DUBLIN CITY UNIVERSITY

DOCTORAL THESIS

Handshape Recognition using Principal Component Analysis and Convolutional Neural Networks applied to Sign Language

Author:
Marlon OLIVEIRA BSc. MSc.

Supervisor:
Dr. Alistair SUTHERLAND

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

School of Computing

January, 2018

Declaration of Authorship

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: 

ID No.: 13211361

Date: January, 2018

“A man must travel. By his own, not through stories, pictures, books or TV. He’s got to travel by himself, with his eyes and feet, to understand what is his. To one day plant his own trees and value them. Knowing the cold to enjoy the heat. And the opposite. Feel the distance and absence of shelter to be well under his own roof. A man must travel to places that he does not know to break this arrogance that makes us see the world as we imagine it, and not simply as it is or can be. That makes us teachers and doctors of what we have not seen, when we should be students, and simply go see ”

Amyr Klink

Acknowledgements

Throughout and before this process called PhD God put some people in my way, thank God I concluded my studies and have a chance to say thank you out loud:

My parents Leodovina and Donirio (in memoriam) for all the support in my educational process. None of this would be possible without their guidance.

My supervisor, Dr. Alistair Sutherland who gave me this opportunity providing me with the support to guide my research, for agreeing with my crazy ideas for papers and research visits to different parts of the planet.

My former supervisors Christine Scarpato (undergraduate) and Pedro Serranho (master) for helping me in previous projects and for showing me the research path.

All my friends, especially Cristiani Eccher, Dr. Plamen Petkov, and Dongyun Nie (Robin) who helped me recording the dataset; Adline Peres (who paid my Spotify during the PhD), Ademar Crotti Junior (always up to help), Samuel Brognoli (always up to a new trip), Cristiani Eccher (at this time for the coffees and chats at The Helix), Luana Muller (walking the master and PhD ways together), for sure every single friend along the way has a special space in my heart.

My colleagues and former colleagues, Dr. Houssem Chatbri, Dr. Dexmont Pena, Dr. Alireza Deghani, Dr. Mohamed Farouk, Dr. Dahai Yu and Ms. Fiona Dermody, your example and suggestions were very important for improving my work. As well as all guys in DCU, for sharing moments together: Dr. Jim, Manoj, Conor, Fouad, Michael, Viviana, Suzanne, Aleksas, Dr. Zohreh, Dr. Marija, Liting Zhou (Tingting), Qishan Yang, Rashmi, Dr. Giovanni, André, Barbara, Jonathan, Flávio and Thayse. Staff and lectures who always supported me with paper works, tutoring and findings; Patricia Lacey, Dr. Rory O'Connor, Dr. Liam Tuohey and Dr. Andrew Way.

Last but certainly not least, THANK my wife Paula, for all the support throughout these years and years of studies and difficulties, for always having a nice word to keep me up, a warm hug to relieve the stress and amazing food always waiting for me at home.

This research was funded by CAPES/Science without Borders - Brazilian Program - Process: 9064-13-3

Contents

List of Figures

List of Tables

List of Publications

This section introduces the publications where this work has been published:

Marlon Oliveira, Houssem Chatbri, Suzanne Little, Noel E. O'Connor, and Alistair Sutherland. A comparison between end-to-end approaches and feature extraction based approaches for sign language recognition. In *Image and Vision Computing New Zealand (IVCNZ), 2017 International Conference on (in press)*.

Marlon Oliveira, Houssem Chatbri, Ylva Ferstl, Suzanne Little, Noel E. O'Connor, and Alistair Sutherland. Irish sign language recognition using principal component analysis and convolutional neural networks. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA) (in press)*, 2017a.

Marlon Oliveira, Houssem Chatbri, Ylva Ferstl, Mohamed Farouk, Suzanne Little, Noel E. O'Connor, and Alistair Sutherland. A dataset for irish sign language recognition. In *Irish Machine Vision & Image Processing Conference*, 2017b.

Marlon Oliveira, Alistair Sutherland, and Mohamed Farouk. Manifold interpolation for an efficient hand shape recognition in the irish sign language. In *International Symposium on Visual Computing*, pages 320–329. Springer, 2016a.

Marlon Oliveira, Alistair Sutherland, and Mohamed Farouk. Two-stage pca with interpolated data for hand shape recognition in sign language. In *Applied Imagery Pattern Recognition Workshop*. IEEE, 2016b.

Marlon Oliveira and Alistair Sutherland. Interpolating eigenvectors from second-stage pca to find the pose angle in handshape recognition. In *Irish Machine Vision & Image Processing Conference*, pages 114–117, 2015.

Abstract

Handshape Recognition using Principal Component Analysis and Convolutional Neural Networks applied to Sign Language

by Marlon OLIVEIRA

Under the supervision of Dr. Alistair SUTHERLAND

Submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy at the School of Computing, Dublin City University

Handshape recognition is an important problem in computer vision with significant societal impact. However, it is not an easy task, since hands are naturally deformable objects. Handshape recognition contains open problems, such as low accuracy or low speed, and despite a large number of proposed approaches, no solution has been found to solve these open problems. In this thesis, a new image dataset for Irish Sign Language (ISL) recognition is introduced. A deeper study using only 2D images is presented on Principal Component Analysis (PCA) in two stages. A comparison between approaches that do not need features (known as end-to-end) and feature-based approaches is carried out.

The dataset was collected by filming six human subjects performing ISL handshapes and movements. Frames from the videos were extracted. Afterwards the redundant images were filtered with an iterative image selection process that selects the images which keep the dataset diverse.

The accuracy of PCA can be improved using blurred images and interpolation. Interpolation is only feasible with a small number of points. For this reason two-stage PCA is proposed. In other words, PCA is applied to another PCA space. This makes the interpolation possible and improves the accuracy in recognising a shape at a translation and rotation unknown in the training stage.

Finally classification is done with two different approaches: (1) End-to-end approaches and (2) feature-based approaches. For (1) Convolutional Neural Networks (CNNs) and other classifiers are tested directly over raw pixels, whereas for (2) PCA is mostly used to extract features and again different algorithms are tested for classification. Finally, results are presented showing accuracy and speed for (1) and (2) and how blurring affects the accuracy.

Chapter 1

Introduction

This chapter introduces the central concept of the research and provides an explanation of the motivation and approach for the research. In Section 1.1 this field of research is reviewed briefly. Then, the motivation behind this thesis is presented in Section 1.2. The research questions are raised in Section 1.3. The significance and contributions presented in this work are summarised in Section 1.4. Finally, Section 1.5 outlines the thesis contents.

1.1 Research Review

Human Computer Interaction depends strongly on the new developments in Computer Vision (CV), and handshape recognition is an active research area in this field.

Handshape or gesture recognition is a challenging task, because hands are deformable objects. This means that depending on many facts, such as rotation, scale, translation and illumination, every single detail can make a hand look like a different object.

According to Mitra and Acharya (2007) some applications of handshape recognition are listed as follows:

- developing aids for the hearing impaired

- enabling very young children to interact with computers
- designing techniques for forensic identification
- recognizing sign language
- medically monitoring patients' emotional states or stress levels
- lie detection
- navigating and/or manipulating in virtual environments
- communicating in video conferencing
- distance learning/tele-teaching assistance
- monitoring automobile drivers' alertness/drowsiness levels, etc

This thesis focuses on handshape recognition applied to sign language. CV plays an important role in this matter by helping people who use sign language with systems for everyday life problems, e.g. automatic transcription, human-machine interaction and machine translation.

Irish Sign Language (ISL) is an indigenous language that is used by around 5,000 Deaf people in the Republic of Ireland and 1,500 in Northern Ireland. In addition, it is known by 50,000 non-Deaf people Leeson and Saeed (2012). ISL is not based on English or Irish, it is a language in its own right.

ISL contains more than 5,000 signs. Each sign consists of a handshape and a motion in 3D space. There are around 23 basic, most common handshapes in ISL and each of them is labelled with a different letter of the alphabet. These handshapes can be seen in a wide range of possible angles in 3D space. The remaining three letters of the alphabet, 'J', 'X' and 'Z' are used to label gestures involving motion and actually use one of the 23 handshapes.

There are basically two different ways to recognise handshapes, glove-based approaches and vision-based approaches. Glove-based uses a physical glove (hardware) with sensors attached to the human arm/hand. This kind of method is very precise because they can provide

precise information about the hand configuration. However, it implies wearing a mostly uncomfortable device which does not allow a natural interaction. Vision-based approaches provide a natural interaction since they use only cameras to obtain hand configuration information. Nonetheless, this kind of approach is more complex and might not be very precise.

Vision-based approaches are highly studied and there are several manners to obtain hand configuration from cameras/images. Sensors, multiple cameras and depth cameras are examples of these manners. Nevertheless, this thesis focuses only on a 2D vision-based approach with a single camera.

1.2 Motivation

A proof of how tough handshape recognition is that it has been researched for years and still there is not an usable application for this matter. Many different theories have been proposed and studied. Each of these theories uses a different approach, e.g. gloves, sensors, lasers, cameras, depth cameras, 3D cameras, or just the most difficult one, a single regular camera.

The idea of using just a regular camera is important because these cameras are inexpensive. Everyone nowadays has easy access to a camera. Thus, if people have these cameras easily accessible, why not focus on research for this kind of device? In addition, there are a huge number of videos uploaded every day to the internet, e.g. YouTube, Vimeo. YouKu. These videos are 2D and contain no depth information.

The key question is how to make computers recognise hands from 2D images? There are several technologies and approaches to do it. Therefore, all of them use basic steps, i.e. train a classifier and then classify. The training and testing stages can be either done with features extracted from images (known as a feature-based approach) or straight from pixels (known as an end-to-end approach).

CV provides the technology to assist people who use sign language with tools such as automatic transcription, human-machine interaction, machine translation, etc. In order to design such tools, large amounts of data are necessary for training and testing the system. In this thesis, a new image dataset for ISL recognition is introduced.

Working with a dataset of images of real hands is challenging because the exact pose angle is unknown. It makes the task of classifying the new object difficult because when working with more than one shape, each shape can be slightly rotated with respect to the others.

Principal Component Analysis (PCA) is an efficient technique for dimensionality reduction and feature extraction Han and Liu (2014). It uses the covariance matrix of the data to create a space called an eigenspace. Each dimension in this space is represented by one eigenvector. The number of eigenvectors used to represent the full data is always considerably lower than the dimensionality of raw data.

Convolutional neural networks (CNNs) specialise in recognising patterns. They are widely known for robustness to distortion and having minimal or no preprocessing. They have been used for detection and recognition of different objects, including hands Nagi et al. (2011) and LeCun et al. (2015).

Some approaches to gesture recognition extract features from images and use these features to train the classifier. Several techniques are used to extract features e.g. edge detectors, contours, corners, Haar-like features, interest points, Fourier transform, PCA and more. These approaches are called feature-based approaches. In contrast to feature-based, the approaches that do not use features are called end-to-end approaches.

The final step is always the classification. In order to classify an image either from raw pixels or from features a classification algorithm has to be applied. Machine Learning is widely used in this process. Techniques such as CNN, linear discriminant analysis (LDA), support vector machines (SVM), a multilayer perceptron (MLP), decision trees, and a k-Nearest Neighbour (k-NN) are examples of classifiers.

Among the large number of techniques to extract features or/and classify objects, this thesis focuses mainly on Principal Component Analysis and Convolutional Neural Networks. In addition, interpolation over PCA manifolds and eigenspaces are proposed.

One big challenge in this work is a considerable quantity of parameters. For PCA: the level of blurring, the size of each image, the number of eigenvectors in each PCA level, the number of dimensions used in the interpolation, the quantity of points to be interpolated and used in the reconstruction. For CNN: the size of the filters, the number of feature maps, the number of hidden layers, the activation function, etc. Every single parameter affects the accuracy of the recognition.

1.3 Research Questions

The research proposed in this thesis raises the following main questions:

- **RQ1:** Is it possible to use two-stage PCA and interpolation to generate artificial data which can augment a sparse dataset?
- **RQ2:** Does the use of interpolated data increase the recognition accuracy on a sparse dataset?
- **RQ3:** How do parameters such as blurring level, number of eigenvectors or sampling interval affect the accuracy?
- **RQ4:** How do feature-based and end-to-end algorithms compare in recognition accuracy on a dense dataset?
  - **RQ4(a):** Should features be extracted?
  - **RQ4(b):** Which classifier is the most accurate for the ISL dataset?
  - **RQ4(c):** Which classifier is the fastest?

1.4 Significance and Contributions

In this thesis three different contributions are presented to the field of handshape recognition applied to Irish Sigh Language (ISL). The first contribution is a new dataset for ISL with a redundancy filter. The second is the use of interpolation over PCA manifolds and PCA eigenspaces. The third contribution is a comparative study between feature-based approaches and end-to-end approaches.

- **New Irish Sigh Language alphabet dataset** The new ISL dataset contains 468 videos, filmed from 6 different human subjects, resulting in more than 58,000 frames representing the 26 alphabet letters. Frames in this dataset were filtered with a proposed redundancy filter resulting in 50,000 images for the 23 static gestures in ISL. This dataset builds on a previous dataset with only 20 handshapes and a rather smaller number of images.

- **PCA with interpolation** PCA is used to represent data in a space with a reduced number of dimensions. This representation of the data allows us to select subsets and apply PCA for the second time. Having the dimensionality of the data reduced twice it is possible to interpolate in order to create synthetic data from the original data. PCA data can be interpolated by fitting a cubic spline in N-dimensions. This raises the question: Can the interpolated data be used to improve the recognition of an unknown object in a *sparse* dataset?

- **Comparison between CNN (and other algorithms) and PCA approaches** A deep study is proposed with end-to-end approaches (mainly CNN) and feature extraction based (PCA and Kernel PCA), and other different classifiers. The most important questions are: Should features be extracted? Which classifier is the best for the data available? Which are the faster? Experiments done in this thesis try to answer these questions.

1.5 Thesis Outline

Following on from the above, the thesis is organised as follows:

- **Chapter 2** discusses the concepts in different areas related to handshape recognition as well as the state-of-the-art approaches in this field.
- **Chapter 3** presents a new dataset for Irish Sign Language and a filter to remove frames with high similarity. Finally, it is shown how datasets are split into a *sparse* dataset and a *dense* dataset.
- **Chapter 4** discusses two-stage Principal Component Analysis and interpolation by splines, including the influence of blurring, applied to the *sparse* dataset.
- **Chapter 5** explores mainly Convolutional Neural Networks and Principal Component Analysis, applied to a *dense* dataset, including recognition accuracy, speed and influence of blurring. In addition, other classifiers are tested and results for accuracy and speed are shown as well.
- **Chapter 6** summarises the implications and contributions, and discusses the possibilities for future work

Chapter 2

Literature Review

As stated earlier in Chapter 1, handshape recognition is an important topic in computer vision, which has been discussed for a long time. The importance and popularity of this field of research, as well as the number of published papers are surveyed in this chapter.

2.1 Introduction

Humans can easily recognize handshapes using their eyes. Size, shape, colour, angle, distance, and other aspects of the human hand, generally, do not prevent a human matching a hand gesture. However, for computers it is completely different; every single detail turns a hand image into a different image. This is the main challenge in this field: how to make hand gestures be correctly classified by computers?

This chapter presents the state-of-the-art in handshape recognition. There are different forms of recognition in the field of pattern recognition. Basically they are divided into two sub-areas: vision and non-vision based approaches.

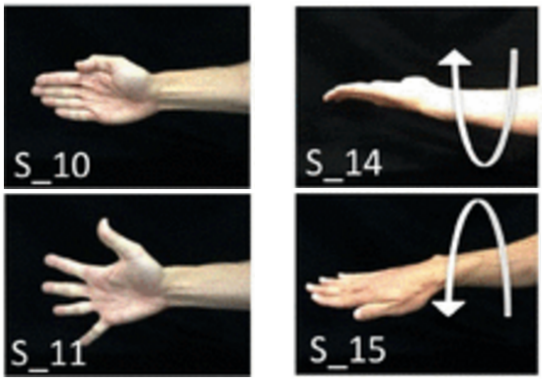
Non-vision-based approaches include all the techniques that use gloves and/or sensors to obtain information about the shape of the hand. In these approaches data gloves equipped with sensors and/or accelerometers are normally used to capture the rotation and movement of the

hand and fingers. These gloves use hardware that usually obtains information about the hand through electrical signals or electromagnetic interference Naoum et al. (2012). In Section 2.2 in this chapter some non-vision-based methods are described.

Vision-based approaches are those that use cameras to obtain information about the hand. The Kinect camera, web-cams and time-of-flight cameras are examples of cameras used in this kind of approach.

A significant quantity of vision-based hand gesture recognition techniques have been introduced in the literature, where images of different kinds and from different numbers of cameras are used to achieve gesture recognition.

Hand gestures can be classified into two main categories: static and dynamic. Dynamic gesture are those that include movement of the hand, e.g. turning palm up and down. Static gestures are those that do not need movement to be recognised, e.g. an open hand. Figure 2.1, from Wolf et al. (2013), shows examples of static gestures 2.1(a), and dynamic gestures 2.1(b).



(a) Static gestures (b) Dynamic gestures

FIGURE 2.1: Examples of static and dynamic hand gestures

Beyond the methods of how images are obtained, most of the researches includes a classification stage. k-Nearest Neighbour, Support Vector Machines and Deep Learning are some examples of classifiers used in hand gesture recognition. The following sections explain some of the theories behind gesture recognition and related works in the field.

2.2 Non-Vision-based Approaches

Non-Vision-based approaches are those that usually use data-gloves and/or sensors to map the human hand. In this section some examples of these hardwares and what sort of pattern recognition algorithms are used to recognise the gestures are presented.

2.2.1 Glove-based Handshape Recognition

Glove-based handshape recognition normally involves the user wearing some device and a certain quantity of cables to connect this device to a computer. This makes these methods a difficult and non-natural way to interact with the computer Mitra and Acharya (2007). Data glove devices use electricity or electromagnetic interference to obtain information about the hand, which is sufficient to provide a description of a hand gesture Naoum et al. (2012). Researchers refer to data gloves in different ways, e.g. CyberGlove and Accele Glove.

Figure 2.2 shows the location of the sensors in a data glove proposed by Bedregal et al. (2007). Basically a sequence of frames can represent any movement. Thus, a sequence of hand configurations represents a hand movement using a data glove. In Bedregal et al. (2007) a random generated hand configuration was used to simulate the data transfer. Each instant of the handshape is represented by a tuple of interval angles from each sensor. The recognition was applied to Brazilian Sign Language (LIBRAS), using Fuzzy logic. As shown in Table 2.2 the accuracy rate and the number of gestures/persons was not provided by Bedregal et al. (2007).

Another kind of glove to extract gesture features is called the CyberGlove. This method was applied to American Sign Language (ASL) in Wang et al. (2006) as in Figure 2.3. The Cyberglove is a glove equipped with strain gauges for detecting finger bending, abductions[1]/adductions and shape. These gloves have been used to obtain the joint values which are then digitised to 8 bits. It has 18 sensors based on a linear, resistive bend sensing technology Sahoo et al. (2014).

[1]The movement of a finger away from the midline, the opposite of adductions Press (2017).

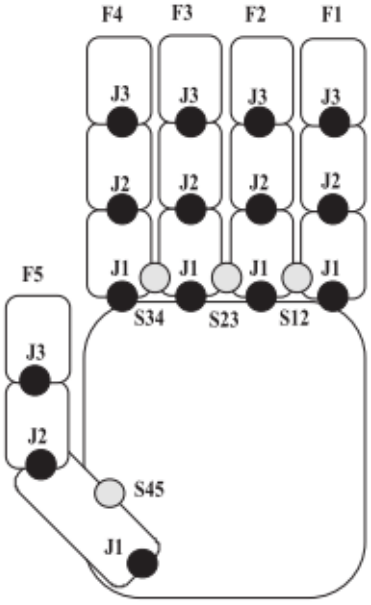


FIGURE 2.2: A data glove scheme with sensors

FIGURE 2.3: A Cyberglove with sensors by Wang et al. (2006)

Bui and Nguyen (2007) developed a similar device called the Accele Glove. These gloves use a microelectronic mechanical system (MEMS) to extract hand configuration. They have been applied to Vietnamese Sign Language for 23 gestures with Fuzzy logic. As shown in Table 2.2 the accuracy was divided by handshape, with an overall 98% accuracy. The relative angles between fingers and the palm is the data obtained from the sensing device. The glove contains six accelerometers and a BASIC Stamp microcontroller as in Figure 2.4 Sahoo et al. (2014) and Phil et al. (2015).

FIGURE 2.4: A sensor glove by Bui and Nguyen (2007)

The Polhemus FASTRAK provides 6 degrees-of-freedom tracking Sahoo et al. (2014) and it is used to collect a sequence of positional co-ordinates of both hands which are measured with the magneto metric sensor at 60 samples per second. It has been used to collect data in Maebatake et al. (2008) for American Sign Language recognition. Polhemus FASTRAK is highly reliable and has near non-latency, what makes it useful for virtual reality interfacing and simulators. It converts the obtained data into the most common computer graphics software format Sahoo et al. (2014).

Abdulla and Manaf (2016) proposed a system for a sign-to-speech/text for deaf people, applied to Arabic Sign Language. This system includes the design and implementation of a smart glove. One of the advantages of this glove is that it does not depend on light conditions, which means it works even in dark environments. These gloves are low cost, low power and have full mobility as well. Another advantage of these gloves is that they have flex sensors which have a wireless interface to a microcontroller.

Zhang et al. (2011) have created their own sensor and applied it to Chinese Sign Language recognition. This sensor uses an accelerometer (ACC) placed at the wrist and electromyography (EMG) placed at the forearm (close to the elbow), that provide two potential technologies for gesture sensing. ACC can measure both dynamic acceleration and static acceleration such as gravity. EMG measures the electrical potentials generated by muscle cells. Thus, the main goal was a framework for hand gesture recognition using decision trees and Hidden Markov Models (HMM) for fusion of the ACC and EMG sensors.

2.3 Vision-based Gesture Recognition

In this section vision-based approaches are presented. Vision-based approaches use cameras to obtain information about the hand. The Kinect, web-cams, time-of-flight cameras and stereo-cameras are examples of cameras used in these kind of approaches.

2.3.1 Example-based Approaches

Example-based methods depend on a large set of examples for which the parameter values are known Farouk (2015). They deduce the label values for the input example (or image) from the known values in similar examples. This approach does not need to model the global structure of the input data, which is just assumed to be sufficient to make such inference. Therefore, example-based approaches are efficient for parameter estimation problems when the system is not complex and/or the dimensionality of the input is not high. However, for complex and high-dimensional problems the number of required examples and the computational complexity become exhaustively high Shakhnarovich et al. (2003).

As with any other classification technique, example-based applications involve two phases: training and testing. In the training stage, the user feeds into the system one or more examples of a specific handshape. For instance in the work of Naoum et al. (2012) the system forms and stores the corresponding orientation histogram. In the testing stage, the system compares the orientation histogram of the current image with each of the training images and picks the category of the best matches. This method can be robust to small information changes such as difference in the size of the hand. However, it would be sensitive to changes in hand orientation. Other common metrics systems are Euclidean distance, interest point matching, moments, Fourier Descriptors, etc.

Some characteristics of example-based learners are as follows:

- They capitalise on the availability of a large set of examples for which the parameter values are known Shakhnarovich et al. (2003)

- They compare the observed image with a database of samples Moeslund et al. (2006)
- The set of typically interesting poses is far smaller than the set of anatomically possible ones, which is good for robustness den Bergh et al. (2009)
- They represent the mapping between image and pose space providing a powerful mechanism for directly estimating 3D pose Moeslund et al. (2006)
- They do not involve domain-specific knowledge Mitra and Acharya (2007)
- They can easily be used to locate any object composed of distinct identifiable parts that are arranged in a well-defined configuration Mohan et al. (2001)

Some classic examples of example-based approaches are k-NN and locally-weighted regression (LWR). These methods are widely used because of their simplicity and the excellent results provided Shakhnarovich et al. (2003). However, these techniques have a high computational complexity due to the similarity search. When working with large datasets the high-dimensional spaces can make it infeasible. Shakhnarovich et al. (2003) proposed a new example-based approach using Locality-Sensitive Hashing (LSH) to overcome this problem. In this case the training examples are indexed by a number of hash tables.

Figure 2.5 shows handshapes followed by the orientation histograms. This is one example of how example-based approaches can be used for gesture recognition Freeman et al. (1998).

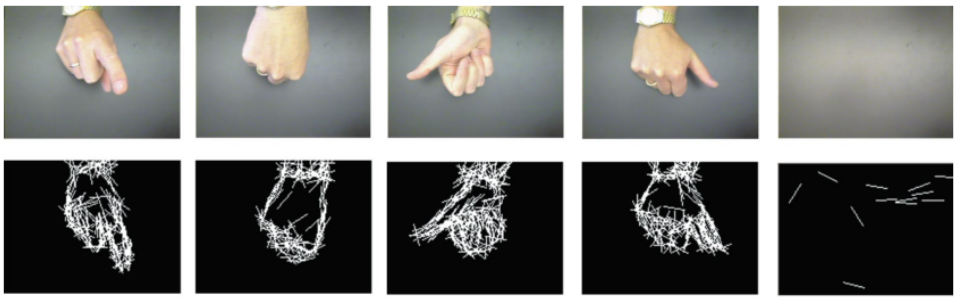


FIGURE 2.5: Histograms of oriented gradients in an example-based approach

2.3.2 Model-based Approaches

Model-based methods utilise an explicit 2D or 3D kinematic hand model to detect hands in images Li and Jain (2011). The 2D or 3D model used has to have enough degrees of freedom (DoF) in order to adapt to different dimensions of the hand(s) Zabulis et al. (2009). In addition, Zabulis et al. (2009) states that full 3D methods may allow view-independent detection in contrast to a 2D model of the hand that only works when the hand is in a certain pose e.g. with the palm facing the camera.

The feature-model correspondence depends on which features and which type of models are used. Point and line features are part of the kinematic models to recover the shape of the hand. Thus, hand gestures are estimated according to the correspondence between the model and the observed image features Zabulis et al. (2009).

A considerable number of model-based gesture recognition techniques use successive approximation methods to estimate parameters. Gesture recognition should be invariant to rotation, and for this reason intrinsic parameters such as joint angles are most utilised Zabulis et al. (2009). In addition, model-based methods enable tracking whilst maintaining estimates of model parameters and features not directly observable at a certain moment Rautaray and Agrawal (2012).

In gesture recognition, a set of gesture patterns composed of sequences is modeled by a generative or discriminative dynamic probabilistic model, i.e. HMM (Huang and Jeng (2001)) or a Hidden Conditional Random Field. Turk and Hua (2013). Ren and Zhang (2009) proposed a hand gesture recognition system using SVM combined with minimum enclosing ball (MEB).

Huang and Jeng (2001) state that model-based vision is an efficient approach for locating and recognizing objects in motion in the real scene with a significant number of spatial-temporal variations. Erol et al. (2007) concluded that 3D model-based approaches have a high computational cost and this cost depends on the DoF. In Figure 2.6, Erol et al. (2007) proposed a block diagram where a search is executed at each frame, in order to set the best parameter to minimise the error rate.

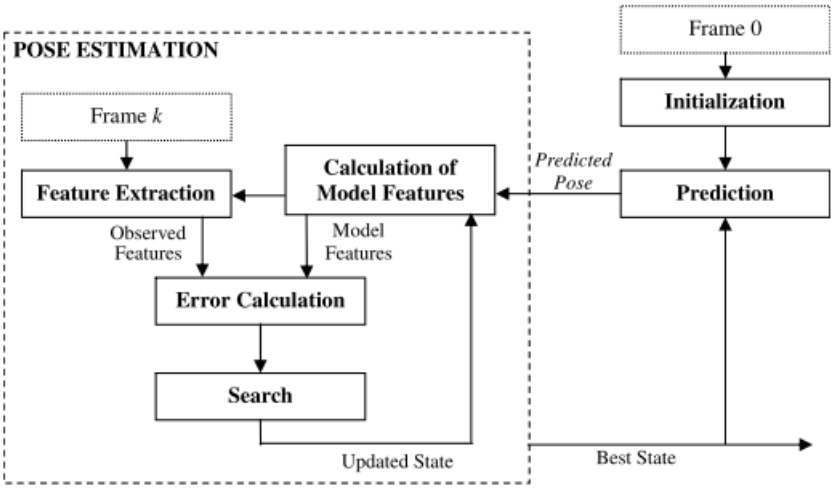


FIGURE 2.6: A typical model-based tracking system design

2.3.3 2D Methods

The methods that do not use depth images (3D) are considered 2D methods. They are more challenging than 3D, because they have less detail in the image. However, they are more appropriate for real time applications since they are less computationally expensive. Basically 2D methods directly use image pixels or features derived from image pixels Li and Jain (2011).

Chen et al. (2007) proposed one of the first hand gesture recognition systems. Haar-like features and the AdaBoost learning algorithm were applied over 2D images. This system was able to recognise only 4 hand gestures, such as two-fingers, palm, fist posture and little finger. Dataset images were acquired by a low-cost Logitech QuickCam web-camera that provides video capture with the resolution of 320×240, 15 frames-per-second. Images at different scales were used in the training phase. The average accuracy was over 90%.

Cui and Weng (2000) proposed an approach to recognizing hand signs using motion recognition merged with spatial recognition. Multidimensional discriminant analysis was used to automatically choose the most discriminant feature of each gesture for classification. Classification is made by a recursive partition tree approximator. This system was able to recognise 28 hand gestures with 93.2% accuracy. A regular web-cam was used for the input of ASL handshapes images.

Mistry et al. (2009) introduced WUW - wear Ur world, a wearable gestural interface. This system attempts to project information out into the real world. A tiny projector and a camera attached to a hat or in a pendant are used. WUW can project onto any surface, such as walls and physical objects, and allows the user to interact with the projected information by natural hand gestures. WUW is able to recognize gestures such as ”zoom in”, ”zoom out” or ”pan” in a map application or to flip though documents or images using the movements of the hand or index finger. In addition, users can draw on any surface using the movement of the index finger.

Naoum et al. (2012) proposed a new Arabic Sign Language recognition system using k-Nearest Neighbour. They proposed a system able to recognise 28 Arabic hand gestures. Naked hand and different colour gloves were tested. The accuracy was 50% for naked hand, 75% for red glove, 65% for black glove and 80% for white colour glove.

Kim et al. (2017) proposed a dataset for ASL fingerspelling, recorded with 2 cameras. They tested different approaches for handshape recognition using videos and deep neural network-based features. Accuracy was about 92% for letters and single-signer and 83% for letters and multi-signer.

2.3.4 Feature Extraction

Some of the 2D methods for handshape recognition rely on features. Feature extraction is related to dimensionality reduction, since the idea is to classify an image by a set of features instead of the entire image (raw pixels). There are several ways to extract features from an image, e.g. contours, corners, Haar-like features, interest points, Fourier transform and more.

Contours are one way to extract features from images. It consists of selecting salient edges in an image. It assumes that the contour is the most salient, or maximum gradient edge in the image Chen et al. (2010). Liu and Kehtarnavaz (2016) used finger spelling contours using a morphological approach for hand gesture recognition.

Mokhtarian and Suomela (1998) state that corner detection is important in various computer vision systems. Applications are not limited to motion tracking, object recognition, and stereo matching. Corner points of an image are defined as points where edges have their maxima of absolute curvature. Some corner detector techniques depend on contours and edges.

Haar-like features focus more on the information of a certain area of the image instead of each pixel. Haar-like feature-based systems are generally faster than a pixel-based system. In addition, these approaches are robust to noise and lighting changes because the grey level difference is computed between the sums of the pixel intensities in each adjacent rectangular region Chen et al. (2007).

Interest points (IPs) have their origin in the notion of corner detection, where authors were looking for features in order to have a robust, stable and well-defined set of features for object recognition. However, most of the corner detectors are actually sensitive to local image regions with a high level of variation. IPs aim to detect a point of interest in the image for later analysis or classification. The main characteristic of IPs is that they can be selected regardless of scale, illumination, translation, rotation or noise. Two important techniques to select IPs are SIFT (Lowe (2004)) and SURF (Bay et al. (2008)), both have demonstrated high precision for this purpose with many successful applications Lindeberg (2013) and Lindeberg (2015).

Charfi et al. (2017) applied SIFT to handshape and palmprint modalities on two databases: the Indian Institute of Technology of Delhi (IITD) hand database (1,150 hand images from 230 different subjects, number of handshapes was not informed) and the Bosphorus hand database (615 subjects, two training images and 1 testing image per subject). The results showed the method yielded high accuracy (99.57%) compared to other popular bimodal hand biometric methods.

The Fourier Transform is an image processing technique widely used to decompose an image into sine and cosine components. The transformation output represents the image in the Fourier domain, whereas the input image is in the corresponding spatial domain. In the Fourier domain, each point represents a frequency contained in the spatial domain image Easton (2010).

Harding and Ellis (2004) proposed a system to recognise a hand gesture trajectory. Fourier Descriptors (FD) were used to recognize 5 subjects performing the same sequence of 6 gestures using neural networks for classification obtaining 90% accuracy. FDs are a manner of encoding the shape taking the Fourier transform of the boundary and every point is mapped to a complex number.

Finally, Principal Component Analysis can be considered as a kind of feature extraction. This is because it reduces the dimensionality and the classification stage is made over the reduced data instead of the entire image.

2.3.5 Mobile-based Methods

Song et al. (2014) have proposed one of the first gesture recognition systems running on a mobile device. Song et al. (2014) uses only a RGB unmodified mobile camera to obtain the shapes for classification. The idea was not to replace the touch screen, instead the idea was to boost it. Basically this system is able to recognize five hand gestures and no-gesture, such as pinch, point, gun, splayed hand and flat hand. The classification phase was made in stages, first the hand is classified into three different coarse depth bins (using depth classification forest), next the handshape is classified into 6 classes, finally, the fingertip parts are detected by a part classification forest. Twenty users were used to test the efficacy and the mean accuracy was 98%.

Lahiani et al. (2015) introduced a real time hand gesture recognition system for Android devices. The system was designed to recognise 10 hand gestures, corresponding to the numbers from 0 to 9. The average accuracy reported was 93%. However, it was only tested with 100 images, with 10 images for each gesture. Images were captured from five different people in different lightning conditions and different backgrounds. The classifier used was SVM.

Several mobile applications for gesture recognition can be found on the internet, e.g. YouTube. Most of them are apply to Android devices. The majority are unpublished works

and exploratory, then meaning it was coded by amateurs or even by experiment programmers just for curiosity. Only a few papers can be found published on this context.

2.3.6 3D Sensors

In this section studies related to 3-dimensions (3D) techniques, applied to handshape recognition, are presented. The 3D methods are those that do not rely in a single camera, rather a sensor (or more than one) and/or multiple cameras are used.

Taylor et al. (2017) used an Intel SR300 depth camera to detect 24 handshapes of ASL performed by 10 individuals. They used the publicly available Sphere-Mesh (Tkach et al. (2016)) hand tracking algorithm to collect and recognize the handshapes. The classifier used was a simple naïve Bayesian. The classification accuracy reported was in average 69.9%.

2.3.6.1 Infrared Cameras

Some studies have been done in applying hand gesture recognition for cars Althoff et al. (2005). The BMW group is the pioneer. In Althoff et al. (2005) they use an infrared camera to detect the hand in an interaction area. The main gestures are applied to control a music player. The system is able to classify 17 different hand gestures using HMM and getting 90% accuracy, Table 2.3. Althoff et al. (2005) do not mention the brand or the technical specification of the camera.

2.3.6.2 Nimble VR

Nimble VR, previously known as 3Gear Systems, is a relatively new vision-based system. It consists of the Microsoft Kinect sensor and estimates the hand gesture via queries to a preprocessed database that relates the detected contour of the hand to its 3D shape Arkenbout et al. (2015). The main goal of Nimble VR was to create a comfortable hand interaction with virtual reality. However, it focuses only on a few hand poses and gestures, e.g. pinching and

pointing. Gestures outside of the database might be poorly classified. In addition, Arkenbout et al. (2015) mixes the use of the Nimble VR, Kinect and data gloves.

Nimble VR was bought by Oculus[2], that is now owned by Facebook[3].

2.3.6.3 Leap Motion

The Leap Motion controller is a compact device that can be connected to a personal computer via USB. It can sense hand movements in the space above it, and these movements are recognised and translated into actions for the computer to perform. The Leap Motion controller is known to be highly sensitive to small movements, and is able to map movements of the full hand Potter et al. (2013), Figure 2.7 shows a leap motion sensor being used to play Fruit Ninja game by an individual with stroke Khademi et al. (2014).

Quesada et al. (2017) used Leap Motion and Intel RealSense to recognise fingerspelling for ASL. Accuracy was reported up to 100% for shape classification, using SVM and data from 50 individuals.

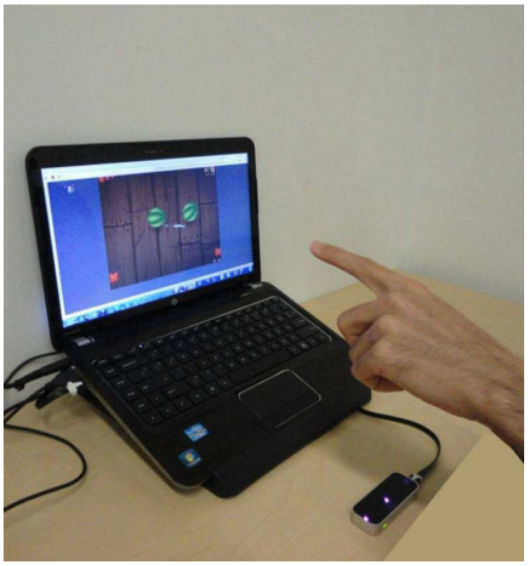


FIGURE 2.7: A Leap Motion connected to a laptop by Khademi et al. (2014)

[2]http://nimblevr.com/
[3]https://www.oculus.com/

The Leap Motion controller can be classified as optical tracking system based on Stereo Vision. This is given by the fact that the controller uses three IR (Infrared Light) emitters and two IR cameras Weichert et al. (2013). The maximum distance from which the controller can capture the hand is about one metre. It runs at around 200 frames per second with 0.7 millimetres accuracy Weichert et al. (2013).

This controller has been applied to Australian Sign Language (Auslan) in Potter et al. (2013). An artificial Neural Network was used to classify Auslan shapes. However, some weaknesses have been discovered as well, e.g. difficulty in recognising a gesture when hands move into a place that is not seen by the controller. In addition, the detection does not work when fingers are brought together.

2.3.6.4 Time-of-flight Cameras

Time-of-flight (ToF) cameras produce depth and reflectance images[4] at significant frames per second. These cameras have sensors that compute the distance based on the speed of the light, by measuring the time of the light signal from the camera until it is received by a CCD/CMOS chip, for each pixel Droeschel et al. (2011). It is a non-invasive way of obtaining sensory information about a scene. This kind of camera is more robust than other sensors because it is not sensitive to illumination Ganapathi et al. (2010).

ToF cameras have been successfully used for real-world applications, e.g. robotics to identify human pointing gestures Droeschel et al. (2011). In addition, they have shown a good performance when applied to hand gesture recognition. Oprisescu et al. (2012) were able to recognise 9 hand gestures with over 93% accuracy. Malawski (2014) used ToF cameras for 3D medical data analysis using a Support Vector Machine (SVM) classifier.

Even before Kinect and Leap Motion come to dominate the market, ToF cameras were considered a low-cost device and widely used in HCI Cheng et al. (2015).

[4]reflectance: the portion of light reflected by each point within the scene

2.3.6.5 SoftKinetic

SoftKinetic is a Sony group company leading in 3D vision technology. SoftKinetic develops hardware and software for gesture recognition in real-time, e.g. time-of-flight cameras. They offer DepthSense® sensors, cameras and modules. In addition, SoftKinetic offer an IISU® middleware. This platform tracks full body and hands precisely. IISU is part of DepthSense® Libraries Softkinetic (2017).

SoftKinetic is the first company to implement gesture recognition in a car. BMW now allows drivers to operate some functions using hand movements. Gestures such as 'swiping' or 'pointing' are identified in the recognition area above the centre console and trigger functions such as accepting or rejecting an incoming telephone call. In addition, circular motions of the index finger can adjust volume, and simple two-finger pointing can be configured for more functions Softkinetic (2017). Figure 2.8 shows a SoftKinetic camera DS525 and Figure 2.9 shows a SoftKinetic embedded in a BMW car. It is not possible to identify if this system is the same as the showed in Section 2.3.6.1 because Althoff et al. (2005) do not mention the brand and specific kind of camera used.

FIGURE 2.8: A SoftKinetic camera DS525

FIGURE 2.9: A SoftKinetic embedded in a BMW car

A study has been conducted by Safaei and Wu (2015) in order to evaluate 3D hand motion with SoftKinetic. Hidden Markov Models were used to conduct the classification. The proposed method uses a framework to capture RGB and depth data from a SoftKinetic camera and a UV map[5] generated to merge the RGB image with the depth image to create a 3D model.

Chaczko and Alenazy (2016) discuss modelling and simulation of gesture recognition technologies. The initial plan was to use a Kinect camera. However, this plan was changed to use the SoftKinetic DepthSense 325 camera due to ease of use and compatibility with the IISU middleware. The goal was the development of a web application that can monitor users, who are making gestures in front of the camera(s), and that can provide controls for other possible applications. However, it is not clear what kind of gestures the web application is able to recognise nor how the process of classification is made.

In video-based techniques one of the problems to be addressed is locating the hands and segmenting them from the background. Suarez and Murphy (2012) have researched how depth images can help in solving these problems, specially when there are occlusions, lighting changes or rapid motion. Some of the hardware available are Microsoft Kinect, ASUS Xtion, or Mesa SwissRanger, or as an alternative any image taken from stereo video cameras.

One solution for low complexity hand skeleton tracking without calibration has been proposed by the company called Gestigon[6] Coleca et al. (2015). The idea is to apply self-organizing maps (SOM) for hand and full body tracking. The proposed algorithms proved efficient and robust with good tracking results. In addition, they can be easily adapted to any depth image supplied, such as Kinect, since it provides enough accuracy.

2.3.6.6 Stereoscopic Cameras

Stereoscopic or stereo camera systems consist of two or more standard cameras capturing the images of the same object form different angles at the same time. Image matching methods

[5]UV mapping is the 3D modelling process of projecting a 2D image onto a 3D model's surface for texture mapping

[6]http://www.gestigon.com

are used to create the disparity map that approximates per-pixel depth. These cameras need to be calibrated and produce lower-fidelity depth images than ToF cameras. In addition, this kind of system requires a computational overhead to solve the image registration problem for each pair of images. However, it works well in good illumination conditions and it is financially inexpensive Suarez and Murphy (2012).

Cerlinca and Pentiuc (2011) have used stereo cameras for gesture recognition. When using a 3D approach an object can be correctly identified even if it shows an uneven distribution of colour components. The Cerlinca and Pentiuc (2011) idea was to detect the hand and then grow a 3D region. It showed that the hand could be detected in almost any illumination and scale condition. However, the accuracy was not mentioned nor were gesture classifications computed.

Liu and Kehtarnavaz (2016) proposed a robust real-time hand gesture recognition using stereo images. Stereo cameras were used to increase the robustness of hand detection, because they are inexpensive and provide a faster detection leading to a real-time application. Merging the information from the left and the right camera made Liu and Kehtarnavaz (2016) obtain an average of 93% accuracy for seven motion hand gestures and 92% average for finger spelling. All images were obtained with real illumination conditions and different backgrounds.

Konda (2012) have created a system for directing robots using stereo images. These stereo images are used to feed into a Convolutional Neural Network in order to detect the gestures. The basic gestures that the robot was able to recognise were forward, backwards, right, left, follow me and stop.

2.3.6.7 Kinect-based Methods

The Microsoft Kinect has two versions. The fist version, Kinect V1, was released in 2010 and it is composed of a QVGA (320x240) depth camera and a VGA (640x480) video camera, both cameras capture at 30 frames per second (fps). The depth camera, developed by PrimeSense (nowadays part of Apple Inc.), is composed of an Infrared (IR) light emitter and works on

the principle of structured light Suarez and Murphy (2012). The second version, Kinect V2, was released in 2013 and is composed of a 1920x1080 pixels camera, and a time-of-flight sensor developed by Microsoft Ahmed et al. (2016). Kinect V1 can detect 20 bones of human skeleton (1 in the hand). However, Kinect V2 can sense 26 bones including 2 more bones in each hand (3 in total: hand, hand-tip, thumb).

Since the Kinect has been made available a large amount of research has been done on gesture recognition using Kinect.

Pedersoli et al. (2014) proposed the first open source package for the Kinect sensor for hand pose and gesture recognition. American Sign Language was the goal of this work. Classification was done in two stages: one with a Support Vectors Machine and the other with Hidden Markov Models. The classifiers come already trained on ASL alphabet and 16 unistroke dynamic gestures. The average accuracy for hand gesture classification was above 70% for 16 gestures.

Marin et al. (2014) proposed a system that combines the use of Leap Motion with Kinect. In this work a comparison of the two devices is shown. In addition, accuracy of the combination of the two devices into the same system is shown. A Support Vector Machine was considered for classification. Accuracy when using only Leap Motion is around 80% and 89.7% for only Kinect. However, the accuracy increases to 91.2% when using the combination.

Yao and Fu (2014) have used Kinect to achieve more reliable and accurate tracking under unconstrained conditions in hand gesture recognition. However, colour gloves for colour-based motion tracking were used and a hand contour model is used to simplify the gesture matching process, in order to reduce the computational complexity of gesture matching. The average accuracy of hand parts classification was 74.65% for four hand gestures.

Ren et al. (2011) created two interesting applications using Kinect V1. The first application was the rock-paper-scissors game and the second was the for basic arithmetic computation (+-*/). Finger-Earth Mover's Distance (FEMD) was used to distinguish between different handshapes. The mean accuracy showed was 90.6% for 10 different gestures.

Biswas and Basu (2011) have used Kinect V1 to detect a human hand from a depth image and classify eight handshapes. The eight shapes consist of clapping, call someone, greeting with folded hands, waving hand, shaking head sideways – "NO", tilting head up and down – "YES", clasped behind head and chin resting on hand. The training and classification stages were done using an SVM. The mean accuracy was 80.87%, the lowest being 54.22% and the highest 95.56%.

Kinect V2 has been used for gesture recognition in Ahmed et al. (2016). They proposed a method for converting sign language to speech. The accuracy shown was 84%. Unfortunately Ahmed et al. (2016) do not describe what kind of gestures the system is able to recognise nor how many.

Table 2.1 shows a comparison among Kinect V1, Leap Motion and Kinect V2, adapted from Cheng et al. (2015). In addition, Weichert et al. (2013) stated that Leap Motion has the best accuracy.

TABLE 2.1: Comparison among devices available for depth images

Sensor	Resolution	Range	Accuracy	Description
Kinect V1	320×240	0.8-4.0m	4mm	20 body joints
Leap Motion	640×240	25-600mm	0.01mm	27 hand joints
Kinect V2	1920×1080	0.8-4.5m	1mm	26 body joints

2.4 Considerations about Vision and Non-vision Approaches

Table 2.2 summarises the works cited until here for non-vsion based approaches, including sensor used, classifier, application, accuracy number of persons and number of gestures. Table 2.3 show the works cited for vision based approaches. It shows a lack of works using only one regular camera.

TABLE 2.2: Comparison among non-vision gesture recognition types, SL = Sign Language

Authors	Camera/Sensor	Classifier	Application	Accuracy	Signers	Gestures
Bedregal et al. (2007)	Data gloves	Fuzzy logic	Brazilian SL	n/a	n/a	n/a
Wang et al. (2006)	Cyberglove	HMM	American SL	95%	5	26
Bui and Nguyen (2007)	Data glove/MEMS	Fuzzy logic	Vietnamese SL	98%[7]	5	23
Maebatake et al. (2008)	Polhemus FASTRAK	HMM	SL	75.6%	4	183
Abdulla and Manaf (2016)	Smart glove	Arduino Software	Arabic SL	n/a	n/a	28
Zhang et al. (2011)	ACC and EMG	HMM/Decision tree	Chinese SL	95.3%	10	18

TABLE 2.3: Comparison among vision gesture recognition types, SL = Sign Language

Authors	Camera/Sensor	Classifier	Application	Accuracy
Althoff et al. (2005)	IR camera	HMM	Gesture recog.	90%
Huang and Jeng (2001)	SONY XC7500	PCA/HMM	Gesture recog.	92%[8]
Arkenbout et al. (2015)	Nimble/Kinect/DataGlove	Kalman filter	Hand motion track.	n/a
Potter et al. (2013)	Leap Motion	ANN	Australian SL	n/a
Droeschel et al. (2011)	ToF camera	HMM	Gesture recog.	n/a
Ganapathi et al. (2010)	ToF camera	Bayesian network	Motion capture	n/a
Oprisescu et al. (2012)	ToF camera	Decision tree	Gesture recog.	93%
Malawski (2014)	ToF camera	SVM	Gesture recog.	95%
Safaei and Wu (2015)	SoftKinetic	HMM	Motion recog.	86.72%[9]
Chaczko and Alenazy (2016)	SoftKinetic	IISU SDK	Gesture recog.	n/a
Coleca et al. (2015)	Kinect	SOM	Hand/boddy track.	n/a
Cerlinca and Pentiuc (2011)	Stereo cameras	n/a	Gesture recog.	n/a
Liu and Kehtarnavaz (2016)	Stereo cameras	Optical flow/HMM	Gesture recog.	92%
Konda (2012)	Stereo cameras	CNN	Robotics	n/a
Ahmed et al. (2016)	Kinect V2	n/a	SL	84%
Marin et al. (2014)	Kinect/Leap Motion	SVM	Gesture recog.	91.2%
Yao and Fu (2014)	Kinect V1	n/a	SL	74.65%
Ren et al. (2011)	Kinect V1	FEMD	Gesture recog.	90.6%
Biswas and Basu (2011)	Kinect V1	SVM	Gesture recog.	80.87%

[7] Overall average, as authors show the accuracy by shape
[8] 89% for dynamic gestures
[9] Average accuracy fro 3 motions and 3 speeds

Chen et al. (2007)	Camera	Haar-like/AdaBoost	Gesture recog.	90%
Cui and Weng (2000)	Webcam	Recursive partition	American SL	93.2%
Mistry et al. (2009)	Projector / camera	n/a	Gesture recog.	n/a
Naoum et al. (2012)	Webcam	k-NN	Arabic SL	50.0%[10]
Song et al. (2014)	Mobile camera	Random forests	Gesture recog.	98%
Lahiani et al. (2015)	Mobile camera	SVM	Gesture recog.	93%

2.5 Pattern Recognition Techniques

In this section a discussion on techniques related to hand gesture recognition is presented.

2.5.1 Principal Component Analysis

Principal Component Analysis is a technique for dimensionality reduction and feature extraction, Han and Liu (2014). It uses the covariance matrix of the data to create a space called an eigenspace. Each dimension in this space is represented by one eigenvector. The number of eigenvectors used to represent the full data is always lower than the dimensionality of the raw data.

PCA is sensitive to scale. In other words, different sizes of the same handshape may produce different projections in the eigenspace. Chomat et al. (2000) stated that the position of a projected image in the PCA space changes according to the appearance of the image, translations, variable background and/or illumination. Hence PCA requires object detection, segmentation and precise normalisation in intensity, size and position.

In the same work, Chomat et al. (2000) proposed some techniques to avoid that sensitivity to scale, translation and illumination. It was proposed to use local appearance-based methods

[10] Accuracy for naked hand, this value increase for 80% when wearing a white colour glove

that describe the appearance of neighbourhoods. The influence of background is minimised by considering small neighbourhoods. The matter of object position may be solved by mapping the locally connected structures into multi-dimensional histograms in a space of local appearances. Illumination invariance is obtained by energy normalization during the image projection to the appearance space.

PCA has been widely applied in handshape recognition. In Farouk et al. (2013) PCA was applied over blurred images to recognise handshapes in Irish Sign Language. In addition, PCA was successfully used for face recognition where the eigenvectors were called eigenfaces Turk and Pentland (1991).

PCA will be explained in detail and used in Chapter 4.

2.5.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a variation of Fisher's linear discriminants, used in pattern recognition and machine learning to find a linear combination of features that characterises or separates two or more classes of objects or events, Aran and Akarun (2010). LDA is used in this thesis in Chapter 5

Zhao et al. (1998) has concluded that applying LDA over raw data for face recognition did not respond well. However, combining LDA and PCA improved the performance. Yang et al. (2000) proposed a new LDA algorithm for face recognition and obtained an accuracy around 90%. Transferring the full rank requirement from S_w to S_b, the algorithm avoided losing the most discriminant dimensions because of removing the null space of S_w.

Aran and Akarun (2010) had proposed a multiclass classification strategy for Fisher scores and applied to sign language recognition. Overall, the accuracy obtained by PCA and LDA was quite similar.

2.5.3 Multidimensional Scaling

Multidimensional Scaling (MDS) is a type of linear dimensionality reduction. MDS maps the original high dimensional space to a lower dimensional space, attempting to preserve pairwise distances. Thus, each object has coordinates assigned in each of the N-dimensions. MDS addresses the problem of constructing a configuration of t points in Euclidean space by using information about the distances between the t patterns Ghodsi (2006). A threshold is set to determine the accuracy of the low-dimensional embedding Saxena and Gupta (2004). MDS has a different mathematics to PCA. However, they produce similar results Ghodsi (2006). Additionally, MDS is known as Principal Coordinates Analysis (PCoA), Torgerson Scaling or Torgerson–Gower scaling Barth et al. (2008).

2.5.4 k-Nearest Neighbour

k-Nearest Neighbour (k-NN) is a method widely used for classification in CV. The input consists of the k closest training examples in the feature space. The output is a class where an item is classified by a majority vote of its neighbours, k is a positive integer, generally low Naoum et al. (2012).

The most common distance metric used in k-NN is Euclidean.

$$d(a,b) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}.$$

Where a and b are two points, $x_1, x_2, x_3 \ldots x_n$ are the coordinates of a and $y_1, y_2, y_3 \ldots y_n$ are the coordinated of b.

2.5.5 Perpendicular Distance

Perpendicular distance is a metric distance used to calculate the distance between a point and an eigenspace. The point is in a space with a higher number of dimensions than the

eigenspace. The point and the eigenspace are embedded in the same higher dimensional space. The shortest perpendicular distance from a projected object to an eigenspace can indicate the nearest neighbour class where each class is represented by a different eigenspace Coogan (2007).

$$dis = \sqrt{\sum(p-o)^2 - \sum(p*v - o*v)^2} \tag{2.1}$$

Where o is the origin (e.g. mean of all projected images at the same angle), v is the set of eigenvectors and p is the point. Equation 2.1 returns the distance between a new image and a space. Figure 2.10 shows a diagram of the perpendicular distance from a point p to an eigenspace v.

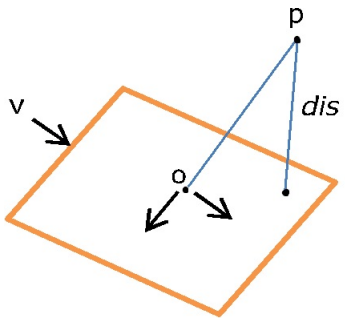


FIGURE 2.10: Perpendicular distance from a point p to an eigenspace v with origin o

Perpendicular distance tends to show improved results when the classes are well separated. If the classes are curved or overlap, this kind of distance will not separate them.

2.5.6 Manifolds

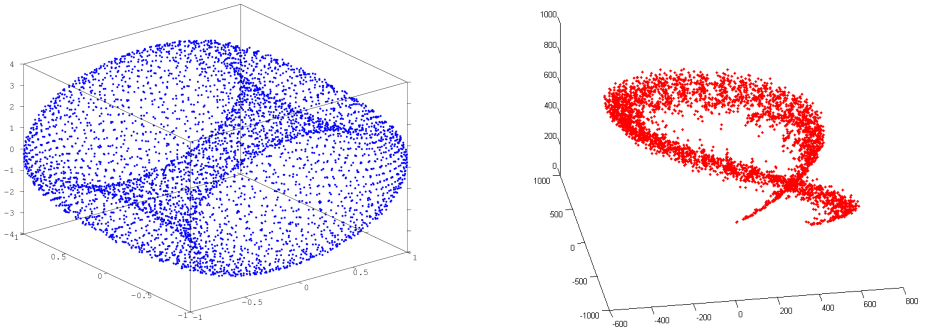
Manifolds are spaces that locally look like Euclidean space Lee (2012). A curve, a circle and a parabola are examples of one-dimensional manifolds, whereas spheres, tori, paraboloids, ellipsoids and hyperboloids are examples of a two-dimensional manifolds Seung and Lee (2000).

Manifold learning is one form of representing data in a lower dimensional space. The problem with this is to understand how it changes in terms of their basic modes of variability,

i.e., the pose and expression of a human face, or the rotation and scaling of an object. These problems are common in computer vision and pattern recognition. For instance Weinberger and Saul (2006) define that an image can be seen as a point in a high dimensional space whose dimensionality is equal to the number of pixels in the image. In the case that the images are effectively parameterised by a lower number of dimensions, then they will lie close to a low dimensional manifold. Manifolds can be associated with structures in ensembles of images, e.g. clusters or parts of objects.

Elgammal and Lee (2007) state that in human activities, such as gesturing, most of the gestures are one-dimensional manifolds. These manifolds can be twisted and/or self-intersect in such a high-dimensional visual space. PCA is widely used in appearance modelling to discover subspaces for appearance variations, i.e. for face recognition and to model the appearance manifold and view manifold for 3D object recognition.

Figure 2.11 shows two examples of manifolds. Figure 2.11(a) shows point clouds in spaces with categorical attributes Baryshnikov and Figure 2.11(b) shows a manifold for handshapes of the ISL in a PCA space. Beyond linear manifold techniques there are others non-linear. Non-linear manifold learning techniques will be discussed in Section 2.5.19



(a) (b)

FIGURE 2.11: Example of manifolds, (a) point clouds in spaces with categorical attributes, (b) manifold for handshapes of the ISL projected into a PCA space

2.5.7 Image Blurring

The process of applying convolution with a Gaussian kernel to the pixels of an image is commonly known as Gaussian blur. Gaussian blur has the effect of reducing details from images and smoothing the edges. Thus, the convolution process is usually used as a low-pass filter Farouk et al. (2013).

The equation of a Gaussian function in one dimension is shown in the Equation 2.2

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{2.2}$$

Considering a 1D image, Figure 2.12, the pixel located in the middle has the highest weight and the weight of its neighbours decreases as the spatial distance between these and the centre pixel increases.

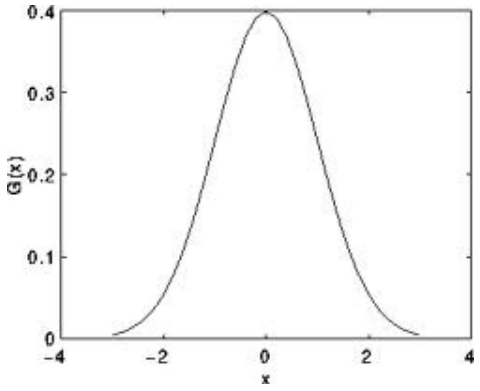


FIGURE 2.12: One dimensional curve for Gaussian filter blurring

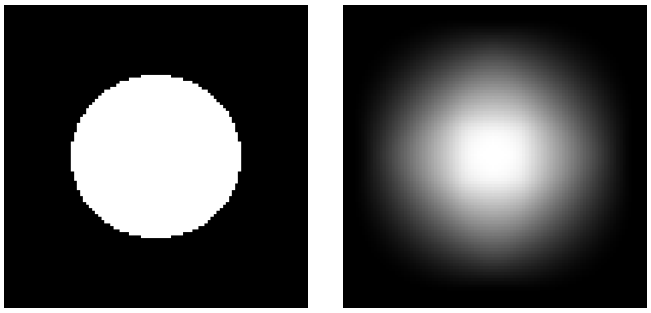
In two dimensions, it is the product of two Gaussians, shown in the Equation 2.3

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.3}$$

Where σ is the standard deviation of the distribution. In 2 dimensions this equation creates concentric circles with the distribution from the centre point. The values of this distribution are

used to build a convolution kernel that is applied to the original image Stockman and Shapiro (2001).

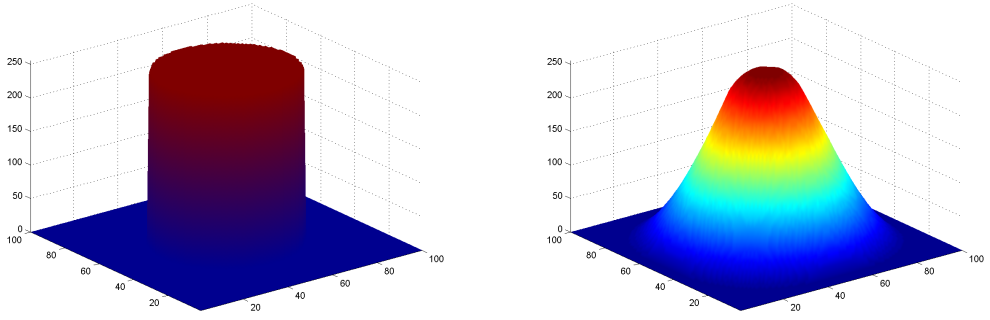
In order to illustrate the Gaussian blur in 2D, an image of a white circle on a black background can be used. Figure 2.13 shows this image before and after blurring been applied. σ has been set to 60.



(a) A circle image (b) A circle image blurred

FIGURE 2.13: A circle image and the same image blurred by a Gaussian filter

The same images produce surface plots, as shown in Figure 2.14

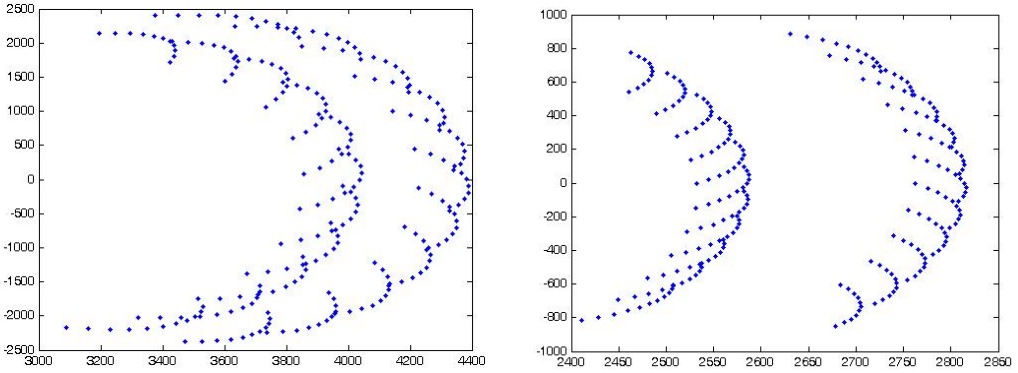


(a) A surface plot of the circle image (b) A surface plot of the circle blurred image

FIGURE 2.14: A surface plot of a circle image and the same plot for the blurred image

Farouk et al. (2013) used this kind of blurring in the classification process. The incoming object and the training samples were blurred by the same Gaussian kernel. The classification process of a new blurred incoming image is less complicated, because blurred images have less details.

In the PCA process blurring helps to reduce the non-linearity in the manifolds within the eigenspaces. It is more efficient to analyse a flat manifold than a curved one, because curves tend to overlap. Blurring applied over images shows much less detail. However, it still has enough quality for recognition, since PCA uses every pixel for computation. Figures 2.15 Farouk et al. (2013) shows an illustration of how manifolds appears before and after blurring. Note that after blurring manifolds appear more separated and less curved, reducing overlapping.



(a) PCA manifold for non-blurred images (b) PCA manifold for blurred images

FIGURE 2.15: PCA manifolds for blurred and non-blurred images, note that for blurred images manifolds are flatter

2.5.8 Data Pyramids

In Farouk (2015), four algorithms using different pattern recognition techniques were proposed. The first algorithm was based on using perpendicular distance to measure the distance between new patterns and the nearest eigenspace. The second algorithm was based on using supervised multidimensional grids. The third algorithm used unsupervised multidimensional grids (MDG) to divide the space into cells containing similar objects, shown in Figure 2.16. The fourth algorithm was based on training a set of simple architecture multi-layer neural networks at the different levels of the pyramid to map new patterns to the closest class. The proposed algorithms were categorized as example-based approaches where a set of computer-generated images were used to densely sample the space.

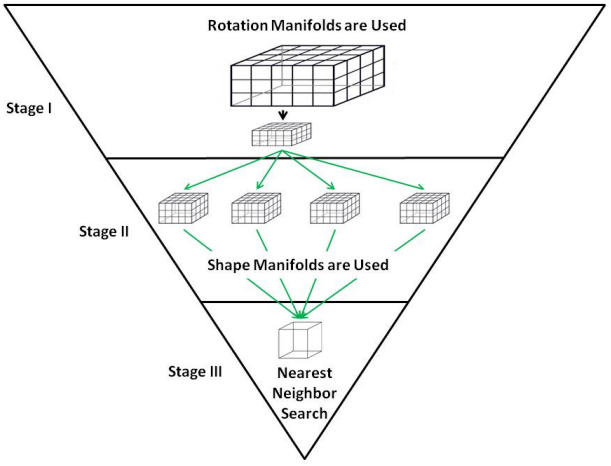


FIGURE 2.16: Multistage Hierarchy Using MDGs proposed by Farouk (2015)

2.5.9 Multi-Dimensional Grids

Farouk et al. (2013) proposed algorithms using a multistage hierarchical strategy for handshape recognition. The proposed multistage hierarchy analyses new patterns by projecting them into the different levels of a pyramid. This pyramid consists of different principal component spaces. Image blurring was used to reduce the non-linearity in manifolds generated by a set of example images. Flattening the space helps in classifying different handshapes more accurately.

The experimental results showed that the proposed algorithms are applicable for real time applications with high accuracy measures. They can achieve frame rates of more than 10 frames per second and accuracy up to 98% in the test data Farouk (2015).

2.5.10 Coarse-to-fine Search

Coarse-to-fine search is an efficient algorithm suitable for implementation in single processor systems. It uses a number of iterations to reduce the computing and storage requirements of the standard procedure. In other words, this technique tries to divide a solution into parts instead

of solve it at once. It finds a coarse solution at the first iteration and then uses that as a starting point to find a finer solution at the next iteration - and so on.

Coarse-to-fine search was successfully applied in Han (2004) for a hand-based personal authentication. They created a threshold to check if a testing sample passed or not, if not it has to be rechecked by a fine-level verification phase.

In Farouk et al. (2009) coarse-to-fine search was used for handshape recognition in the supervised and unsupervised multidimensional grid algorithms. Basically, a pyramid structure with different stages was used as a coarse-to-fine search technique to provide an estimate of the possible location of the target. The search starts at the top level of the pyramid with a reduced resolution version of the images. The search proceeds through the pyramid levels.

2.5.11 Support Vector Machines

Support vector machines (SVM) are a classification technique that uses machine learning theory to maximise the accuracy of prediction. SVMs use a kernel function that can be either linear or non-linear. SVMs are used in this thesis in Chapter 5.

SVM classifiers have been used in a significant number of works in gesture recognition. Biswas and Basu (2011) and Pedersoli et al. (2014) have applied SVM to recognise shapes using the Kinect sensor. Marin et al. (2014) used this technique for images obtained from both Kinect and Leap Motion. Malawski (2014) applied SVM for gestures recognition with ToF camera for 3D medical data analysis. Chan et al. (2015) used SVM for PCANet (Section 2.5.18). Wang et al. (2016) applied SVM to detect human falls in surveillance videos. Wang et al. (2015) utilised deep learning and SVM to recognize traffic lights. Roberto and Pizzolato (2013) and De Souza et al. (2013) have used SVM for gesture recognition of Brazilian Sign Language (LIBRAS).

2.5.12 Neural Network

It is known that human brains have hundreds of billions of interconnected neurons in order to process information. Based on this, researchers are trying to demonstrate a similar kind of intelligence on machines. For example in machine translation and pattern recognition Wang (2003) and Goodfellow et al. (2016).

A neural network (NN), or artificial neural network (ANN), is composed of an input layer of artificial neurons, one or more hidden layers of neurons, and finally a layer of output neurons. Figure 2.17 shows one common scheme for one hidden layer, each connection has a weight. The output, h_i of neuron i in the hidden layer is computed by the Equation 2.4

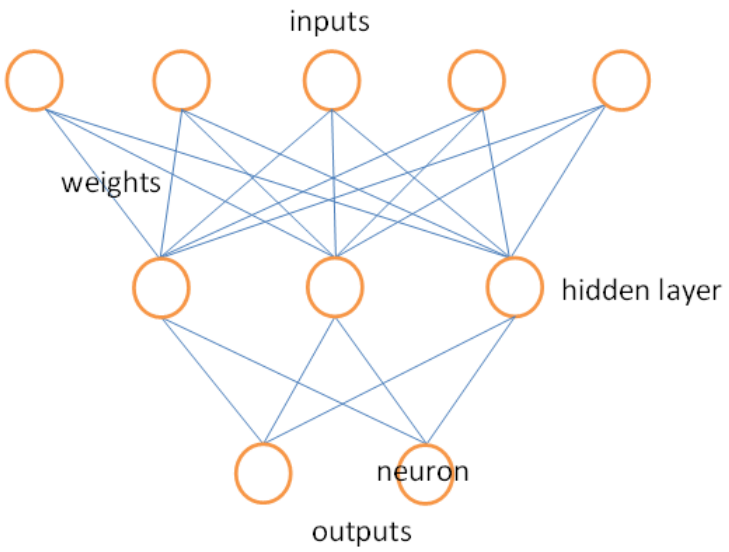


FIGURE 2.17: Example of neural network architecture with one hidden layer

$$h_i = \sigma\left(\sum_{j=1}^{N} V_{ij}x_j + T_i^{hid}\right), \tag{2.4}$$

where σ is the activation function, N is the number of input neurons, V_{ij} the weights, x_j the input neurons and T_i^{hid} the threshold terms of the hidden neurons. The activation function introduces non-linearity into the neural network. The sigmoid function is one example of the activation function, Equation 2.5, Wang (2003).

Figure 2.18 shows an example of sigmoid activation function Networks (2013)

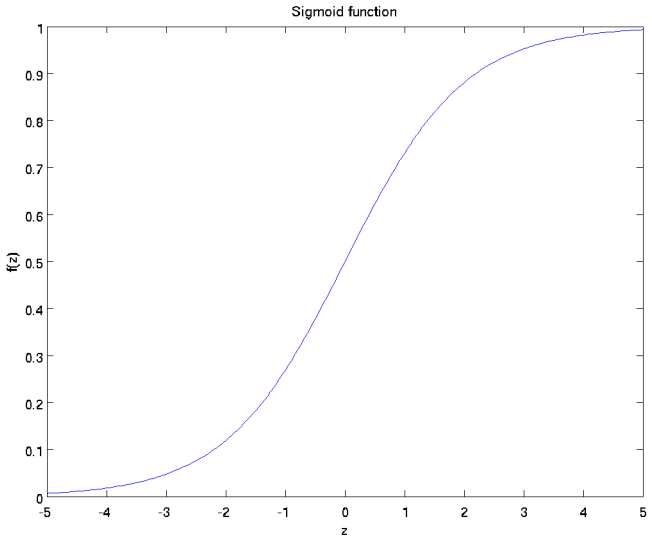


FIGURE 2.18: Illustration of a sigmoid activation function

$$\sigma(u) = \frac{1}{1+exp(-u)}. \tag{2.5}$$

A neural network built in this way may approximate any computable function to an arbitrary precision. The input neurons are independent variables. However, the output ones are dependent variables of the function that is being approximated by the network. Inputs and outputs of the neural network can have multiple output units, Boolean (1 or 0), or symbols e.g. colours Wang (2003).

Farouk (2015) has proposed, in his fourth algorithm, a multistage hierarchical technique for handshape recognition using ANN. A pyramid using a set of neural networks at the different levels was created, where the network was fed with eigenspaces instead of raw images. The pyramid shown in Figure 2.19 has three stages. It organises different eigenspaces in a supervised way according to the pose and the shape of the hand. The first stage is used to map a new image to an estimated rotation angle of the arm. The second stage, is where the classification of the incoming shape happens. The third and last stage, the translation position is estimated based on the nearest neighbour.

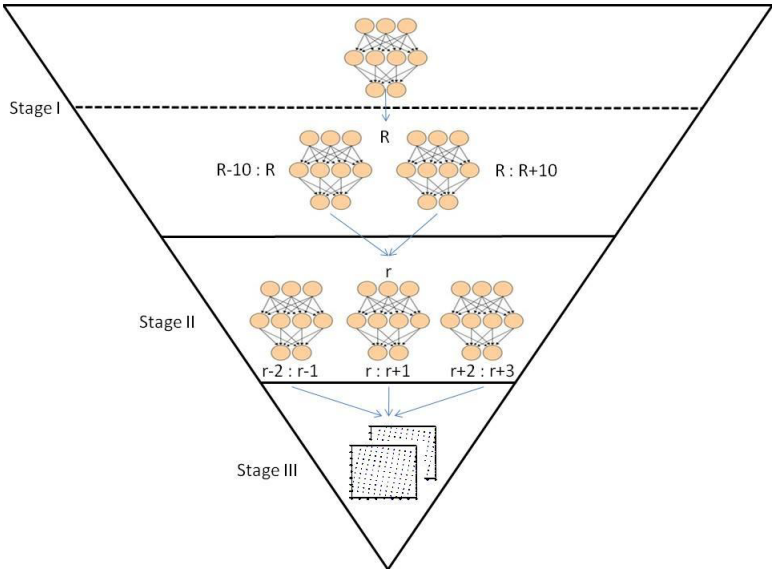


FIGURE 2.19: ANN multistage hierarchy proposed by Farouk (2015)

Two hidden layers were used in Farouk (2015). The number of input neurons is equal to the number of eigenvectors chosen to build the input eigenspace. This number of eigenvectors was chosen by trial and error. Gaussian blur was applied over the images. The average accuracy was 87.52%.

An artificial neural network was used in order to classify Australian Sign Language (Auslan) shapes by Potter et al. (2013). A Leap Motion controller was used to obtain the hand-shapes. Each symbol (shape) is trained before use and attempted recognition. The neural network was able to assess a symbol and output with a range of certainty about a particular sign between 0% and 100%. This network was trained by data obtained from the Leap Motion API. The network inputs training data which has no limits or expectations on the data format.

2.5.13 Multilayer Perceptron

Multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of layers of nodes. Each node is a neuron that uses a non-linear activation function. The output of a neuron is scaled by the connecting weight and fed forward to be an input to the

neuron in the next layer. MLP utilises a supervised learning technique known as backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron, what makes MLP able to classify data that is not linear Licciardi et al. (2014).

The most common activation functions are: *identity* useful to implement linear bottleneck $f(x) = x$; *logistic* sigmoid function $f(x) = 1/(1 + exp(-x))$; *tanh* hyperbolic tan function $f(x) = tanh(x)$ and *relu* the rectified linear unit function $f(x) = max(0, x)$. MLP can learn through training, it uses a set of training data that consists of a series of input and output vectors Gardner and Dorling (1998).

Backpropagation is an algorithm where the weights in the network are initially set to small random values and then the algorithm calculates the local gradient of the error surface and changes the weights in the direction of steepest local gradient Gardner and Dorling (1998).

2.5.14 Deep Learning

Deep neural networks (DNNs) have received significant attention recently. DNNs are a set of techniques that learn features from data of interest. They offer more advanced techniques for extracting features, than traditional techniques such as PCA. The concept of deep learning is to discover multiple levels of representation, with the hope that higher-level features represent more abstract semantics of the data LeCun et al. (2015). This representation tends to provide more invariance to intra-class variability Chan et al. (2015). Deep Networks are nowadays one of the main research topics in shape recognition.

2.5.15 Convolutional Neural Network

Convolutional neural networks (CNNs) or ConvNet specialise in recognising patterns. They are widely known for robustness to distortion and having minimal or no preprocessing. CNNs are one kind of deep learning with multiple layers and feature maps created by convolving an input image with a filter. CNN will be explained in detail in Section 5.2.1.2.

Figure 2.20 shows the architecture used by Nagi et al. (2011). This scheme has 1 input layer, 3 convolutional layers and 2 pooling layers (6 hidden layers all together) and 20 feature maps.

Nagi et al. (2011) proposed combining convolution and max pooling (MPCNN) for classification of human hand gestures by mobile robots. Coloured gloves have been used for colour segmentation. In total 6 gestures are classified with 96% accuracy.

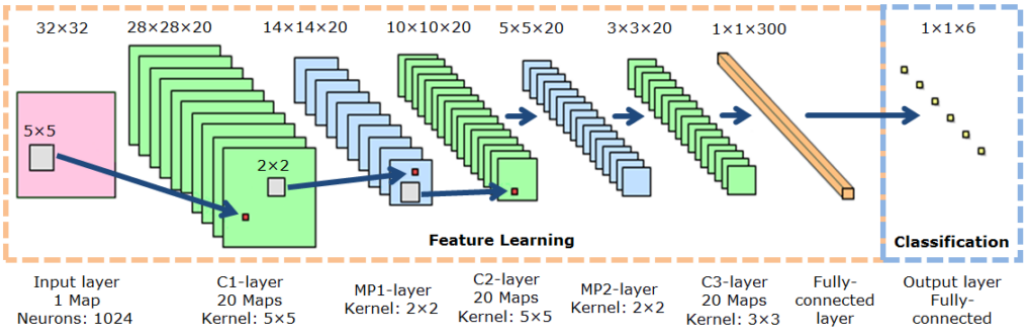


FIGURE 2.20: CNN architecture with different convolutional layers (c), max pooling layers (MP), fully-connected and output layers; proposed by Nagi et al. (2011)

One of the first successful applications of Convolutional Networks was LeNet. LeNet has had different improvements over time, starting from LeNet 1, LeNet 4, LeNet 5 and Boosted LeNet 4. The LeNet architecture was used to read handwritten zip code digits LeCun et al. (1995). At that time, days were needed to train a dataset.

After the success of LeNet, AlexNet was introduced into the field. A large convolutional neural network was proposed to classify the 1.2 million high-resolution images in the ImageNet dataset. ImageNet is composed of over 15 million high-resolution images belonging to an average of 22 thousand categories Krizhevsky et al. (2012). Images were not preprocessed apart from resizing to keep all the images the same size.

More recently Szegedy et al. (2014) proposed a new improvement for CNN and won the ImageNet challenge in 2014. The utilisation of the computing resources inside the network has been improved. The depth and width of the network was increased and the computational effort was kept. The decisions were based on the Hebbian principle (neurons that fire together, wire together). The 22 layers deep network (or 27 layers including pooling) was called GoogLeNet.

The main contribution of Szegedy et al. (2014) was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). In addition, it uses average pooling instead of the fully connected layers at the end of the CNN, eliminating a large amount of parameters that seem less relevant.

2.5.16 Recurrent Neural Network

Recurrent Neural Networks (RNN) are characterised by having feedback connections between the layers and within the layer. This feedback makes the network have local memory characteristics that store activity patterns. These patterns are presented to the network more than once. Thus, at any time, the network output is calculated by propagating the input pattern through the network, and the recurrent activations are propagated back to the extra layer, known as the context layer, that copies the activation pattern from the output on the last part Maraqa and Abu-Zaiter (2008)

LeCun et al. (2015) states that RNNs are often more appropriate for tasks that involve sequential inputs, such as language and speech recognition.

2.5.17 Long Short Term Memory

Long Short Term Memory (LSTM) is based on the idea of facilitating the storage of the information for a longer time with an explicit memory. This seems to be complicated in traditional RNNs. LSTM networks use special hidden units, with the goal of remembering inputs for a longer time LeCun et al. (2015).

It has been proved that LSTM is more efficient than the conventional RNN, mainly because the LSTM have a greater quantity of layers than RNN for each time step LeCun et al. (2015).

2.5.18 PCA Network

PCANet, a Deep Neural Network for PCA, was proposed by Chan et al. (2015). This technique consists of the use of PCA for feature extraction in 2 stages and a hashing and histogram for indexing and pooling. It has been applied to face recognition obtaining 99.58% accuracy on the Extended Yale B dataset.

The architecture of the network is shown in Figure 2.21. PCANet is composed of three stages: the first two stages are PCA and the last stage is hashing and histogram.

In every layer of the first step, the input images are convolved with a filter as in CNN. The most significant difference between CNN and PCANet is how the filter is trained. PCANet does not use backpropagation for training and it is replaced with a solved optimization problem Huang and Yuan (2015) The non-linear layer is the most straightforward quantization which is considered as hashing.

Assume that there are N training images of size $m \times n$. In each image, a patch of size $k_1 \times k_2$ is taken around each pixel. Thus, all the patches are collected, vectorised and combined into a matrix of $k_1 \times k_2$ rows and $(m-k_1+1) \times (n-k_2+1)$ columns.

For the i_{th} image I_i, a matrix X_i and the patch mean from each patch is obtained:

$$\mathrm{X} = \left[\overline{X_1}, \overline{X_2}, \ldots, \overline{X_N}\right] \in R^{k_1 k_2 \times Nc}$$

where c denoting the number of rows of X_i. Thus, the eigenvectors of XX^T are obtained and those corresponding to the L_1 largest eigenvalues are saved as the PCA filters, expressed as:

$$W_l^1 = q_l\left(XX^T\right) \in R^{k_1 k_2}, l = 1, 2, \ldots, L_1$$

The principal eigenvectors capture the highest variation of the training patches.

At the second stage, a similar process to stage 1 is implemented. The input images I_i^l of stage 2 are:

$$I_i^l = I_i * W_l^1, i = 1,2,\ldots,N$$

the boundary of I_i is zero-padded and I_i^l have the same size of I_i. All the patches of I_i^l are collected and the patch mean from each patch is obtained:

$$Y^l = \left[\overline{Y_1^l}, \overline{Y_2^l}, \ldots, \overline{Y_N^l}\right] \in R^{k_1 k_2 \times Nc}, l = 1,2,\ldots L_1$$

and the Y^l is combined together as a matrix:

$$Y = \left[Y^1, Y^2, \ldots, Y^{L_1}\right] \in R^{k_1 k_2 \times L_1 Nc}$$

Thus, the eigenvectors of YY^T are obtained, storing the corresponding eigenvectors to the L_2 largest eigenvalues as the PCA filters of the second stage

$$W_\ell^2 = q_\ell\left(YY^T\right) \in R^{k_1 k_2}, \ell = 1,2,\ldots,L_2$$

Finally, the last step, the pooling feature is obtained by the block-wise histograms of binary codes Chan et al. (2015), Wang et al. (2016):

$$T_i^l = \sum_{\ell=1}^{L_2} 2^{\ell-1} H\left(I_i^l * W_\ell^2\right), l = 1,2,\ldots L_1$$

The function H binarizes these outputs. For each of the L_1 images $T_i^l, l = 1,2,3,\ldots,L_1$ partitioned into B blocks, that size is $k_1 k_2 \times B$, and compute the $2^{L_2} \times B$ histogram matrix in each block ranging from $[0, 2^{L_2} - 1]$, followed by vectorising the matrix into a row vector $Bhist(T_i^l)$. Finally, concatenate the $Bhist(T_i^l)$ of $T_i^l, l = 1,2,3,\ldots,L_1$ as the feature

$$f_i = \left[Bhist\left(T_i^1\right), \ldots, Bhist\left(T_i^{L_1}\right)\right]^T \in R^{\left(2^{L_2}\right) L_1 B}$$

The model parameters of PCANet include the patch size k_1, k_2, the filters number L_1, L_2, the number of stages and the block size for histograms.

2.5.19 Non-linear Manifold Learning Techniques

In this section non-linear manifold techniques are described. Manifolds created from hand gesture images are not linear, thus a non-linear technique to reduce the dimensionality is needed.

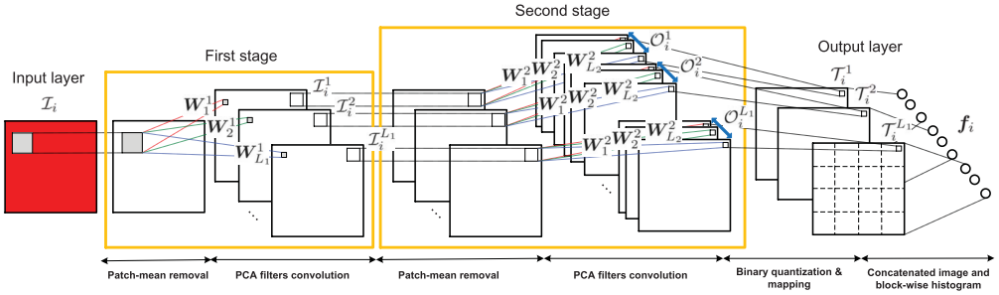


FIGURE 2.21: Architecture of PCANet with 2 stages of PCA instead of convolution layers, proposed by Chan et al. (2015)

2.5.19.1 Non-linear Principal Component Analysis

Non-linear Principal Component Analysis (NLPCA) provides a mapping between low and high dimensional spaces and it is a generalisation of standard Principal Component Analysis that turns PCA into a non-linear technique for dimensionality reduction. NLPCA extracts the principal components as curves instead of straight lines. It helps in visualising the non-linear data as an aspect of data analysis by mapping the data from the original space to a lower dimensional space using an artificial neural network Barth et al. (2008) and Scholz et al. (2008).

NLPCA extends PCA by replacing the linear encoder and decoder by non-linear functions, e.g. neural networks. It optimises NN in an autoencoding setup. The goal of the autoencoder is to learn a representation for a set of data. The embedded manifold appears just implicitly as the decoded image of the input space Barth et al. (2008). In addition, auto-encoding is known as autoassociative, replicator network, bottleneck or sandglass type network Scholz et al. (2008).

The encoder extracts features from the images, and the decoder does the opposite, i.e. reconstruct the images from the features Gutmann et al. (2008).

The auto-encoding network performs an identity mapping. The output $\hat{x}$ is equal to the input x with high accuracy. It is achieved by minimising the squared reconstruction error shown in Equation 2.6 Scholz et al. (2008).

$$E = \frac{1}{2} \left| |\hat{x} - x| \right|^2 \tag{2.6}$$

This is important because there is a *bottleneck* in the middle: a layer with fewer units than in the input or output layer. Therefore, the data have to be projected into a lower dimensional representation Z. The network is composed by two parts: the first, represents the extraction function, and the second represents the inverse function. The hidden layer in each part enables the network to fulfil non-linear mapping functions. These hidden layers allow PCA to perform non-linearity Scholz et al. (2008).

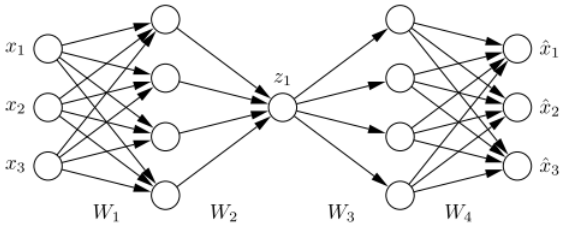


FIGURE 2.22: Auto-encoding Neural Network example

Figure 2.22 shows that the output $\hat{x}$ and the input x are equal. The architecture shown is 3-4-1-4-3 where a data of three-dimensions are compressed to a component z in the middle. The second and fourth layers (four non-linear units) enable the network to fulfil non-linear mappings Scholz et al. (2008).

Licciardi et al. (2014) have applied NLPCA to detect and classify Synthetic Aperture Radar (SAR) images. They compared NLPCA with other techiniques and NLPCA provides the best accuracy. The accuracy of recognising water, buildings and vegetation from satellite images was minimum 86.82% and maximmum 91.67% for different datasets.

2.5.19.2 Kernel Principal Component Analysis

Kernel Principal Component Analysis (Kernel PCA or KPCA) is a non-linear extension of PCA in the sense that it can carry out PCA in feature spaces of arbitrary large dimension. KPCA is used in this thesis and it will be discussed in Chapter 5.

Algorithm 1 Laplacian Eigenmaps

Precondition: Data in high dimensionality

1: constructing the graph, either by choosing points within a radius or computing the k nearest neighbour
2: choosing the weights of the edges, either by heat kernel or by simple-minded
3: eigenmaps: compute the eigenvalues and eigenvector for the generalised problem

Licciardi et al. (2015) compared NLPCA and KPCA among other techniques for forecast in ground horizontal irradiance (GHI) from satellite images. As expected, both non-linear methods (NLPCA and KPCA) provided similar results. They perform better than approaches such as independent component analysis (ICA) and the classic PCA. The accuracy metric used was mean square error (MSE) format, and NLPCA and KPCA showed the best results for the years 2011 and 2012.

2.5.19.3 Laplacian Eigenmaps

Laplacian Eigenmaps are a local approach for non-linear reduction of dimensionality. This approach tries to preserve the local geometry of the data by approximating each point on the manifold with a linear combination of its neighbours, and using the same weights to compute a low-dimensional manifold Saxena and Gupta (2004).

Given k points $t_1, \ldots, t_k$ in n-dimensional space, Laplacian Eigenmaps starts by constructing a weighted graph with k nodes and a set of edges connecting neighbouring points. The first step is constructing the graph, where the neighbourhood map can be constructed by finding the k nearest neighbours or by picking points within a fixed radius. The second step is to choose the weights, either by heat kernel $W_{ij} = e^{-\frac{\left\|x_1 - x_j\right\|^2}{t}}$ or by simple-minded $W_{ij} = 1$ if and only if vertices i and j are connected by an edge. The final step is to build the eigenmap by computing the eigenvalues and eigenvectors $Ly = \lambda Dy$, where D is the diagonal weight matrix sums of W, $D_{ji} = \sum_j W_{ji}$. $L = D - W$ is the Laplacian matrix Belkin and Niyogi (2001).

Algorithm 1 shows the pseudo-code proposed by Belkin and Niyogi (2001).

Figure 2.23 shows a comparison between classical PCA (right) and Laplacian Eigenmaps (middle) for vertical and horizontal bar images(left). Dots (blue) correspond to vertical bars and + (red) signs correspond to horizontal bars.

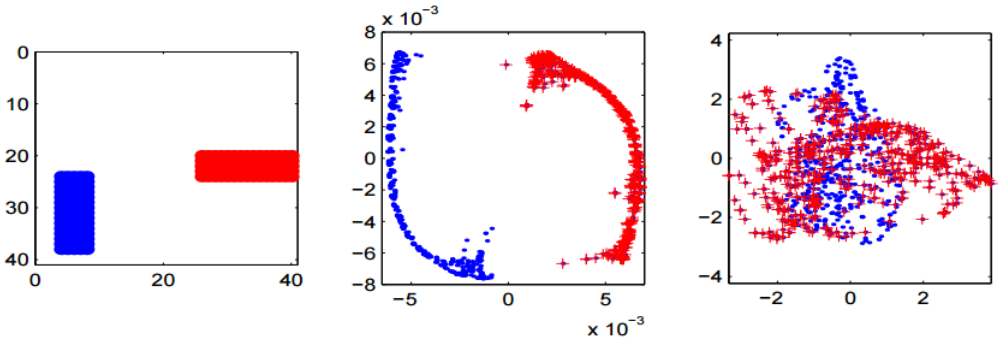


FIGURE 2.23: Comparison between PCA and Laplacian Eigenmaps for vertical and horizontal bar images

Wu et al. (2013) applied Laplacian Eigenmaps for visual object recognition. The accuracy for recognising objects like aeroplanes, bicycles, birds, boats, etc. was 64.1%, slightly above the other techniques compared for the benchmark datasets.

2.5.19.4 Locally Linear Embedding

Such as in Laplacian Eigenmaps, Locally Linear Embedding (LLE) is a local approach and is applied to non-linear dimensionality reduction. It maps the high-dimensional data into a single global coordinate system in a way that preserves the relationships. Algorithm 2 shows the steps for compute this reduction Teng et al. (2005).

LLE consists of finding a set of nearest neighbours of each point. Thus, computes a set of weights for each point that best describe the point as a linear combination of its neighbours. Finally, it uses an eigenvector-based optimisation technique to find the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbours Roweis and Saul (2001).

The input data has D dimensions and the goal of LLE is reducing the dimensionality to d. The W_{ij} weights are the same that reconstructs the data points i in the input D. This data point is

Algorithm 2 LLE algorithm

Precondition: Data in high dimensionality

1: compute the neighbours of each data point $\overrightarrow{X_i}$
2: compute the weights W_{ij} that best reconstruct each data point $\overrightarrow{X_i}$ from its neighbours, minimising the cost by constrained linear fits
3: compute the vectors $\overrightarrow{Y_i}$, best reconstructed by the weights W_{ij} minimising the quadratic form by its bottom nonzero eigenvectors

used to reconstruct the same point in the lower d dimensional space. Finally, a neighbourhood preserving map is created. Each point X_i in the D dimensional space is mapped onto a point Y_i in the d dimensional space, see Algorithm 2 Roweis and Saul (2001).

Figure 2.24 shows an examples of manifolds in 3-dimensions being reduced to 2-dimensions. The black mark in B and C show the neighbourhood of a single point Roweis and Saul (2001).

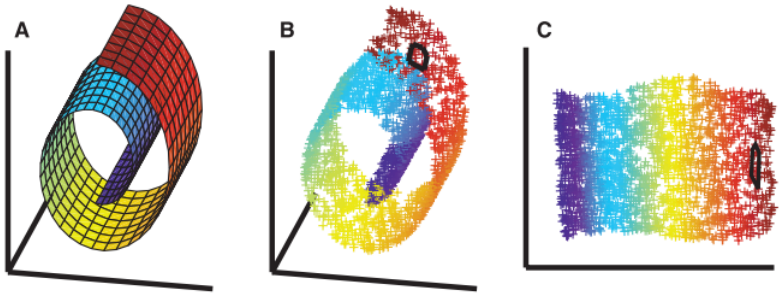


FIGURE 2.24: Example of swiss roll manifold with dimensionality redu by LLE

Elgammal and Lee (2007) have applied LLE for synthesis, pose recovery, reconstruction and tracking moving objects. The accuracy measured for pose recovery was over 5 people dataset, and it was shown 93.05% accuracy for a single frame and 99.63% when five frames were used for each person.

Teng et al. (2005) used LLE to extract features and create a hand gesture recognition for Chinese Sign Language. The system used only a regular webcam and was able to recognise 30 gestures with 92.2% accuracy.

Algorithm 3 Isomap algorithm

Precondition: Data in high dimensionality

1: determine the neighbours of each point
2: construct a neighbourhood graph
3: compute shortest path between two nodes
4: compute lower-dimensional embedding
5: multidimensional scaling

2.5.19.5 Isometric Feature Mapping

Isometric feature mapping (Isomap) is a global method for non-linear dimensionality reduction. It is basically an extension of MDS that creates a pairwise geodesic distance matrix D_G from the input data samples $\overrightarrow{X_i}$. First a neighbourhood graph G is determined. Thus the shortest paths in G are computed for all pairs of data points. In other words, Isomap attempts to preserve the global relations, mapping close points on the manifold to close points, and far points to far points in the low-dimensional space. It is predicted to find an improved representation of data's global structure Raytchev et al. (2004).

Different algorithms can be applied to compute the shortest path, such as Floyd–Warshall's algorithm and Dijkstra's algorithm. k-Nearest Neighbour is the most common used for determining the neighbour of each point. Algorithm 3 shows an Isomap algorithm proposed in Saxena and Gupta (2004).

Figure 2.25 shows a dimensionality reduction by Isomap algorithm for a *fishbowl* dataset and for a manifold of face images. k refers to the number of nearest neighbours as in k-NN, in this example it was set to 15 Silva and Tenenbaum (2003).

Raytchev et al. (2004) proposed to apply Isomap for 3D view representation and head pose estimation. They compared the average angle error of three mehtods: Linear subspace, Locality Preserving Projections (LPP) and Isomap. Considering only 8 dimensions Isomap showed the lowest error rate.

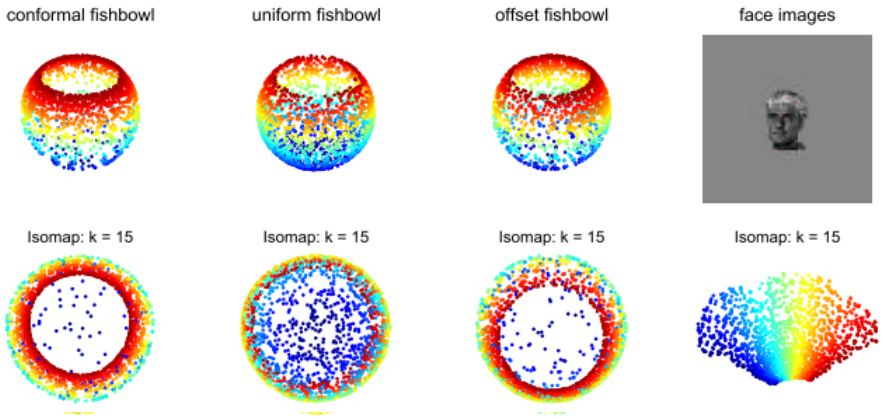


FIGURE 2.25: Example of dimensionality reduction for different manifolds by Isomap

2.5.19.6 Comparison between Non-linear Methods

Ham et al. (2004) states that Isomap, Laplacian eigenmaps and LLE are similar techniques. All three approaches have common characteristic in the way that they first induce a local neighbourhood structure on the data, and use this local structure to globally map the manifold to a lower dimensional space. These are basically extensions of Kernel PCA, with different kernel matrices. The Dijkstra shortest path distance between the points is used in the kernel matrix of Isomap approach; commute times for Laplacian eigenmaps kernel; and a specially constructed graph operator for LLE kernel.

Elgammal and Lee (2007) compared Isomap and LLE and obtained qualitatively similar results in the manifold embedding.

Figure 2.26 shows a comparison between Isomap (B), Laplacian eigenmaps (C) and LLE (D) for a set of 600 points (A) sampled from the S-curve manifold. For this example k was set to 6 for the number of nearest neighbours Ham et al. (2004).

2.5.20 Interpolation

Interpolation is a technique that aims to construct new data points within the range of a set of known points. There are different methodos for computing an interpolation. The most common are linear interpolation and spline interpolation. Linear interpolation uses a linear

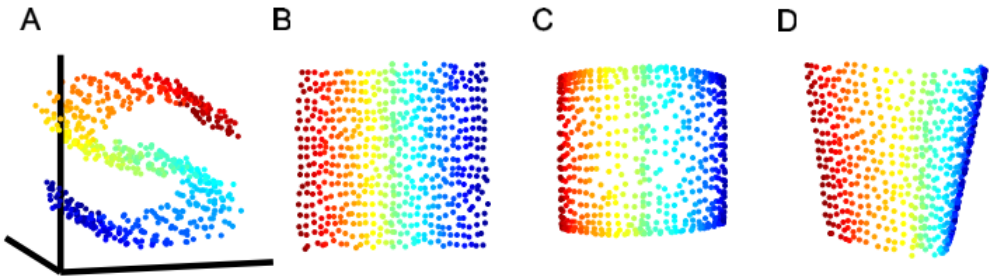
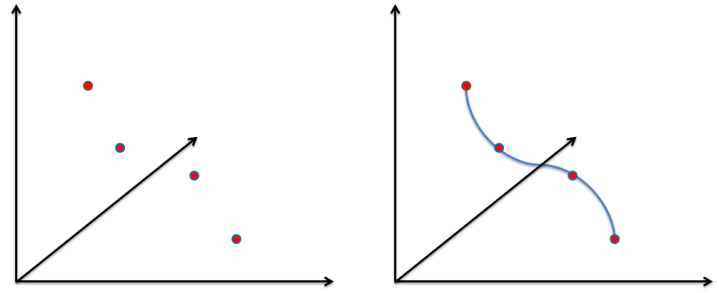


FIGURE 2.26: Comparison between Isomap, Laplacian Eigenmaps and LLE dimensionality reduction

function for each interval, therefore it can only interpolate straight lines. Whereas, splines use low-degree polynomials in each of the intervals (piece-wise polynomial) Johnson et al. (1993).



(a) Points in red are the original point data

(b) The blue curve connecting them is the interpolation

FIGURE 2.27: Illustration of interpolation in 3 dimensions

Figure 2.27 shows an interpolation in 3D using splines. In 2.27(a) red dots are the original data and in 2.27(b) the blue curve is the interpolation. Interpolation is possible in any number of dimensions. In D'Errico a Matlab function able to interpolate in any dimension and with different methods has been launched.

In Elgammal and Lee (2007) radial basis function interpolation has been successfully used for creating 3D body poses.

2.6 Discussion of the Literature Review

In this chapter, the literature review and the related work were presented. Hand gesture recognition has improved over the last few years. New technologies have been applied trying to solve the challenges in the area. However, old theories are still relevant in the recognition process.

Glove-based approaches seem not to have a high relevance nowadays. Wearing a glove or any other device does not allow a smooth interaction with computers. However, it is still on-going research. The advantage of wearing a glove with sensors is the accuracy and the robustness to illumination changes.

Vision-based techniques are more intuitive for the user. It allows gesture recognition from images and from videos without the need of a device attached to the arm or hand.

Some of a large number of techniques for vision-based hand gestures recognition have been explained in this literature review. Most of the vision-based approaches use a depth sensor, or infrared, or even more than one camera. There are fewer researches in recognition using only one standard camera.

For instance Farouk (2015) proposed 4 algorithms for handshape recognition applied to ISL. However, all the algorithms are based on similar ideas and over a small dataset. Most of the work in Farouk (2015) is done with computer-generated images of 20 handshapes and in all algorithms only PCA is used with different classification techniques. Farouk (2015) did not used any kind of interpolation or deep classifier nor any non-linear manifold learning algorithm.

There is a strong belief that non-linear manifold learning algorithms provide improved results for handshape recognition. In addition, deep learning is trending nowadays in the CV field. The efficacy of this method has been proved in a large number of research articles. However, it is still computationally expensive. In contrast PCA is computationally cheaper. The question is which is the most appropriate technique for ISL handshape recognition?

The purpose of this thesis is to carry out research on hand gesture recognition using only one regular camera. Thus, in future it could potentially be used for any kind of image or video, e.g. taken from the Internet or a recording made by any mobile device, such as smartphones. Finally, a study and application of PCA, non-linear PCA, deep learning and other techniques is shown.

Chapter 3

Irish Sign Language Dataset

This chapter presents a new dataset for Irish Sign Language (ISL) followed by a frame redundancy filtering.

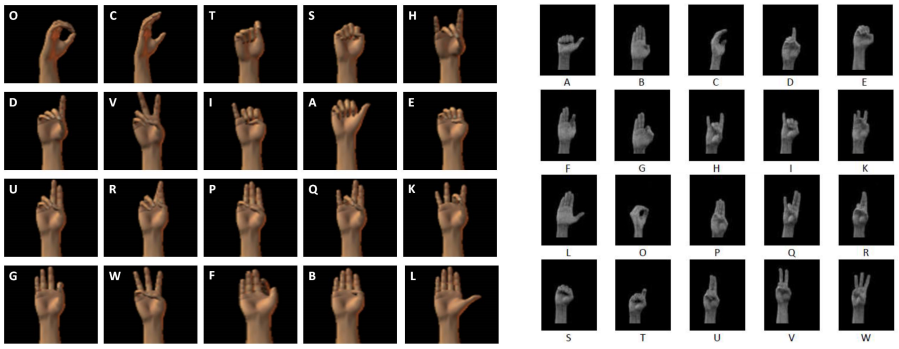
3.1 Introduction

In this chapter a new dataset for Irish Sign Language (ISL) is introduced. The dataset contains real hand images for the 23 most common ISL handshapes. Compared to previous works on ISL, the proposed dataset is larger and contains all handshapes. In addition, a filter is proposed in order to avoid redundancy and discard frames with high similarity.

Earlier works in this area have used rather smaller datasets. For instance, Farouk (2015) proposed two ISL datasets. The first dataset is composed of computer-generated images, produced by the Poser software by SmithMicro; the total number of images is 920. The second dataset is composed of real hands, and has a total of 1,620 images. Both datasets represent only 20 ISL handshapes (excluding 'M', 'N' and 'Y and the dynamic shapes 'J', 'X' and 'Z'), see Figure 3.1.

The number of real hand images of the dataset proposed in this thesis has been increased from 1,620 to 52,628 with 6 different human subjects. Three shapes were added to the 20 used

by Farouk. Finally, an iterative method is designed in order to select the images that keep the dataset diverse by removing redundant frames.



(a) Computer generated (b) Real hands

FIGURE 3.1: Datasets for ISL created by Farouk (2015) with 20 static gestures

Apart from Irish Sign Language there are other datasets available for sign language recognition. Quesada et al. (2017) state that there are more than 137 different sign languages around the world. However, this thesis will focus only on ISL.

Zheng et al. (2017) have presented some datasets for Sign Language recognition. Some examples of datasets available are:

- *American Sign Language Lexicon Video Dataset* (ASLLVD) with more than 3,000 signs, from 1 to 6 signers and around 9,800 tokens Neidle et al. (2012)
- *MSR Gesture 3D dataset* with 12 dynamic shapes for ASL, 10 signers, recorded 2 or 3 times each Wang et al. (2012)
- *Auslan Signbank* with about 7,797 sign words and 26 finger spellings, each sign is repeated 5 times Signbank (2017)
- *LTI-Gesture Database* was created by the Chair of Technical Computer Science at the RWTH Aachen, containing 14 dynamic gestures in videos Ney et al. (2005)
- *RWTH German Fingerspelling Database* with 35 gestures representing the letters of the alphabet, German umlauts, and the numbers from one to five, 20 different signers and 2 times each, totalizing 1160 images Dreuw et al. (2006)

- *DEVISIGN (Chinese)* with 26 letters and 10 numbers performed by 8 different subjects, 4 subjects and 2 times each with a single camera Chai et al. (2014)

- *Indian Sign Language dataset*, designed for Leap Motion and Kinect sensors, contains about 7,500 sign gestures, being 50 dynamic sign gestures, done by 10 signers, repeated up to 15 times each, consists of both single and double hand dynamic sign gestures (28 words single hand and 22 were performed using both hands) Kumar et al. (2017)

Ronchetti et al. (2016) proposed a dataset for handshape recognition of Argentinian Sign Language (LSA) called *LSA16*. The LSA16 handshape dataset is freely available. It consists of 16 different handshapes of the most common in the LSA, with n 10 subject performing 5 different poses of each handshape, with a total of 800 frames. The interpreters wore coloured gloves to facilitate the segmentation. Quiroga et al. (2017) compared different CNN architectures for the task of handshape recognition for LSA.

Sections 3.2.1 and 5.2.1.2 were developed in collaboration with Dr. Houssem Chatbri, who is the second author in the paper titled: "Irish Sign Language Recognition Using Principal Component Analysis and Convolutional Neural Networks", where this work has been published Oliveira et al. (2017a).

3.2 The Irish Sign Language Handshape (ISL-HS) Dataset

The ISL-HS dataset contains real hand images, unlike synthetic images used in previous works. ISL-HS is composed of 23 handshapes combined with different motions.

To build the dataset short videos were recorded from six people (3 males and 3 females) performing the finger spelling ISL handshapes. Each shape was recorded 3 times in front of a dark background. Videos recorded only the arm and the hand of the subject. The arm is useful to help to detect the rotation angle. The dark background was chosen to avoid the need for segmentation, since that is not the focus of this thesis.

Each of the 23 handshapes was performed by moving the arm in an arc from the vertical to the horizontal position. This was performed to simulate rotated handshapes that can occur in real word conversations. For the 3 motion gestures 'J', 'X' and 'Z' there was no rotation, only the motion indicated in Figure 3.2. All the handshapes in the dataset, apart from the 3 with motion, are rotated in a plane.

The videos were converted into frames. Frames were converted to grayscale and the background was removed from the frame using a pixel-value threshold. This produced frames containing only the arm and the hand.

The number of frames for each video depends on the time taken by the human subject to perform the gesture. Videos were recorded at 30 frames per second (fps) and a resolution of 640×480 pixels. The device used to record the videos was an Apple iPhone 7. The videos were saved with *.mov* extensions. The video format is RGB24.

The illumination sources were a combination of natural and artificial. Therefore, the important point is that the effect of the illumination (shades) on the appearance of the arm changes according to the position of the arm. Videos were recorded in a laboratory for post-graduate computing students. Illumination was different for each person, because they were recorded at different times of the day and on different days.

In total 468 videos were recorded. From these videos a total of 58,114 frames were obtained, consisting of 52,688 frames for the rotated shapes and 5,426 for the 'J', 'X' and 'Z'. Figure 3.2 shows cropped images of the ISL-HS dataset, and Figure 3.3 shows the class distribution across the image dataset. The variation observed in Figure 3.3 is due to the speed variation among the subjects when performing the ISL handshapes and rotating them. Note that the letter '*X*' has the lowest number of frames because this is a dynamic feature with a short motion.

The dataset was released online[1] and provides both videos and images.

[1] https://github.com/marlondcu/ISL

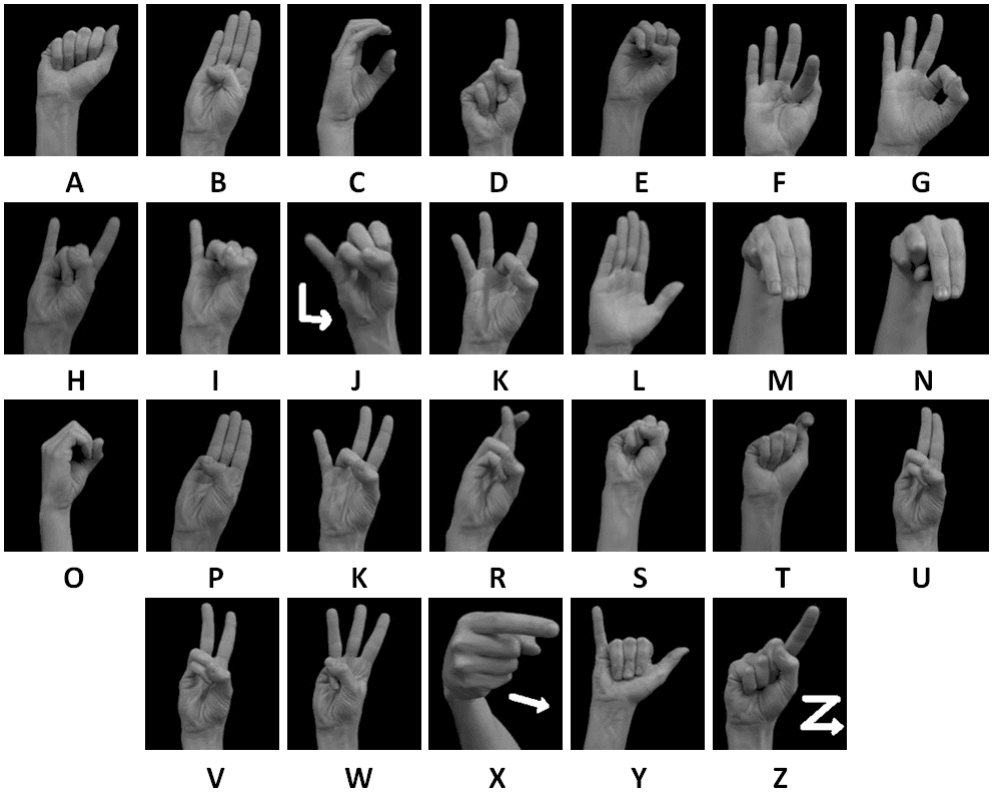


FIGURE 3.2: Cropped images from the new Irish Sign Language handshapes alphabet

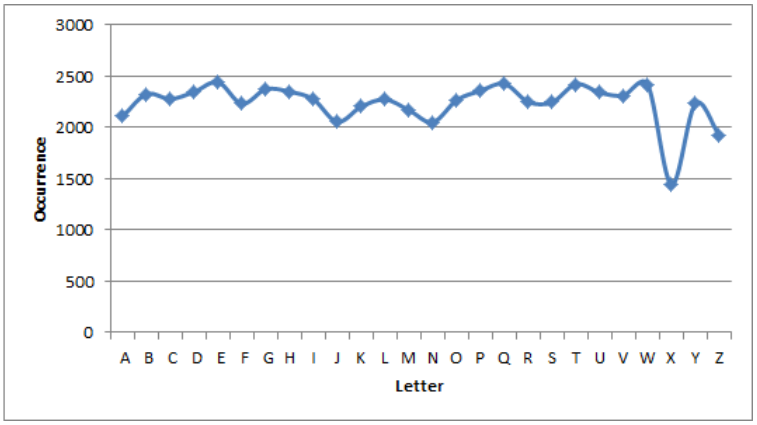


FIGURE 3.3: Ocurrence of the different handshapes in the new ISL dataset

3.2.1 Redundant Frame Filtering

Since all frames were extracted from the videos, numerous frames are similar due to the speed variation of the subjects. For this reason, a method to filter redundant frames was designed (i.e. frames with insignificant difference). The method works as follows: Each image I_u of

the original dataset is represented with a compact feature vector $\overrightarrow{V_u}$ (Section 3.2.1.1). Then, a *diversity score* is introduced to express image heterogeneity, and images are iteratively selected to optimise the *diversity score* (Section 3.2.1.2). This method of selecting frames was inspired by Geiger et al. (2012).

3.2.1.1 Feature Extraction

From each image I_u of the original dataset, a feature vector $\overrightarrow{V_u}$ is extracted by splitting the image into a $K \times K$ grid and calculating the number of foreground pixels in each cell of the grid after edge detection with a 3×3 Laplacian convolutional kernel Nahar and Ali (2014) as following: (Figure 3.4):

$$\phi = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.1}$$

Therefore, the feature vector $\overrightarrow{V_u}$ for an image I_u is a 2D histogram and it is expressed as follows:

$$\overrightarrow{V_u} = (V_{i,j}), 0 \leq i, j \leq K \tag{3.2}$$

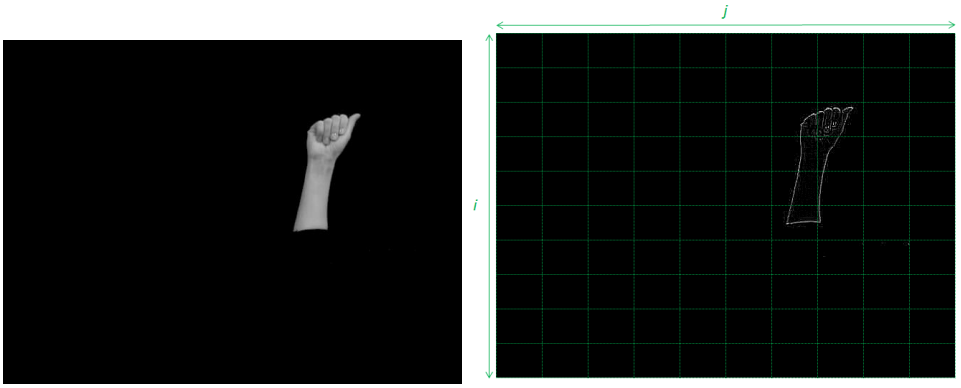
where $V_{i,j}$ denotes the number of foreground pixels in bin (i, j) divided by the total number of foreground points.

Thus, computing the dissimilarity between two images I_u and I_v is done by accumulating the differences between the histogram bins of their feature vectors $\overrightarrow{V_u}$ and $\overrightarrow{V_v}$ as follows:

$$d(\overrightarrow{V}, \overrightarrow{V_i^A}) = \frac{1}{K^2} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} (V_{i,j}^u - V_{i,j}^v)^2 \tag{3.3}$$

where d is the distance between two feature vectors. The dissimilarities between the histogram bins is amplified by adding the square distances between every two corresponding

histogram bins, $V^u_{i,j}$ and $V^v_{i,j}$. Therefore, small differences in the image details are penalised. Other distance measures could be used as well (e.g. Earth Mover distance Shu and Wu (2011)). However, they might be slower to compute, which is important in this step (Section 3.2.1.2).



(a) (b)

FIGURE 3.4: (a) a sample image from the ISL-HS dataset, and (b) the feature extraction grid

3.2.1.2 Iterative Image Selection using the Diversity Score

After extracting a feature vector $\overrightarrow{V_u}$ from each image I_u in the original dataset, an iterative selection process is performed to select images based on a *diversity score* that is defined as follows:

$$\gamma(I,A) = \frac{1}{N_A} \sum_{i=0}^{N_A-1} d(\overrightarrow{V}, \overrightarrow{V_i^A}) \tag{3.4}$$

where A is an image set that contains N_A images (originally $N_A = 0$), I is an image from the original dataset and that not belong to A, $\overrightarrow{V}$ and $\overrightarrow{V_i^A}$ are the feature vectors of I and I_i^A respectively, $\gamma(I,A)$ is the *diversity score* of one image I to images in the set A.

The iterative process is illustrated in Algorithm 4. First, an image is selected randomly from the original image dataset DB_{orig} and put into the final dataset A. Thus, an iterative process selects the image from D_{orig} for all images N that has the largest *diversity score* (i.e. the image that is the most different to images of A), where *diversity score* is calculated with

Algorithm 4 Diversity-based image selection

Precondition: Original dataset DB_{orig} : containing all N images

function DIVERSITYBASEDSELECTION(DB_{All})
    A: final dataset, initially empty
    I : one frame selected randomly from DB_{All}
    $A \leftarrow \{I\}$
    for $i \leftarrow 1$ to N **do**
        $k_{max} \leftarrow \underset{0 \leq k \leq |DB_{orig}|-1}{\text{argmax}} \gamma(I_k, A)$
        $I \leftarrow I_{k_{max}}$
        $A \leftarrow \{I\}$
        Remove I from DB_{All}
    end for
end function

Equation 3.4. When this process finishes, The curve of *image diversity score* at each selected image is shown in Figure 3.5.

In order to perform this process in a reasonable time, it was tweaked by applying the iterative image selection from a subset of D_{orig} with a limited size equal to 100 images selected randomly, instead of the whole D_{orig}. During the process, it was observed that this does not lead to a selection bias. However, it makes the process achievable in a reasonable time.

After plotting the curve of Figure 3.5 which shows the plot of the curve of *image diversity score* at each selected image, the first 50,000 images are selected (i.e. roughly just before the sharp decrease in *diversity score*). Figure 3.6 shows the class distribution in this dataset. The dataset is available online as well [2].

Table 3.1 shows the distribution of number of frames by person.

3.2.2 Dense Dataset

The *dense* dataset consists of the 50,000 frames after filtering and contains images for the 23 most common static handshapes for ISL. This dataset is taken as the final dataset used for all experiments in Chapter 5.

[2] https://github.com/marlondcu/ISL_50k

TABLE 3.1: Frame distribution (occurrence) per person

Person	No of frames
1	7,958
2	8,284
3	8,530
4	8,180
5	9,134
6	7,914

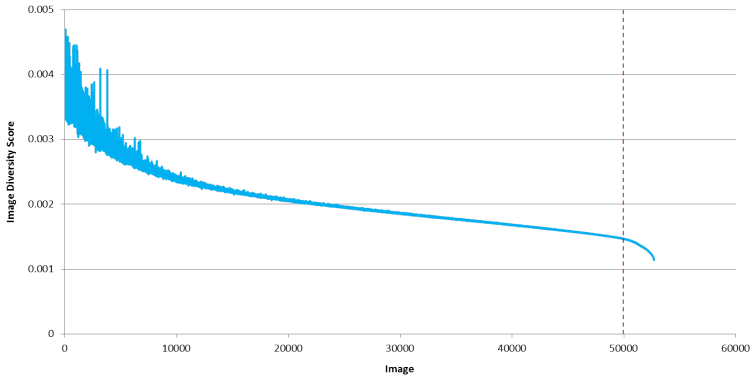


FIGURE 3.5: The curve of image diversity score, note the steep descent after 50,000 frrames

3.2.2.1 Variations of the Dataset

Two sets of testing and training dataset are proposed. The training N_{train} and testing N_{test} datasets contain 25,000 images each.

The first dataset is created by iterating through the images and assigning every image to either the training or the testing set in an alternating manner, and it is called DB_i; in this dataset all frames labels are sorted by person, shot and frame number. Then, in a sequential way the first frame is selected as training, the second frame as testing, the third frame as testing and so on.

The second set is created by random selection algorithm and it is called DB_r. The random selection algorithm is used to avoid overfitting Scikit. It is common practice when performing a supervised machine learning experiment to hold out part of the available data as a test set. The $scikit-learn$ python code is used to create a random split into training and testing.

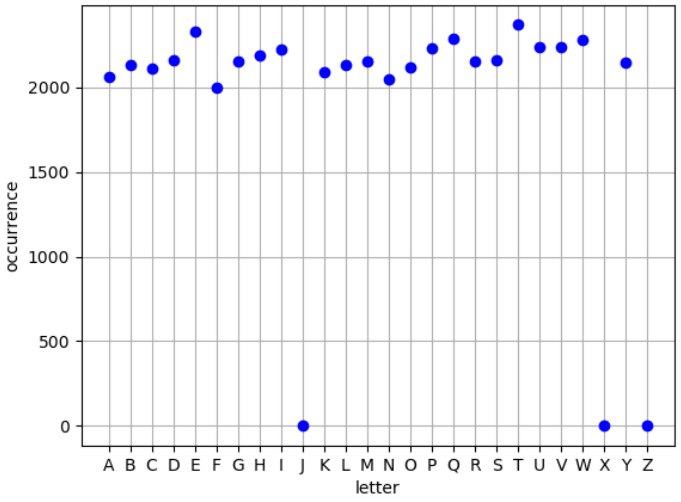


FIGURE 3.6: Rate of occurrence by shape, after filtering (50,000 frames)

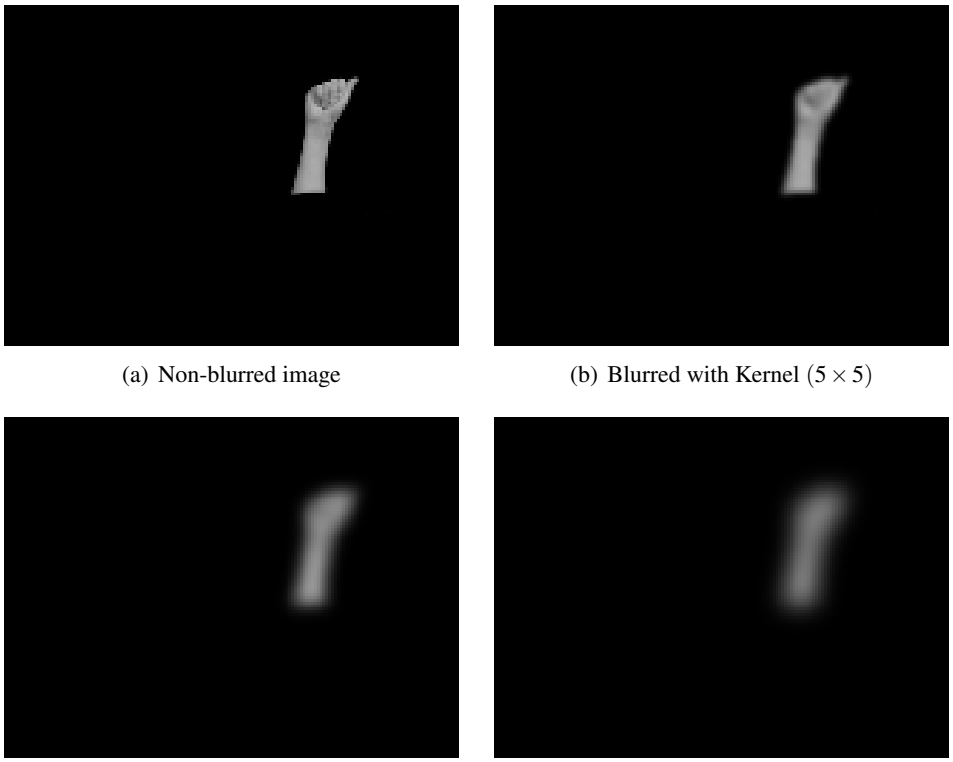
Each image in the dataset has 640×480 pixels originally. However, in order to manipulate them in a CPU all images were resized to 160×120 pixels.

3.2.2.2 Blurred Images Dataset

Images were blurred with two-dimensional Gaussian blurring in order to test how the classifiers behave on blurred images. This was motivated by earlier results by Farouk (2015), which showed that such image filtering is beneficial for PCA accuracy. Therefore, it will be shown how other approaches respond over blurred images. In this stage, a kernel of different sizes were used and the standard deviation is computed according to $\sigma = 0.3 * ((ksize - 1) * 0.5 - 1) + 0.8$. Kernel sizes were tested as $ksize$ = 5, 15 and 25. These datasets are called DB_b. Figure 3.7 shows a non-blurred image followed by the different kernel sizes used.

3.2.2.3 Different Persons Dataset

Finally, the last variation in the dataset is removing persons from the training set and using them in the test stage. As the full dataset is composed by 6 persons, three possibilities were tested. The first was removing one person out of the training set then using five to train and



(a) Non-blurred image

(b) Blurred with Kernel (5×5)

(c) Blurred with Kernel (15×15)

(d) Blurred with Kernel (25×25)

FIGURE 3.7: Images of shape A, non-blurred and blurred with Gaussian filters of different kernel sizes

the test stage was with only one person. For this specific case the training was done with every third image, because of memory availability and it is called DB_1. The second was removing two persons out of the training, then using 4 to train and 2 to test, this dataset is called DB_2. The third dataset was removing 3 persons out of the training using then 3 to train and 3 to test, this dataset is called DB_3. This leads to different numbers of images for N_{train} and N_{test}, because the number of frames depends on each person. Section 5.3.4 will show the results for these experiments and the number of N_{train} and N_{test} for each case.

3.2.3 Sparse Dataset

The *sparse* dataset has certain translations and rotations removed and so can be considered a *sparse* dataset. The *sparse* dataset is used in all experiments in Chapter 4.

TABLE 3.2: Gaussian filter blurring levels with different kernel sizes

Blur level	Kernel size	Standard deviation
1	[24,24]	40
2	[36,36]	60
3	[48,48]	80

The entire dataset was recorded from 6 persons and each video was recorded 3 times. The following steps were taken to select the *sparse* dataset from the videos of the 23 static shapes of ISL.

- Select one person (person 3)
- Select the longest video out of 3 for each shape
- Extract the frames
- Count how many frames and select the middle one
- Select 40 frames on each side of the middle and label them
- Convert to greyscale

Finally, a dataset with $N_{shape} = 23$ shapes and 80 frames for each shape is ready for working with PCA. Note that these 80 images are only labeled in order to represent the motion of the arm over a range of 90 degrees, the label does not represent the exact angle, it is only illustrative. The middle image was labeled as 40 and the other ones decreasing or increasing, depending on the side. Figure 3.8 shows cropped images of the dataset.

In order to make the algorithm more robust to translation, each image was translated plus and minus 5 pixels horizontally and vertically, creating $N_{tra} = 121$ images for each handshape. In total there are N_{im} images for each rotation angle, where $N_{im} = N_{shape} \times N_{tra} = 2,783$.

In the *sparse* dataset only 3 levels of blurring were tested, because it was not the focus of this work, given by the fact it was previously explored by Farouk (2015). The blurring level are shown in Table 3.2.

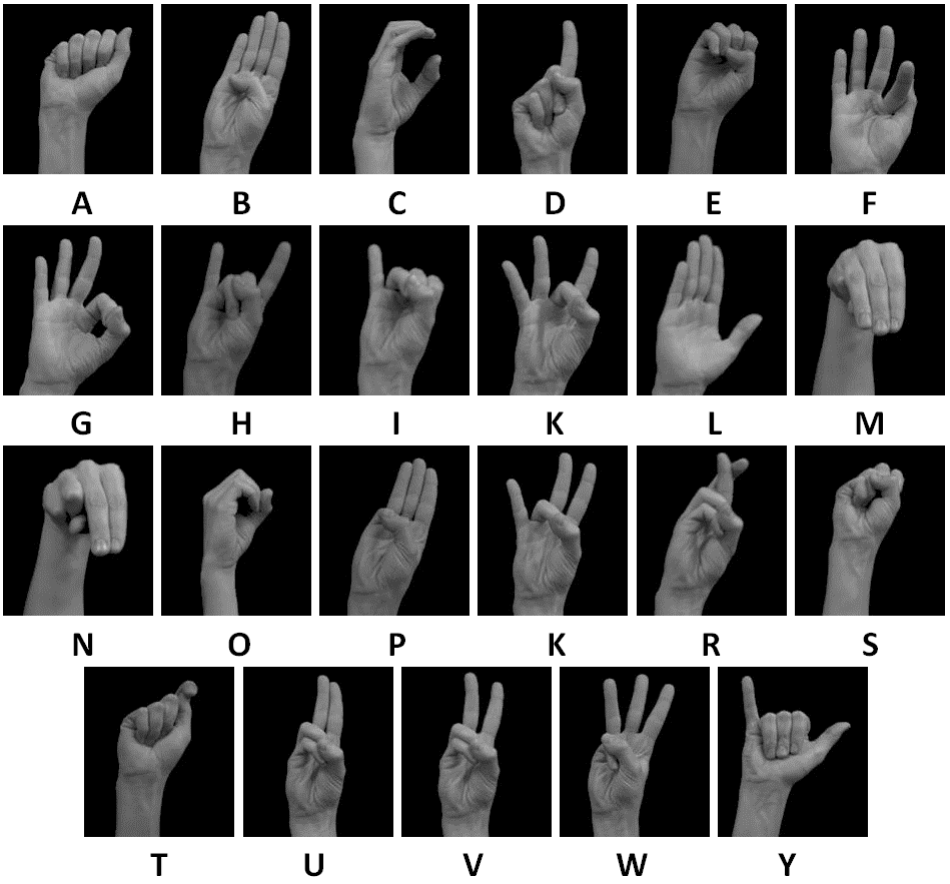
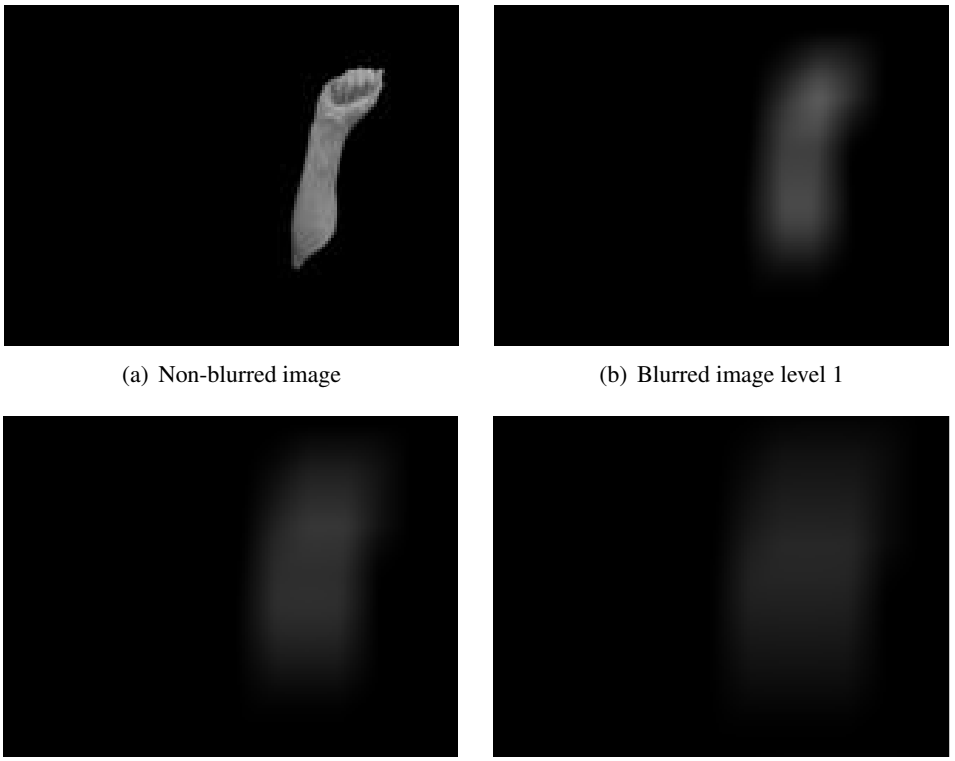


FIGURE 3.8: Cropped images from the Irish Sign Language static handshapes

Each image in the dataset has 480×640 pixels. In this stage, each of these images are resized to 80×60 pixels, in order to manipulate these images in a personal computer. Figure 3.9 shows the shape for the letter *A* in greyscale before and after blurring with different kernel sizes.

3.2.4 Training Dataset

The training dataset consists of N_{im} images for each rotation angle. This dataset contains only the images labelled in the range from 0 to 80 at intervals of d. Thus, 0 is considered the most vertical image and 80 the most horizontal one. The size of the interval d was chosen to be 6, 8 or 10.



(a) Non-blurred image

(b) Blurred image level 1

(c) Blurred image level 2

(d) Blurred image level 3

FIGURE 3.9: Images of shape A, non-blurred and blurred with different kernel sizes, see Table 3.2

For $d = 6$ the selected images were labeled as 0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72 and 78, resulting in 14 different rotation angles $N_{ang} = 14$. For $d = 8$, angles labels are 0, 8, 16, 24, 32, 40, 48, 56, 64, 72 and 80, providing $N_{ang} = 11$. Finally, for $d = 10$ angles are labeled as 0, 10, 20, 30, 40, 50, 60, 70 and 80 giving $N_{ang} = 9$. The total of images in the training dataset is given by $N_{train} = N_{im} \times N_{ang}$, then for $d = 6$ $N_{train} = 2,783 \times 14 = 38{,}962$, for $d = 8$ $N_{train} = 2,783 \times 11 = 30{,}613$ and for the last $d = 10$ $N_{train} = 2,783 \times 9 = 25{,}047$. Results for how d influences the accuracy will be shown in Section 4.3.

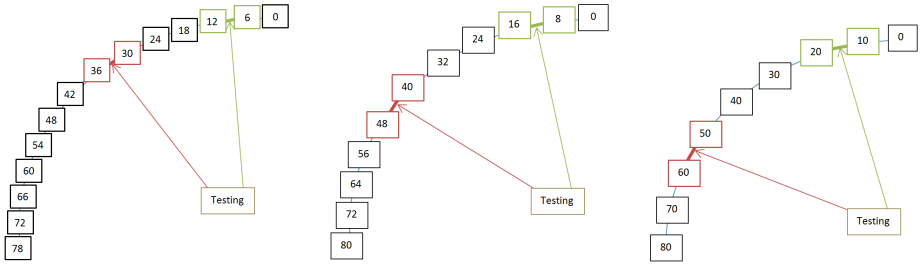
3.2.5 Testing Dataset

In the testing dataset images of 1 frame apart were selected. The test images are selected from the original set in between two pairs of consecutive images from the training set. Therefore, two testing datasets are constructed in order to avoid testing within all the intervals in the

training dataset (for computational reasons). This means, when $d = 6$ images of the testing dataset are selected from the range 7 to 11 and from the range 31 to 35; for $d = 8$ images are labeled selected from the range 9 to 15 and from 41 to 47; for $d = 10$ images are selected from the range 11 to 19 and from 51 to 59.

The number of images in the testing dataset depends on the value of d, such as $N_{test} = d \times N_{im}$ e.g. for $d = 6$ $N_{test} = 6 \times 2,783 = 16,698$, $d = 8$ $N_{test} = 8 \times 2,783 = 22,264$, and finally for $d = 10$ $N_{test} = 10 \times 2,783 = 27,830$.

Figure 3.10 shows an illustration of the training and testing datasets according to different values of d. The squares represent the training sets and the testing sets are in two of the intervals between the squares.



(a) $d = 6$ (b) $d = 8$ (c) $d = 10$

FIGURE 3.10: Illustration of the training and testing datasets according to d, note each d produces a different number of subsets; two subsets are used for testing for each d

Chapter 4

PCA for Sign Language Recognition

In this chapter, experiments with Principal Component Analysis are presented. All the experiments are made over a real hands dataset called a *sparse* dataset, presented in Chapter 3.

4.1 Introduction

As stated in Chapter 1 handshape recognition is a hard task because hands are deformable objects. Computer Vision (CV) plays an important role in helping Deaf people to communicate. Sign language is basically a visual language. Thus, CV is exactly the computing area that can help whose who need assistance to improve their lives.

As shown in Chapter 2 there are different approaches to address this problem. From this high number of techniques a few of them have been chosen to be applied in this thesis to the new Irish Sign Language dataset proposed in Chapter 3.

PCA is a widely used technique for dimensionality reduction and feature extraction Jolliffe (2014). It means that a huge amount of data can be represented by fewer dimensions of these data. It makes tasks such as image processing easier, since images have a considerable amount of data. In addition, PCA allows images to be projected into eigenspaces, creating

manifolds. Manifolds can be manipulated, and interpolation can be used in order to create artificial manifolds which might represent projections of data not present in the training set.

Experiments in this chapter are done over the sparse dataset, presented in Chapter 3, Section 3.2.3.

In this chapter a deeper study on PCA applied to ISL is proposed. PCA is applied over from the ISL *sparse* dataset and interpolation is proposed to make this dataset more robust to missing translations and rotations. Two-stage PCA is used to facilitate the interpolation. At the second stage, PCA is applied to subsets of the data from the first stage. This helps to reduce the dimensionality further, the main motivation for using PCA in more than one stage is that reducing the data dimensionality even more makes the interpolation a faster process since it is made with fewer data.

If interpolation was tried in the first-stage PCA space, where each point represents a single image, then interpolation would have to be done over 50,000 points. Whereas, if created a second stage PCA space, where each point represents a different subset of points in the first-stage space (e.g. in this case each subset represents all the possible 121 translations of single handshape) then it dramatically reduce the number of points to be interpolated (in this case down 431). This produces a great increase in efficiency. There is an additional increase due to the fact that the second-stage PCA space tends to have even fewer dimensions than the first-stage space. See Section 4.2.2 and Figure 4.4.

4.2 Principal Component Analysis

A common problem when working with any kind of images is the huge quantity of data. For that reason it is important to have a technique to reduce its dimensionality. Images are multidimensional data, where each image is represented by a point in an N-dimensional space where N is the number of pixels in each image. To analyse and apply recognition over this enormous amount of data is quite computationally expensive. Principal Component Analysis

(PCA) can be used for feature extraction and dimensionality reduction Han and Liu (2014); Jolliffe (2014). In addition, PCA is known as the discrete Karhunen–Loève transform (KLT).

Basically PCA replaces the original set of data with a reduced set of derived data. Frequently it is possible to represent most of the variability in the original data with a smaller number of variables.

Observe in Figure 4.1 an illustration of the PCA process in 2D. The mean values of the variables are represented by μ_x and μ_y. u_1 and u_2 present the directions of the greatest variation in the data. It is very clear that the most variance occurs in u_1 direction, and the second in the y_2 direction. The same idea applies for any number of dimensions in space.

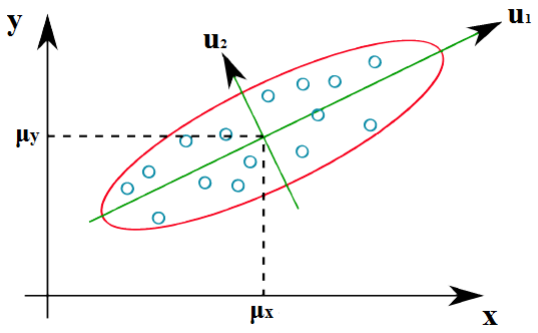


FIGURE 4.1: Illustration of an eigenspace showing the direction of 2 eigenvectors

The space created by PCA is called an eigenspace. Each dimension in an eigenspace is called an eigenvector. Each eigenvector is referred to as a principal component. In the case of the Figure 4.1 the eigenvectors are represented by u_1 and u_2. Every eigenvector is associated with an eigenvalue, which is equal to the variance of the data along the eigenvector. In Figure 4.1 the eigenvalues are represented by the widths squared of the ellipse along the two eigenvectors. In addition, the eigenvectors are orthogonal to each other.

The number of these principal components is less than or equal to N. The eigenvectors are ranked in order of the size of their eigenvalues which tend to zero at some point. This is particularly the case if images are related to each other, as handshapes are. Figure 4.2 shows a typical plot of the eigenvalues in decreasing order. It is very important to know how

many eigenvectors to use to obtain a satisfactory representation of the dataset. The goal is to select a set of eigenvectors that gives a good representation of the data. This set consists of those eigenvectors whose eigenvalues are significantly greater than zero. Eigenvectors whose eigenvalues are close to zero carry none or minimal information about the data.

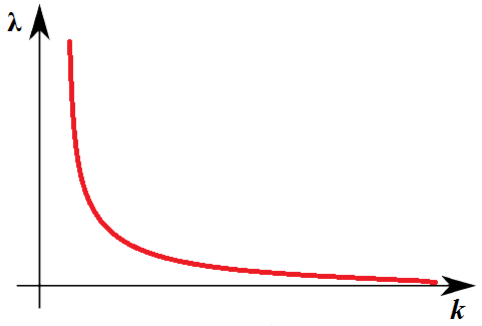


FIGURE 4.2: Eigenvalues in descending order, each eigenvalue represents one eigenvector, the higher the eigenvalue the higher its significance

In the case where the data are images, the input consists of a set of those images. Each image is vectorised and then concatenated to create a matrix of images where each row contains one image. In Equation 4.1 x_1 represents the first pixel of the image, x_2 the second, until x_N which is the last pixel.

$$X = \{x_1 x_2 x_3 \ldots x_N\} \tag{4.1}$$

Thus, each X represents one image. Since there is more than one image, they can be put all together creating a matrix such as Equation 4.2, where N_{im} is the number of images:

$$Z = \begin{pmatrix} X_1 \\ X_2 \\ \cdots \\ X_{N_{im}} \end{pmatrix} \tag{4.2}$$

Z is the input to PCA. After PCA has been computed, the output is the original data represented with respect to the eigenvectors, which are found from the covariance matrix. The covariance matrix is an $N \times N$ matrix computed according to Equation 4.3.

$$A = Z^T Z \tag{4.3}$$

After computing the covariance matrix, eigenvalues and eigenvectors can be calculated according to Stockman and Shapiro (2001), Equation 4.4.

$$Au_k = \lambda_k u_k \tag{4.4}$$

where u_k is the k_{th} eigenvector and λ_k is the k_{th} eigenvalue. If A is an NxN matrix then there will be N eigenvectors and N eigenvalues. Each vector u_k will have N components. The eigenvalues λ_k are scalars. U is a matrix in which each column is an eigenvector u_k.

PCA has two separate functions - encoding and decoding - to transform the observed input to the Principal Component (PC) space and then back to the observed space. PCA performs linear computation over the input data Z to form a representation H that has a lower dimensionality than the data. This stage is known as linear encoding. The action of transforming a data H in the lower dimensional space back to the input space Z is known as decoding, which results in a reconstruction Z Kambhatla and Leen (1997). H is a matrix where each row represents the coordinates of a point in the PC space.

Equation 4.5 shows how to project data Z onto eigenvectors U, the encoding stage.

$$H = ZU \tag{4.5}$$

The procedure represented in Equation 4.5 is equivalent to computing the scalar product of the data with each of the eigenvectors, this process is known as encoding. Figure 4.3 shows an example of data projected onto 3 dimensions (3 eigenvectors). Each eigenvector represents

a different mode of variation in the set of data. This will be discussed in detail in this thesis in the Section 4.2 and the Section 5.2.2.1.

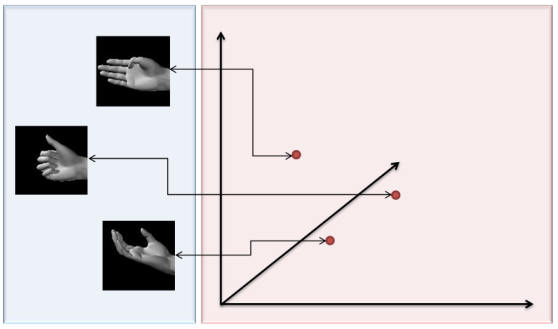


FIGURE 4.3: Example of data projected into a PCA space in 3 dimensions

Equation 4.6 shows how to reconstruct data Z from the reduced dimensional data H and eigenvectors U, at the decoding stage. In addition, this process is known as "back projection" and will be discussed again in Section 4.2.4.

$$Z = HU^T \tag{4.6}$$

PCA concepts can be used for handshape recognition, where the original data are images of hands. Thus, the question is, given a new image, which shape from the original dataset is the closest? The way this is done is by finding the distance between the new image and images in original dataset. However, instead of being along the original axes, the distance is calculated along the new axes derived from the PCA.

In this section experiments with Principal Component Analysis (PCA) are shown. PCA is applied in more than one stage in order to make interpolation feasible.

Algorithm 5 creates a global eigenspace where each image is represented by a point; Algorithm 6 creates a second eigenspace where each point represents a subset of points from the first eigenspace; Algorithm 7 interpolates between points in the second eigenspace, in order to represent missing rotation angles. This algorithm uses back projection in order to recreate the manifolds of the missing angles in the first stage, it is an important step because

Algorithm 5 PCA over training dataset

Precondition: Images in the training dataset

```
for each angle do
    for each handshape do
        for each translation do
            [I1] ← vectorise image of the training dataset and add to [I1] matrix
        end for
    end for
end for
S1 ← PCA over [I1]
P1 ← projection of [I1] into the eigenspace S1 with D1 eigenvectors
```

the classification is made in the first stage PCA and the second stage is used only in order to interpolate data; Algorithm 8 creates a subspace for each rotation angle in the training dataset; Algorithm 9 creates a subspace for each rotation angle and interpolates in between its manifolds; Algorithm 10 creates a subspace for each rotation angle in the training dataset and interpolates in between eigenvectors; finally Algorithm 11 interpolates between translations in order to fill gaps due to missing ones.

4.2.1 First Stage PCA - Global Eigenspace

The first step in this approach is to combine all the images $[I_1]$ from the training dataset into the same matrix and then compute PCA. Since each image has 60×80 pixels when vectorised it becomes 4800 pixels in a vector. As a result, a 4800×4800 covariance matrix is obtained. By applying PCA to the covariance matrix an eigenspace S_1 with 4800 eigenvectors is created.

By projecting the images $[I_1]$ from the training set into the most significant D_1 eigenvectors, a D_1-dimensional space S_1 is obtained, containing N_{im} points for each rotation angle. Each point represents an image. Figure 4.4(a) shows 2 dimensions (axes) of these D_1 where each point represents one image of the training dataset. Different values for D_1 will be discussed in Section 4.3. Algorithm 5 shows the steps to compute PCA over the training dataset.

Algorithm 6 Second stage PCA to represent points in P_1

Precondition: The points P_1 representing the training images projected into first stage PCA

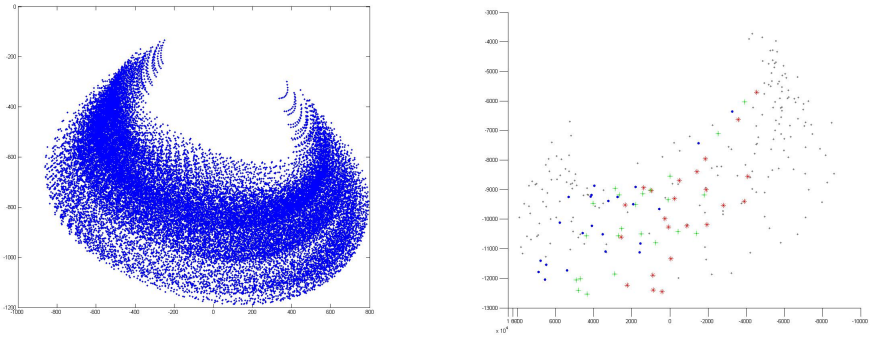
1: **for** each angle **do**
2: **for** each handshape **do**
3: $[I_2] \leftarrow$ concatenation of the $D_{tra} \times D_1$ coordinates into a vector and add to $[I_2]$ matrix
4: **end for**
5: **end for**
6: $S_2 \leftarrow$ PCA over $[I_2]$
7: $P_2 \leftarrow$ projection of $[I_2]$ into the eigenspace S_2 with D_2 eigenvectors

4.2.2 Second Stage PCA - Subsets of the First-stage

At the second stage a new eigenspace S_2 is constructed where each point in the space represents a subset of the points P_1 in the first stage PCA. Each subset consists of the set of points for a particular handshape at a particular rotation angle in the training dataset. In other words, each point represents the translation manifold for a particular handshape at a particular angle. These points will then be used in Section 4.2.3 to interpolate between angles.

The members of the input dataset are the translation manifolds in the first stage eigenspace for each handshape for each rotation angle. The pseudocode is shown in Algorithm 6. Each shape has N_{tra} images at each rotation angle. Hence each handshape at each rotation angle is now represented by $[I_2]$ matrix $N_{tra} \times D_1$, where D_1 is the number of coordinates. The final step is to compute the PCA for these new data. As a result a new eigenspace S_2, as shown in Figure 4.4(b), is obtained. Different values for D_2 will be discussed in Section 4.3.

In Figure 4.4(b) each point represents the translation manifold of a handshape at a particular angle. The points fall into clusters, each of which represents the set of handshapes at one particular angle. Figure 4.4(b) contains the same information as Figure 4.4(a). However, it is much less cluttered. Points represented as $+$ (green) $*$ (red) and $\bullet$ (blue) highlight 3 different angles in the first stage PCA. In Figure 4.4 each axis represents one dimension out of D_2 dimensions in each stage.



(a) Projection of training images I_1 into the first-stage PCA S_1

(b) Projection of points I_2 into the second-stage PCA S_2

FIGURE 4.4: Projection of training dataset images into the first-stage S_1 and second stage-PCA S_2, note that the same data is represented with fewer points when projected into S_2

Note that for all the different algorithms presented in this chapter, each only represents 2 levels of PCA. D is used as the number of dimensions (eigenvectors), i.e. D_1 is the number of dimensions in the first stage PCA and D_2, D_3 and D_4 are the number of dimensions in the second stages of respectively Algorithms 6, 8 and 9. In this thesis D_2, D_3 and D_4 are all set to the same value.

4.2.3 Interpolating Angles in the Second Stage PCA

In order to interpolate subspaces, a curve in a space of any dimensionality is created by applying splines. Splines are applied in the second-stage PCA projection P_2 in D_2 dimensions. Therefore, the interpolation is done in D_2 dimensions. Figure 4.5 shows the original and interpolated data in the second stage PCA, in between groups of three neighbours.

Having a spline interpolated between one angle and another it is possible to determine any angle in between. Note in Figure 4.5, a projection of the eigenspace for the second level PCA, points represented as $\bullet$ (blue), $+$ (green) and $*$ (red) highlight the three different manifolds in the first stage PCA and the grey dots are the interpolated points between those angles.

The pseudocode for interpolation is shown in Algorithm 7. Note that the number of points n determines how many points is going to be interpolated in between training angles N_{ang}. In

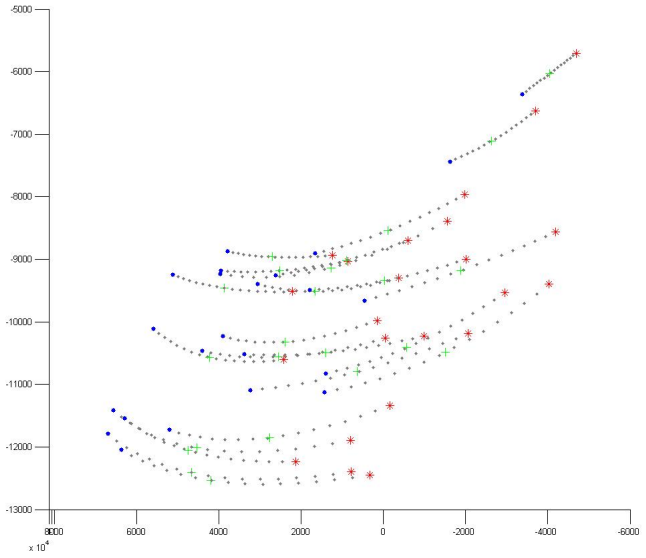


FIGURE 4.5: Missing angles interpolated by splines, red, green and blue symbols represent the landmark points and grey points the interpolated data, each axis represents one dimension D_2

Algorithm 7 Pseudocode for angle interpolation by splines

Precondition: Points P_2 in the second stage PCA

1: **for** each shape **do**
2: $P_{2(inter)} \leftarrow n$ points interpolated by a spline over the points (angles) representing that shape
3: $P_{1(back)} \leftarrow$ back projection of $P_{2(inter)}$ to S_1
4: $P_{1(inter)} \leftarrow$ reshape of $P_{1(back)}$
5: **end for**

this case n depends on d, e.g. $d = 8$ gives $N_{ang} = 11$, as from labels 0 to 80 there are 80 points, then n is set to 80; in case of $d = 6$ n is set to 78, see Figure 3.10.

4.2.4 Back Projection from the Second Stage PCA to the First

From the coordinates of any point in the second stage PCA P_2 or $P_{2(inter)}$ it is possible to reconstruct the corresponding manifold in the first stage PCA. Equation 4.7 shows the back projection process of computing the dot product of each coordinate of the point P_2 or interpolated point $P_{2(inter)}$ with the corresponding set of eigenvectors of the second-stage PCA D_2.

Algorithm 8 Second stage PCA for Rotation Subspaces

Precondition: The points representing the training images projected into first stage PCA P_1

1: **for** each angle i **do**
2: $[I_3^i] \leftarrow$ subset of N_{im} points from P_1 corresponding to the angle
3: $S_3^i \leftarrow$ PCA over $[I_3^i]$
4: $P_3^i \leftarrow$ projection of $[I_3^i]$ into the eigenspace S_3^i with D_3 eigenvectors
5: **end for**

This is an example of the decoding process which was mentioned in Section 4.2 in Equation 4.6.

$$P_{1(back)} = P_{2(inter)} U_2^T \tag{4.7}$$

where U_2^T are eigenvectors of S_2. By reshaping $P_{1(back)}$ back to P_1 shape, artificial manifolds of the first stage PCA are obtained as $P_{1(inter)}$. Figure 4.17 shows examples of back projected manifolds.

Different values of D_2 (number of eigenvectors) used in the back projection (reconstruction) affect the quality of the reconstruction of the manifolds. The quality of the reconstruction improves as D_2 increases. Different values of dimensions D_2 will be tested and discussed in Section 4.3.

4.2.5 Second Stage PCA - Rotation Subspaces

In order to have subspaces for each rotation angle it is necessary to compute PCA separately for each angle. Algorithm 8 shows the pseudocode for rotation subspaces. Note that at this stage, one subspace for each group of shapes and translations at the same rotation angle will be created (S_3^i where i represents the number of the subspace).

Rotation subspaces will be used to compute perpendicular distance from a new image to the subspaces, in order to find the interval in which the new image lies, see Section 4.3.1. In addition, rotation subspaces are used to interpolate data and eigenspaces, Section 4.2.7 and 4.2.6 show algorithms for this case.

Algorithm 9 Pseudocode for rotation sub-manifolds interpolation by splines

Precondition: Projection of points into the second stage PCA P_3^i

for each handshape **do**
    $[I_4] \leftarrow$ concatenation of $N_{im} \times D_3$ dimensions in a vector and add in $[I_4]$ matrix
end for
$S_4 \leftarrow$ PCA over $[I_4]$
$P_4 \leftarrow$ projection of $[I_4]$ into the eigenspace S_4 with D_4 eigenvectors
for each handshape **do**
    $P_{4(inter)} \leftarrow n$ interpolated points along the spline over the points representing that shape
end for
for each interpolated point in $P_{4(inter)}$ **do**
    $P_{4(back)} \leftarrow$ back projection of $P_{4(inter)}$ to S_3^i
    $P_{3(inter)^i} \leftarrow$ reshape of $P_{4(back)}$ to $D_3 \times D_3$ matrix
end for

4.2.6 Interpolating Rotation Sub-manifolds

In a similar manner to that shown in Section 4.2.2, it is possible to interpolate new manifolds in between rotation manifolds of the second stage PCA. These new manifolds consist of artificial data recreated through interpolation by splines. The number of dimensions in the artificial manifold depends on D_4.

Algorithm 9 shows how manifolds can be interpolated at the second stage. In order to interpolate between rotation manifolds P_3 the first step is to vectorise each of P_3^i creating one vector for each set of shapes and translations at each rotation angle and add to a matrix called $[I_4]$; the second step is to apply PCA over this matrix creating a new space S_4; the third step is project $[I_4]$ into S_4; the fourth step is to interpolate between points for all D_4 eigenvectors creating $P_{4(inter)}$; the fifth step is to back project this $P_{4(inter)}$ into S_3^i creating $P_{4(back)}$; finally reshape these back projected vectors back to a $D_3 \times D_3$ matrix called $P_{3(inter)}$.

The number of points n determines how many points are going to be interpolated in between training angles N_{ang}. In this experiment $d = 8$ is used, therefore $N_{ang} = 11$. As from labels 0 to 80 there are 80 points, n was set to 80.

$P_{3(inter)}$ consist of n new manifolds, one for each rotation angle label, from 0 to 80.

Algorithm 10 Pseudocode for eigenspace interpolation by splines

Precondition: Eigenvectors of the second stage PCA S_3^i

for each subspace in S_3^i **do**
    $S_{3(mat)^i} \leftarrow$ vectorise each eigenspace with $D_3 \times D_3$ elements and add to $S_{3(mat)^i}$
end for
$S_5 \leftarrow$ PCA over $S_{3(mat)^i}$
for each i in $S_3(mat)^i$ **do**
    $S_{5(proj)} \leftarrow$ projection of $S_{3(mat)^i}$ into S_5
end for
$S_{5(inter)} \leftarrow n$ points interpolated along the spline over elements of each eigenvector of $S_{5(proj)}$
for each n in $S_{5(inter)}$ **do**
    $S_{5(back)} \leftarrow$ back projection of $S_{5(inter)}$ to S_3^i
    $S_{3(inter)^i} \leftarrow$ reshape of $S_{5(back)}$ to $D_3 \times D_3$ matrix
end for

4.2.7 Interpolating Rotation Sub-eigenspaces

Having N_{ang} rotation subspaces S_3^i, it is possible to interpolate new spaces in between. These spaces consist of artificial eigenspaces recreated through interpolation by splines. The number of dimensions in the artificial spaces depends on D_3.

Algorithm 10 shows how eigenspaces can be interpolated. In order to interpolate between eigenspaces S_3^i the first step is to vectorise each of S_3^i creating one vector for each and concatenating in a matrix, called $S_{3(mat)}$; the second step is to apply PCA over this matrix creating a new space S_5; the third step is to project $S_{3(mat)}$ into S_5; the fourth step is to interpolate between elements for all D_5 eigenvectors creating $S_{5(inter)}$; the fifth step is to back project this $S_{5(inter)}$ to S_3^i creating $S_{5(back)}$; finally reshape these back projected vectors back to a $D_3 \times D_3$ matrix called $S_{3(inter)}$.

The number of points n determines how many points are going to be interpolated in between training angles N_{ang}. In this experiment $d = 8$ is used, therefore $N_{ang} = 11$. As from labels 0 to 80 there are 80 points, n was set to 80.

Figure 4.6 shows N_{tra} real images for the shape A projected into interpolated eigenspace in blue and projected into the real eigenspace in red.

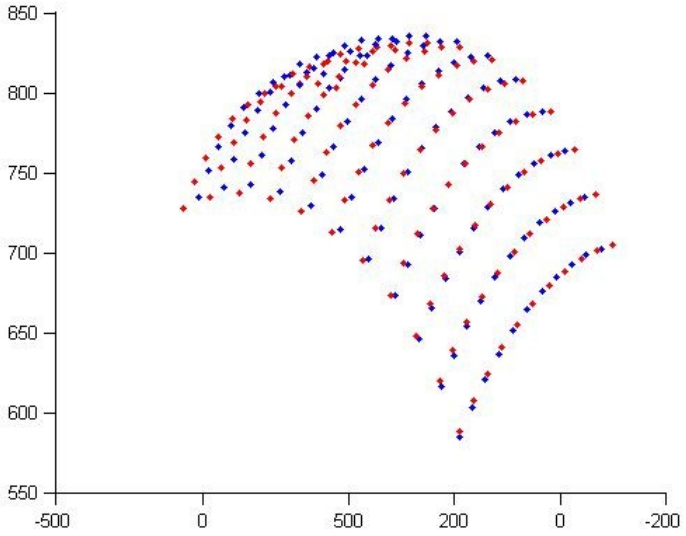


FIGURE 4.6: Projection of images into interpolated space (blue) and into real space (red), note they are very similar

Figure 4.7 shows the projection of 3 elements of the first eigenvector for all spaces. Black dots represent the landmarks (real spaces) and the green dot represent interpolated eigenspaces.

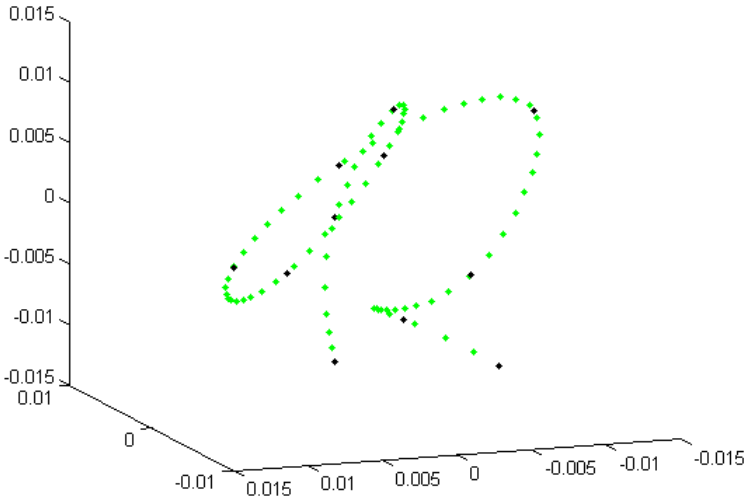


FIGURE 4.7: Interpolation of 3 elements of the first eigenvector, black dots are the landmarks and green dots the interpolated points

4.2.8 Results with Interpolated Rotation Sub-eigenspaces

It is possible to recognise a shape using interpolated eigenspaces $S_{3(inter)}$ and interpolated rotation data $P_{3(inter)}$. The first step is to find the interval where the new shape fits. Section 4.3 shows how to find the interval of a new image. After finding the interval the goal is to find the closest interpolated space. In order to do that, perpendicular distance is used to find the closest interpolated eigenspace in this interval. Them, the new image is projected into the closest eigenspace and k-NN is used to classify the correct shape within the corresponding interpolated rotation manifold.

Figure 4.8 shows the average accuracy for shape classification using interpolated spaces and interpolated data. Different number of eigenvectors and different blurring levels were tested and the best accuracy is 82.83%, for $d = 6$, blurring level equal to 2 and D_3 and D_4 = 17. Note that the minimum number of eigenvectors tested in the second stage is 11 because according to the test which will be discussed in Section 4.4 the minimum number of eigenvectors for a good back projection is 11. The possible reason for this low accuracy is because perpendicular distance does not seem to work well for extremely close subspaces. Nevertheless, blurring level 6 kept the best accuracy. For $d = 6$ n is set to 78 and for $d = 8$ n is set to 80.

The average time to recognise a shape with interpolated eigenspaces and interpolated manifolds is 0.0177 seconds.

4.3 Finding the interval

In this section, for each image in the testing dataset, the closest subspace in the training dataset is classified. The images in the testing dataset lie within two separate intervals of the training dataset. This dataset contains images in two intervals ($d = 8$) out of the total 11. One from 8 to 16 and the other from 40 to 48.

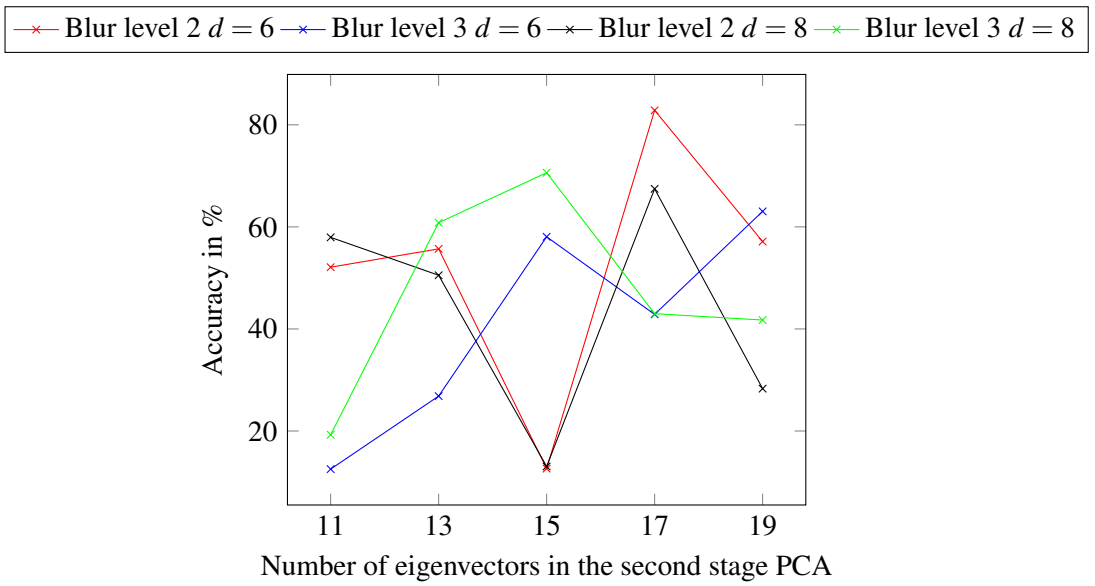


FIGURE 4.8: Accuracy of shape classification within interpolated spaces and interpolated manifolds, according to the number of eigenvectors

All experiments in this section are done using Matlab 2014a, running on a personal computer, with 16GB memory RAM, Intel Core i7-2600 CPU @3.4GHz and Microsoft Windows 7 Professional 64 bits.

4.3.1 Perpendicular Distance

In order to compute perpendicular distance, each subspace S_3 is used and the origin point is being used as the mean of all images at the same rotation angle. Section 2.5.5 shows the equation for perpendicular distance, where v is the subspace, o is the origin image and p is the image to be classified.

The accuracy of one image being classified into the correct subspace can be measured taking the shortest perpendicular distances from this image to each sub-eigenspace S_2. These distances are saved in a vector t sorted from 1 to N_{ang}, in this case 11, each element represents one rotation angle. In this way, when a new image comes in, the closest subspace can be classified, e.g. angle 0 is element 1, angle 8 is element 2 and so on.

It is marked as correct if the images from angle 8 to 16 have the shortest distance to the second or the third sub-eigenspaces and images from angle 40 to 48 to the sixth or seventh sub-eigenspaces.

4.3.1.1 Influence of the Number of Eigenvectors

Figure 4.9 shows the accuracy rate, as a percentage, in identifying the shortest distance. Each set of these images is at an intermediate rotation angle and it shows the mean and standard deviation of the accuracy. The number of images in the test set depends on n. In addition, 2 different intervals were tested, as shown in Section 3.2.5.

This accuracy was measured after computing perpendicular distance of each image against all N_{ang} 11 sub-eigenspaces ($d = 8$) and taking the shortest one. In Figure 4.9 each bar represents different number of eigenvectors used in the first stage PCA (D_1), the x-axis represents the number of eigenvectors used in the second stage and y-axis represents the accuracy as a percentage. Accuracy is computed out of N_{im} 2,783 images for each rotation angle.

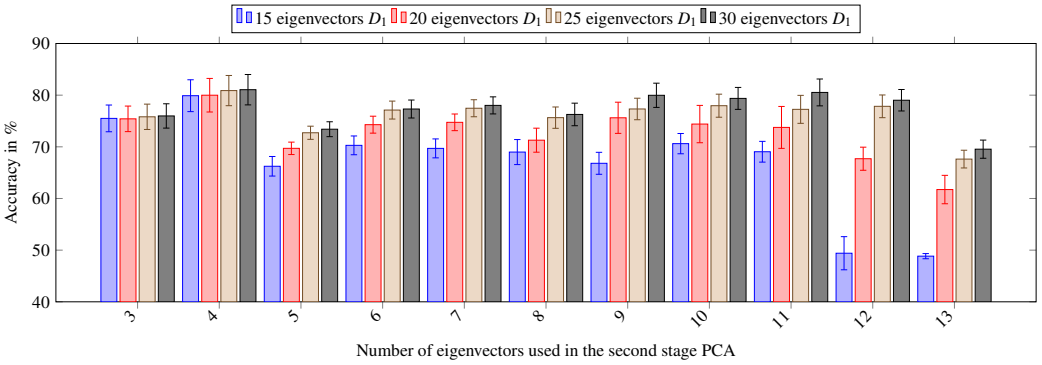


FIGURE 4.9: Accuracy according to different number of eigenvectors for the first and the second-stage PCA

The best accuracy was found with only 4 eigenvectors in the second-stage PCA, e.g. 79.89% for $D_1 = 15$; 79.99% for $D_1 = 20$; 80.88% for $D_1 = 25$ and finally 81.05% for $D_1 = 30$. This is an interesting result, because the combination of the parameters D_i can improve or decrease the accuracy drastically.

4.3.1.2 Time for Perpendicular Distance

The time for classification of the correct interval using perpendicular distance is measured. Figure 4.10 shows the average time to classify one image according to the number of eigenvectors in the first stage PCA. Note that time increases according to the number of eigenvectors in D_1. Time was measured according to the number of eigenvectors in the second stage as well. However, it seems to not have a significant variation. Figure 4.11 shows the average time to classify one image for $D_1 = 25$ and different values for D_3.

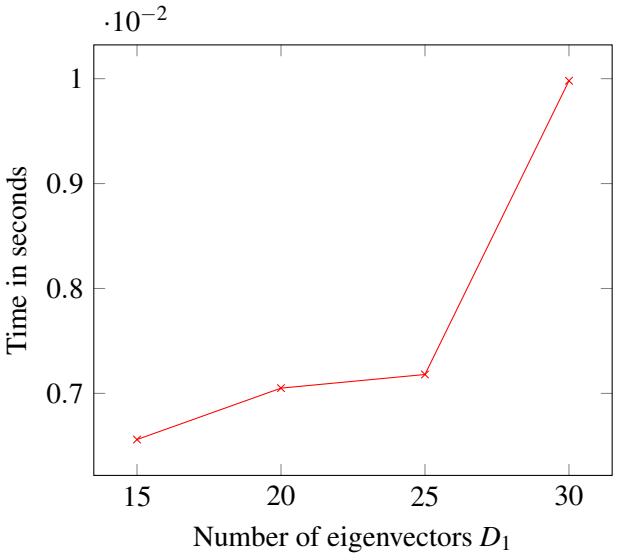


FIGURE 4.10: Average time to classify one image according to the number of eigenvectors in the first-stage PCA D_1

4.3.1.3 Influence of the Interval Size

The size of the interval d affects the accuracy. Accuracy is tested for different values of d (6, 8 and 10). Figure 4.12 shows how accuracy changes according to d. Note that as d increases the accuracy improves, probably because as the further one subspace is from the other the greater is the perpendicular distance.

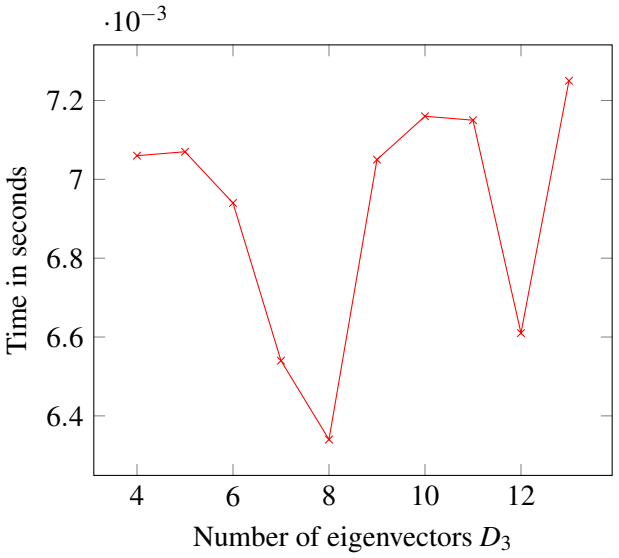


FIGURE 4.11: Average time to classify one image according to the number of eigenvectors in the second-stage PCA, given $D_1 = 25$

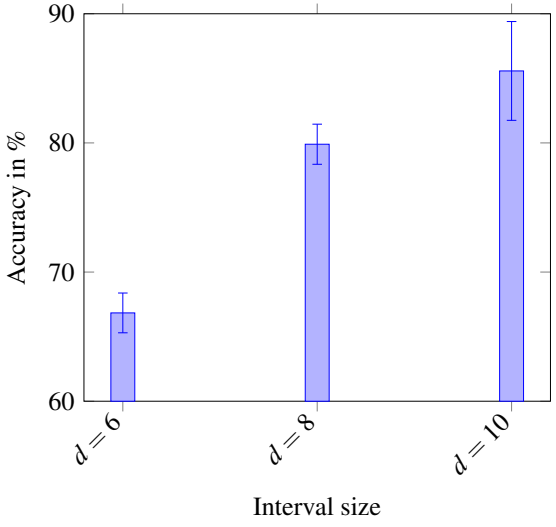


FIGURE 4.12: Average accuracy and standard deviation according to the interval size d, the longer the interval, the higher the accuracy

4.3.1.4 Influence of the Blurring Level

In a similar manner, the level of blurring applied to images influences the accuracy. Figure 4.13 shows how blurring influences the accuracy. Note that a low or a high level seems to not help considerably. However, a medium level tends to improve the accuracy.

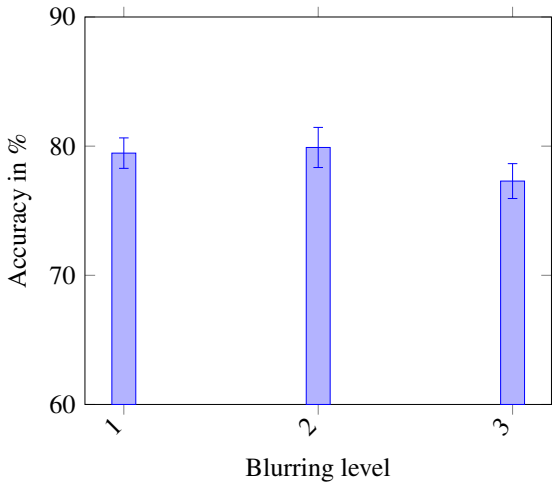


FIGURE 4.13: Average accuracy according to blurring level, level two showed the highest accuracy

4.3.2 k-NN

Another manner of classifying the correct interval for a test image is using k-Nearest Neighbour. As shown in Section 2.5.4 this technique consists of computing the distance from a point to another in any dimension. The most common distance metric used is Euclidean distance. Therefore in this section Euclidean distance is used. In addition, k is set as 1 because only one neighbour is needed to classify one subspace as the closest.

In order to classify the closest angle, Euclidean distance is computed from a point (new image) to a matrix of all other points (P_1) and the algorithm returns the same size matrix with all the distances. For this experiment points in P_1 were labeled according to the rotation angle. The same style as in perpendicular distance method, it is considered correct if the testing image is classified into one of the 2 neighbouring angles. In other words, the distance from the test image is computed to all the images in all of the key angles.

4.3.2.1 Influence of the Number of Eigenvectors and the Blurring Level

Figure 4.14 shows a plot with the relation between the number of eigenvectors used to compute the Euclidean distance and the blurring level. These distances are computed using interval

$d = 8$. However, Section 4.3.2.2 shows how different interval sizes influence the accuracy.

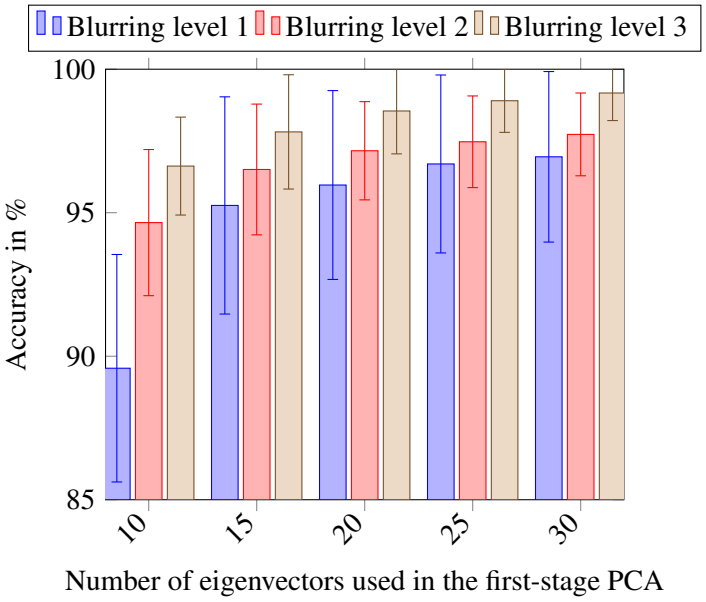


FIGURE 4.14: Accuracy according to the number of eigenvectors and blurring level, using k-NN, for first-stage PCA

4.3.2.2 Influence of the Interval Size

In order to measure how the size of the interval influences the accuracy, Euclidean distance is computed with different values for d. Figure 4.15 shows how the interval size d influences the accuracy of classifying the correct interval. Note that a small value for d gives an improved accuracy. However, the difference in accuracy is near irrelevant among $d = 8$ and $d = 10$. These distances are measured with $D_1 = 30$ and blurring level equal to 2. Finally, note that the the standard deviation increases as d increases, due to the fact that the higher the value of d the further the data in the testing dataset is from the training dataset, specially for the middle ones.

4.3.2.3 Classification Time for k-NN

Figure 4.16 shows the time in seconds to classify the nearest subspace according to the number of eigenvectors used for D_1. Note that speed does not change dramatically, the difference

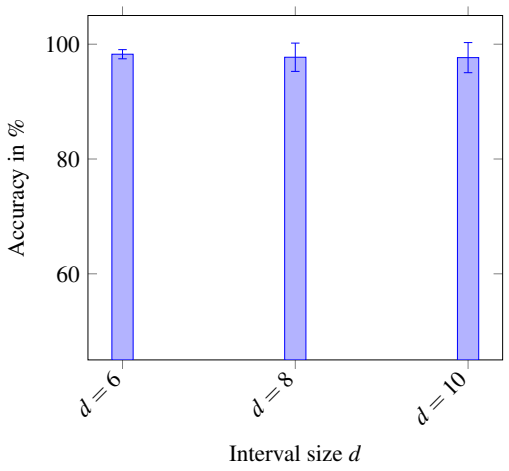


FIGURE 4.15: Average accuracy and standard deviation according to the interval size d, using k-NN, the variation in accuracy is low, varying more the standard deviation

between 10 and 30 eigenvectors is around 0.003 seconds.

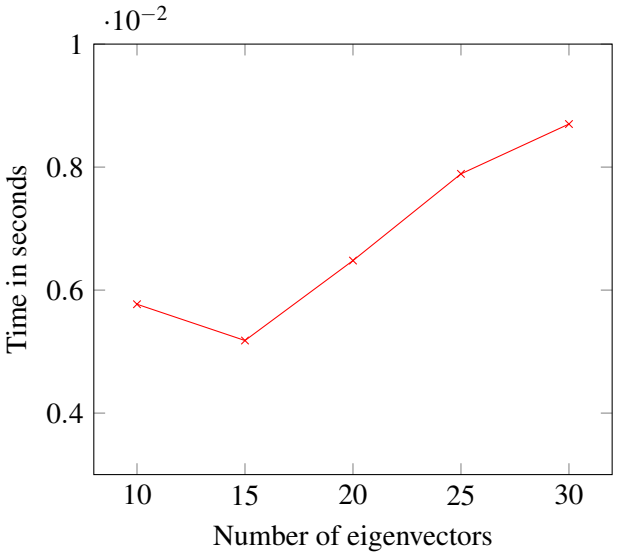


FIGURE 4.16: Average time to find the correct interval for one image by k-NN according to the number of eigenvectors

4.3.3 Conclusion of Results in Finding the Interval

In order to classify the closest subspace S_3^i, perpendicular distance and k-NN were tested. Considering time and accuracy, k-NN with Euclidean distance provided improved results for both. It is probably given by the fact that manifolds are not linear and they overlap each

other, perpendicular distance is a rough measure and seems to not work well for non-linear overlapping manifold. The average time to classify one image was slightly longer for k-NN.

4.4 Back Projection

Back projection of one point in the second-stage PCA P_2 back to first-stage P_1 depends only on the quantity of eigenvectors considered (S_2). Figure 4.17 shows the quality of the back projection according to the number of eigenvectors used, where the blue dots are real projection and the red dots are back projection of points in P_2 back to first-stage P_1. For this experiment only real data was considered. However, the same process applies to interpolated data. Note that less than 13 eigenvectors does not provide a good back projection. In this thesis the number of eigenvectors used was the same as D_2, it means different values were tested and accuracy shown.

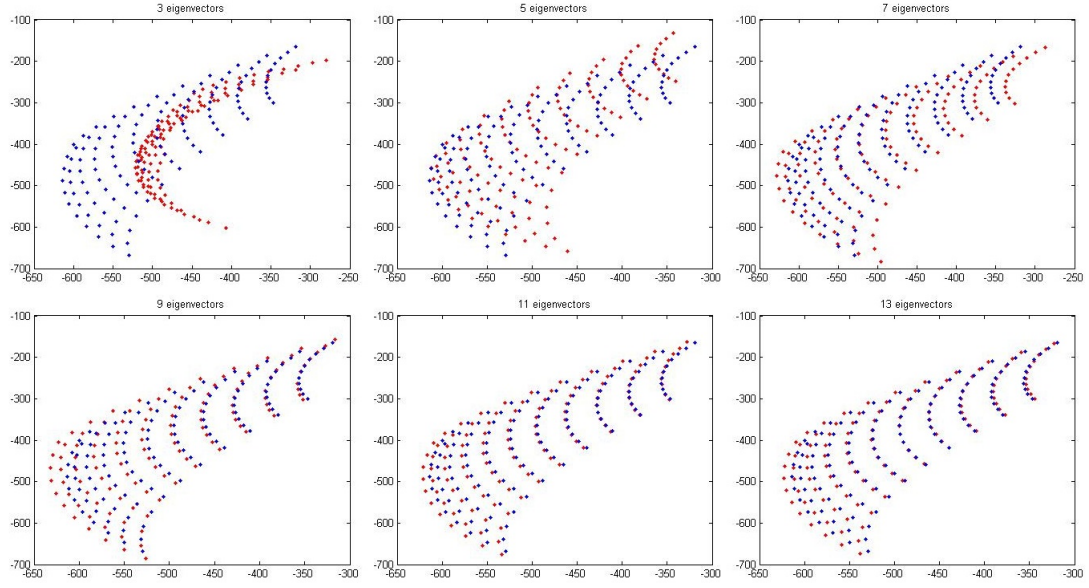


FIGURE 4.17: Quality of manifolds back projected according to the number of eigenvectors

4.5 Handshape Classification with k-Nearest Neighbour

In order to classify the correct shape of an unknown image, the k-Nearest Neighbour (k-NN) algorithm is used. The distance metric considered is Euclidean. Basically, in order to classify a new image from the testing dataset, it has to be projected into the training eigenspace S_1 and then the distance from this point (P_1) to all the other interpolated points has to be computed. The shape with the shortest distance is taken as the object shape. Note that in this section only interpolated data ($P_{1(inter)}$) has been used, which is different from Section 4.2.8 where interpolated spaces and interpolated data were used.

4.5.1 Knowing the Correct Interval

In this section it is assumed that the correct interval is known. In other words the interval classification is not needed. Table 4.1 shows the accuracy of recognising the correct shape given the correct closest subspaces are known. The accuracy is out of N_{im} (2,783 images). D_1 is set to 30 and different values for D_2 are used in the vectorisation and reconstruction process. As D_2 increases, the accuracy increases as well. However, the greater the value of D_2 the more computationally expensive it becomes.

4.5.2 Not Knowing the Correct Interval

In a second experiment, a general classifier is made, assuming that the correct interval is unknown. In this case the interval can be the one on the right side or the one on the left side of the subspace classified as the closest. Therefore, the k-NN search is carried out in both intervals, meaning two intervals at each time, except when the closest subspace is classified as the first. The case where the search is done in two intervals is more computationally expensive than when the correct interval is known. Table 4.1 shows the average accuracy and the standard deviation for shape classification by not knowing the correct interval.

TABLE 4.1: Average accuracy and standard deviation according to D_2 using k-NN when the correct interval is known and unknown

D_2	Known		Unknown	
	Accuracy	Stdv.	Accuracy	Stdv.
3	58.544	2.749	52.279	1.474
5	87.116	2.130	83.099	2.577
7	96.278	1.310	93.812	2.290
8	97.231	1.269	95.593	2.146
11	97.749	1.583	96.066	2.410
13	98.498	1.866	97.285	2.322
15	**98.845**	1.441	**97.631**	2.075

TABLE 4.2: Average accuracy and standard deviation according to the interval size d

d	Accuracy	Stdv.
6	**98.264**	$\pm$ 1.052
8	97.631	$\pm$ 1.666
10	95.710	$\pm$ 3.154

4.5.3 Classification Time for Known and Unknown Interval

Time is measured for classification when knowing and without knowing the correct interval. Note that the speed for classifying the interval and shape is slower due to two classifications being needed. The average time to recognize one shape knowing the correct interval is 0.03458 seconds and without knowing the interval is 0.07379 seconds, which can be considered to be fast enough to be used in real time.

4.5.4 Influence of the Interval Size

In order to measure how the size of the interval influences the accuracy of the final shape classification, Euclidean distance is computed with different values for d. Table 4.2 shows how the interval size d influences the accuracy of classifying the correct shape. Note that a small value for d gives an improved accuracy, as d increases the accuracy decreases, it is given by the fact that small intervals have less data in between, being then closer to the training set. These distances are measured with $D_1 = 30$ and blurring level equal to 3.

TABLE 4.3: Average accuracy and standard deviation according to the blurring level

Level	Accuracy	Stdv.
1	93.453	$\pm$ 1.463
2	96.838	$\pm$ 2.613
3	**97.631**	$\pm$ 1.666

4.5.5 Influence of the Blurring Level

In a similar manner, the level of blurring applied to images influences the accuracy. Table 4.3 shows how blurring influences the accuracy. Note that a high level of blurring improves the accuracy. Different from perpendicular distance (Section 4.3.1.4) where a high level of blurring decreases the accuracy, in k-NN it improves, it probably because the more blurred the images are, the flatter the manifolds.

4.6 Translation Interpolation

In addition to interpolating rotation angles and eigenspaces, it is possible to interpolate in between translations. In this section certain translations are missing from the training dataset and reconstructed by splines.

4.6.1 Sub-dataset for Missing Translations

In order to compute and test interpolation of missing translations, a testing dataset with $d = 8$ and blurring level equal to 3 was chosen. Given the fact that the training dataset has N_{tra} translations for each image at each rotation angle, from the N_{tra} 121 images two sub-datasets were created, the training dataset and the testing dataset. The training dataset consists of the images at every third point in a column of the translation array. Each curve has 11 points in total. Thus, 8 points are training and 3 are testing. In total, for each 121 images, 88 are training and 33 are testing. Figure 4.18 shows the manifold for one shape at one rotation angle where the red dots represent the training images and the blue dots represent the testing images.

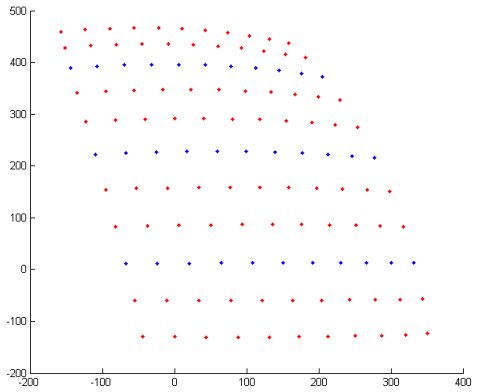


FIGURE 4.18: Projection of training (red dots) and testing (blue dots) sub-datasets into PCA space for translation interpolation

Algorithm 11 Pseudocode for translation interpolation by splines

Precondition: Manifolds of the first stage PCA P_1

for each shape in N_{tra} **do**
  for each angle in N_{ang} **do**
    $P_{(tra)} \leftarrow$ Points of the 1st, 2nd, 4th, 5th, 7th, 8th, 10th and 11th translations
    $P_{(tes)} \leftarrow$ Points of the 3rd, 6th and 9th translations
  end for
end for
for each angle in N_{ang} **do**
  $S_{2(tra)^i} \leftarrow$ PCA over $P_{(tra)}$
end for
for each manifold in $P_{(tra)}$ **do**
  $P_{t(inter)} \leftarrow n$ points interpolated along a spline over elements of each $P_{(tra)}$
end for

4.6.2 Interpolating Missing Translations

Algorithm 11 shows the steps to interpolate in between translations of the translation sub-dataset. Different values of n influence the accuracy of recognising a new shape. Note that for this experiment each manifold is a set of all translations for each shape at each rotation angle (N_{tra}). In this experiment $S_{2(tra)^i}$ is not used, the classification is done directly on the interpolated data $P_{t(inter)}$.

Figure 4.19 shows one of these manifolds. In this case, $P_{(tra)}$ is a manifold with missing translations, and $P_{(tes)}$ are the missing translations.

Figure 4.19 show the anchor points in red and the interpolated points in cyan. In this figure $n = 45$ was considered.

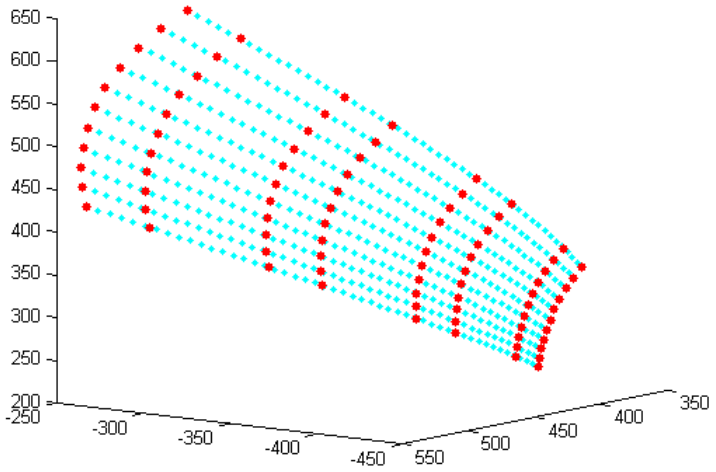


FIGURE 4.19: Manifold of the training sub-dataset (red) and interpolated translations (cyan)

4.6.3 Results for Classifying Shapes with Missing Translations

In order to classify the shapes, the first step is to classify the closest subspace. For this purpose perpendicular distance (distance from the new image to $S_{2(tra)^i}$) and Euclidean distance (distance from the new image to $P_{(tra)}$) are used. For each metric, two approaches were considered, the first is to make a hard decision, i.e. to choose only one subspace and the second one is to pick the closest distance plus its two neighbours. Figure 4.20 shows the average accuracy for both cases. Note these measures were out of 3 subspaces only.

Figure 4.21 shows the accuracy of recognising the shape, given that the closest subspace ± 1 is known. Note that the accuracy of recognising a shape without any interpolation is quite low when compared with the interpolated translation. The size of n influences directly the accuracy as well. As can be seen in Figure 4.21 4 eigenvectors and n equal to 25 is enough to obtain a considerable high accuracy.

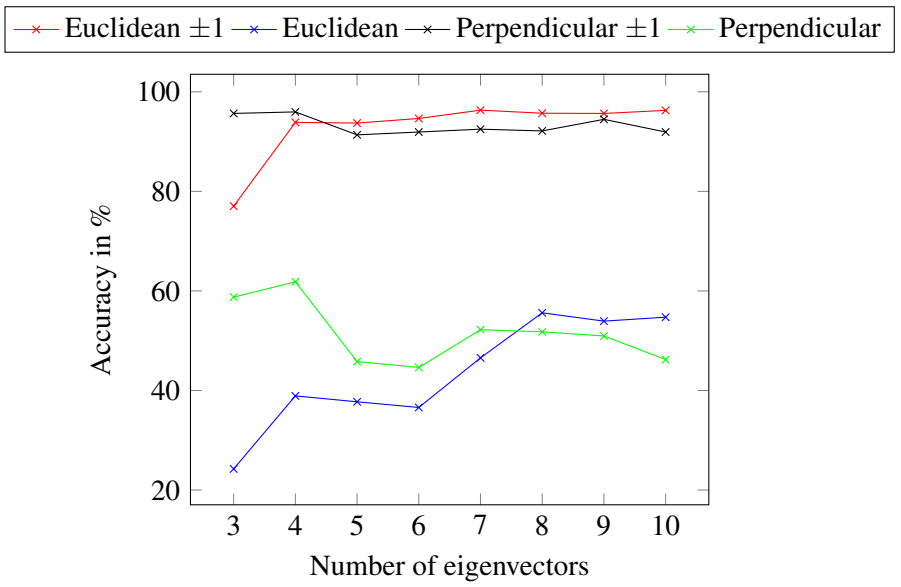


FIGURE 4.20: Average accuracy for subspace classification using Perpendicular Distance and K-NN with Euclidean distance according to the number of eigenvectors

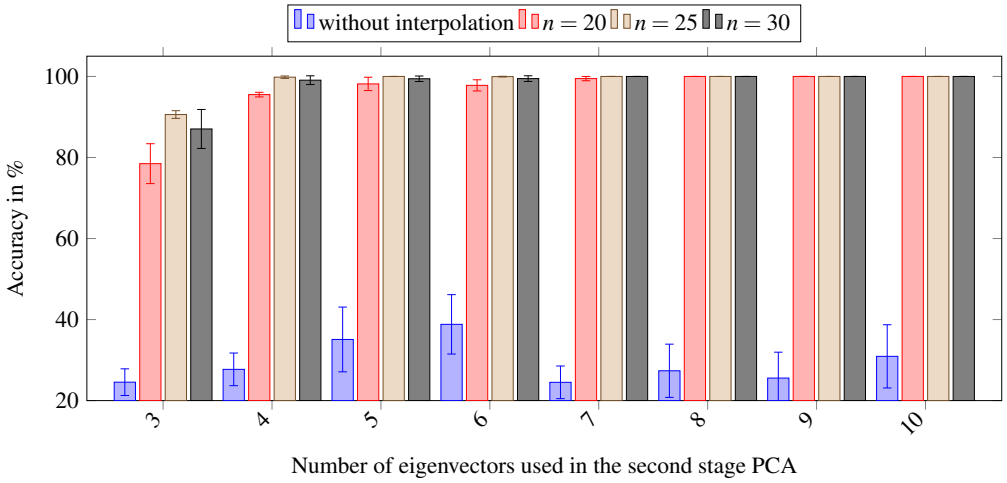


FIGURE 4.21: Shape classification with different values for D_2, different values of n and without interpolation

Figure 4.22 shows the overview of the entire process for $d = 10$, perpendicular distance and k-NN. The first step is take a new image from the testing dataset and project it into the main eigenspace (first-stage); second step is to compute perpendicular distance to find the closest subspace; third step is to interpolate missing translations within the manifolds of the closest subspace and its neighbours; the final step is to compute k-NN and match the correct shape along all interpolated points.

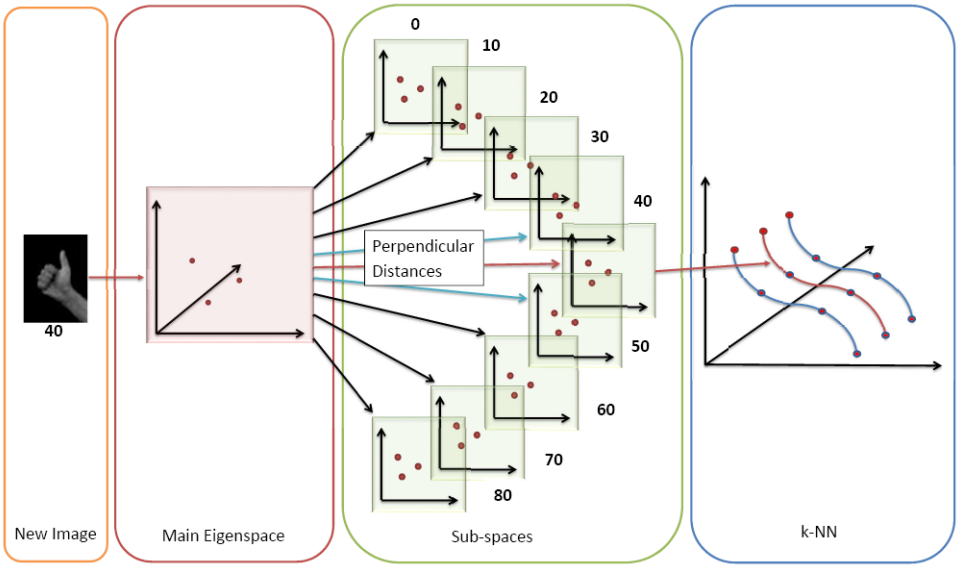


FIGURE 4.22: Flowchart for the overall process for shape classification with missing translations, with perpendicular distance followed by k-NN

Finally, the entire process for shape classification with $D_1 = 15$, $d = 8$ and $n = 25$ was tried. The overall average classification accuracy is 96.08% and standard deviation of $\pm$ 1.83 for perpendicular distance with $D_2 = 4$, ± 1 subspace of freedom and $D_2 = 7$ for k-NN, the average time for classification is 0.21 seconds. Nevertheless, with hard decision, using perpendicular distance the average accuracy is 62.93% with a standard deviation of $\pm$ 8.99 and average time for classification 0.076 seconds. It is clear that a hard decision speeds up the classification. However, it decreases considerably the accuracy.

4.6.4 Shape Classification in the First-stage

In the first stage PCA it is possible to classify shapes without knowing the angle. Figure 4.23 shows manifolds with different colours for different shapes. Note that they are spread over all the rotation angles. However, they keep the same layer.

Table 4.4 shows the accuracy for $d = 6$ and $d = 8$, using k-NN with projection into 20 eigenvectors, blurring level was set to 3, because it is already known that this level of blurring provides improved accuracy for k-NN. Note that the closer the testing images are to the training images, the higher the accuracy.

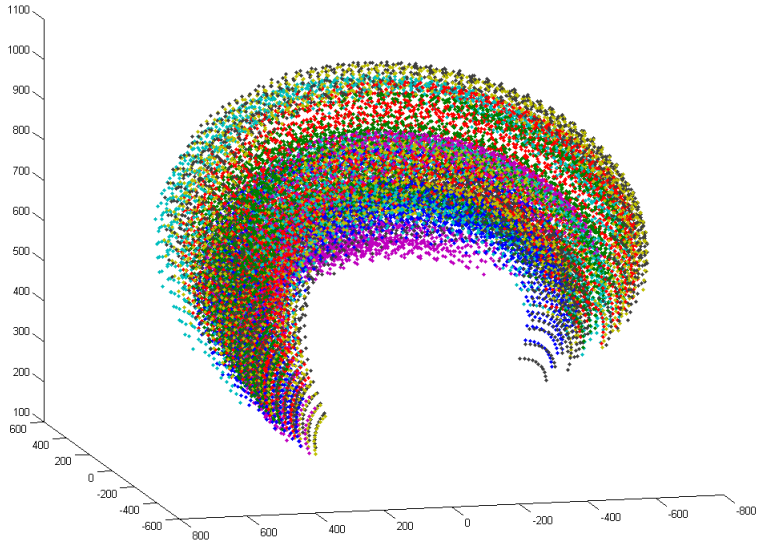


FIGURE 4.23: Projection of images into first-stage PCA (manifold) with different colours for different shapes

TABLE 4.4: Average accuracy according to the interval size, and for each intermediate rotation angle

Test set	Accuracy	
	$d = 8$	$d = 6$
1	99.28%	88.46%
2	78.11%	82.28%
3	73.73%	76.82%
4	66.00%	75.92%
5	76.32%	95.65%
6	84.87%	-
7	93.71%	-
Mean	81.72%	83.83%

4.6.5 Subspaces for Shapes

As stated in Section 4.6.4 each shape appears in a curved layer in the first stage PCA manifold. This leads to the idea of creating one subspace for each shape instead of one subspace for each rotation angle. Figure 4.24 shows the 23 manifolds projected into the same space. The manifolds appear in reverse orientation because eigenvectors can have reverse sign, i.e. if an eigenvector is multiplied by -1 it is still an eigenvector. It is possible to reverse them back

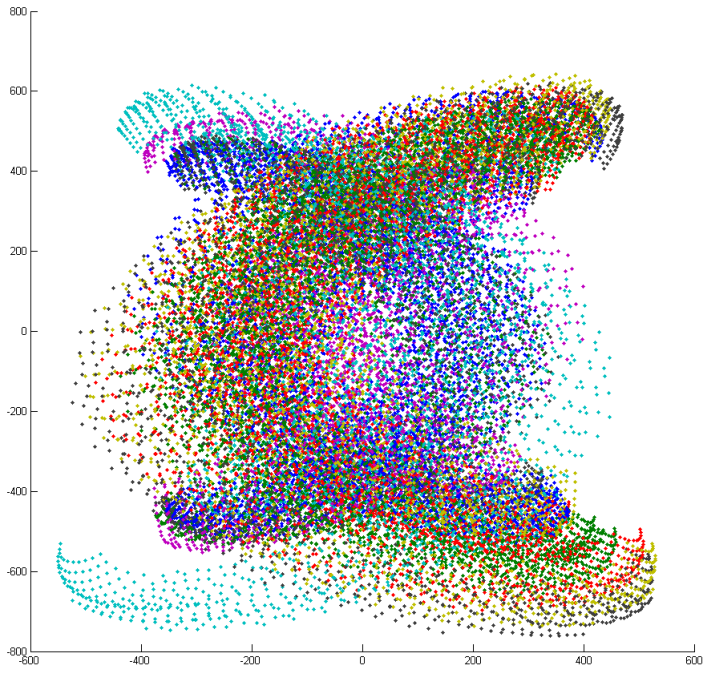


FIGURE 4.24: Second-stage PCA subspace manifolds with different colours for different subspace shapes (each colour is the projection into one different space)

again by multiplying by -1 and re-align with the other manifolds. Each colour represents one shape.

The question that rises from this shape manifolds is how to classify. Perpendicular distance is tested and the accuracy shown is quite low. Probably because shapes overlap each other in different rotation angles. Table 4.5 shows the accuracy of recognising the correct shape using Perpendicular Distance, $D_1 = 20$ and blurring level 3. Note the best accuracy is achieved with 5 eigenvectors.

4.7 Discussion and Conclusion

This chapter showed mainly how PCA can be applied to sign language recognition, especially by manipulating manifolds and sub-eigenspaces. Therefore, different manifold manipulation

TABLE 4.5: Accuracy according to D_2 using k-NN for shape subspaces classification

D_2	Accuracy %
3	11.385
5	**18.207**
7	13.921
8	15.025
11	14.296
13	10.082
15	7.659

techniques were proposed in order to increase the accuracy of recognising an unknown sign language handshape image.

Firstly, it was shown that the interval of an incoming object may be classified by computing the perpendicular distance or by k-NN (Euclidean distance) between this point (new image) and all subspaces.

Secondly, splines were used to interpolate between manifolds to create artificial data. This interpolation is important because the illumination changes according to the arm rotation. Therefore, rotating images by software in order to make the dataset more robust would not be a solution for a real problem.

Finally k-NN was used to find the nearest point within interpolated manifolds and classify the correct shape.

Figure 4.25 shows a comparison of accuracy between using only real data and using interpolated data. The k-NN with $k = 1$ and Euclidean distance was used in this experiment. The number of eigenvectors used (horizontal axis) is the D_1 used in the k-NN classification stage. For interpolation D_2 was set to 30 and a blurring level 3 was used. Note that by using interpolated data the accuracy of recognising the correct shape increases from 83.84% to 97.631% for 15 eigenvectors. However, time for classification increases, the average time to classify one image using real data is 0.0069 seconds and for interpolated data is 0.073 seconds.

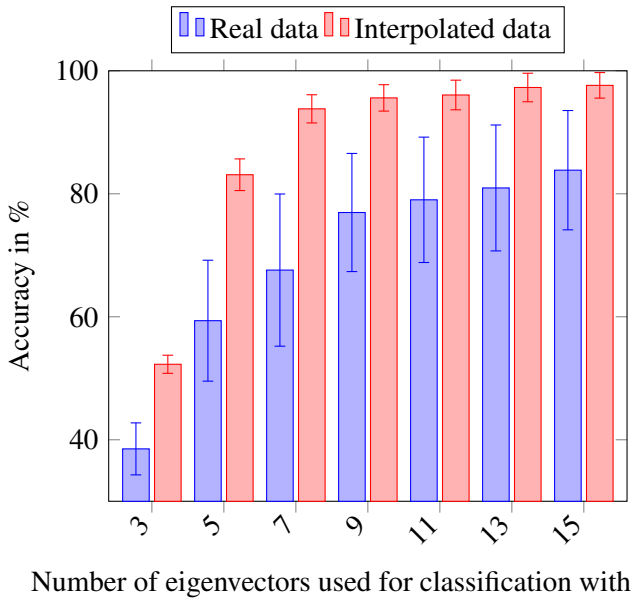


FIGURE 4.25: Comparison of accuracy with only real data (without interpolation) and interpolated data, according to the number of eigenvectors

The lower accuracy of the real data is because of the missing data of the sparse dataset and the interpolation is filling in the missing values and therefore it increases the accuracy. This answers **RQ2**, proving that interpolated data increases the accuracy in the sparse dataset.

Chapter 5

Deep and Shallow Methods

This chapter presents experiments with different classification techniques for the dense dataset proposed in Chapter 3, Section 3.2.2. In this chapter two sets of approaches are proposed, end-to-end and feature-based. End-to-end approaches are techniques that do not need to extract any features from images beforehand. Therefore, classification is made straight from the image pixels. Feature-based approaches are techniques that require feature extraction to be carried out on the images before input to the classification technique.

For end-to-end approaches, in this chapter both deep and shallow methods are used. For feature-based approaches, in this chapter only shallow methods are used.

Deep methods are those, which use many layers of processing before the final classification stage, e.g. CNNs. Shallow methods perform classification directly on the input data e.g. k-NN and SVM.

5.1 Introduction

In this chapter experiments with different classification techniques are tested, mainly with PCA and CNN.

In contrast to Chapter 4 in this chapter a *dense* dataset containing all the rotations is used for experiments, and the goal is to test different techniques applied to this dataset.

PCA was introduced in Chapter 4 and is used in this chapter to extract features for the feature-based approaches. Different classifiers are presented for the handshape recognition process, including k-Nearest Neighbour (k-NN), decision trees, multilayer perceptrons (MLP), support vector machines (SVM), and linear discriminant analysis (LDA). In addition, an experiment with a Kernel PCA (non-linear manifold learning) is presented.

CNNs are multi-layered neural networks specialised in recognising patterns directly from images. CNNs are widely known for robustness to distortion and having minimal or no preprocessing. They have been used for detection and recognition of different objects, including hands Nagi et al. (2011) and LeCun et al. (2015). In addition to CNN, the end-to-end approaches include decision trees, LDA, SVM, MLP, and k-NN. Furthermore, a comparison of end-to-end approaches and feature-based approaches on the new dataset of ISL is reported.

5.2 Handshape Classification

In order to classify handshapes two different approaches are compared. The first approach is end-to-end (Section 5.2.1), and the second approach is based on feature extraction followed by different classifiers (Section 3.2.1.1). Table 5.1 shows the different methods used.

TABLE 5.1: Recognition techniques for end-to-end and feature-based approaches used in this chapter

Approach	Models
End-to-end	Decision Trees; CNN; LDA; MLP; SVM; k-NN
Feature-based	PCA+k-NN; PCA+SVM; PCA+MLP; PCA+LDA; PCA+Decision Trees; KPCA+k-NN

5.2.1 End-to-end Approaches

For this approach different models were used, including a convolutional neural network (CNN), linear discriminant analysis (LDA), support vector machines (SVM), a multilayer perceptron (MLP), decision trees, and a k-Nearest Neighbour (k-NN). Inputs of each model are raw image pixels (blurred or non-blurred).

5.2.1.1 Support Vector Machines

Support vector machines (SVM) are a classification technique that uses machine learning theory to maximise the accuracy of prediction. There are two types of SVM. The first, in which training data is linearly separable in the input spaces and the second is where training data is not linearly separable and they map the input space into a high-dimensional feature space to enhance linear separability in that feature space. Figure 5.1 shows an SVM in the simplest form. It is a hyperplane that separates the training data by a maximal margin. Vectors lying on one side of the hyperplane are classified as -1, and all vectors lying on the other side are classified as 1. The training instances that lie closest to the hyperplane are known as *support vectors* Tong and Koller (2001).

In this thesis, the SVMs were set with a one-vs-rest (ovr) decision function and polynomial kernel (degree 3). For instance, radial basis function (rbf) Gaussian kernel, linear and sigmoid kernel were tested as well.

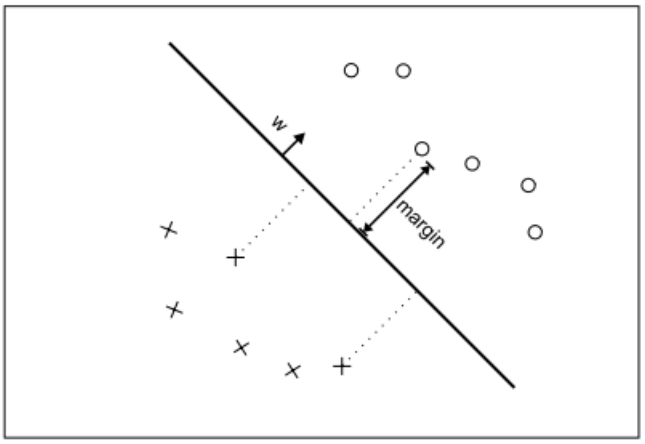


FIGURE 5.1: An example of a simple SVM with a linear function

The kernel of an SVM can be either linear, as shown in Tong and Koller (2001) or non-linear, as in Anguita et al. (2012). In the case of non-binary classification, it uses different schemes for multiclass problems. Anguita et al. (2012) have used the One-Vs-All (OVA) method for human activity recognition on smartphones. Sections 5.3.1.1 and 5.3.2.1 show how SVM were used in this thesis.

5.2.1.2 Convolutional Neural Network

A Convolutional Neural Network is a multistage method specialised on recognizing patterns straight from images. CNN is well known for robustness to distortion and minimal or no preprocessing. CNNs have been used for detection and recognition of objects such as faces, hands, logotypes, text, with a very high accuracy and real-time performance Nagi et al. (2011). The input and output layers are called feature maps. A typical CNN is composed of one or more stages followed by a fully connected layer or classification module LeCun et al. (2010). Each stage is composed of layers such as: convolutional layer and pooling layer.

The convolutional layer has the role of detecting local conjunctions of features from the previous layer, while the pooling layer merges semantically similar features into one. The classification layer has one output neuron per class in the classification task Nagi et al. (2011).

In a CNN each convolutional layer is organised in feature maps. The output of one filter bank applied to the previous layer is called a feature map, e.g. if the input is a colour image, each feature map would be a 2D array containing a colour channel (red, green or blue) of the input image LeCun et al. (2010). Each feature map is connected to local patches in the feature maps of the previous layer with different weights. The result of this local weighted sum is sent to an activation function non-linear node, e.g. rectified linear unit (ReLU). Every feature map in the same layer shares the same kernel size. However, different layers can have different kernel sizes LeCun et al. (2015).

The filter draws across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. Each filter detects a particular feature at every location on the input. Hence spatially translating the input of a feature detection layer translates the output and otherwise leaves it unchanged. In other words, starting from top-left corner of the input image, each patch is moved from left to right, one pixel at a time, as shown in Figure 5.2. Once it reaches the top-right corner, the patch is moved one pixel in the downward direction, and again the patch is moved from the left to the right, one pixel at a time. This process repeats until the patch reaches the bottom-right corner of the image Samer et al. (2015).

In this thesis a *filter bank* is considered to be one set of filters in one layer, the *kernel* is the square matrix with the weights used in the convolution and the *filter* is the process of scanning the kernel across the image. Note that different authors have different definitions, some of them consider three things to be all the same: filter, filter bank and kernel.

Figure 5.3 shows one example of one input layer and one convolutional layer of one image with 32×32 pixels and 32 feature maps (M) Zaccone (2016). Note that the input layer has a kernel, that is considerably smaller than the image size, and in the convolutional layer there is a new kernel that can be the same or different size as the previous layer. Therefore, as in Figure 5.3, a 32×32 image, with a kernel of size 5×5 receptive field across the input image data with a stride width of 1 results in a feature map of $28 \times 28 (32-5+1 \times 32-5+1)$ output values or 784 distinct activations per image Samer et al. (2015).

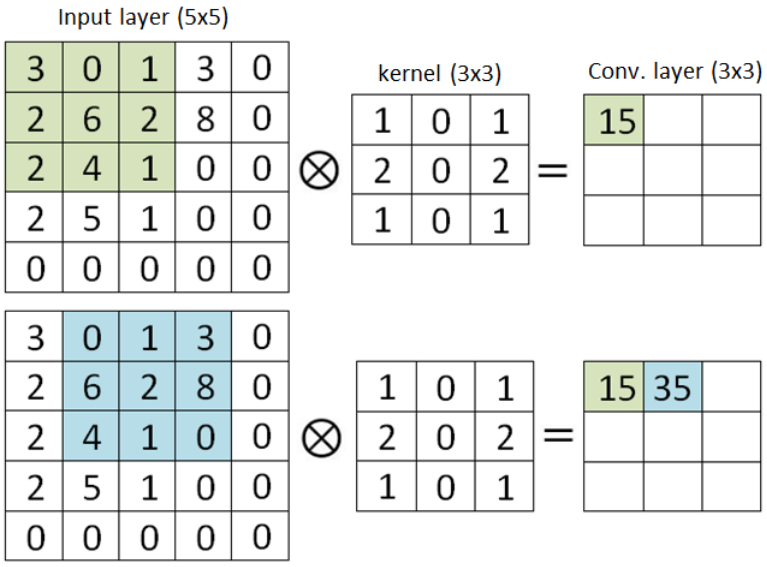


FIGURE 5.2: Example of an input layer being convolved by a kernel and creating an convolutional layer

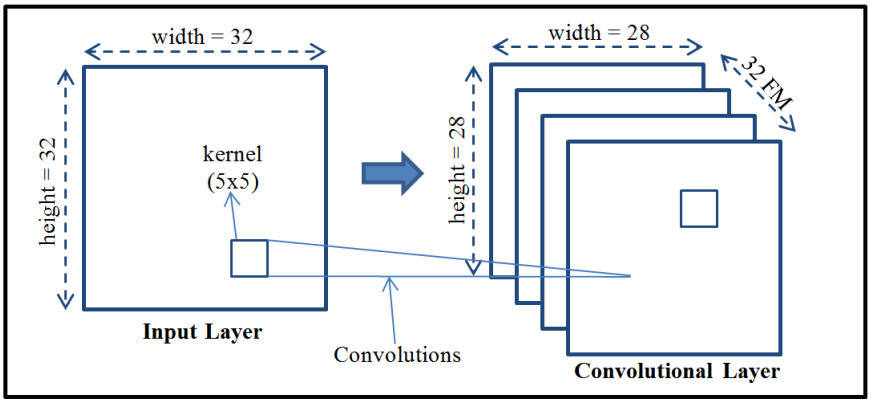


FIGURE 5.3: Example of an input layer, convolutional layer and feature maps

Figure 5.4 shows ReLU activation function, which is zero when $x < 0$ and then linear with slope 1 when $x > 0$. ReLu has become very popular in the last few years. It computes the function $f(x) = max(0,x)$. In other words, the activation is simply thresholded at zero. Krizhevsky et al. (2012) showed that a CNN with ReLU trains several times faster than other activation functions. The number of iterations required to reach 25% training error is considerably smaller than for the equivalent activation functions, e.g. *tanh* or *sigmoid*.

The number of feature maps, kernel size and size of the maps are the parameters for the convolutional layer. Each layer has M maps of equal size (M_x, M_y). The kernel of size (K_x, K_y)

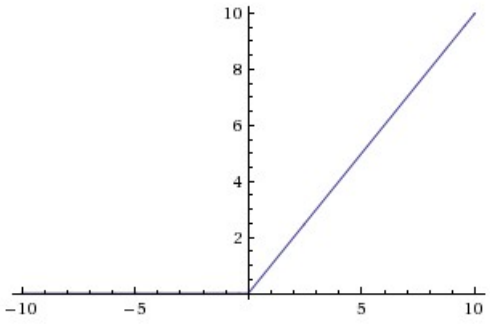


FIGURE 5.4: ReLu activation function

is scanned over a region of the input image. In Layer L^n each map is connected to all maps in layer L^{n-1}. Each map has neurons that share the kernel weights. However, the input fields are different Nagi et al. (2011).

Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Pooling replaces the sub-sampled layer and can be done by taking the max value or the average value from a cluster of neurons at the previous layer.

The maximum activation over non-overlapping rectangular regions of size (K_x, K_y) is the output of the max pooling layer. The objective is to down-sample an input representation, reducing its dimensionality by a factor of K_x and K_y in each direction, allowing a faster convergence rate and improving generalisation performance Nagi et al. (2011). In summary, for each of the regions represented by the filter, the max of that region is used to create a new output matrix, where each element is the max of a region in the original input.

Figure 5.5 shows one example of max pooling layer, where the previous layer is a 4×4 matrix and after max pooling is turned into a 2×2. Cireşan et al. (2011) have shown that max pooling can lead to faster convergence.

After several convolutional and max pooling layers, the high-level reasoning in the neural network is done by the fully connected layer Nagi et al. (2011). All neurons in the previous layer are now connected to every single neuron. These layers are not spatially located anymore.

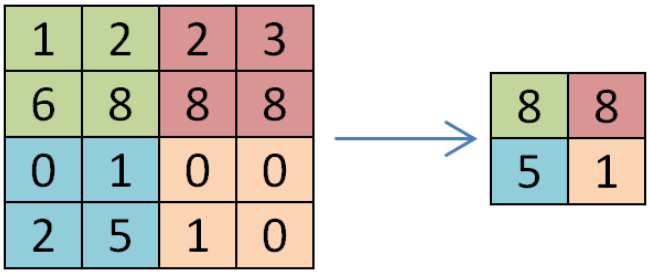


FIGURE 5.5: Example of max pooling layer (4x4) to (2x2)

Thus, there is no convolutional layer after a fully connected layer. The fully connected layer leads to a classification step (output).

The architecture of the CNN model used in this thesis is inspired by Krizhevsky et al. (2012). The structure shown in Figure 5.6 is an adaption of the CNN due to Krizhevsky et al. (2012) where layers have been removed and the high accuracy has been kept. The size of the frames and filters were adapted according to images in the *dense* dataset. The architecture used in this thesis is as follows: It has 4 convolutional layers with ReLU non-linearity, and ends with 2 fully connected layers with 128 and 23 neurons in each layer, and Relu and Softmax non-linearity respectively. The first convolutional layer has 32 feature maps of size 114×154 and kernel size 7×7; the second convolutional layer has 64 feature maps of size 53×73 and kernel size 5×5; the third convolutional layer has 128 feature maps of size 24×34 and kernel size 3×3 the last (fourth) convolutional layer has 256 feature maps of size 22×32 and kernel size 3×3. Each max pooling layer has kernel = 2×2. Dropout layers are used to prevent overfitting (Srivastava et al. (2014); Wan et al. (2013)). The third and fourth convolutional layers are connected to one another without any intervening pooling or normalization layers.

The 23 output neurons are activated correspondingly to the image class. An Adadelta optimizer Zeiler (2012) is used with a learning rate of 1.0, and a categorical cross entropy is used as a loss function. The filter size of the convolutional layers decreases throughout the model because it has been proven to be beneficial Krizhevsky et al. (2012).

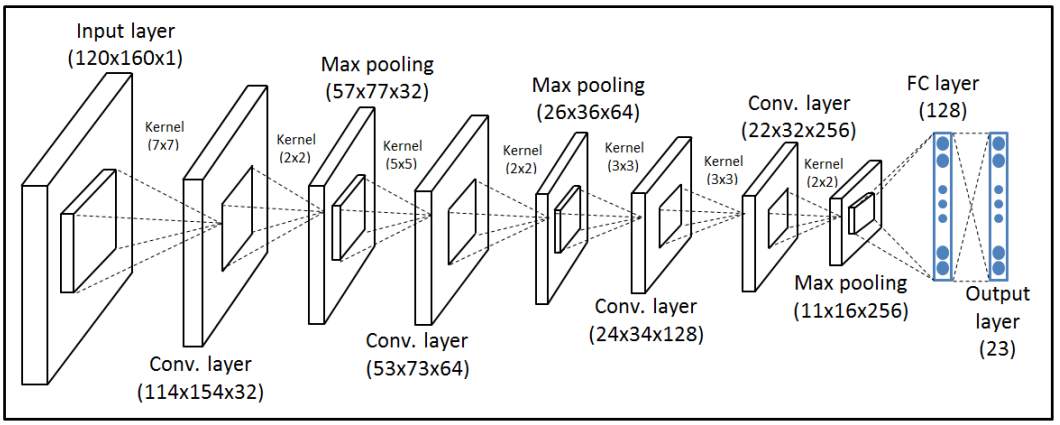


FIGURE 5.6: Architecture of the convolutional neural network used in this thesis, with 1 input layer, 4 convolutional layers, 3 max pooling layers, 1 fully connected layer and 1 output layer; adapted from Krizhevsky et al. (2012) architecture

5.2.1.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) constructs an optimal linear discriminant function $f(x) = W^T x$ that maps the input into the classification space where the class identification of this sample is decided based on some kind of metric, i.e. Euclidean distance (simple or weighted). A typical LDA training stage is done with within and between-class scatter matrices analysis Zhao et al. (1998). Within and between-class scatter matrices are computed as follows:

$$S_w = \frac{1}{M} \sum_{i=1}^{M} Pr(C_i) \sum_i \tag{5.1}$$

$$S_b = \frac{1}{M} \sum_{i=1}^{M} Pr(C_i)(m_i - m)(m_i - m)^T \tag{5.2}$$

Where S_w is the within-class scatter matrices and S_b is the between-class scatter matrices. $Pr(C_i)$ is the prior probability of class C_i; m is the overall mean vector; $\sum_i$ is the average scatter of the sample vectors of different classes C_i around their representative mean vector m_1. Different from PCA, the maximum dimensionality that can be achieved with LDA is the number of classes minus 1 Aran and Akarun (2010). In this thesis a singular value decomposition solver is used.

5.2.1.4 Other Classifiers

The other classifiers are configured as follows: The MLP has a first layer area that is connected to all pixels of the input image, a single hidden layer has 256 neurons, the output layer has 23 neurons to indicate the class of the handshape, and the activation function is ReLu. The k in k-NN is set to 1 and distance metric to Euclidean.

5.2.2 Feature-based Approaches

For this category of methods, features were extracted via Principal Component Analysis (PCA). These features were used as input to the classifiers k-NN, LDA, MLP, SVM, and decision trees. In addition, one experiment is done by Kernel PCA followed by k-NN.

5.2.2.1 PCA-based approach

In order to apply PCA over the training dataset all the images were combined into the same array and then PCA was computed. After vectorization every image is represented by a vector with 19,200 (pixels) entries. As a result, an eigenspace with 19,200 dimensions is obtained.

By projecting the images from the training set into the most significant D_i eigenvectors, a D_i-dimensional space is obtained containing (N_{im}) points for each pose angle. Each point in the space represents an image. In this thesis different number of eigenvectors will be shown and how they affect the accuracy.

Figure 5.7 shows the 2 dimensions (axes), where each point represents one image in the training dataset. The dataset used in this picture is the DB_i dataset.

In the same way, images from the testing dataset were projected into the eigenspace, in order to have both in the same space.

Figure 5.8 shows the manifolds for different people. Note that every person has a different manifold shape.

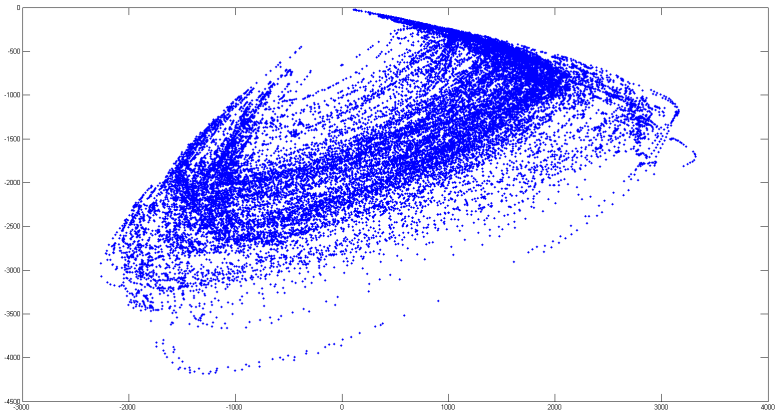


FIGURE 5.7: Projection of training dataset (DB_i) images into the PCA space in 2D

5.2.2.2 Other approaches

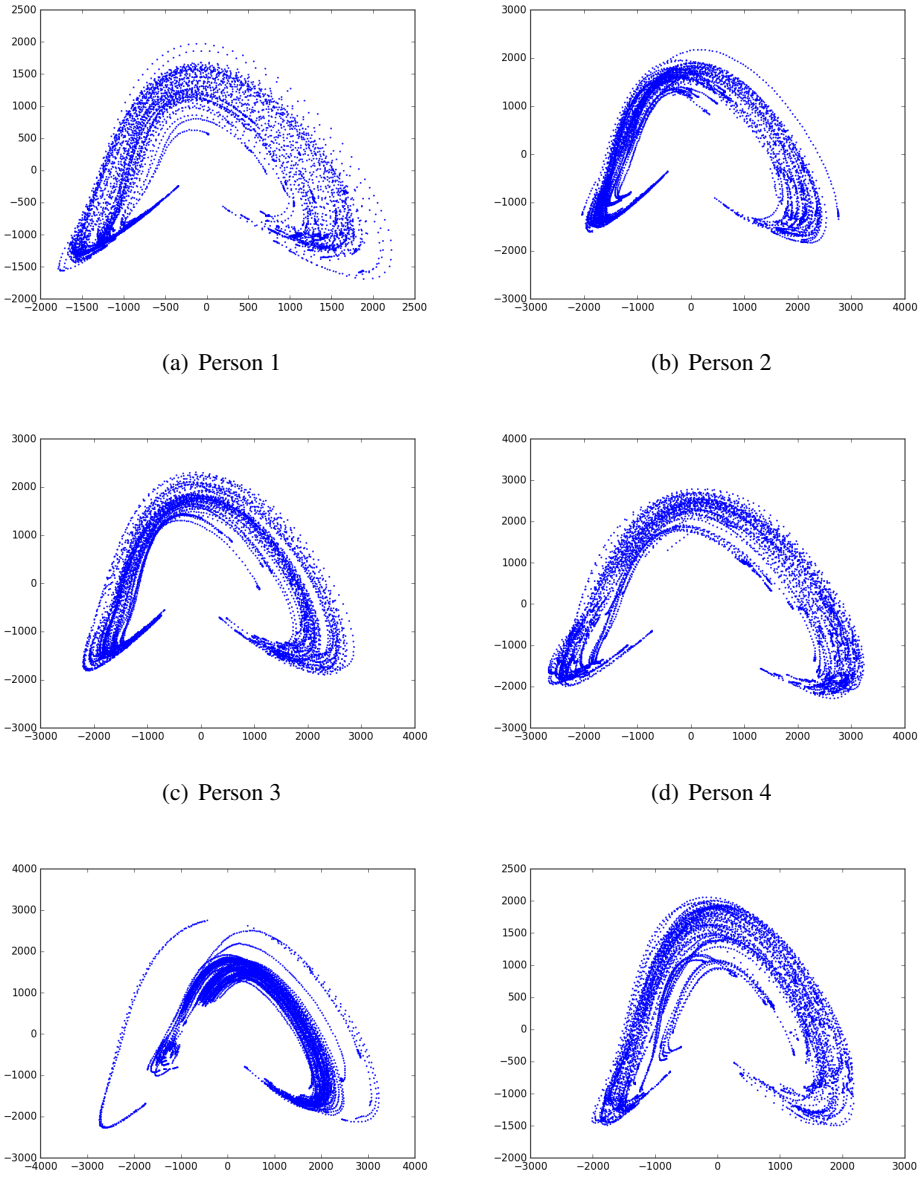
At this point, PCA is applied to the dataset and then used with the aforementioned classifiers to measure accuracy.

The classifiers are configured as follows: for decision trees the minimum number of samples required to split an internal node is set to 2 and the minimum number of samples required to be at a leaf node set to 1. k-NN, LDA and MLP kept the same configuration as the end-to-end approach, apart from taking PCA feature vectors as the inputs. The SVM classifier was set with an one-vs-rest (ovr) decision function and different kernels, such as end-to-end approach with SVM.

5.2.2.3 Outputs of the PCA Approach

PCA allows each eigenvector to be displayed independently as an image. Each eigenvector represents some features of the set of images. Figure 5.9 represents the 3 first eigenvectors (in order: first, second and third) of the 25,000 training images of DB_i. As can be seen visually all of them represent variation in rotation. This variation makes it more difficult to identify what else the eigenvector represents.

In order to make it easier to identify what each eigenvector represents it was considered only the 9 first images of each person, each shape and each shot, in total 3,304 images were



(a) Person 1 (b) Person 2

(c) Person 3 (d) Person 4

(e) Person 5 (f) Person 6

FIGURE 5.8: Different manifold shapess of the PCA approach according to the person, graph (e) shows an outlier probably because one shape was performed in a different position

considered and PCA performed again. This should avoid a significant variation in rotation. Figure 5.10 show plots representing the 3 first eigenvectors.

The 3,304 non-rotated images in the sub-dataset were projected into the first 3 eigenvectors and are illustrated in Figure 5.11. This includes only the 9 first frames for each person,

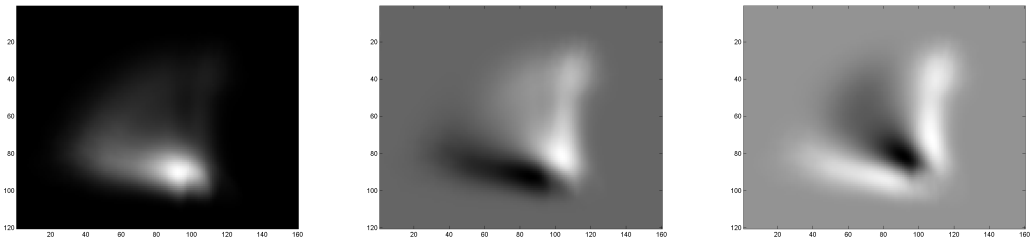


FIGURE 5.9: Visualization of the first 3 eigenvectors for the full training dataset, including all persons and all rotations

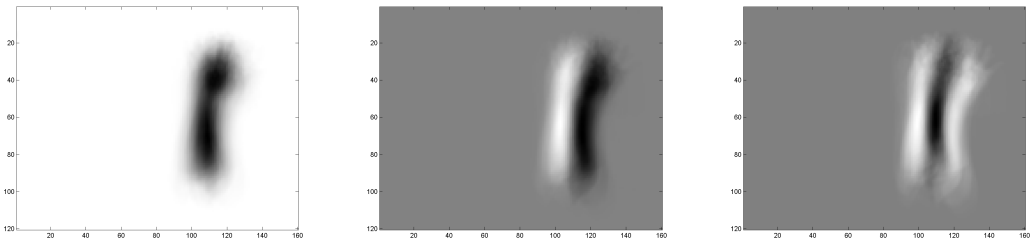


FIGURE 5.10: Visualization of the first 3 eigenvectors for the non-rotated sub-dataset, including only the 9 first frames of each person/shape/shot

each shape and each shot. Therefore, there is no (or minimum) rotation included.

Thus, images are selected from each extreme, two extremes for each dimensions (3 dimensions), 6 extremes in total. Colours are used to distinguish these image extremes. Note in Figure 5.11, yellow and green are the extremes of the first dimension; cyan and magenta for the second, and black and yellow for the third. Finally, images are shown to analyse what the extremes mean.

Figure 5.12 shows two different images at the extremes of the first eigenvector. The image in the left is one of the yellow dots of the plot in Figure 5.11 and the right image, the green dots. In a similar way, images in Figure 5.13 refer to cyan (left image) and magenta (right image). Finally, Figure 5.14 show the left image corresponding to yellow dots and the right image corresponds to black dots of the plot in Figure 5.11.

It can be inferred from Figures 5.12, 5.13 and 5.14 that the first eigenvector is concerned with size of the hand/arm and illumination, the second mostly about translation and the third

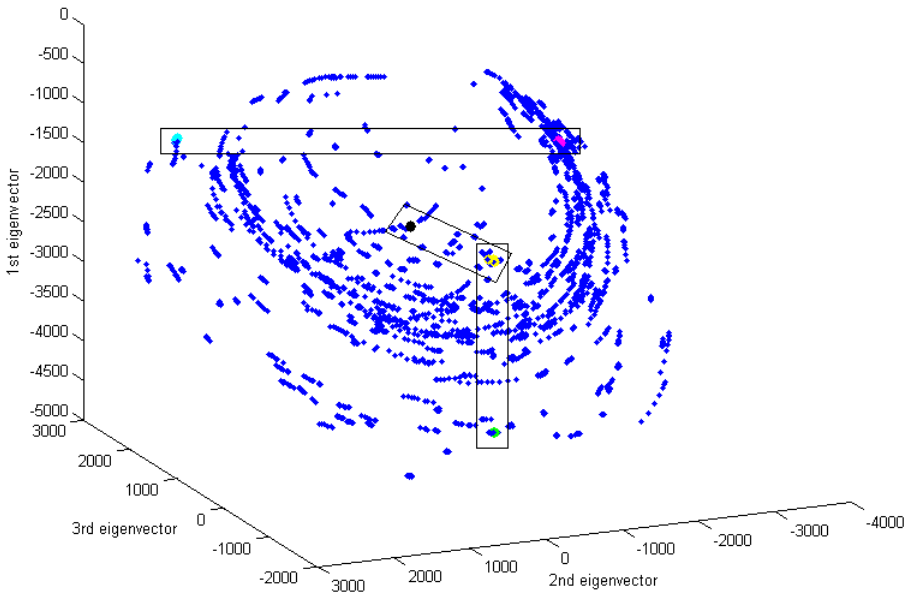


FIGURE 5.11: Plot of the images projected into PCA space (3 first eigenvectors) for the 3,304 images of the 9 first frames of each person/shape/shot

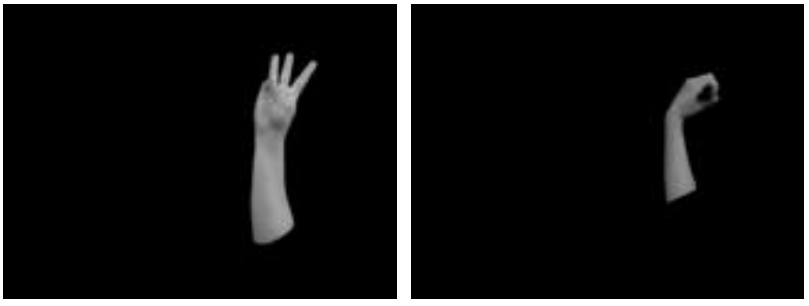


FIGURE 5.12: Visualization of the images corresponding to the variation along the first eigenvector (see Figure 5.11)

about shape and size.

In order to understand how the manifolds behave according to different shapes and different people the colour have been changed of some points in the plot. Figure 5.15 shows the manifolds, with different colours for different people. Note that each person forms a unique manifold at different places in space.

Figure 5.16 shows the manifolds, with different colour for different handshapes, with A

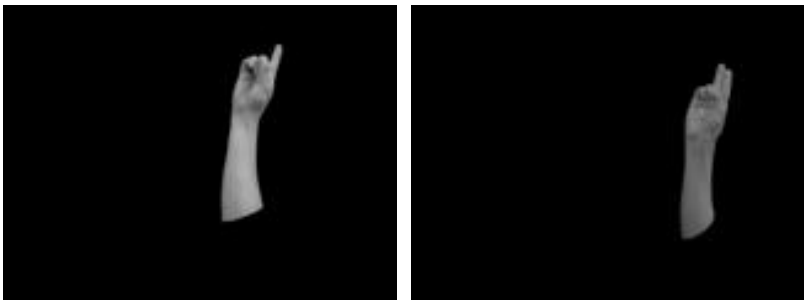


FIGURE 5.13: Visualization of the images corresponding to the variation along the second eigenvector (see Figure 5.11)

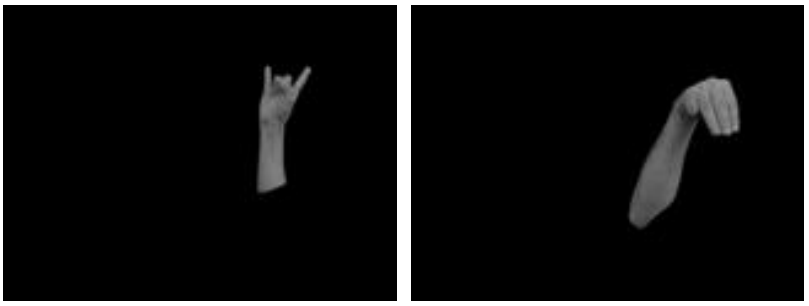


FIGURE 5.14: Visualization of the images corresponding to the variation along the third eigenvector (see Figure 5.11)

= yellow, G = red, M = magenta, R = cyan, W = green and Y = black. It is possible to identify different handshapes falling at different places in space.

5.3 Experimental Results

In this section, results of comparing end-to-end and feature-based approaches applied to Irish Sign Language dataset is reported. Evaluation is shown in terms of *Recognition Accuracy*, which is defined as follows:

$$Recognition\ Accuracy = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (1 - |\overrightarrow{y^i_{true}} - \overrightarrow{y^i_{predicted}}|) \tag{5.3}$$

where $\overrightarrow{y_{true}}$ and $\overrightarrow{y_{predicted}}$ refer to the ground truth and predicted outputs respectively, and $|\overrightarrow{y_{true}} - \overrightarrow{y_{predicted}}|$ is the distance between them. For the specific case of CNN with Keras, the

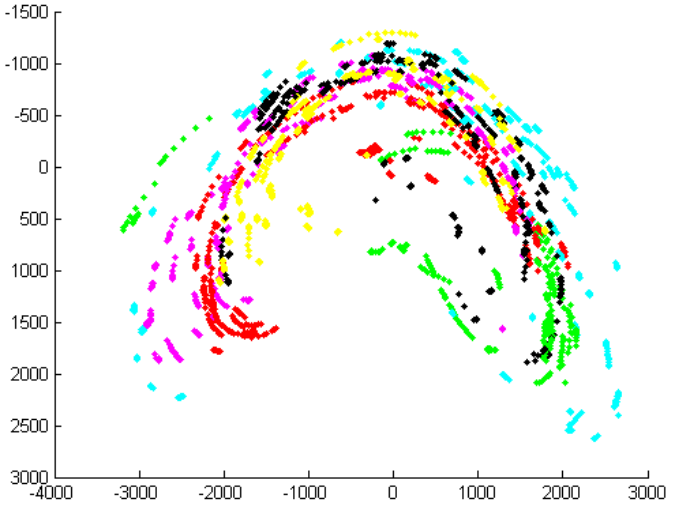


FIGURE 5.15: Manifolds (PCA projection) with different colours for the 6 different people

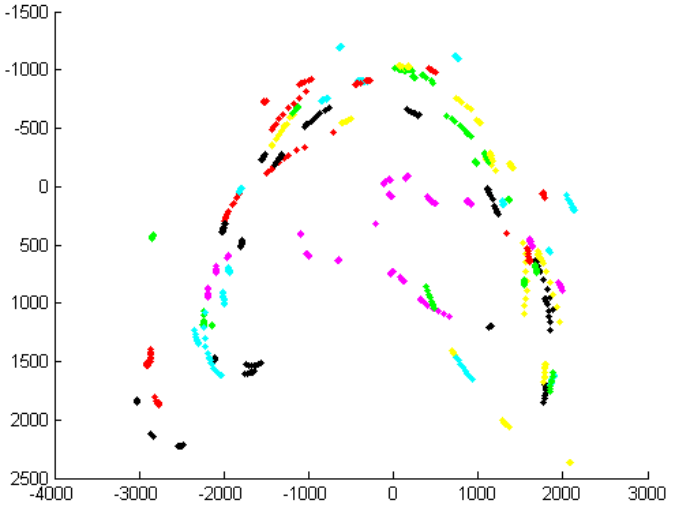


FIGURE 5.16: Manifolds with different colours for different shapes (6 shapes only) for all people

output is a vector of N_{shape} dimensions containing the probabilities of being the y_{true}. In this thesis the maximum probability was taken as the predicted shape $y_{predicted}$.

Experiments in this chapter were made in Python 3.5, scikit-learn 0.19.0, and Keras 2.0.6, running on Windows 7, on an Intel Core i7 CPU @3.4GHz with 16GB RAM. The CNN was trained on an Nvidia Titan X GPU with 12GB RAM.

5.3.1 End-to-end Classification

In the end-to-end approach the images were used without any pre-processing or resizing. For the CNN approach pixels were divided by 255 for normalization. For the other classifiers images were only vectorized. Finally, for each approach/classifier the effect of blurring is tested.

Figure 5.18 illustrates one sample image of the testing dataset. Figures 5.19 show the responses of the CNN model's first convolutional layers and Figures 5.20 the feature maps of the first max pooling layer. It can be seen that the model learns filters that respond to the salient part of the hand, that is the fingers, which results in brighter pixel values in the heat map.

It can be inferred from Figure 5.19 that each map appears to pick out a different component of the hand-shape, e.g. one picks out the right edge of the arm, one picks out the left edge, another picks out the bottom of the arm and there are several that respond to different parts of the hand including one or two that respond to the thumb. Some feature maps appear to respond to global features of the image, e.g. there one or two that respond to the whole area of the image and there is another that appears to respond to the horizontal translation of the hand/arm. Figure 5.20 – the first pooling layer – are simply downsampled versions of the images in Figure 5.19. As the image size gets smaller at each stage it is difficult to visualise the information in the remaining layers.

Figure 5.17 shows the performance progress of the model during the training with 100 iterations, for the DB_i dataset. The model starts to improve in the first few iterations, with only 3 iterations the testing accuracy exceeds 90% and after 43 iterations the testing accuracy exceeds 99%.

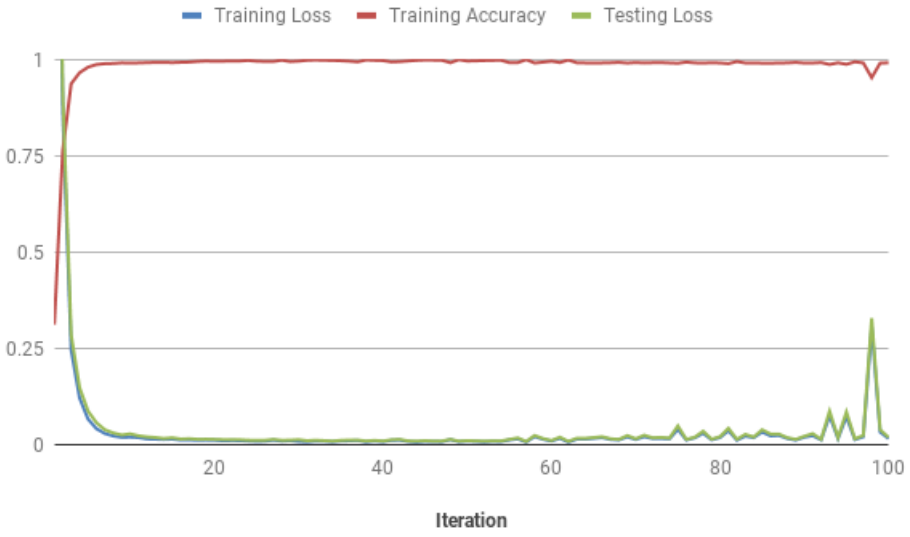


FIGURE 5.17: Training and testing curves of the CNN model, with 100 iterations, for the DB_i dataset, recognition accuracy reaches 90% after only 3 iterations

FIGURE 5.18: Sample of image used to show the result of the layers of the CNN approach, see Figures 5.19 and 5.20

Finally, CNN was tested with blurred images. The kernel size chosen was $(15, 15)$ because for PCA it has shown the optimum accuracy curve (Figure 5.25). It is possible that CNNs carry out their own blurring on the images. Blurring is a convolution operation and it showed an insignificant change in testing accuracy for CNN (Tables 5.4 and 5.5), proving that pre-processing is not important for this technique.

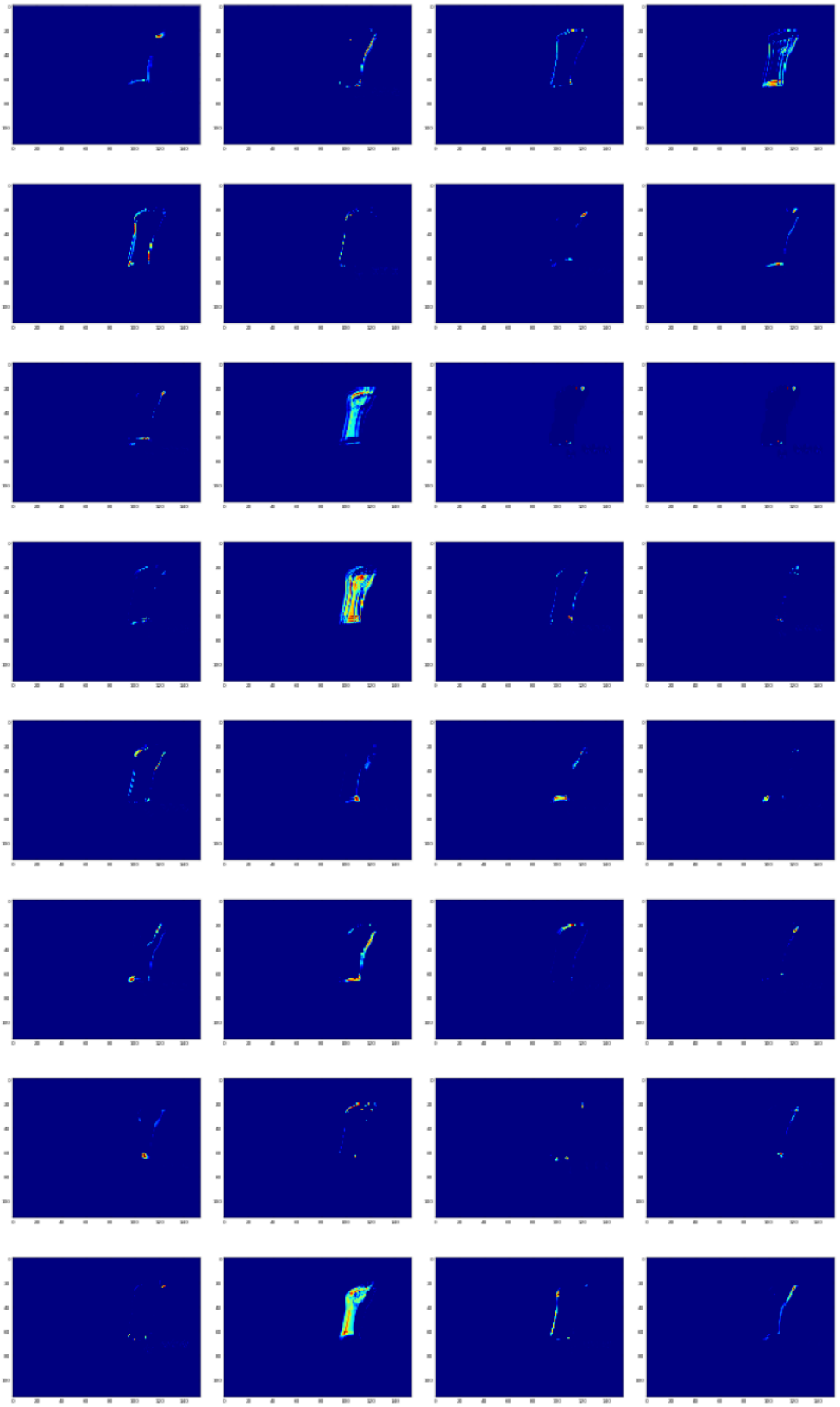


FIGURE 5.19: Feature maps of the CNN's first convolutional layer from the image shown in Figure 5.18

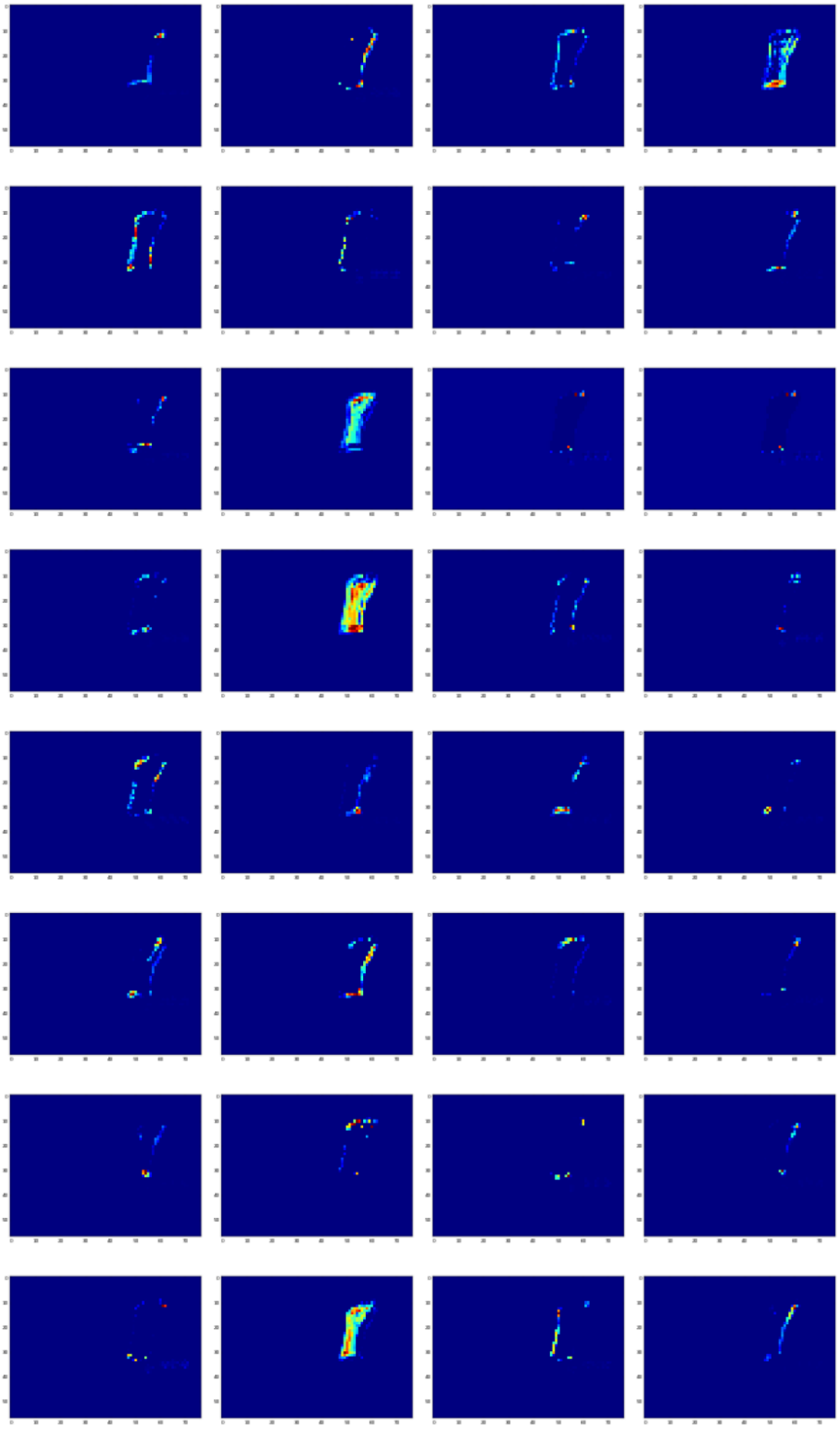


FIGURE 5.20: Feature maps of the first max pooling layer of the CNN's model, from the Figure 5.18

For CNN and the remaining classifiers, such as decision trees, SVM, MLP, LDA and k-NN, accuracy is shown in Table 5.4 for DB_r and in Table 5.5 for DB_i. As can be noted the highest accuracy is with CNN. However, SVM and k-NN showed very high accuracy as well. MLP showed the worst accuracy among the classifiers, because this technique mostly depends on a small number of features instead of raw pixels. Some classifiers showed improved results for blurred images, i.e. k-NN, LDA and SVM. However, CNN seems to not benefit from blurring.

Figure 5.21 shows the accuracy for the iterative dataset DB_i and for the random dataset DB_r for non-blurred images for the end-to-end approach. Note the iterative method showed improved accuracy for decision trees, LDA, SVM and k-NN whereas MLP and CNN showed slightly improved results for random selection.

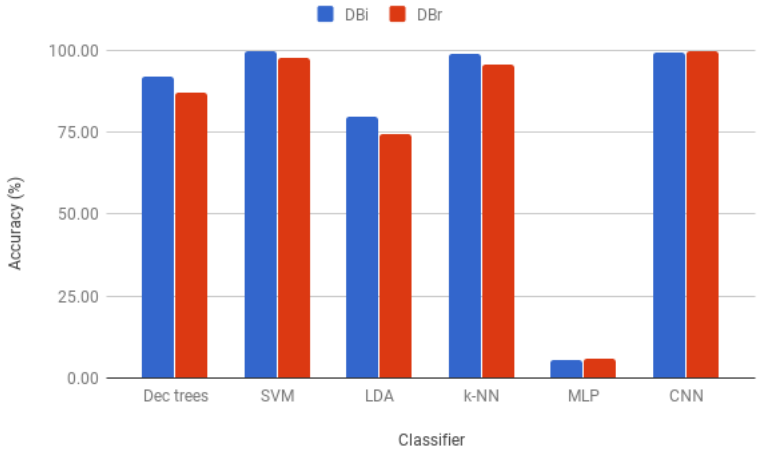


FIGURE 5.21: Accuracy according to dataset (DB_i and DB_r) and classifiers for the end-to-end approaches, note that the best accuracy is given by CNN, followed by SVM and k-NN

Figure 5.22 shows the average time and standard deviation to recognize one image in seconds. Ten images are used to measure the time and the mean is taking. For CNN and MLP 100 iterations are considered. Time was measured for different end-to-end classifiers. Note that the shortest classification time is with decision trees followed by LDA, and the classifier that took the longest was SVM. Taking into account speed and accuracy k-NN provided the best accuracy with a still short time followed by CNN.

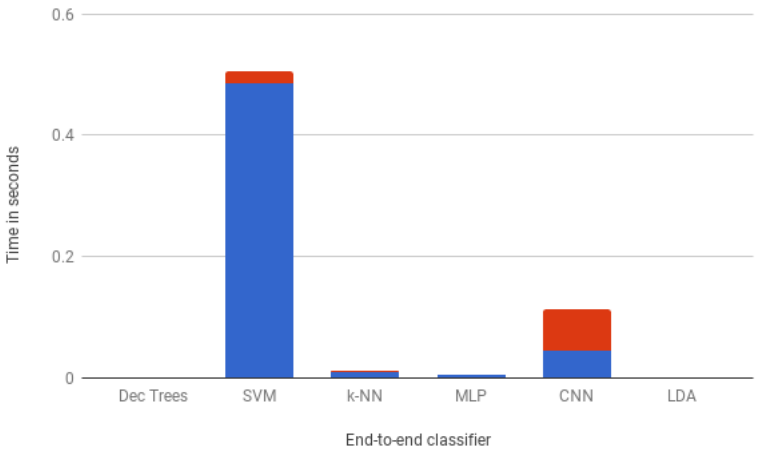


FIGURE 5.22: Average time to classify one image according to the classifier for end-to-end approaches, note that the slowest classifier was SVM followed by CNN

TABLE 5.2: SVM classifier for end-to-end approach with different kernel functions for DB_r, the best accuracy is given by polynomial kernel and blurred images

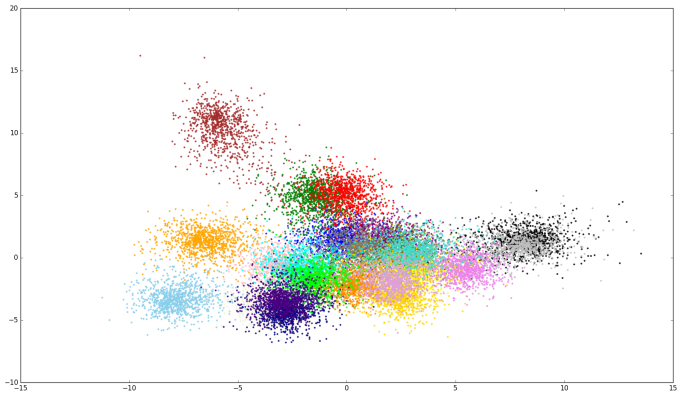
kernel	Non-blurred	Blurred
Poly	**97.86%**	**99.80%**
Linear	84.40%	72.77%
Sigmoid	4.38 %	4.01%

Figure 5.23 shows a projection of the training data over the LDA space. Note that contrary to PCA, blurring does not help to make the handshapes more separated. However, it makes them even closer. It explains why the accuracy for LDA decreases for blurred images.

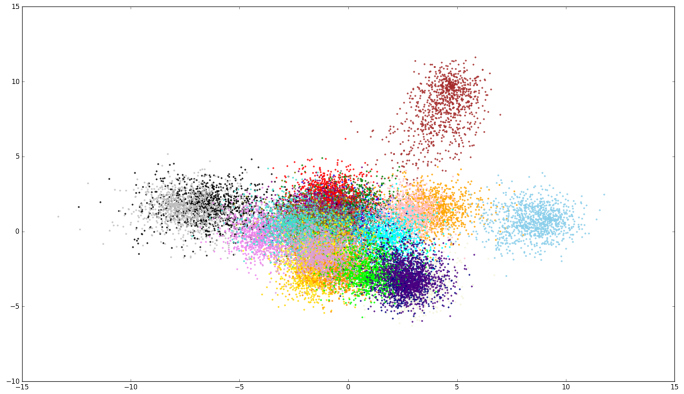
5.3.1.1 SVM Classifier

The SVM classifier was used with different kernels. The first is the polynomial kernel (degree 3); the second is the sigmoid kernel and the third is the linear kernel.

Table 5.2 shows the accuracy according to the kernel functions for DB_r. Note that the best accuracy for the polynomial kernel is 99.80% for blurred images and the lowest is the sigmoid kernel with 4.01%. Linear kernel showed 84.4% accuracy for non-blurred images and 72.77% for blurred images, proving that non-linear kernel is not the best function for SVM in this dataset.



(a) Non-blurred images



(b) Blurred images

FIGURE 5.23: Projection of training data into LDA space, each colour represents one shape, in this case blurring does not help to separate the classes

5.3.2 Feature-based Classification

As stated in Section 3.2.1.1 PCA is used in order to reduce the dimensionality and extract features. The highest number of eigenvectors used was 100.

The first classifier tested was the k-NN algorithm, with $k = 1$ and Euclidean distance. Each testing image was projected into the training dataset eigenspace and classified according to the nearest point (shortest distance). This was run on two different datasets, DB_i and DB_r. In addition, it will be shown in Section 5.3.2 how different levels of blurring affect the accuracy in DB_b.

The accuracy in recognising the correct sign strongly depends on the number of the eigenvectors (dimensions) considered. For example, in DB_i for $D_i = 15$, accuracy is 81.7%. When using more eigenvectors the accuracy increases as well. e.g. for $D_i = 60$ the accuracy is 91.4%.

Figure 5.24 shows the accuracy according to the number of eigenvectors for DB_i and non-blurred images. It is possible to identify that a plateau starts around the 40th eigenvector and above. This means that there is no need to consider a large number of eigenvectors in the recognition process.

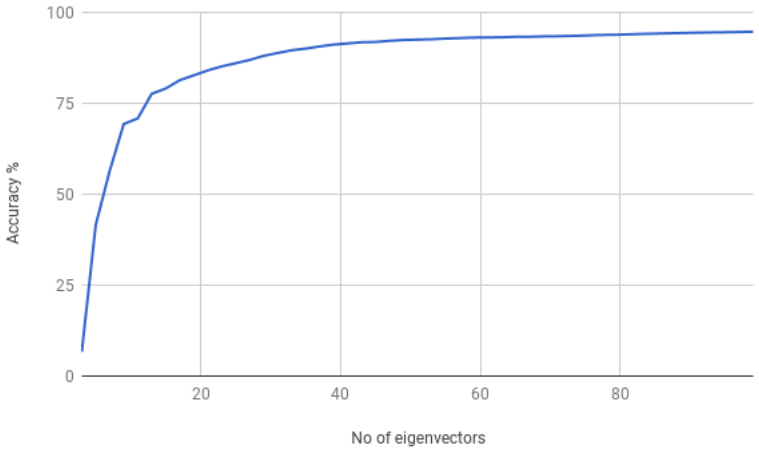


FIGURE 5.24: Recognition accuracy according to the number of eigenvectors for DB_r, note that 40 or more eigenvectors give a improved accuracy

Figure 5.25 shows the accuracy according to the number of eigenvectors and according to the Gaussian blurring kernel size for DB_i and DB_r testing dataset. It is clear that blurring helps to reduce the number of eigenvectors needed to obtain a satisfactory accuracy. This finding is consistent with earlier studies showing the positive effect of image filtering with blurring on PCA by reducing the non-linearity in the manifolds within the eigenspaces Farouk et al. (2013). In addition, it is noticeable that there is an optimal result in the blurring with kernel size $(15, 15)$. For that reason, all other experiments from now on will use this kernel size.

Table 5.3 show the accuracy performances using k-NN according to different values of k on the DB_i for feature-based approaches using PCA with 100 eigenvectors. The Gaussian kernel blurring size was set to (15,15) . It is noticeable that the optimum number of neighbours

TABLE 5.3: Classifier performances using PCA with 100 eigenvectors and k-NN according to k in the DB_i, $k = 1$ has the highest accuracy

k	Non-blurred	Blurred
1	**97.60%**	**98.79%**
2	96.21%	98.01%
3	96.92%	98.46%
4	94.47%	96.04%
5	93.08%	94.76%

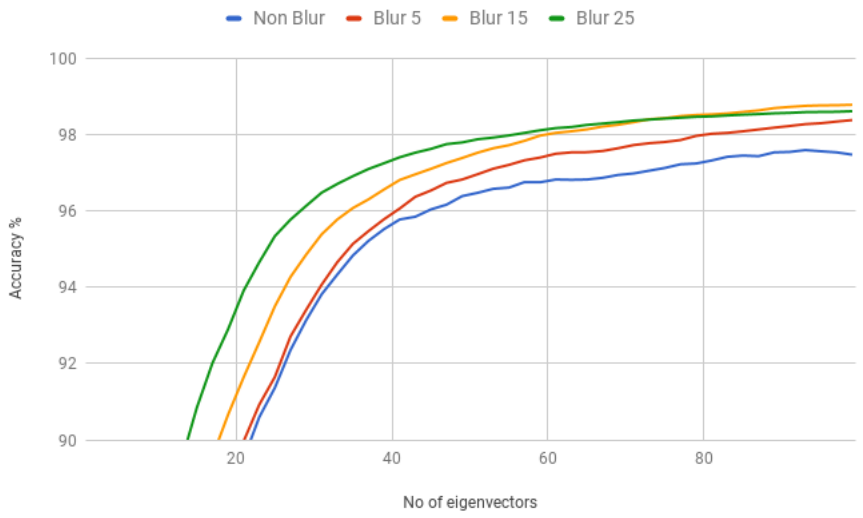
k is only 1. In addition, different metrics for the k-NN were tested, for the Manhattan distance, with $k = 1$ for the same dataset the accuracy was 98.05% for non-blurred images and 99.38% for blurred images.

Figure 5.26 shows the accuracy for the iterative dataset DB_i and for the random dataset DB_r for non-blurred images and feature-based approaches with 100 eigenvectors. Note that iterative dataset did not show improved accuracy for PCA+SVM, PCA+LDA and KPCA+k-NN, for PCA+MLP the difference is insignificant.

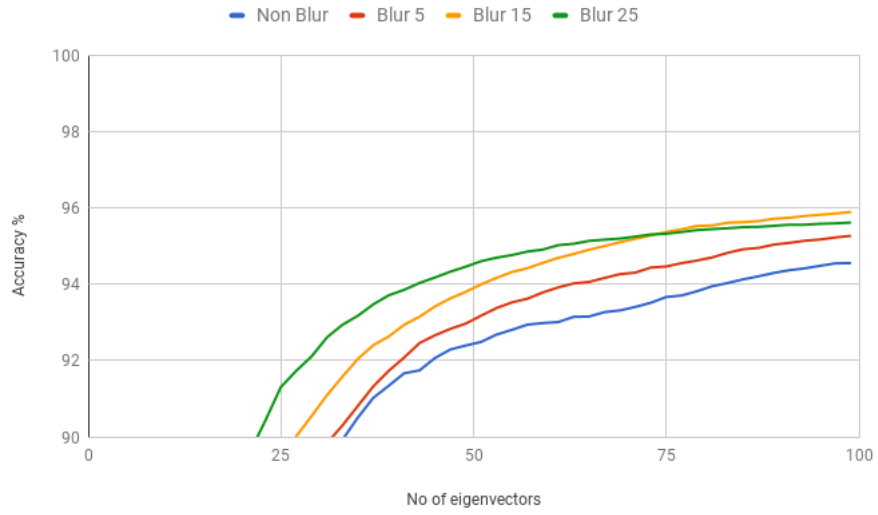
In addition to k-NN, different classifiers were tested over PCA data, such as decision tree, LDA, SVM and MLP. Table 5.4 and 5.5 report the accuracy for each classifier, over the DB_r and DB_i datasets, for either blurred and non-blurred images.

For this experiment, 100 eigenvectors were used and Gaussian kernel blurring size was (15,15) for the blurred images. Note that blurring helps all tested classifiers and the best accuracy is obtained with PCA+SVM for the feature-based approach. PCA+LDA showed the lowest accuracy, probably because a linear classifier is not the best to classify overlapping manifolds, compared to the other techniques (see Table 5.2). Different to what Al-Taie et al. (2017) believe, in this case LDA showed a lower accuracy compared to PCA. A non-linear manifold learning technique was tested with Kernel PCA + k-NN which showed lower accuracy than PCA+k-NN.

Table 5.4 and Table 5.5 show how classifiers behave with blurred and non-blurred images for DB_i and DB_r respectively. For this experiment Gaussian kernel blurring size was (15,15) for blurred images and 100 eigenvectors for PCA. Note that the use of blurring over images



(a) DB_i testing dataset



(b) DB_r testing dataset

FIGURE 5.25: Accuracy according to the blurring level and number of eigenvectors for DB_i and DB_r using PCA+k-NN

improved the accuracy for decision trees, SVM and k-NN. The classifiers CNN and MLP depend on the dataset DB_i and DB_r. However, the difference in accuracy is extremely low. In all the cases that blurring improved upon the results with blurred images, the improvement was considerably low. MLP showed the lowest accuracy over all the techniques, probably because this classifier there are a plethora of parameters to be tuned for use with high dimension data,

TABLE 5.4: Classifier performances in the DB_r, 100 eigenvectors for PCA and kernel blurring size (15,15), polynomial kernel for SVM and 100 iterations for CNN; the best accuracy is given by CNN and SVM

Model	End-to-end accur		Feature-based accur.	
	Non-blur.	Blurred	Non-blur.	Blurred
CNN	**99.80%**	99.58%	-	-
MLP	6.01%	5.61%	98.56%	99.50%
Decision trees	87.21%	88.81%	76.45%	77.36%
k-NN (k=1)	95.50%	96.91%	94.56%	95.90%
LDA	74.41%	53.44%	32.72%	38.82%
SVM	97.86%	**99.80%**	**99.61%**	**99.87%**
KPCA + k-NN	-	-	91.74%	95.14%

TABLE 5.5: Classifier performances in the DB_i, 100 eigenvectors for PCA and blurring size (15,15), polynomial kernel for SVM and 100 iterations for CNN; the best accuracy is given by SVM followed by CNN

Model	End-to-end accur		Feature-based accur.	
	Non-blur.	Blurred	Non-blur.	Blurred
CNN	99.28%	99.78%	-	-
MLP	5.48%	6.00%	98.56%	99.50%
Decision trees	92.05%	93.97%	85.09%	85.56%
k-NN (k=1)	98.80%	99.28%	97.60%	98.79%
LDA	79.79%	56.78%	31.46%	39.56%
SVM	**99.88%**	**99.99%**	**98.76%**	**100.00%**
KPCA + k-NN	-	-	87.25%	91.05%

such as these images without any extraction of features. The second lowest accuracy was with LDA, probably because a linear classifier is not the best for this kind of data with such high dimensions (see Table 5.2). CNN showed the best accuracy for non-blurred image for DB_r, it reinforces the idea that CNN convolves a input image with a filter such as Gaussian blurring.

Figures 5.27 show the average time and standard deviation to recognize one image in seconds. Ten images are used to measure the time and take the mean for PCA approach with 100 eigenvectors. Time was measured for feature-based approaches and different classifiers. Note that the shortest classification time is with for PCA+LDA followed by PCA+decision trees, and the classifier that took the longest is PCA+SVM. Taking into account speed and accuracy PCA+MLP provided the best accuracy with a still short time.

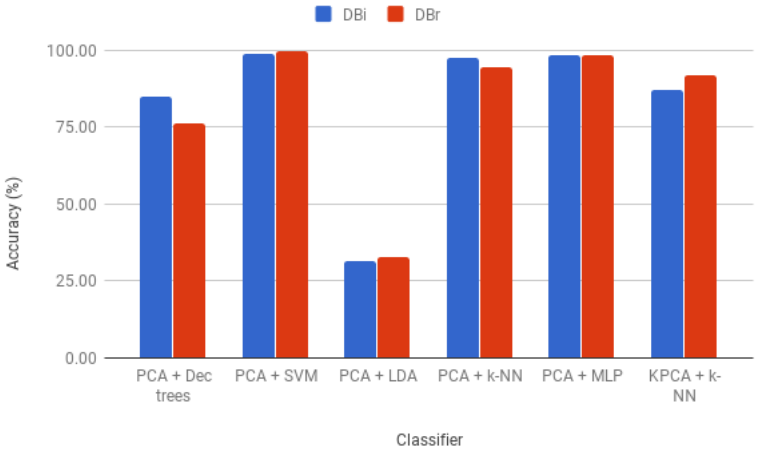


FIGURE 5.26: Accuracy according to dataset (DB_i and DB_r) and classifiers for feature-based approaches, 100 eigenvectors for PCA and non-blurred images; the highest accuracy was with MLP and SVM

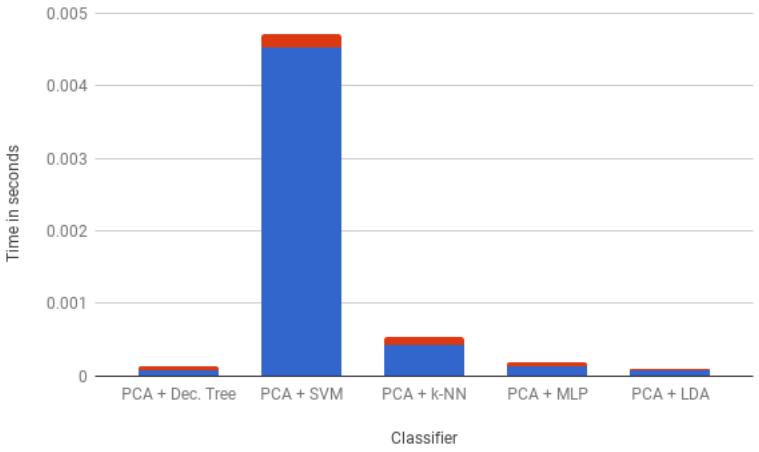


FIGURE 5.27: Average time to classify one image according to the classifier for feature-based approach; the slowest classifier was PCA+SVM followed by k-NN

Considering time (Figure 5.27) and accuracy (Figure 5.26), PCA + MLP has the best combination of accuracy and speed.

5.3.2.1 PCA+SVM Approach

At this stage three main kernel types were tested for PCA+SVM approach. The first is the radial basis function kernel or RBF kernel (Gaussian), the second is the sigmoid kernel and

the third one is the polynomial kernel (degree 3).

Figure 5.28 shows the accuracy according to the value of gamma value for SVM classifier over eigenvectors with Gaussian kernel for DB_r considering 100 eigenvectors. Note that the best accuracy for Gaussian kernel is 94.89%, still lower than the sigmoid kernel with 99.61%.

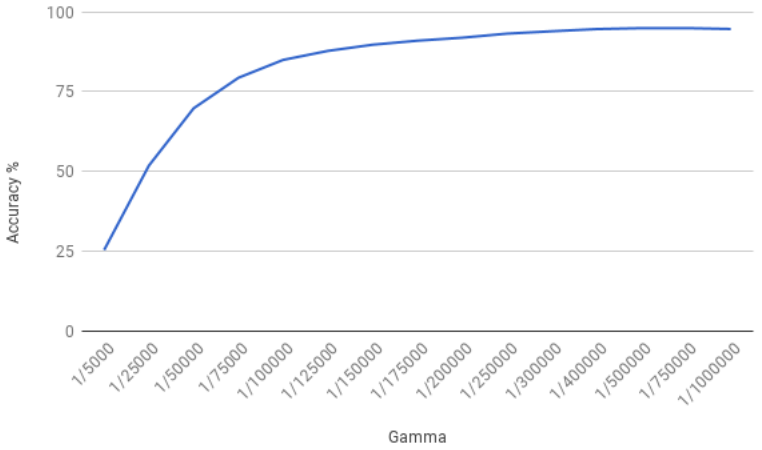


FIGURE 5.28: Accuracy for PCA+SVM according to the value of Gamma for Gaussian kernel

5.3.3 Analogy between PCA and CNN

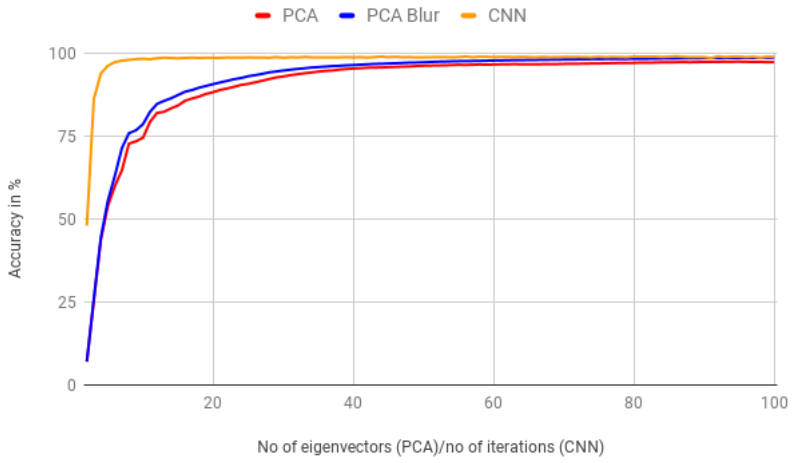


FIGURE 5.29: Comparison of accuracy, PCA with number of eigenvectors (blurred and non-blurred images) and CNN iteration number, both for DB_i; note the accuracy for CNN reaches a very high value with just 3 iterations, whereas PCA depends on certain number of eigenvector for approximate CNN curve

While understanding that PCA and CNN are different approaches, it is possible to see their similarities.

PCA uses eigenvectors and CNN uses feature maps and iterations. Using more eigenvectors for PCA results in preserving more information from the original data whereas more feature maps for CNN results in more information preserved as well. Each eigenvector detects a feature of the data. That is similar to the convolutional filters in the feature maps in CNN, note that images displayed in Figure 5.19 are comparable with the images in Figure 5.9 and Figure 5.10 which show similar responses for different eigenvectors.

Layers in the CNN are comparable to stages in multi-stage PCA. Because a layer in a CNN operates on the output from the previous layer, in the same way the second-stage PCA operates on the output from the previous stage.

The manifolds/interpolation in PCA space are similar to the non-linear parts of the CNN, i.e. the fully-connected layers.

It has been shown that blurring helps PCA accuracy and it is possible that in the similar way CNN convolves an image with a kernel obtaining a similar effect. In that way the number of iterations help in adjusting the weights of the filters by backpropagation.

Basically, the difference between PCA and CNNs is that in PCA the human user has to choose which points to interpolate to create the manifolds, whereas in CNNs the algorithm does that. Moreover, in creating the second-stage PCA the human user has to select which subsets of the points in the first stage are going to be used for the second-stage, whereas the CNN does that itself.

Figure 5.29 shows a comparison of accuracy of PCA and CNN. The y-axis shows the accuracy in % and the x-axis represents the number of eigenvectors for PCA and iteration number for CNN. Note that CNN has shown an improved accuracy.

Figure 5.30 shows the time to recognize one image in seconds. A hundred images were used to measure the time. For PCA it used 100 eigenvectors and k-NN for classification, CNN

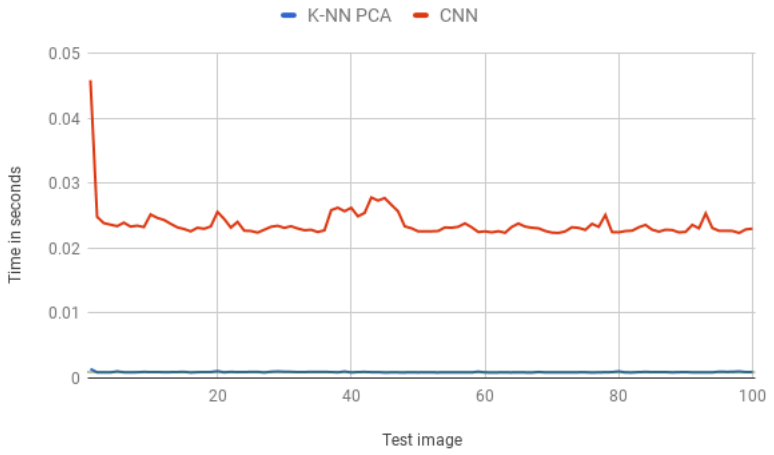


FIGURE 5.30: Comparison of time to recognise 100 images (one at each time) using PCA+k-NN and CNN, note time for PCA is considerable shorter

was iterated 100 times because this number provided a good accuracy as an empirical result. Note that the time is shorter for the PCA+k-NN approach.

5.3.4 Results for Missing Persons in the Training Stage

In this section the performance of the PCA and CNN approaches are presented for the missing persons datasets DB_1, DB_2 and DB_3.

Figure 5.31 shows the accuracy for the CNN approach according to the number of iterations whereas Figure 5.32 shows the accuracy for the PCA+k-NN approach according to the number of eigenvectors. Note that in both cases the accuracy is quite low. However, CNN showed improved results for these case scenarios.

The best accuracy for the CNN approach when removing 3 persons from the training dataset was 34.27%, removing 2 persons 35.80% and removing one person 31.24%. For the PCA+k-NN approach when removing 3 persons from the training dataset the accuracy was 12.70%, for two persons out 9.55% and for 1 person out 10.28%. The best accuracy for CNN was for DB_2 and for PCA+k-NN for DB_3. It is to be expected that the person chosen to be out of the dataset influences the accuracy. However, only a few of the possible experiments are tested.

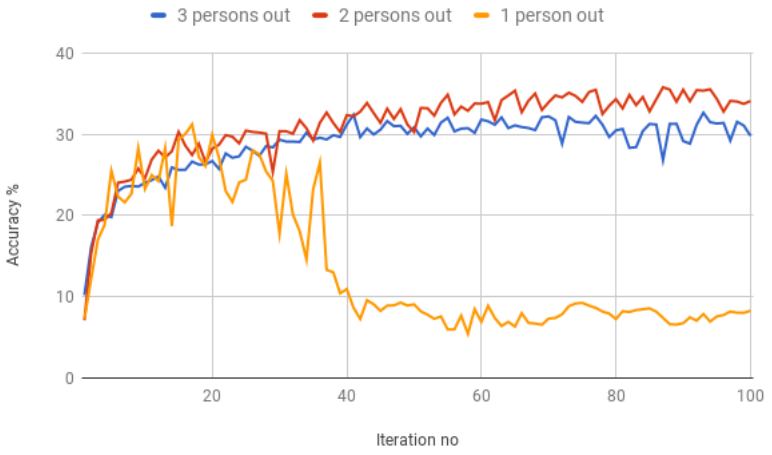


FIGURE 5.31: Classification accuracy for persons out of the training (DB_1, DB_2 and DB_3) using CNN approaches, even with low results CNN still performs better than PCA+k-NN (Figure 5.32)

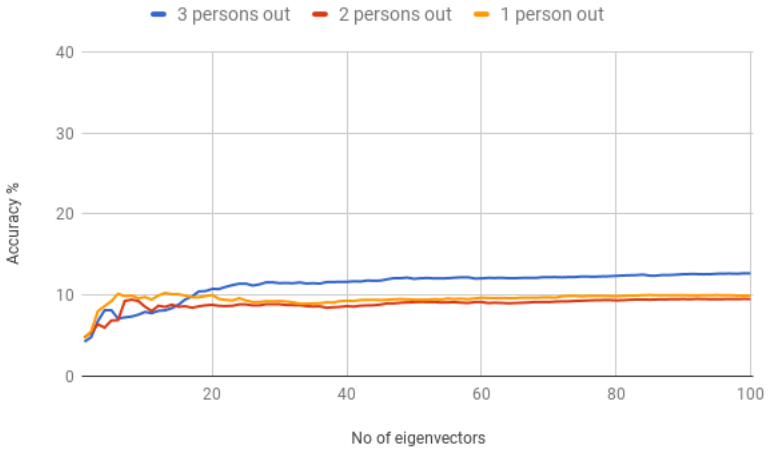
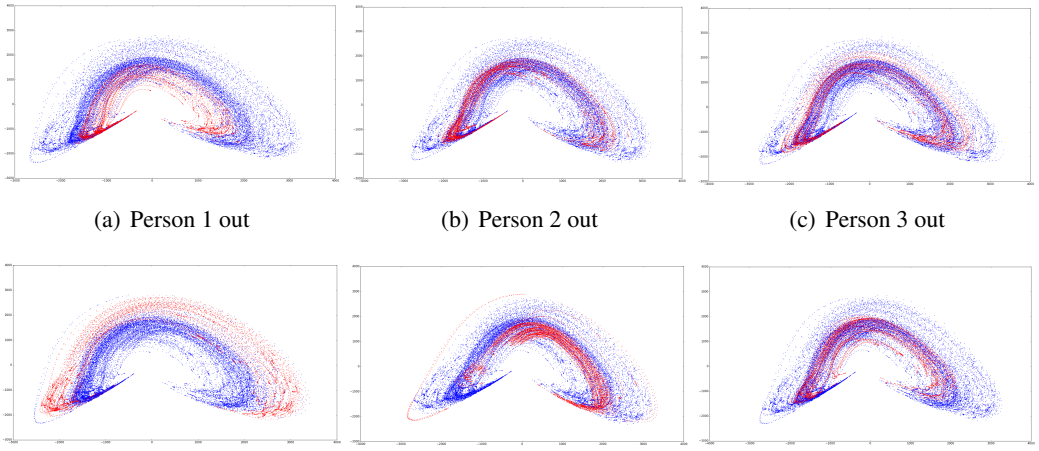


FIGURE 5.32: Accuracy for persons out of the training dataset (DB_1, DB_2 and DB_3) using PCA+k-NN approach, note accuracy is lower than CNN (Figure 5.31)

Figure 5.33 shows PCA manifolds for different persons out of the training set DB_1. For this experiment 100 eigenvectors were used, applied over non-blurred images. The classifier used was k-NN with $k = 1$. Blue dots represent the training set and the red dots the person used to test. Note that testing points appear in different position in the space, causing different accuracy. Table 5.6 shows the accuracy for each person and the number of training and testing images. In addition, the same table shows accuracy for DB_2 and DB_3 with different persons out of the training set.



(a) Person 1 out (b) Person 2 out (c) Person 3 out

(d) Person 4 out (e) Person 5 out (f) Person 6 out

FIGURE 5.33: PCA manifolds for different persons out of the training set, variations of DB_1

TABLE 5.6: Classifier performances in the DB_1, DB_2 and DB_3 according to person(s) out (PCA with 100 eigenvectors, non-blurred images and PCA+k-NN

Person(s) out	Accuracy (%)	N_{train}	N_{test}
1	9.18	21,043	7,914
2	8.45	20,433	9,134
3	12.57	20,910	8,180
4	14.10	20,735	8,530
5	**23.76**	20,858	8,284
6	11.22	21,021	7,958
1, 3	10.49	33,512	16,488
2, 4	17.52	33,536	16,464
3, 5	11.47	32,336	17,664
4, 6	11.48	33,906	16,094
1, 5, 5	14.32	25,622	24,378
2, 4 ,6	10.12	24,378	25,622

5.3.5 Kernel PCA

Kernel Principal Component Analysis (Kernel PCA or KPCA) is a non-linear extension of PCA. Generally, Kernel PCA can provide an improved recognition rate compared to classical PCA. The reasons for this improvement are: Kernel PCA uses an arbitrary number of non-linear components, whereas classical PCA uses just a limited number of linear principal components; Kernel PCA has more flexibility than ordinary PCA since it can choose different kernel functions, e.g. Gaussian kernel and polynomial kernel, for different recognition tasks

Wu et al. (2015).

The idea of Kernel PCA is to map the input space into a feature space by non-linear mapping and compute the principal components in that feature space. In any algorithm that can be expressed simply in terms of dot products, this kernel method enables the construction of different non-linear versions of the original PCA algorithm. When compared to other non-linear methods, one advantage of Kernel PCA is that it does not involve non-linear optimisation. It only requires linear algebra, making it as simple as classical PCA Lee et al. (2004).

Figure 5.34 shows the relation between a linear PCA and a Kernel PCA. The basic idea is to use a kernel function k instead of a dot product. Thus, kernel PCA is performed in a possibly high-dimensional space F. The dotted lines are lines of constant feature value. This kernel function is similar to the kernel function used in SVM Scholkopf et al. (2012). Therefore, in the case of a polynomial kernel, rather than using the dot product function $k(x,y) = (x \cdot y)$ the function $k(x,y) = (x \cdot y)^d$ is used, where d is the degree of the polynomial kernel, this function changes according to the chosen kernel.

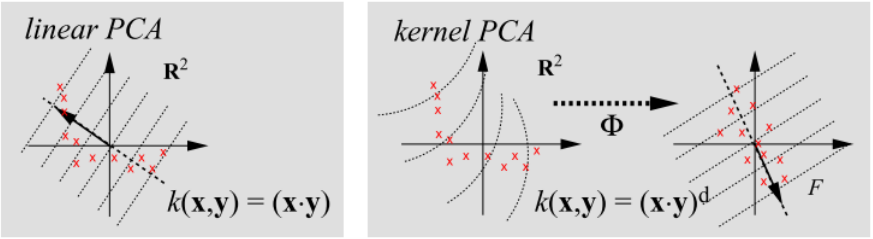
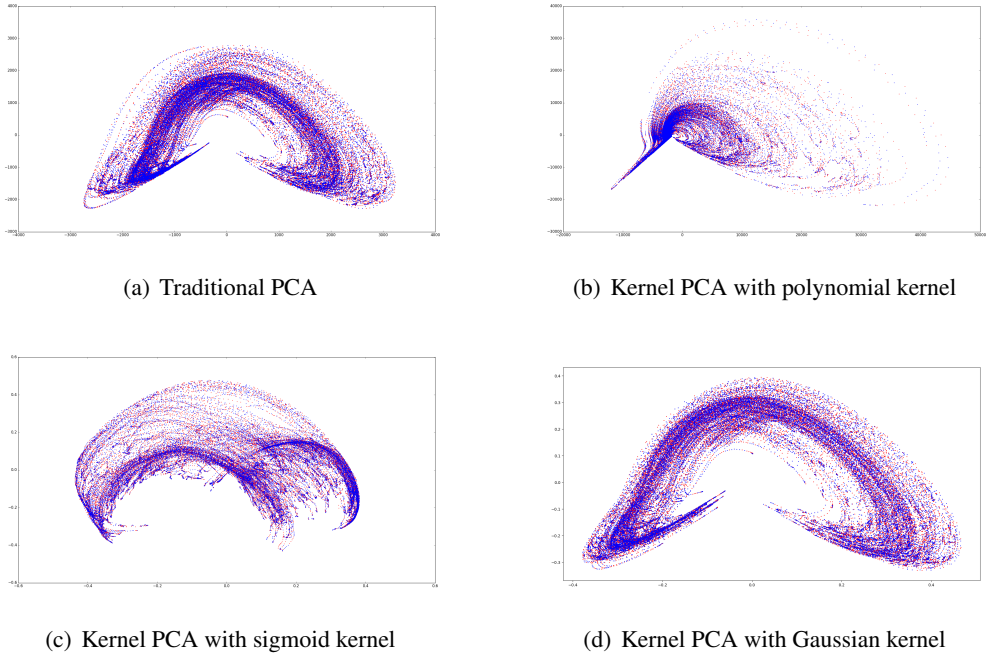


FIGURE 5.34: Illustration of linear PCA and Kernel PCA

In this section Kernel PCA is applied over the DB_r dataset. For this approach three different kernels were tested and the results compared to traditional PCA. The classifier used was k-NN with $k = 1$ and Euclidean distance. The first kernel is the polynomial kernel with degree 3, the second kernel is the sigmoid kernel, and the third is the Gaussian kernel.

Figure 5.35 shows manifold for traditional PCA and for the three different kernels in KPCA. Note that the shape depends completely on the kernel used for the Kernel PCA approach. Note that traditional PCA and Kernel PCA showed very similar shapes. Blue dots

represent the training images and the red dots the testing images for DB_r and non-blurred images.



(a) Traditional PCA

(b) Kernel PCA with polynomial kernel

(c) Kernel PCA with sigmoid kernel

(d) Kernel PCA with Gaussian kernel

FIGURE 5.35: Comparison of manifolds for different dimensional reduction approaches, PCA and KPCA with different kernel functions

Figure 5.36 shows the accuracy for Gaussian Kernel PCA + k-NN according to gamma for DB_r and non-blurred images. Note that as gamma decreases the accuracy increases until it reaches a plateau.

Figure 5.37 shows the accuracy for PCA + k-NN approach and Kernel PCA + k-NN with different kernels for DB_r and non-blurred images. Note that the best accuracy is still given by traditional PCA followed by Gaussian kernel with gamma = 1/100000000.

Table 5.7 shows the accuracy for DB_i and DB_r using Kernel PCA with polynomial kernel (degree 3) and k-NN classifier with $k = 1$ and Euclidean distance. Note that accuracy is improved for the iterative selection of training and testing split dataset DB_i, probably because it tends to have more similar frames. In addition, both datasets show improved accuracy for blurred images.

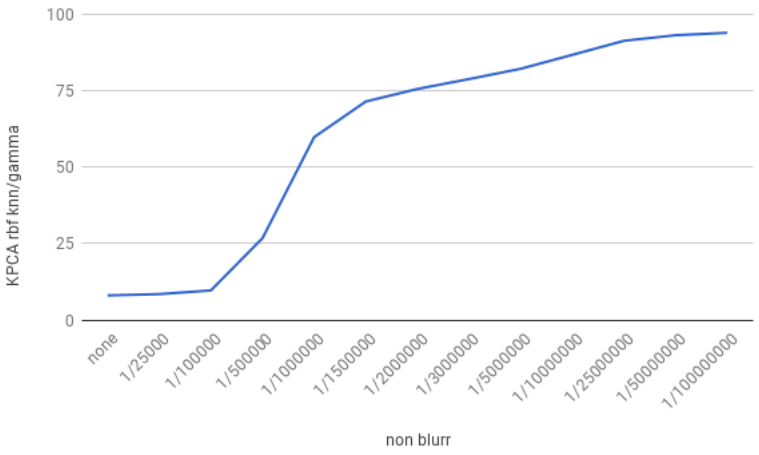


FIGURE 5.36: Accuracy according to the value of gamma for Gaussian kernel in KPCA

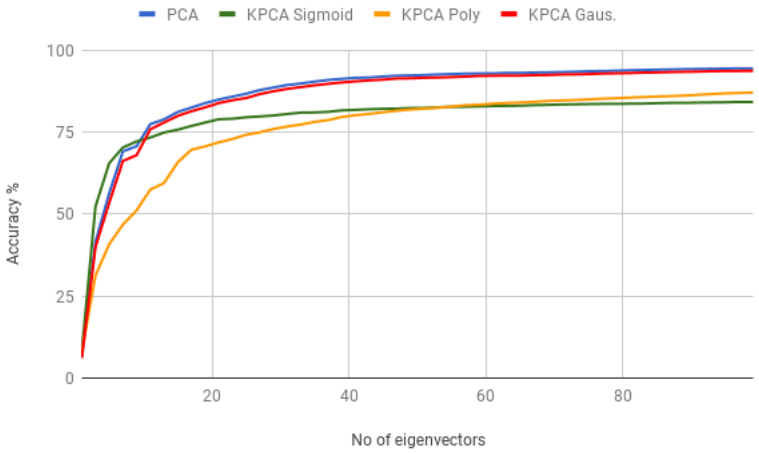


FIGURE 5.37: Accuracy according to the kernel function in KPCA and traditional PCA + k-NN, PCA performed the highest accuracy

TABLE 5.7: Classifier performances for DB_i and DB_r with KPCA, blurred and non-blurred images

DB	Non-blur.	Blurred
DB_i	**91.74%**	**95.14%**
DB_r	87.24%	91.05%

5.4 Conclusions

In this chapter a new dataset for Irish Sign Language (ISL) was introduced. Furthermore, a filter to select the most distinct frames was proposed. Finally, different techniques were tested

over this large dataset containing 50,000 images and 23 handshapes for ISL. This paves the way for more in-depth research in this area that has been hindered by the lack of large public datasets.

Two manners of separating the testing and the training dataset were proposed as well. The first was an iterative method where every second image was a test image. The second method was a totally random selection. It was followed by Gaussian blurring being applied over images and accuracy tested over non-blurred and blurred cases. Blurring showed improved accuracy in the majority of the cases.

A comparative study of using end-to-end approaches and feature-based approaches in recognising handshapes were reported. Results showed that, contrary to common belief, handcrafted features are still strongly competitive against deep features extracted using convolutional neural networks (CNN).

Mainly, experiments were done using Principal Components Analysis (PCA) and CNNs for handshape recognition. On the one hand, CNN showed more accurate performances than PCA/KPCA for non-blurred images in DB_r. On the other hand PCA+SVM showed a higher accuracy for non-blurred images among the feature-based approaches for the same DB_r. Although the improvement is slight, it is worth noting that CNN did not need any pre-processing, while for PCA blurring was needed to improve the accuracy. Furthermore, time for classification is considerably shorter for the PCA-based approach than for CNN. In addition, some outputs of the eigenvectors of the PCA approach were shown and an initial analogy between PCA and CNN was traced.

The overall best accuracy was shown with PCA+SVM for blurred images in the DB_i dataset. Most probably because this iterative selection method of splitting testing and training images is vulnerable to overfitting. Still for DB_i, the SVM classifier showed the best accuracy in all cases. However, SVM is the slowest classifier followed by CNN. PCA+MLP showed 99.32% accuracy for DB_i and k-NN applied direct over images showed 99.28% with $k = 1$ for the iterative selection (DB_i). CNN showed extremely high accuracy in all tested cases (all over 99%).

The worst case scenario was for the end-to-end approach using MLP with accuracy around 5.5%, which can imply that MLP depends on the kind of features. The second worst case was for the classifier LDA with or without PCA. Considering time for classification, the best accuracy and speed was given by PCA approaches especially for PCA+MLP.

One last experiment was done with a non-linear manifold learning technique Kernel PCA. KPCA was applied over the dataset with polynomial kernel and k-NN to classify. Comparing to PCA+k-NN, KPCA+k-NN showed a lower accuracy. It proves that non-linear PCA does not necessarily work better than the traditional linear PCA.

Chapter 6

Conclusions

This chapter concludes this thesis showing a summary of the research, contributions for the community and conclusions taken after a journey in handshape recognition studies. Finally directions for future research are presented.

6.1 Summary

This thesis proposed a new dataset for Irish Sigh Language (ISL) extending the previous dataset proposed by Farouk (2015) from 20 shapes to 23 static shapes plus 3 dynamic shapes. This new dataset is composed of videos and frames from six different human subjects performing hand gestures in a plane rotation.

Furthermore, a sequence of experiments with PCA was introduced with the use of interpolation of manifolds and eigenspaces creating artificial data able to recognise new incoming objects with an accuracy higher than the original *sparse* dataset.

Finally, an extensive comparative study was done with different techniques (feature-based approaches and end-to-end approaches) for hand gesture recognition, using different classifiers; especially comparing PCA with CNN for the *dense* dataset.

6.2 Conclusions

In this thesis a new dataset for Irish Sign Language was proposed. Furthermore, a method was designed to filter the redundant images with an iterative image selection process in order to select the images which keep the dataset diverse. This dataset contains 468 videos and more than 58,000 images for the 26 letters of the ISL alphabet and 50,000 images after the filter been applied to the 23 static handshapes. This paves the way for more in-depth research in this area that was hindered by the lack of large public datasets.

In Chapter 4 Principal Component Analysis (PCA) was applied in more than one stage, over a subset of blurred images of the proposed dataset in order to massive reduce the dimensionality. Interpolation was explored for missing rotations and translations in order to make datasets more robust, able to recognize a shape in any rotation and translation, even if the translated or rotated image is not contained in the training dataset. Splines were used to interpolate between manifolds and eigenspaces creating artificial data. This interpolation was important because the illumination changes according to the arm rotation. The results showed an improved accuracy when compared to the real data, answering Research Questions 1 (Is it possible to use two-stage PCA and interpolation to generate artificial data which can augment a sparse dataset?) and 2 (Does the use of interpolated data increase the recognition accuracy on a sparse dataset?). In addition, was shown how the number of eigenvectors, blurring level and interval size influences the accuracy, answering Reserach Question 3 (How do parameters such as blurring level, number of eigenvectors or sampling interval affect the accuracy?).

In Chapter 5 results of using PCA and Convolutional Neural Networks (CNNs) for handshape recognition were reported. Experiments were made over the handshape images of the new Irish Sign Language *dense* dataset, for blurred and non-blurred images. Improved performances were obtained with CNNs compared to using PCA. Notwithstanding the fact that the increase in accuracy is slight, it is important to note that CNN did not need pre-processing. However, time for classification is considerably shorter for PCA-based approaches than for

CNN and other end-to-end classifiers. This answered Research Questions 4 (How do feature-based and end-to-end algorithms compare in recognition accuracy on a dense dataset?).

A comparative study of using end-to-end approaches and feature-based approaches in recognising handshapes was presented as well. The results showed that, opposed to common belief, handcrafted features are still strongly competitive against deep features extracted using CNN. In addition, an experiment with a non-linear Kernel PCA was done showing that it does not provide necessarily improved separation of the data. Finally, some outputs of the eigenvectors of the PCA approach were shown and an initial analogy between PCA and CNN was traced.

6.3 Research Contributions

This thesis proposed three different contributions to the field of handshape recognition applied to ISL. The first contribution was a new dataset for ISL with a redundancy filter. The second was the use of interpolation over PCA manifolds and PCA eigenspaces. The third contribution was a comparative study between end-to-end and PCA-based approaches.

- **New Irish Sigh Language Alphabet Dataset** A new dataset for ISL was proposed. The dataset contains 468 videos, filmed from 6 different human subjects (3 males and 3 females), resulting in more than 58,000 frames representing the 26 alphabet letters. A selection of 50,000 frames for 23 static gestures were made with a proposed redundancy filtering. This dataset built on previous dataset with only 20 handshapes and a rather smaller number of images and it is public available online.

- **PCA with Interpolation** PCA was applied in two-stages over the *sparse* dataset and translations were artificially added in order to make the dataset more robust. Splines were used to interpolate data and eigenspaces in order to create artificial data from the original one. These data were either the projections of images into the eigenspaces (manifolds) or the eigenspaces themselves (set of eigenvectors). They were used to

improve the recognition accuracy over a *sparse* dataset significantly. In addition, it was shown how parameters such as blurring level, number of eigenvectors or sampling interval affected the accuracy.

- **Comparison between CNN and PCA** A deeper study was proposed with end-to-end approaches (mainly CNN) and feature-based (PCA, KPCA), and different classifiers. Probably for the first time a selection of different approaches and classifiers were applied to one large dataset for ISL, showing the accuracy for each approach and how blurring can affect the recognition process. All these experiments were done over the *dense* dataset. Feature-based approaches are competitive in recognition accuracy with end-to-end approaches having the advantage of time performance.

6.4 Directions for Future Research

Handshape recognition still has a long way to go in the research path, especially for 2D systems. This research offers interesting ideas for future research. Some of these possibilities are described in this section.

- **Dynamic Gesture Recognition** As this thesis focused only on static hand gesture recognition, one simple step forward is to recognize the dynamic shapes for the ISL (J, X and Z). The dataset proposed already have videos and frames for these 3 shapes.

- **Recognition from Videos** Nowadays videos are commonly found on the internet. Actually, the idea of classifying single frames is a start to classifying frames in videos. This could be applied in real time systems. Extending the algorithms proposed in this thesis to video and building an automatic transcript system is an important step forward. For this purpose, it might be interesting to explore sequential models that take into account the time dimension, such as recurrent neural networks and hidden Markov models or a neural architecture combining CNNs and RNNs.

- **Test different techniques for the *sparse* dataset** As was done in Chapter 5 it would be interesting to see how different techniques behave for a *sparse* dataset such as the one used in Chapter 1. Even though it is known that not every technique allows interpolation, at least different classifiers could be tested over the *sparse* dataset and over the interpolated data.

- **Deeper Comparisons** This thesis proposed an initial comparison between PCA and CNN. Is is clearly possible to research deeper in this comparative evaluation with shallow and deep models and a deeper analysis between the outputs of the PCA and CNN approaches.

- **Non-linear Techniques** PCA is well known for dimensionality reduction and widely used in computer vision problems. However, it is known that PCA extracts features linearly which may not be the best way to extract features from handshape images/videos. Therefore, applying non-linear manifold learning techniques such as Isomap, Laplacian eigenmaps, Multidimensional scaling, Isometric Feature Mapping, Locally Linear Embedding and Non-linear Principal Component Analysis can lead this research to a step forward for automatic Sign Language recognition. In this thesis a single experiment was done with Kernel Principal Component Analysis.

- **Extending to 3D** 3D cameras and sensors are getting more common and less expensive every day. This kind of sensor can provide much more information about the hand, making it possible to create more precise systems for sign language real time recognition.

- **Extension of the Dataset** Even though that thesis introduces a new dataset with a rather more frames for Irish Sign Language (Chapter 3), it still does not provide all the possible movements for ISL. Videos with rotation in 3D, words and expressions are examples of how this dataset can be extended.

Bibliography

Abdulla, S. and Manaf, R. (2016). Design and Implementation of A Sign-to- Speech/Text System for Deaf and Dumb People. *5th Int. Conf. Electron. Devices, Syst. Appl.*, pages 3–6.

Ahmed, M., Idrees, M., Mumtaz, R., Khalique, S., and Science, C. (2016). Deaf Talk Using 3D Animated Sign Language. *SAI Comput. Conf.*, pages 330–335.

Al-Taie, I., Azeez, N., Basbrain, A., and Clark, A. (2017). The Effect of Distance Similarity Measures on the Performance of Face , Ear and Palm Biometric Systems. *Digit. Image Comput. Tech. Appl.*, pages 254–260.

Althoff, F., Lindl, R., and Walchshäusl, L. (2005). Robust multimodal hand- and head gesture recognition for controlling automotive infotainment systems. *VDI Berichte*, (1919):187–205.

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. (2012). Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. *Ambient Assist. Living Home Care*, 7657:216–223.

Aran, O. and Akarun, L. (2010). A multi-class classification strategy for Fisher scores: Application to signer independent sign language recognition. *Pattern Recognit.*, 43(5):1776–1788.

Arkenbout, E. A., de Winter, J. C. F., and Breedveld, P. (2015). Robust hand motion tracking through data fusion of 5dt data glove and nimble VR kinect camera measurements. *Sensors (Switzerland)*, 15(12):31644–31671.

Barth, T. J., Keyes, D. E., and Roose, D. (2008). *Principal Manifolds for Data Visualization and Dimension Reduction*, volume 58.

Baryshnikov, Y. DIMENSIONALITY REDUCTION AND MANIFOLD LEARNING. Available from: https://faculty.math.illinois.edu/{~}ymb/nrv/tutorial/part2.html{#}/1.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359.

Bedregal, B., Dimuro, G. P., and Costa, A. C. d. R. (2007). Hand Gesture Recognition in an Interval Fuzzy Approach. *Trends Appl. Comput. Math.*, 8(1):21–31.

Belkin, M. and Niyogi, P. (2001). Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. *Nips*, 14:585–591.

Biswas, K. K. and Basu, S. K. (2011). Gesture recognition using Microsoft Kinect®. *Autom. Robot. Appl. (ICARA), 2011 5th Int. Conf.*, 2:100–103.

Bui, T. D. and Nguyen, L. T. (2007). Recognizing postures in vietnamese sign language with MEMS accelerometers. *IEEE Sens. J.*, 7(5):707–712.

Cerlinca, T. I. and Pentiuc, S. G. (2011). Robust 3D hand detection for gestures recognition. *Stud. Comput. Intell.*, 382:259–264.

Chaczko, Z. and Alenazy, W. (2016). Modelling Gesture Recognition Systems Zenon. *J. Softw. Syst. Dev.*, 2016:11.

Chai, X., Wang, H., and Chen, X. (2014). The devisign large vocabulary of chinese sign language database and baseline evaluations. Technical report, echnical report VIPL-TR-14-SLR-001. Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS.

Chan, T. H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2015). PCANet: A Simple Deep Learning Baseline for Image Classification? *IEEE Trans. Image Process.*, 24(12):5017–5032.

Charfi, N., Trichili, H., Alimi, A. M., and Solaiman, B. (2017). Bimodal biometric system for hand shape and palmprint recognition based on SIFT sparse representation. *Multimed. Tools Appl.*, 76(20):20457–20482.

Chen, Q., Georganas, N. D., and Petriu, E. M. (2007). Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. *2007 IEEE Instrum. Meas. Technol. Conf. IMTC 2007*, pages 1–6.

Chen, Q., Xue, B., Sun, Q., and Xia, D. (2010). Interactive image segmentation based on object contour feature image. *2010 IEEE Int. Conf. Image Process.*, pages 3605–3608.

Cheng, H., Yang, L., and Liu, Z. (2015). Survey on 3D Hand Gesture Recognition. *IEEE Trans. Circuits Syst. Video Technol.*, PP(99):1.

Chomat, O., Colin, V., Hall, D., and Crowley, J. L. (2000). Local scale selection for Gaussian based description techniques. *Eur. Conf. Comput. Vis.*, pages 117–134.

Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. *IJCAI Int. Jt. Conf. Artif. Intell.*, pages 1237–1242.

Coleca, F., State, A., Klement, S., Barth, E., and Martinetz, T. (2015). Neurocomputing Self-organizing maps for hand and full body tracking. *Neurocomputing*, 147:174–184.

Coogan, T. a. (2007). *Dynamic Gesture Recognition Using Transformation Invariant Hand Shape Recognition*. PhD thesis, Dublin City University.

Cui, Y. and Weng, J. (2000). Appearance-Based Hand Sign Recognition from Intensity Image Sequences. *Comput. Vis. Image Underst.*, 78(2):157–176.

De Souza, C. R., Pizzolato, E. B., and dos Santos Anjo, M. (2013). Fingerspelling Recognition with Support Vector Machines and Hidden Conditional Random Fields. *Ibero-American Conf. Artif. Intell.*, pages 561–570.

den Bergh, M. V., Koller-Meier, E., Bosché, F., and Van Gool, L. (2009). Haarlet-based hand gesture recognition for 3D interaction. *Appl. Comput. Vis.*, pages 1–8.

D'Errico, J. Interparc Function. Available from: http://www.mathworks.com/matlabcentral/{%}0Afileexchange/34874-interparc [Accessed 08/04/2017].

Dreuw, P., Deselaers, T., Keysers, D., and Ney, H. (2006). RWTH German Fingerspelling Database. Available from: http://www-i6.informatik.rwth-aachen.de/{~}dreuw/fingerspelling.php [Accessed 07/01/2018].

Droeschel, D., Stückler, J., and Behnke, S. (2011). Learning to interpret pointing gestures with a time-of-flight camera. *Proc. 6th Int. Conf. Human-robot Interact. - HRI '11*, pages 481–488.

Easton, R. L. (2010). *Fundamentals of Digital Image Processing*. Number November.

Elgammal, A. and Lee, C. S. (2007). Nonlinear manifold learning for dynamic shape and dynamic appearance. *Comput. Vis. Image Underst.*, 106(1):31–46.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. (2007). Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.*, 108(1-2):52–73.

Farouk, M. (2015). *Principal Component Pyramids using Image Blurring for Nonlinearity Reduction in Hand Shape Recognition*. PhD thesis, Dublin City University.

Farouk, M., Sutherland, A., and Shokry, A. (2013). Nonlinearity Reduction of Manifolds using Gaussian Blur for Handshape Recognition based on Multi-Dimensional Grids. *ICPRAM*, pages 303–307.

Farouk, M., Sutherland, A., and Shoukry, A. A. (2009). A multistage hierarchical algorithm for hand shape recognition. *IMVIP 2009 - 2009 Int. Mach. Vis. Image Process. Conf.*, pages 105–110.

Freeman, W. T., Anderson, D. B., Beardsley, P. A., Dodge, C. N., Roth, M., Weissman, C., Yerazunis, W. S., Kage, H., Kyuma, K., Miyake, Y., and Tanaka, K. I. (1998). Computer vision for interactive computer graphics. *IEEE Comput. Graph. Appl.*, 18(3):42–52.

Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2010). Real time motion capture using a single time-of-flight camera. *2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 755–762.

Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmos. Environ.*, 32(14-15):2627–2636.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *Comput. Vis. Pattern Recognition)*, pages 3354–3361.

Ghodsi, A. (2006). Dimensionality Reduction A Short Tutorial. Technical report, University of Waterloo, Waterloo.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*.

Gutmann, M., Hyvärinen, A., and Aihara, K. (2008). Learning encoding and decoding filters for data representation with a spiking neuron. *Proc. Int. Jt. Conf. Neural Networks*, (2):243–248.

Ham, J. H., Lee, D. D., Mika, S., and Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. Technical Report 47.

Han, C. C. (2004). A hand-based personal authentication using a coarse-to-fine strategy. *Image Vis. Comput.*, 22(11):909–918.

Han, F. and Liu, H. (2014). Scale-Invariant Sparse PCA on High Dimensional Meta-elliptical Data. *J. Am. Stat. Assoc.*, 109(505):275–287.

Harding, P. and Ellis, T. (2004). Recognizing Hand Gesture using Fourier Descriptors. pages 3–6.

Huang, C.-L. and Jeng, S.-H. (2001). A model-based hand gesture recognition system. *Mach. Vis. Appl.*, 12(5):243–258.

Huang, J. and Yuan, C. (2015). Weighted-PCANet for Face Recognition. *Neural Inf. Process.*, 9492:246–254.

Johnson, S., Stedinger, J. R., Shoemaker, C. A., and Li, Y. (1993). Numerical solution of continuous-state dynamic programs using linear and spline interpolation. pages 484–500.

Jolliffe, I. (2014). *Principal Component Analysis*. John Wiley & Sons, Ltd.

Kambhatla, N. and Leen, T. (1997). Dimension reduction by local principal component analysis. *Neural Comput.*, 9(7):1493–1516.

Khademi, M., Hondori, H. M., Mckenzie, A., Dodakian, L., Lopes, C. V., and Cramer, S. C. (2014). Free-Hand Interaction with Leap Motion Controller for Stroke Rehabilitation. *Proc. Ext. Abstr. 32nd Annu. ACM Conf. Hum. factors Comput. Syst.*, pages 1663–1668.

Kim, T., Keane, J., Wang, W., Tang, H., Riggle, J., Shakhnarovich, G., Brentari, D., and Livescu, K. (2017). Lexicon-free fingerspelling recognition from video: Data, models, and signer adaptation. *Comput. Speech Lang.*, 46:209–232.

Konda, K. (2012). Real Time Interaction with Mobile Robots using Hand Gestures. pages 177–178.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.*, pages 1–9.

Kumar, P., Gauba, H., Pratim Roy, P., and Prosad Dogra, D. (2017). A multimodal framework for sensor based sign language recognition. *Neurocomputing*, 259:21–38.

Lahiani, H., Elleuch, M., and Kherallah, M. (2015). Real Time Hand Gesture Recognition System for Android Devices. In *Intell. Syst. Des. Appl.*, pages 591–596.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. a., Sackinger, E., Simard, P., and Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks Stat. Mech. Perspect.*, pages 261–276.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. *ISCAS 2010 - 2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pages 253–256.

Lee, J. M. (2012). *Introduction to Smooth Manifolds*.

Lee, J.-M., Yoo, C., Choi, S. W., Vanrolleghem, P. a., and Lee, I.-B. (2004). Nonlinear process monitoring using kernel principal component analysis. *Chem. Eng. Sci.*, 59(1):223–234.

Leeson, L. and Saeed, J. I. (2012). *Irish Sign Language : A Cognitive Linguistic Account*. Edinburgh University Press.

Li, S. Z. and Jain, A. K. (2011). *Handbook of face recognition*.

Licciardi, G., Avezzano, R. G., Del Frate, F., Schiavon, G., and Chanussot, J. (2014). A novel approach to polarimetric SAR data processing based on Nonlinear PCA. *Pattern Recognit.*, 47(5):1953–1967.

Licciardi, G. A., Dambreville, R., Chanussot, J., and Dubost, S. (2015). Spatiotemporal pattern recognition and nonlinear PCA for global horizontal irradiance forecasting. *IEEE Geosci. Remote Sens. Lett.*, 12(2):284–288.

Lindeberg, T. (2013). Scale selection properties of generalized scale-space interest point detectors. *J. Math. Imaging Vis.*, 46(2):177–210.

Lindeberg, T. (2015). *Image Matching Using Generalized Scale-Space Interest Points*, volume 52. Springer US.

Liu, K. and Kehtarnavaz, N. (2016). Real-time robust vision-based hand gesture recognition using stereo images. *J. Real-Time Image Process.*, 11(1):201–209.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.*, 60(2):91–110.

Maebatake, M., Suzuki, I., Nishida, M., Horiuchi, Y., and Kuroiwa, S. (2008). Sign language recognition based on position and movement using multi-stream HMM. *Proc. 2nd Int. Symp. Univers. Commun. ISUC 2008*, pages 478–481.

Malawski, F. (2014). Applying Hand Gesture Recognition with Time-of-Flight Camera for 3D Medical Data Analysis. *Challenges Mod. Technol.*, 5(4):12–16.

Maraqa, M. and Abu-Zaiter, R. (2008). Recognition of Arabic Sign Language (ArSL) Using Recurrent Neural Networks. *1st Int. Conf. Appl. Digit. Inf. Web Technol. ICADIWT 2008*, pages 478–481.

Marin, G., Dominio, F., and Zanuttigh, P. (2014). Hand gesture recognition with leap motion and kinect devices. *2014 IEEE Int. Conf. Image Process. ICIP 2014*, pages 1565–1569.

Mistry, P., Maes, P., and Chang, L. (2009). WUW-wear Ur world: a wearable gestural interface. *CHI'09 Ext. Abstr. Hum. factors Comput. Syst.*, pages 4111–4116.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, 37(3):311–324.

Moeslund, T. B., Hilton, A., and Kruger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2-3 SPEC. ISS.):90–126.

Mohan, A., Papageorgiou, C., and Poggio, T. (2001). Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):349–361.

Mokhtarian, F. and Suomela, R. (1998). Robust image corner detection through curvature scale space. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1376–1381.

Nagi, J., Ducatelle, F., Caro, G. A. D., Cires, D., Meier, U., Giusti, A., and Gambardella, L. M. (2011). Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition. *IEEE Int. Conf. Signal Image Process. Appl.*, pages 342–347.

Nahar, M. and Ali, S. (2014). An Improved Approach for Digital Image Edge Detection. *Int. J. Recent Dev. Eng. Technol.*, 2(3):14–20.

Naoum, R., Owaied, H., and Joudeh, S. (2012). Development of a new Arabic sign language recognition using k-nearest neighbor algorithm. *J. Emerg. Trends Comput. Inf. Sci.*, 3(8):1173–1178.

Neidle, C., Thangali, A., and Sclaroff, S. (2012). American Sign Language Lexicon Video Dataset (ASLLVD). Available from: http://www.bu.edu/av/asllrp/dai-asllvd.html [Accessed 07/01/2018].

Networks, N. (2013). Neural Networks. Available from: http://ufldl.stanford.edu/wiki/index.php/Neural{_}Networks [Accessed 25/07/2017].

Ney, I., Dreuw, P., Seidl, T., and Keysers, D. (2005). Appearance-Based Gesture Recognition.

Oprisescu, S., Rasche, C., and Su, B. (2012). Automatic static hand gesture recognition using ToF cameras. *Eur. Signal Process. Conf.*, (Eusipco):2748–2751.

Pedersoli, F., Benini, S., Adami, N., and Leonardi, R. (2014). XKin: An open source framework for hand pose and gesture recognition using kinect. *Vis. Comput.*, 30(10):1107–1122.

Phil, L. T., Nguyen, H. D., Suil, T. T. Q., and Vul, T. T. (2015). A Glove-Based Gesture Recognition System for Vietnamese Sign Language. *15th Int. Conf. Control. Autom. Syst.*, (Iccas):1555–1559.

Potter, L. E., Araullo, J., and Carter, L. (2013). The Leap Motion controller. *Proc. 25th Aust. Comput. Interact. Conf. Augment. Appl. Innov. Collab. - OzCHI '13*, (February 2016):175–178.

Press, O. U. (2017). Oxford University Press. Available from: https://en.oxforddictionaries.com/ [Accessed 17/05/2017].

Quesada, L., López, G., and Guerrero, L. (2017). Automatic recognition of the American sign language fingerspelling alphabet to assist people living with speech or hearing impairments. *J. Ambient Intell. Humaniz. Comput.*, 8(4):625–635.

Quiroga, F., Antonio, R., Ronchetti, F., Lanzarini, L., and Rosete, A. (2017). A Study of Convolutional Architectures for Handshape Recognition applied to Sign Language. *XXIII Congr. Argentino Ciencias la Comput.*, pages 13–22.

Rautaray, S. S. and Agrawal, A. (2012). Vision based hand gesture recognition for human computer interaction: a survey. *Artif. Intell. Rev.*, 43(1):1–54.

Raytchev, B., Yoda, I., and Sakaue, K. (2004). Head Pose Estimation by Nonlinear Manifold Learning. *IEEE Int. Conf. Pattern Recognit.*, pages 462–466.

Ren, Y. and Zhang, F. (2009). Hand Gesture Recognition Based on MEB-SVM. *2009 Int. Conf. Embed. Softw. Syst.*, pages 344–349.

Ren, Z., Meng, J., Yuan, J., and Zhang, Z. (2011). Robust Hand Gesture Recognition with Kinect Sensor. *Proc. 19th ACM Int. Conf. Multimed.*, pages 759–760.

Roberto, d. S. C. and Pizzolato, E. B. (2013). Sign Language Recognition with Support Vector Machines and Hidden Conditional Random Fields. *Int. Work. Mach. Learn. Data Min. Pattern Recognit.*, pages 561–570.

Ronchetti, F., Quiroga, F., Estrebou, C., and Lanzarini, L. (2016). Handshape recognition for Argentinian Sign Language using ProbSom. *J. Comput. Sci. Technol.*, 16(1):1–5.

Roweis, S. T. and Saul, L. K. (2001). Nonlinear dimensionality reduction by locally linear embedding. *Science (80-.).*, 290(1994):2323–2326.

Safaei, A. and Wu, Q. M. J. (2015). Evaluating 3D Hand Motion with a Softkinetic Camera. *Proc. - 2015 IEEE Int. Conf. Multimed. Big Data, BigMM 2015*, pages 290–291.

Sahoo, A. K., Mishra, G. S., and Ravulakollu, K. K. (2014). Sign Language Recognition : State of the Art. *Asian Res. Publ. Netw.*, 9(2):116–134.

Samer, C. H., Rishi, K., and Rowen (2015). Image Recognition Using Convolutional Neural Networks. *Cadence Whitepaper*, pages 1–12.

Saxena, A. and Gupta, A. (2004). Non-linear dimensionality reduction by locally linear isomaps. *Neural Inf. Process.*, pages 1038–1043.

Scholkopf, B., Smola, a. J., and Muller, K. R. (2012). Kernel Principal Component Analysis. *Comput. Vis. Math. Methods Med. Biomed. Image Anal.*, 1327:583–588.

Scholz, M., Fraunholz, M., and Selbig, J. (2008). Nonlinear principal component analysis: neural network models and applications. *Princ. manifolds data Vis. Dimens. Reduct.*, pages 45–68.

Scikit, P. 3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.19.1 documentation. Available from: http://scikit-learn.org/stable/modules/cross{_}validation.html [Accessed 24/10/2017].

Seung, H. S. and Lee, D. D. (2000). The Manifold Ways of Perception. *Science (80-.).*, 290(5500):2268–2269.

Shakhnarovich, G., Viola, P., and Darrell, T. (2003). Fast pose estimation with parameter-sensitive hashing. *Proc. Ninth IEEE Int. Conf. Comput. Vis.*, (April):0–11.

Shu, X. and Wu, X. J. (2011). A novel contour descriptor for 2D shape matching and its application to image retrieval. *Image Vis. Comput.*, 29(4):286–294.

Signbank, A. (2017). Auslan Signbank. Available from: http://www.auslan.org.au/ [Accessed 07/01/2018].

Silva, V. D. and Tenenbaum, J. B. (2003). Global Versus Local Methods in Nonlinear Dimensionality Reduction. *Adv. Neural Inf. Process. Syst.*, 15:705–712.

Softkinetic (2017). SOFTKINETIC - 3D Vision Leader. Available from: https://www.softkinetic.com/ [Accessed 01/03/2017].

Song, J., Sörös, G., Pece, F., Fanello, S. R., Izadi, S., Keskin, C., and Hilliges, O. (2014). In-air gestures around unmodified mobile devices. *Uist 2014*, pages 319–329.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Stockman, G. and Shapiro, L. G. (2001). *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Suarez, J. and Murphy, R. R. (2012). Hand gesture recognition with depth images: A review. *Ro-Man, 2012 Ieee*, pages 411–417.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Hill, C., and Arbor, A. (2014). Going Deeper with Convolutions.

Taylor, B., Dey, A., Siewiorek, D., and Smailagic, A. (2017). Real-Time Depth-Camera Based Hand Tracking for ASL Recognition. *Proc. 19th Int. ACM SIGACCESS Conf. Comput. Access. - ASSETS '17*, pages 337–338.

Teng, X., Wu, B., Yu, W., and Liu, C. (2005). A hand gesture recognition system based on local linear embedding. *J. Vis. Lang. Comput.*, 16(5):442–454.

Tkach, A., Pauly, M., and Tagliasacchi, A. (2016). Sphere-meshes for real-time hand modeling and tracking. *ACM Trans. Graph.*, 35(6):1–11.

Tong, S. and Koller, D. (2001). Support Vector Machine Active Learning with Applications to Text Classification. *J. Mach. Learn. Res.*, pages 45–66.

Turk, M. and Hua, G. (2013). *Vision-based interaction*, volume 4.

Turk, M. and Pentland, A. (1991). Face Recognition Using Eigenfaces. *Comput. Vis. Pattern Recognit.*, pages 586–591.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. *Icml*, (1):109–111.

Wang, H., Leu, M. C., and Oz, C. (2006). American Sign Language Recognition Using Multi-dimensional Hidden Markov Models. *J. Inf. Sci. Eng.*, 22:1109–1123.

Wang, J., Liu, Z., Chorowski, J., Chen, Z., and Wu, Y. (2012). MSR Gesture3D dataset. Available from: http://users.eecs.northwestern.edu/{~}jwa368/my{_}data.html [Accessed 07/01/2018].

Wang, S., Chen, L., Zhou, Z., Sun, X., and Dong, J. (2016). Human fall detection in surveillance video based on PCANet. *Multimed. Tools Appl.*, 75(19):11603–11613.

Wang, S.-C. (2003). Artificial Neural Network. In *Interdiscip. Comput. java Program.*, pages 81—-100. Springer, US.

Wang, Z., Bi, Z., Wang, C., Lin, L., and Wang, H. (2015). Traffic Lights Recognition Based on PCANet. *Chinese Autom. Congr.*, pages 559–564.

Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D. (2013). Analysis of the accuracy and robustness of the Leap Motion Controller. *Sensors (Switzerland)*, 13(5):6380–6393.

Weinberger, K. Q. and Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comput. Vis.*, 70(1):77–90.

Wolf, M. T., Assad, C., Stoica, A., You, K., Jethani, H., Vernacchia, M. T., Fromm, J., and Iwashita, Y. (2013). Decoding static and dynamic arm and hand gestures from the JPL BioSleeve. *IEEE Aerosp. Conf. Proc.*

Wu, D., Wu, J., Zeng, R., Jiang, L., Senhadji, L., and Shu, H. (2015). Kernel principal component analysis network for image classification. *arXiv Prepr. arXiv1512.06337*, page 7.

Wu, R., Yu, Y., and Wang, W. (2013). SCaLE: Supervised and cascaded laplacian eigenmaps for visual object recognition based on nearest neighbors. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 867–874.

Yang, J., Yu, H., and Kunz, W. (2000). An Efficient LDA Algorithm for Face Recognition. *sixth Int. Conf. Control. Autom. Robot. Vis.*, pages 34—-47.

Yao, Y. and Fu, Y. (2014). Contour model-based hand-gesture recognition using the kinect sensor. *IEEE Trans. Circuits Syst. Video Technol.*, 24(11):1935–1944.

Zabulis, X., Baltzakis, H., and Argyros, a. (2009). Vision-based hand gesture recognition for human-computer interaction. *Univers. Access . . .*, pages 1–56.

Zaccone, G. (2016). *Getting Started with TensorFlow*. Packt Publishing Ltd.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method.

Zhang, X., Chen, X., Li, Y., Lantz, V., Wang, K., and Yang, J. (2011). A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, 41(6):1064–1076.

Zhao, W., Chellappa, R., and Krishnaswamy, S. (1998). Discriminant analysis of principal components for face recognition. *Face Recognit. From Theory to Appl.*, pages 73–85.

Zheng, L., Liang, B., and Jiang, A. (2017). Recent Advances of Deep Learning for Sign Language Recognition. *Digit. Image Comput. Tech. Appl.*, pages 454–460.