred. In addition, a crouching elderly person might also need assistance at getting back up again after crouching, and then a robot might want to give a helping hand in any way.

4.3 Parameter and method selection

The pose recognition problem is highly non-linear and there are many parameters and methods to choose from. Table 4.2 lists the results for a selected set of typical values. The table is divided into four sections. The number of edge points are varied in the first section (1–5) and shows that there is no additional performance gain by using more than 64 edge points. The second section (6–9) shows similar results, but for very small shape contexts with just one radial bin. Hence, 64 or possibly 128, edge points can be used in the algorithm. The third section (10–12) shows that as the shape context size increases with additional number of radial bins, there is not much performance gain.

The last section (13–15) shows output using a neural network (NN) instead of a linear regression as classifier. The hidden layers are shown in parenthesis. If there are no hidden layers, the NN performance is really poor. With one or two hidden layers, the performance is similar to that of the linear regressor. Therefore, the linear regressor should be used in the final classifier, because the regression matrix can be calculated almost instantly using optimized linear algebra algorithms as opposed to the iterative backtracking algorithm for the neural network.

With a rather limited validation dataset, the specificities and sensitivities in table 4.2 vary between runs. For example, as there are 34 true falls in the dataset, one misclassification corresponds to about 3 % change in the sensitivity. More exact results regarding method selection could be achieved by using a much larger dataset. This also accounts for the large variation in these measures.

The small histograms in samples (6–9) capture only the very local distribution of points, as they cover only 15 % of the shape size. Yet, the method is capable of a rather accurate classification. This indicates that the local distribution of edge points is important in the classification process. The difference between the small shape contexts and the larger ones is that the outer bins include remote edge points, and thus give a representation of where on the shape the current point is located.

As all shape contexts are clustered together, each histogram is associated with its nearest cluster, and the shape is represented by the number of points connected to each of these clusters. To make the algorithm robust to noise, it is desirable to have several histograms (i.e. edge points) associated with a few clusters, than one edge point with each cluster. Hence, it is desirable to keep the number of clusters relatively low (0.5–1.0 times the number of edge points), so that only a few characteristic shape properties are represented in the clusters. Additional experiments reveal slight overfitting if an excessive amount of cluster centers are used.

All steps used in the methods of this report have performed “hard” classifications. For instance, only the nearest cluster (the “winner”) is selected during the quantization. An alternative approach is to perform the quantization “softly” by voting into the set of nearest clusters using suitable weights. Similarly, the index of the largest value in the output vector is taken as the classified pose. Here, another option could have been to let the output vector represent the probabilities of the shape being each pose. In addition, the current classifier is trained to differentiate between four different standing poses and three lying. Since the training output vector consists of only zeros except for the correct pose, the algorithm is

Table 4.2: Performance of the algorithm on validation set, as minimum and maximum values of 5 subsequent runs.

Test index	# of edge points	# of radial bins	# of angular bins	Shape context size (%)	# of cluster centers	Classifier	Fall sensitivity (%)	Fall specificity (%)
1	16	3	12	50	20	linear	76.5–83.5	95.1–98.8
2	32	3	12	50	40	linear	82.4–91.2	96.3–98.8
3	64	3	12	50	60	linear	85.3–94.1	100.0–100.0
4	128	3	12	50	80	linear	91.2–100.0	97.6–100.0
5	256	3	12	50	100	linear	88.2–94.1	97.6–100.0
6	64	1	4	15	10	linear	82.4–97.1	93.9–98.8
7	64	1	8	15	20	linear	88.2–91.2	97.6–100.0
8	64	1	12	15	40	linear	91.2–97.1	98.8–100.0
9	64	1	16	15	40	linear	82.4–91.2	97.6–100.0
10	64	2	12	30	40	linear	82.4–88.2	96.3–100.0
11	64	3	12	40	60	linear	91.2–94.1	96.3–100.0
12	64	4	12	50	60	linear	79.4–88.2	98.8–100.0
13	64	3	12	50	60	NN (–)	20.6–76.5	85.4–98.8
14	64	3	12	50	60	NN (30)	85.3–97.1	96.3–98.8
15	64	3	12	50	60	NN (40,20)	85.3–94.1	93.9–100.0

trained to classify a standing person as “standing”, but not the almost similar “walking slowly”. A soft voting mechanism could use the sum or product of the classifier output vector entries for poses in the same class for increasing predictability.

Regarding the classifier, the linear least squares one can suffer from overfitting. This can be reduced by adding a damping term and solve the damped least squares problem instead [1]. Overfitting also occured when training a feed-forward neural network, especially when it consists of many hidden layers. When the network has been trained by holdout validation, it has been optimized for both the training and validation set. With the relatively small dataset used here, the solution was to use a mirrored copy of the validation set for holdout validation, and the real validation set for testing. Since the mirrored copy of an image dataset is not independent of its original, this might induce some higher validation results than a completely unseen test set.

Chapter 5

Conclusions

In this thesis, a method for detecting poses has been devised, with the aim of being able to detect a fallen person. A dataset of sample images has been created, and segmented manually. The segmented human silhouettes have then been used for training an algorithm at recognizing poses, and detects a person lying down with 97 % accuracy.

By detecting pose accurately, the algorithm should be able to realize whether there is a person lying down on the floor or not. Because the method does not rely on constant monitoring and capture of the actual fall event, it can be fitted on a moving household that can aid the elderly in their homes. In case of an accidental fall, that statistically occurs among one third of the elderly population during one year, the robot can either give a helping hand or raise the alarm and call for further assistance.

Even though this is a successful pose detection method, it still relies on a good segmentation method. Simple background subtraction methods are not good options for two reasons. Firstly, the camera is non-stationary so the background will constantly change. Secondly, if the background is updated while the camera moves and a fallen person is lying on the floor, that lying person could become a part of the background model. A better method should be to build a simple 3-D model of the environment and generate a background from it. Such a method could also make use of infrared and depth field cameras to provide with further data for the model. Additionally, the background model has to adapt to large changes, such as a chair being moved.

Given the large amounts of laboratory research on fall detection (see section 1.4), further research should focus more on the gathering of real fall data from elderly, rather than optimizing currently developed methods to simulated fall data. This data could be gathered using video surveillance or by interviews with elderly people who have experienced a fall. It is also important that researchers on fall detection technology work together with researchers in medicine, sociology and psychology for this, as technological solutions both have to fulfill ethical criteria and be accepted by elderly people. As an example, there are cases where a fallen person chose not to activate a call alarm and get help, because this person wanted to remain independent of other people [14].

References

[1] A. Agarwal and B. Triggs. "3D human pose from silhouettes by relevance vector regression". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2004). ISSN: 1063-6919.

[2] A. Agarwal and B. Triggs. "A local basis representation for estimating human pose from cluttered images". In: *Computer Vision–ACCV 2006* (2006), pp. 50–59.

[3] D. Anderson et al. "Recognizing falls from silhouettes". In: *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE. 2006, pp. 6388–6391. ISBN: 1424400325.

[4] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[5] E. Auvinet et al. "Fall detection using multiple cameras". In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE. 2008, pp. 2554–2557.

[6] S. Belongie, J. Malik, and J. Puzicha. "Shape matching and object recognition using shape contexts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002), pp. 509–522. ISSN: 0162-8828.

[7] J. Boyle and M. Karunanithi. "Simulated fall detection via accelerometers". In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE. 2008, pp. 1274–1277.

[8] J. Canas et al. "Visual fall detection for intelligent spaces". In: *IEEE International Symposium on Intelligent Signal Processing*. IEEE. 2009, pp. 157–162.

[9] H. Costin et al. "TELEMON—A Complex System for Real Time Telemonitoring of Chronic Patients and Elderly People". In: *4th European Conference of the International Federation for Medical and Biological Engineering*. Springer. 2009, pp. 1002–1005.

[10] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), pp. 886–893. ISSN: 1063-6919.

[11] N. Doulamis. "Iterative motion estimation constrained by time and shape for detecting persons' falls". In: *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. ACM. 2010, pp. 1–8.

[12] A. Elgammal, D. Harwood, and L. Davis. "Non-parametric model for background subtraction". In: *Computer Vision—ECCV 2000* (2000), pp. 751–767.

[13] P. Felzenszwalb and D. Huttenlocher. "Pictorial structures for object recognition". In: *International Journal of Computer Vision* 61.1 (2005), pp. 55–79. ISSN: 0920-5691.

[14] J. Fleming and C. Brayne. "Inability to get up after falling, subsequent time on floor, and summoning help: prospective cohort study in people over 90". In: *BMJ: British Medical Journal* 337 (2008).

[15] H. Foroughi, A. Rezvanian, and A. Paziraee. "Robust Fall Detection using Human Shape and Multi-class Support Vector Machine". In: *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE. 2008, pp. 413–420.

[16] H. Foroughi et al. “An eigenspace-based approach for human fall detection using Integrated Time Motion Image and Neural Network”. In: *9th International Conference on Signal Processing*. IEEE. 2008, pp. 1499–1503.

[17] G. Fuller. “Falls in the elderly.” In: *American Family Physician* 61.7 (2000), p. 2159. ISSN: 0002-838X.

[18] R. Gonzalez and R. Woods. *Digital Image Processing*. Pearson Education International, 2008. ISBN: 0-13-505267-X.

[19] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation*. Vol. 1. Westview press, 1991.

[20] P. Hodgetts. “Falls in the Elderly: Assessment and prevention in the community”. In: *Canadian Family Physician* 38 (1992), p. 2413.

[21] A. Householder. “Unitary triangularization of a nonsymmetric matrix”. In: *Journal of the ACM (JACM)* 5.4 (1958), pp. 339–342. ISSN: 0004-5411.

[22] B. Huang, G. Tian, and H. Wu. “A method for fast fall detection based on intelligent space”. In: *IEEE International Conference on Automation and Logistics*. IEEE. 2008, pp. 2260–2265.

[23] B. Jansen and R. Deklerck. “Context aware inactivity recognition for visual fall detection”. In: *Pervasive Health Conference and Workshops, 2006*. IEEE. 2006, pp. 1–4. ISBN: 1424410851.

[24] X. Kolovou and I. Maglogiannis. “Video-surveillance and context aware system for activity recognition”. In: *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. ACM. 2010, pp. 1–2.

[25] D. Leister et al. “Large-scale evaluation of plant growth in Arabidopsis thaliana by non-invasive image analysis”. In: *Plant Physiology and Biochemistry* 37.9 (1999), pp. 671–678. ISSN: 0981-9428.

[26] D. Lowe. “Object recognition from local scale-invariant features”. In: *iccv*. Published by the IEEE Computer Society. 1999, p. 1150.

[27] *Math.NET Numerics*. Math.NET Project Team. URL: `http://numerics.mathdotnet.com/`.

[28] S. Miaou, P. Sung, and C. Huang. “A customized human fall detection system using omni-camera images and personal information”. In: *1st Distributed Diagnosis and Home Healthcare Conference, 2006*. IEEE. 2006, pp. 39–42. ISBN: 1424400589.

[29] *MSDN Library*. Microsoft Corporation. URL: `http://msdn.microsoft.com/en-us/library`.

[30] A. Nasution and S. Emmanuel. “Intelligent video surveillance for monitoring elderly in home environments”. In: *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*. IEEE. 2007, pp. 203–206.

[31] J. Perry et al. “Survey and evaluation of real-time fall detection approaches”. In: *6th International Symposium on High-Capacity Optical Networks and Enabling Technologies (HONET)*. IEEE. 2009, pp. 158–164.

[32] Q. Pham et al. “Video monitoring of vulnerable people in home environment”. In: *Smart Homes and Health Telematics* (2008), pp. 90–98.

[33] H. Qian et al. “Home environment fall detection system based on a cascaded multi-SVM classifier”. In: *10th International Conference on Control, Automation, Robotics and Vision*. IEEE. 2008, pp. 1567–1572.

[34] N. Rao, H. Di, and G. Xu. “Panoramic background model under free moving camera”. In: *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 1. IEEE. 2007, pp. 639–643.

[35] R. Rosales and S. Sclaroff. “Inferring body pose without tracking body parts”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE. 2000, pp. 721–727.

[36] B. Schulze, M. Floeck, and L. Litz. “Concept and Design of a Video Monitoring System for Activity Recognition and Fall Detection”. In: *Ambient Assistive Health and Wellness Management in the Heart of the City* (2009), pp. 182–189.

[37] G. Shakhnarovich, P. Viola, and T. Darrell. “Fast pose estimation with parameter-sensitive hashing”. In: *IEEE International Conference on Computer Vision* (2003).

[38] M. Sonka, V. Hlavác, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Thomson Learning, 2008. ISBN: 0-495-24438-4.

[39] C. Thurau and V. Hlavác. "Pose primitive based human action recognition in videos or still images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008.

[40] M. Tinetti, W. Liu, and E. Claus. "Predictors and prognosis of inability to get up after falls among elderly persons". In: *JAMA: the journal of the American Medical Association* 269.1 (1993), p. 65. ISSN: 0098-7484.

[41] B. Töreyin, Y. Dedeoğlu, and A. Çetin. "HMM based falling person detection using both audio and video". In: *Computer Vision in Human-Computer Interaction* (2005), pp. 211–220.

[42] H. Tzeng, M. Chen, and J. Chen. "Design of fall detection system with floor pressure and infrared image". In: *System Science and Engineering (ICSSE), 2010 International Conference on*. IEEE. 2010, pp. 131–135.

[43] M. Wahde. *Biologically inspired optimization methods*. WIT Press, Billerica, MA, 2008.

[44] C. Wang and A. Hunter. "Robust Pose Recognition of the Obscured Human Body". In: *International journal of computer vision* (2010), pp. 1–18. ISSN: 0920-5691.

[45] D. Winter. *Biomechanics and motor control of human movement*. John Wiley & Sons Inc, 1990.

[46] *World Population Ageing*. New York, USA: United Nations, 2009. URL: `http://www.un.org/esa/population/`.

[47] M. Yu, S. Naqvi, and J. Chambers. "Fall detection in the elderly by head tracking". In: *Statistical Signal Processing, 2009. SSP'09. IEEE/SP 15th Workshop on*. IEEE. 2009, pp. 357–360.

[48] Z. Zhang et al. "Experiments with computer vision methods for fall detection". In: *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*. ACM. 2010, pp. 1–4.

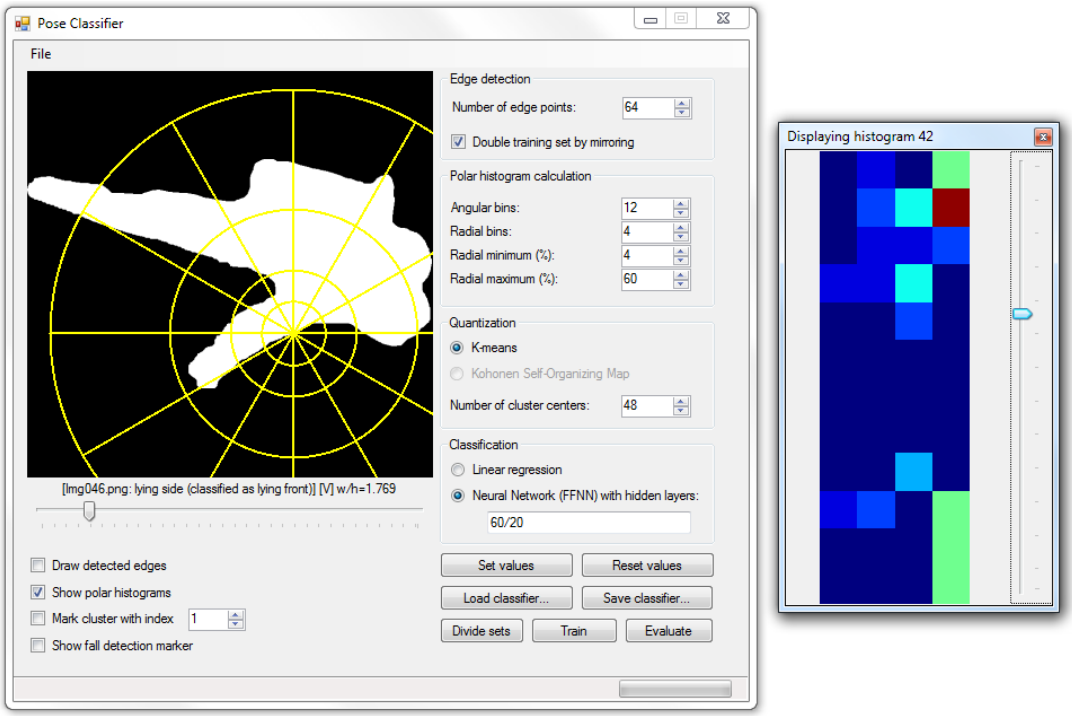
[49] Q. Zhou et al. "HMMs-based human action recognition for an intelligent household surveillance robot". In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2009, pp. 2295–2300.

[50] A. Zweng, S. Zambanini, and M. Kampel. "Introducing a statistical behavior model into camera-based fall detection". In: *Advances in Visual Computing* (2010), pp. 163–172.

Appendix A

Description of C# implementation

The algorithm is implemented as a C#.NET application with a graphical user interface (GUI). The application reads a file in XML format that specifies the dataset, loads the stored images and traces the edges in them. The human-readable XML format allows a user to easily extend the database of images, or switch between several datasets. In addition, the program design allows for most parts of the training process run in parallel on several CPU cores, by using the built-in functions in the .NET API for parallel code execution [29].



A screenshot of the produced application showing how the polar histograms can be investigated. Note the information given below the image: file name, true pose, classified pose, set belonging and bounding box ratio.

The GUI makes it easy for a user to change the parameters of the model and visualizes the shape context bins while their parameters are changed. In addition, it can calculate the histograms and visualize them, while simultaneously superposing the log-polar bins on the current edge point, as seen in the figure above. The edge points belonging to a particular cluster can be highlighted, and the corresponding histogram values can be visualized to show their similarity.

Additionally, the application displays valuable information about the current shape, such as the file name, the set belonging (T for training and V for validation) and width to height ratio. If a classifier is loaded or trained, the classification result is also shown. The performance output is displayed as confusion matrices in an alert textbox.