GENERATING SYNTHETIC DATA BY MORPHING TRANSFORMATION FOR HANDWRITTEN NUMERAL RECOGNITION (WITH v-SVM)

Sapargali Kambar

A Thesis

in

The Department

of

Computer Science & Software Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

August 2005

Your file Votre référence
ISBN: 0-494-10288-8
Our file Notre référence
ISBN: 0-494-10288-8

ABSTRACT

Generating Synthetic Data by Morphing Transformation for Handwritten Numeral Recognition (with ν-SVM)

Sapargali Kambar

The amount of training data is one of the critical factors affecting the performance of handwritten numeral recognition system. One way to increase the training data size is adding synthesized data. In this study, synthetic data generation using morphing transformation with convex evolution is investigated. This technique uses a pair of original samples as the source and the target, and generates the synthetic samples by evolving the source towards the target.

We aim to balance the data distribution. Normally, the training data has poor distribution due to the data redundancy and sparseness caused by frequent and rare samples, respectively. In terms of data clusters, some clusters are small, some are large and filling the gap between these clusters with synthetic data should smooth the clusters. Using the Support Vector Machines method, the rare samples, also called support vectors, are determined. Then, the number of rare samples is increased using morphing transformation.

Using this technique a recognition rate of 99.19% has been achieved, while the initial performance without morphing was 99.07%. Morphing transformation generated more representative synthetic samples, which cannot be obtained by the other data synthesis methods such as affine and elastic distortions.

Он екіден бір гүлі ашылмай о дүниелік болған
бөлелерім Медет пен Мекенбайға естелік ретінде

Acknowledgements

To my Supervisor Dr. Ching Y. Suen: I would like to thank you for your continual guidance and support as well as for your valuable suggestions and encouragement throughout my studies.

To Dr. Richard Zanibbi, Dr. Wumo Pan and Mr. Nicola Nobile: Special thanks to my best friends and colleagues for your competent arguments, critiques, and judgements, which pushed me to meet with the highest standards. Thank you again for believing in me.

To my colleagues at CENPARMI: I enjoyed life with all of you and I am happy to have wonderful friends such as Chun Lei He, Fabien Lauer, Javad Sadri, Joanna Rokita, Ning Wang, Ping Zhang, Wu Ding, Yan Zhang and Yun Li, without whom my life at CENPARMI would not have been the same.

To Halina Monkiewicz, Beverley Abramovitz, Brigitte Léonard, Monica Etwaroo, Guiling Guo and Shira Katz: I am grateful for your endless help during my stay at Concordia University.

To Ms. Kadisha Dairova, Dr. Peter Schneider and Ekaterina Loban: Thank you for your continuous support and encouragement. Thank you to the Government of Kazakhstan and "Bolashak" Program, who gave me a chance to study in Canada.

To my family: Thank you for waiting for me so long. Without your patience and support, this just could not have happened.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Optical character recognition (OCR) is a research field that has been drawing scientists' attention across the world for several decades [29]. Automatic document scanning, language identification and handwriting recognition on tablet PCs are a few examples of successful OCR applications. Handwriting recognition (HWR) is a specific OCR subject that covers a wide range of challenging problems such as handwritten character recognition, handwritten numeral recognition, signature recognition and handwritten mathematical formulae recognition. Handwritten numeral recognition is a popular problem due to its applicability to industrial problems such as automatic cheque processing and postal mail sorting. In this thesis, we considered how transforming the appearance of numerals can better train a handwritten numeral recognizer to reduce the number of errors made by the system.

1.1 Background

1.1.1 Prior knowledge

Recent studies in OCR show that using prior knowledge can improve performance [11], [22], [45], [46]. In the literature, *prior knowledge* refers to information about the recognition task that is available in advance. For example, a database of all streets in all cities across the country can be considered as prior knowledge for recognizing postal codes. The dictionary used in word-driven HWR systems is another example.

It is not always possible to obtain prior knowledge in advance. Most of the time, the training data is given. However, there are different methods to derive additional knowledge from the training data. Methods like feature selection and transformation invariance are some of them [11], [38], [44]. Feature selection aims to find the best discriminative features from the given data, whereas transformation invariance focuses on generating various samples of the data.

In this study, transformation invariance is considered. Morphing transformation is applied to infer knowledge to the training data. In the following sections, previously known transformation invariance techniques such as affine, elastic distortions and virtual support vector machines will be discussed. Additionally, HWR methods related to morphing transformation are also reviewed.

1.1.2 Transformation invariance

Transformation invariance is based on the assumption that for a certain transformation no matter how the data is transformed, its decision value remains unchanged. For instance, in rotation invariance, when a certain character in the image is rotated, the human can still recognize it correctly. The data generated by invariant transformation is called artificial or synthetic data. The process of generating artificial data is referred to as *data synthesis*. In practice, the size of the training data is an important issue. Most of the time, there will not be enough training data for the recognition tasks. In addition, the training samples may not cover all of the possible variations of the data. Thus, increasing the amount of the training data by adding artificial samples can be helpful. In principle, when one has a large amount of data, one might derive more knowledge about how to classify unseen data. Therefore, the study of transformation invariance is a problem of practical importance.

Recently, several transformation invariance methods were published in the literature [11], [38], [45], and [46]. These methods can be grouped into two approaches with respect to their data representation spaces used for the data synthesis. In the first approach, the data space is referred to as input space where the raw data is used. In the second approach, the data space is represented by feature vectors in terms of the data attributes, and it is called a feature space.

The first approach is based on a Neural Network (NN) [46]. NN is a powerful learning technique [3]. In this approach, the data involved in data synthesis are the actual samples. So, the data generation process occurs in the input space. This approach has reached the highest performance on the MNIST dataset, [23]. MNIST is a benchmark

dataset of 28x28 size images of isolated handwritten numerals. There are 60,000 training samples and 10,000 testing samples in the dataset. An optimal performance has been reached by two essential practices, extending of the size of training data and convolutional neural networks [22], [46]. In the first practice, the size of the training data is extended by adding artificial samples generated using elastic and affine distortions. *Elastic distortion* is performed by random displacement and Gaussian convolution. In random displacement each pixel of the example is moved in a random direction within a pixel range. Then in the second practice, the resulting image is convolved by a Gaussian filter. *Affine distortion* is a group of transformations such as rotation, translation, and line thickness, which are used in the distortion. Either elastic distortion or affine distortion can be used as the first practice. The final impact comes from the second practice, convolutional neural networks that exploit the spatial structure. These two practices have made significant improvements on MNIST.

In contrast to the first approach, the same type of solution can be obtained by mapping the data into a more suitable space and by changing its representation in the desired space [11]. The second approach is based on Support Vector Machines (SVMs). SVM is a learning technique based on Statistical Theory [51]. Recently, Decoste and Scholkopf (2002) proposed a method for training invariant SVMs. It is called *virtual support vectors*. It is a learning method trained over so-called virtual vectors obtained by applying invariant transformations to the support vectors. Additionally, this method suggests invariant kernel functions. It aims at replacing the kernel functions of SVM. This method has also demonstrated the state-of-the-art performance on the MNIST benchmark dataset of handwritten numerals.

1.1.3 HWR and morphing transformation

During recent years, HWR has been one of the most attractive topics in OCR. It is mainly due to the fast growing hand held devices and tablet PCs in business that target the vast consumer market. The case where the recognition is performed concurrently to the writing process is called online HWR. For the recent implementations of online HWR, one can refer to [1], [28]. Online HWR has gained much interest for its various deformations used in the matching process.

In 1998, Pavlidis et al. introduced a physics-based shape metamorphosis technique for online HWR. The idea was originally proposed by Sederberg et al. (1992) and designed for animation purposes. Shape metamorphosis is also called shape morphing. The study conducted by Pavlidis et al. was one of the first applications of a morphing technique designed for recognition tasks. Shape morphing, based on physics, is used as a matching criterion. It is determined by the *degree of morphing* between an input and reference patterns. The degree of morphing is an abstract quality to measure the energy of morphing from one shape to another. This technique was applied for user-dependent online HWR and resulted in considerable improvements over existing techniques. The technique was later applied to planar shape recognition tasks [47].

Recently, Iga and Wakahara (2004) proposed an interesting work related to character image reconstruction from a feature space. Their character reconstruction method is based on the mesh-model (free-form) morphing technique and genetic algorithms. This method deforms a predefined template so that it fits the given feature vector. Genetic algorithms are employed to search for the optimal morphing parameters. This method is useful for analysing the feature space and searching for more robust

features. The authors also advocated that this might be a useful tool for generating artificial examples.

Morphing transformation (MT) is widely used in Computer Graphics for animation, visualization and blending of graphical objects. MT is also known as warping in the case of planar shapes. There are several morphing techniques available in the literature depending on the purpose of use. We had already mentioned the physics-based morphing. It was designed for shape blending purposes. Another popular morphing technique is feature-based morphing. It was first introduced by Beier and Neely (1992). It is useful for animation purposes where realistic object morphing with feature preservation is required. Morphing techniques were studied by Gomes et al. (1999) and presented as a book and as a chapter in [17] and [16].

Historically, there was a similar study in the late 70s. In a study of OCR based on a human perception model carried by Suen and Shillman (1977), a troublesome pair of letters "U" and "V" was investigated. The most important features in the discrimination of these letters were determined by psychophysical experiments. This was the first time that computer generated artificial characters were considered for recognition tasks. Today's technology permits more sophisticated morphing transformations to be used in the learning process.

1.2 Purpose

The purpose of this study is to investigate the impact of synthetic samples generated by morphing transformation for handwritten numeral recognition.

According to error analysis of Suen and Tan [48], more than sixty percent of the classification errors generated from different classifiers can still be recognized by a human. However, it appears that due to the lack of certain examples in the training set, a classifier cannot derive complete knowledge about handwritten numerals. In addition, redundancy of the most frequent examples in the training set may also be the reason for some complexity. Giving the same type of the character again and again for training will not give new information to the classifier.

A possible solution in this case may be invariances. Invariance techniques based on affine and elastic distortions are necessary to extend the size of the training set. The amount of training examples can be increased by adding the synthetic data generated by invariance transformations. Then, the recognizer can be trained on the dataset consisting of the original training set and the synthetic examples. The impact of data synthesis is significant. However, each of the transformations has certain limitations. Affine distortion consists of the limited number of transformations such as rotation, translation, and line thickness. Affine transformations are applied to each sample of the training dataset. The resulting data samples are rotated, translated, and thickened versions of the handwritten numerals. However, synthetic examples generated by affine transformations will not cover all possible variations of the handwritten numerals. In the case of elastic distortion, the data is distorted too much, and some of the data examples may look unnatural. Learning from these kinds of data examples may mislead the recognizer. Thus,

it is still necessary to find a transformation to represent variations in handwritten numerals that is both complete and representative.

There are many transformations available in the literature that might be useful for data synthesis, e.g. active contours, deformable models, level sets, warping and morphing transformations [16], [17], [40]. In this study, morphing transformation is considered. In Pattern Recognition, it was first applied for online HWR as a matching criterion [32]. The degree of morphing was used as a similarity measure between the test and reference patterns. However, its potential in data synthesis has not been studied yet.

1.3 System Overview

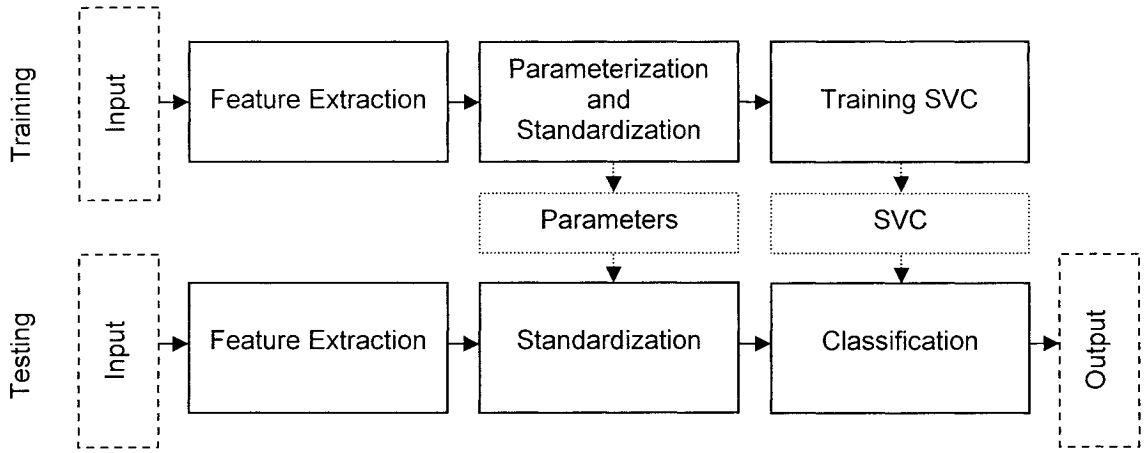
In this study, a system for handwritten numeral recognition was developed. The performance of this system was tested on the MNIST dataset of handwritten numerals [23]. The first part of the system was dedicated to handle handwritten numeral recognition, whereas the second part of the system was designed to generate artificial examples of the handwritten numerals by morphing transformation (Synthetic Data Generation). Furthermore, each of the system parts will be separately described.

1.3.1 Handwritten numeral recognition

Handwritten numeral recognition system contains training and testing modules. The first module is to train support vector classifier (SVC) with the training examples and to estimate the parameters of standardization, a procedure for normalizing and scaling feature vectors. The testing module is designed to recognize unseen examples from the testing set using the derived parameters and SVC from the training module.

The training module has three components: feature extraction, standardization with parameterization, and SVC training. The first component extracts features from an input image using image processing procedures, and it constructs a feature vector. All constructed feature vectors of the training examples are accumulated into the database of feature vectors. Standardization normalizes and scales feature vectors. Standardization requires parameters such as the mean and variance vectors, and a scaling factor. These parameters are estimated during the training process. The last component trains the SVC with the standardized feature vectors.

The testing module is also comprised of three components: feature extraction, standardization, and classification. The first component, feature extraction, is the same as in the training module. It processes input images from the testing set and returns the database of feature vectors to the next component. The standardization component standardizes the database of feature vectors by making use of the parameters derived from the training module. The last component of the testing module is used to classify the standardized feature vectors with respect to their class labels. Classification, performed by the SVC, is also derived from the training module. The final outcome of the system is given as the values of numeral class labels. A diagram of the handwritten numeral recognition system is given in Figure 1.



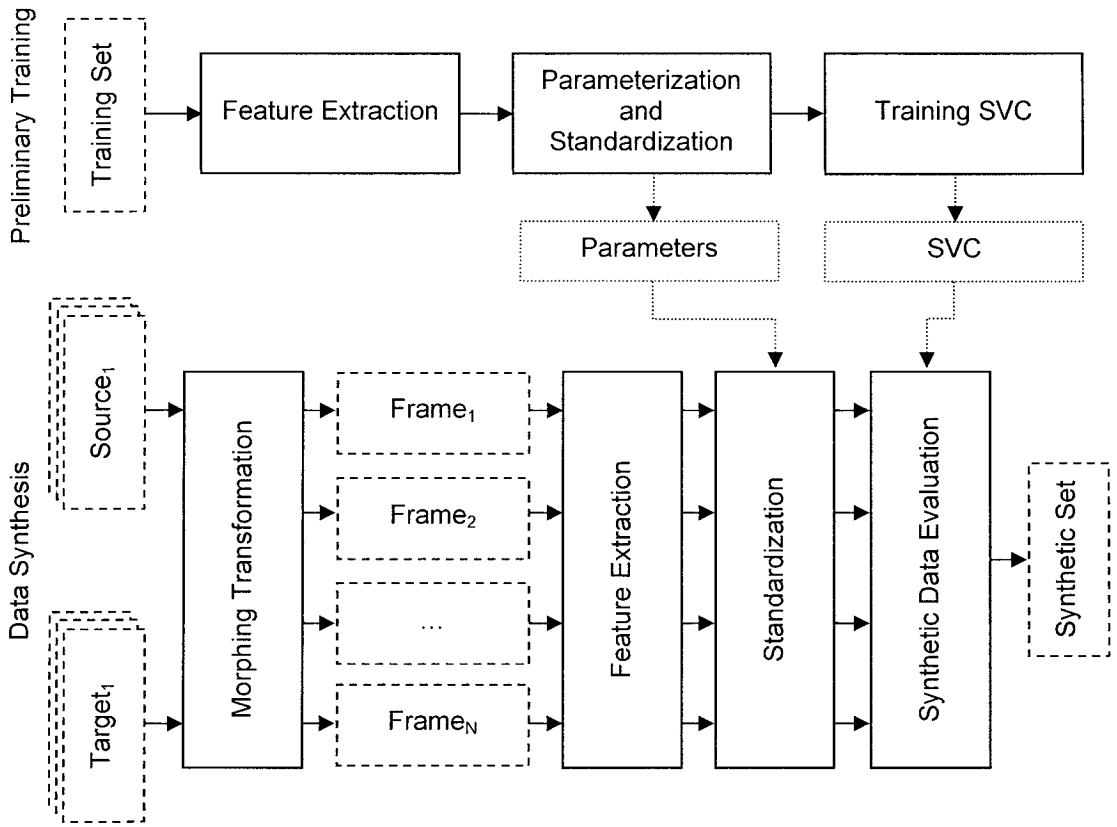
In this figure, both inputs and the output of the system are drawn with dashed rectangles. Solid rectangles represent processes of the system. Solid arrows indicate the data flow. Other data such as parameters and SVC are drawn with dotted rectangles, and connected to the system with dotted arrows.

Figure 1. Data flow of the handwritten numeral recognition system

1.3.2 Synthetic data generation

A synthetic data generation system is dedicated to produce artificial examples of the data, which can be used to balance the training data distribution. The system consists of preliminary training and data synthesis modules. The purpose of the preliminary training is exactly the same as the training module of the handwritten numeral recognition system and it contains the same data flow. The data synthesis module produces artificial examples of the handwritten numerals. It uses the knowledge derived from the preliminary training to control the process of data generation. The parameters and SVC, estimated in the preliminary training, are passed to the data synthesis to standardize the extracted features and evaluate the generated examples of the handwritten numerals. An evaluation of the artificial examples is required to validate the quality of the artificially generated examples.

Now let us describe the data flow of the synthetic data generation. First of all, it generates artificial examples using morphing transformation. Morphing transformation is a complex process that produces synthetic versions of a given dataset making use of techniques such as warping and blending. The dataset for the morphing must contain two different examples of the data, named "source" and "target". Synthetic versions of the given dataset are called frames. Morphing generates artificial examples frame by frame from the source towards the target. Consequently, the frames at the beginning of the morphing transition are assumed to be more similar to the source, whereas the frames at the end resemble the target. One by one the frames generated are passed through the rest of the system. At the next stage, the set of feature vectors are obtained by making use of the feature extraction component from these frames. Standardization is applied to the set of feature vectors. Standardization uses the parameters obtained in the preliminary training. Lastly, synthetic data evaluation is applied. In this stage, the system generates decisions about the quality of the artificial examples. An artificial example is qualified to be either the synthetic version of the source only if it is classified to be in the same category of the source or the synthetic version of the target only if it is classified to be in the same category of the target. Those artificial examples that do not respect the rule above are discarded. A final output of the system is a set of good quality synthetic examples. A diagram of the synthetic data generation is presented in Figure 2.



In this figure, the training and synthetic data sets, the pairs of sources and targets, and the frames are drawn with dashed rectangles. Solid rectangles represent processes of the synthetic data generation system. Solid arrows indicate the data flow. The parameters and SVC are drawn with dotted rectangles. They are connected to the system with dotted arrows.

Figure 2. Synthetic data generation

1.4 Thesis Outline

The remaining chapters are organized as follows:

In the second chapter, the feature extraction techniques shown in Figure 1 are described. Gradient, structural and convexity feature extraction methods will be described. Additionally, image processing procedures used for feature extraction will also be described.

In the third chapter, classification with Support Vector Machines is discussed. The definition of the SVM and the comparison of different versions of the SVM algorithms are given in this chapter. Additionally, the standardization, training and testing algorithms used in the system (see Figure 1) are also described in this chapter.

In the fourth chapter, synthetic data generation by morphing transformation for handwritten numerals is discussed (refer to Figure 2). The basic definitions of morphing transformation and curve evolution warping transformation are given, and the application of the morphing transformation to recognize handwritten numerals is described. Furthermore, the practical issues of morphing transformation in generating synthetic samples are discussed (see also Figure 2). A comparison of the outputs of different synthetic data generation methods is also given in this chapter.

Chapter five describes the experiments conducted, and chapter six draws the conclusion of this study.

CHAPTER 2
FEATURE EXTRACTION

A large number of feature extraction methods for handwritten numeral recognition have been reported in the literature. For the selected references one might refer to [24], [25], [44], and [50]. Selection of a good feature extraction method is one of the important factors in achieving a high recognition performance. In this study, gradient, structural and concavity feature extraction methods are considered. For this combination of the features, we will further refer to them as GSC features in short. This concept, previously proposed by Favata and Srikantan (1996), will be used the feature extraction. Each of these methods is described in section 2.1. For gradient feature extraction, a method introduced in [43] was adapted. It was slightly modified due to our database specifications. The description of the method and the modifications made are detailed in section 2.1.1. For structural feature extraction, there are many possible features, including contour profile, directional, and chain code features as in [31], [50]. We use the chain code feature extraction method for this study. Contour chain code features were extracted using the zoning method [31]. Additionally, a new feature, double-contour, was applied to this method. All of these modifications and the method itself will be discussed in section 2.1.2. For concavity feature extraction, a method similar to [24] was implemented. However, a new pie-like zoning method was applied instead of horizontal crossing counts. The details of the method are in section 2.1.3.

Different feature extraction methods require different image processing procedures. Image processing procedures used for the GSC feature extraction method are described in section 2.2.

2.1 GSC Feature Extraction

2.1.1 Gradient

Recently, Shi et al. proposed a gradient feature extraction method for handwritten numeral recognition [44]. This method was adapted to extract gradient features. According to Shi et al, size normalization has to be applied first. However, the database used in this study (MNIST, [23]) has been normalized already and there is no need for size normalization. Following this step, image smoothing is applied. Lastly, the resulting gray-scaled image is transferred into a matrix of color intensities where the maximum and minimum values are 1 and 0, respectively.

There are several operators that can be applied to estimate the gradients. Operators such as Roberts, Sobel and Kirsh are the most common operators. Roberts operator was used by Shi et al. It is defined as follows:

$$\begin{aligned} \Delta u &= f(x+1, y+1) - f(x, y) \\ \Delta v &= f(x+1, y) - f(x, y+1) \end{aligned} \tag{1}$$

Thereafter, the degree and strength of a gradient are computed as follows:

$$\begin{aligned} \theta(\Delta u, \Delta v) &= \tan^{-1}\left(\frac{\Delta u}{\Delta v}\right) \\ \psi(\Delta u, \Delta v) &= \sqrt{(\Delta u)^2 + (\Delta v)^2} \end{aligned} \tag{2}$$

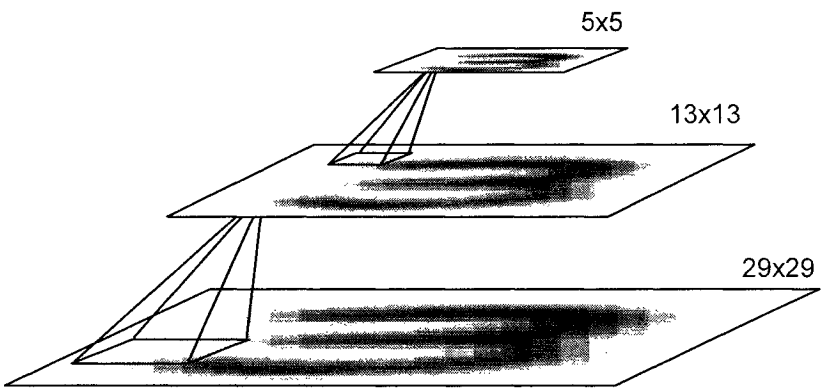


Figure 3. Image resolutions by down sampling

The remainder of our method is the same as in Shi et al. except for variable transformation and spatial resolution reduction. We did not apply variable transformation, and spatial resolution reduction was applied differently. The spatial resolution was reduced by down-sampling, similar to the method described in [44]. The structure of this method is presented in Figure 3. An input image is down sampled by a factor of 2 using the 5x5 Gaussian filter described earlier (see section 2.2). Each time, the image resolution is reduced from n by n to $(n-3)/2$ by $(n-3)/2$. This down sampling rule is applied two times for all MNIST images. In our case, the initial image size is 28x28. However, the nearest value which generates an integer number after two iterations of down-sampling is 29x29, $n=29$. We added an additional blank row and column so that after two iterations the size becomes 5x5 (as shown in Figure 3).

Using the gradient feature extraction method, a feature vector of size 200 was produced. The feature vector consists of 5 horizontal, 5 vertical and 8 directional resolutions.

2.1.2 Structural

For the structural features, we used a chain code feature extraction method due to its popularity among OCR researchers, and due to its use in high recognition performance systems. Contour chain code features were extracted using the zoning method [31].

In this study, a contour chain code extraction method similar to that in [31] was adopted. According to it, contour information is measured from a histogram of eight directional chain codes at each zone of 2x3. However, in this study, double-contour information is used instead of contour information. In addition to 2x3 zones, 2x2 zoning was also added.

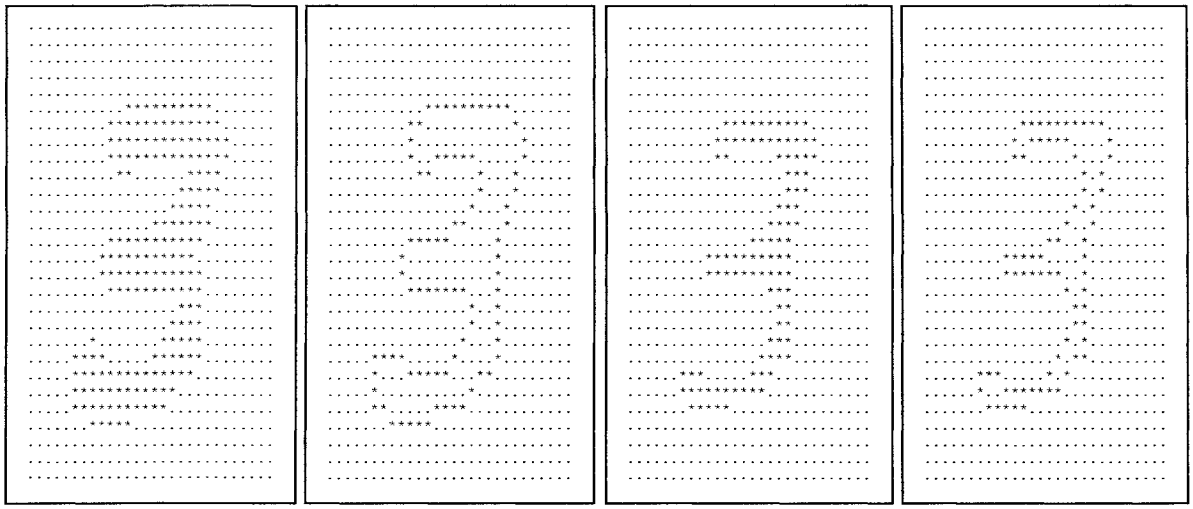
A double-contour image is a sum of two contour images, the first-level contour and second-level contour images (as shown in Figure 4 b) and d), respectively). The first-level contour is a contour extracted from the original image. The second-level contour is a contour extracted from a sub-image. The sub-image is obtained by subtracting the first-level contour image from the original image. Each of the first-level contour and second-level contour images was coded using an eight directional Freeman mask before summing them. The coded versions of the first-level contour and second-level contour images are shown in Figure 4 e) and f). Freeman directions are presented in Figure 5 a). The final double-contour is shown in Figure 4 g). We believe that the double-contour is useful to ease the influence of noisy edges in the numeral. If the original image is sloppy, then its first-level contour is also curvy. However, if the information of the second-level contour is considered, in addition to the first-level contour, then the influence of the noisy edges is reduced.

An additional 2x2 zoning was added to the 2x3 zoning method used in [31]. This helps to retrieve more detailed information with respect to the zone locations. There are ten zones, sized 16x10, overlapping each of the neighbour zones by a two-pixel width. The first group of zones, 2x3 zoning, is shown in Figure 5 b). Another group of zones, 2x2 zoning, is shown in Figure 5 c).

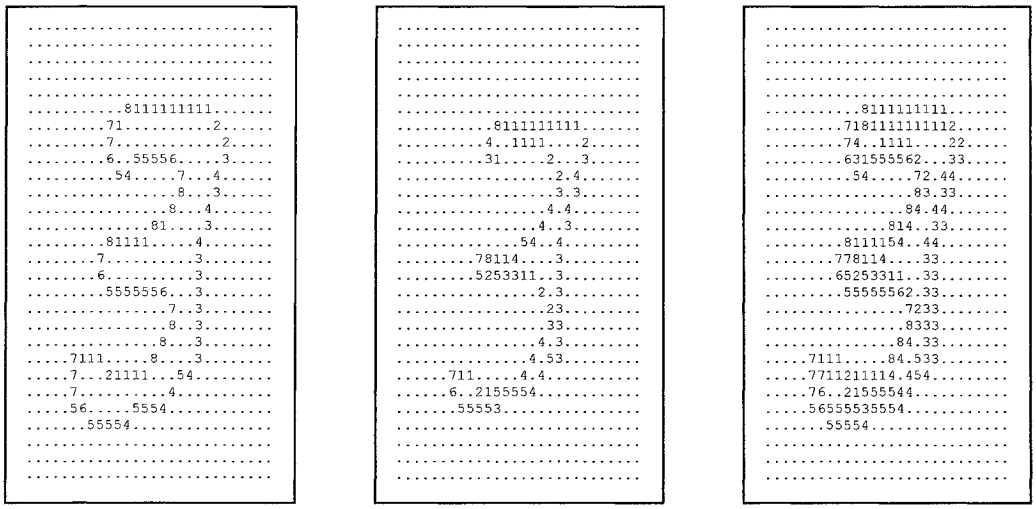
Four features at each of the ten zones are calculated with respect to their directions at 0^0, 45^0, 90^0 and 135^0. The sum of these features in each zone is divided by the total number of contour points. The direction codes "1" and "5" stand for 0^0. Similarly, the codes "4" and "8" stand for 45^0. The codes "3" and "7" stand for 90^0. Lastly, the codes "2" and "6" stand for 135^0. Using this technique, 40 directional features are extracted in total at 4 directions in 10 zones. A simple equation for the calculation of these features is given as follows:

$$
\begin{aligned}
s_i^k &= \frac{1}{N}\sum_{y_i}\sum_{x_i} f^k(x,y), \quad i = 1,2,...,10, \quad k = 1,2,...,4, \\
N &= \sum_{y}\sum_{x}\sum_{k} f^k(x,y), \\
f^k(x,y) &= \begin{cases} 1, & if\ p_{x,y} = k, k+4 \\ 0, & otherwise \end{cases}
\end{aligned}
\tag{3}
$$

where s_i^k stands for the feature value of the i^{th} zone and k^{th} direction and $p_{x,y}$ returns the direction code of the contour at the position x and y. Note that $p_{x,y}$ returns a zero when it is out of the contour.

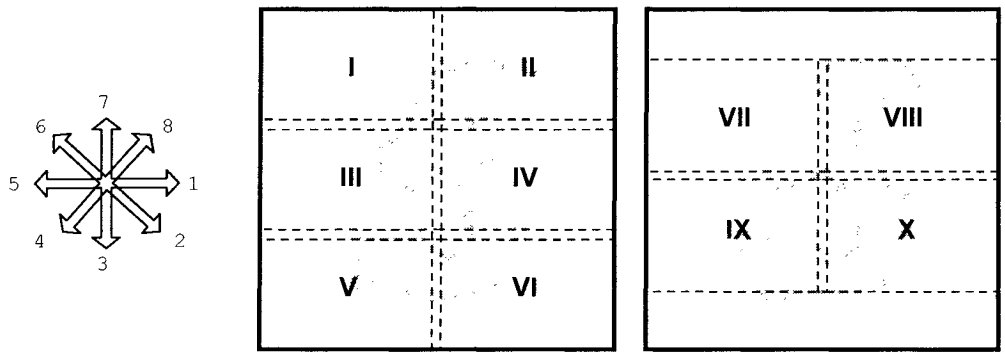


a) Original image b) First-level contour c) Subimage d) Second-level contour



e) Coded first-level contour f) Coded second-level contour g) Coded double-contour

Figure 4. Double-contour extraction



a) Freeman directions b) 2x3 Zoning c) 2x2 Zoning

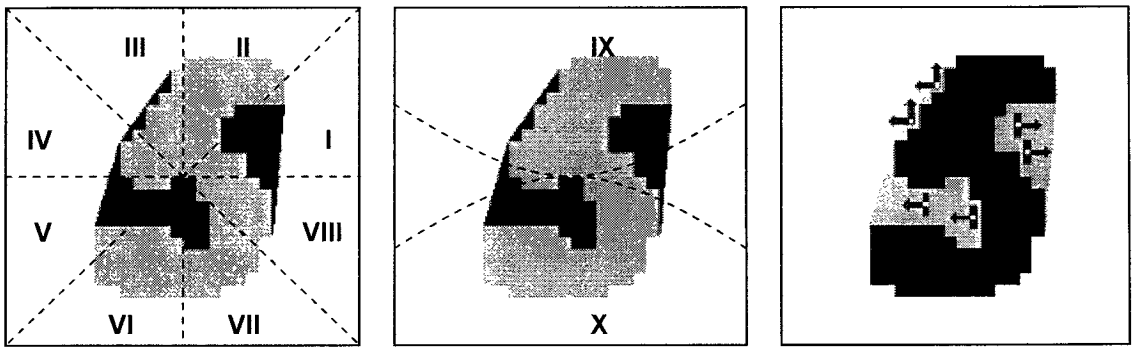
Figure 5. Structural feature extraction

2.1.3 Concavity

We used the modified version of the concavity feature extraction method from [24]. Concavity features were used as background information to complement structural features. Unlike in [24], a pie-like zoning technique and two parabolic zones were used to measure concavity features instead of using horizontal crossing counts. Each of the eight zones of the pie-like zoning overlaps neighbouring zones by a pixel width as shown in Figure 6 a). In addition to the pie-like zones, two parabolic zones as shown in Figure 6 b) were also used to measure features of the upper and lower parts of the image.

Let us now describe the concavity feature extraction methods. In the first step, a contour image is obtained from the binary image as shown in Figure 7 b). Then, a convex hull image is computed making use of this contour image, as in Figure 7 c). A final image with labelling information, as in Figure 7 d), is acquired by cropping the background side with respect to its convex hull. All image pixels inside of the convex hull and outside of the binary image are considered as background information or concavity pixels. All concavity pixels are classified into nine concavity types with respect to a pixel's availability from the left, right, top and bottom sides of the image. The availability of a pixel from each side is determined by tracing the straight lines from the pixel to the left, right, top and bottom sides. A side will be considered as unavailable if a black pixel was reached on the line during the tracing towards this side. Pixels that are available from three or more sides are discarded. Only the points that are facing two or less sides are considered and labelled with respect to their availability from the sides. Examples of the side availability are illustrated in Figure 6 c), where the arrows indicate the available sides. Concavity labels are shown in Figure 6 d). The arrows of label signs point to the

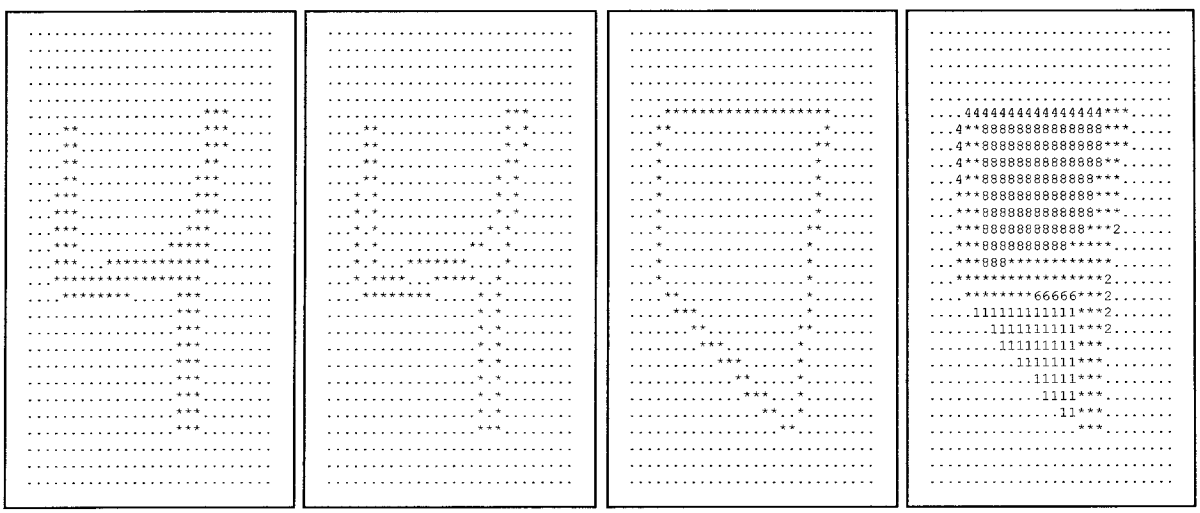
available sides. The pixels that are available from each pair of the left, bottom, top and right sides are labelled as "1", "2", "3" and "4", accordingly. The pixels that are available from only one of the right, bottom, left and top sides are labelled as "5", "6", "7" and "8", respectively. The pixels that are not available from any side are labelled as "9".



a) Pie-like zoning b) Parabolic zones c) Concavity measurements

d) Concavity labels

Figure 6. Concavity feature extraction



a) Binary image b) Contour image c) Convex hull d) Concavity features

Figure 7. The steps of the concavity feature extraction

Making use of concavity labelling rules, six composite concavity features are extracted. The first four features are extracted from the pixels that face the left, right, top and bottom sides. Specifically, the left side features contain pixels with labels "7", "1" and "4". The right side features contain pixels with labels "5", "2" and "3". The top side features contain pixels with labels "8", "3" and "4". The bottom side features contain pixels with labels "6", "1" and "2". The fifth feature is computed explicitly from the pixels with label "9" or the pixels facing no sides at all. The last feature describes the sparseness of the zone. It is extracted only from the black pixels that are not labelled but shown as star shapes "*" as seen in Figure 7 d). The feature values are computed using the same formulation given in (3). In total, there are 60 features extracted from 6 concavity features in 10 zones.

2.2 Image Processing

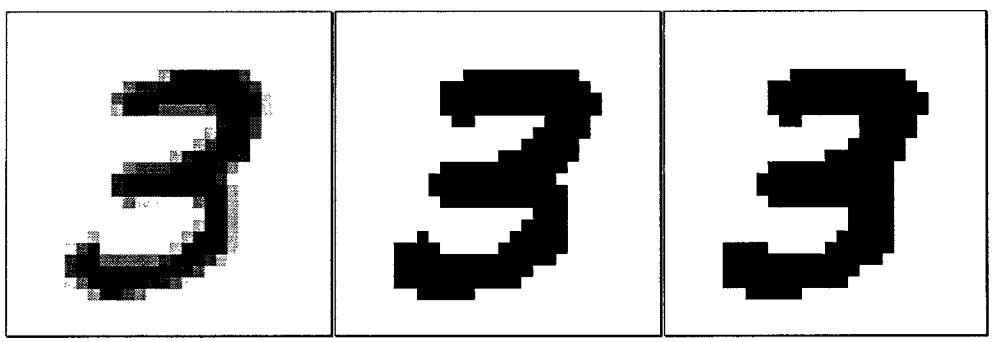
An input image has to be processed before each of the feature extraction methods. Different feature extraction methods require different image processing procedures. For example, gradient feature extraction works with gray-scale images and the input image has to be smoothed in advance. For structural feature extraction, the input image has to be binarized, denoised and the contour image should be computed from the denoised binary image. Lastly, the convex hull image has to be computed in order to extract concavity features. In all, there are five image processing procedures used in this study:

- Binarization
- Denoising
- Contour tracing

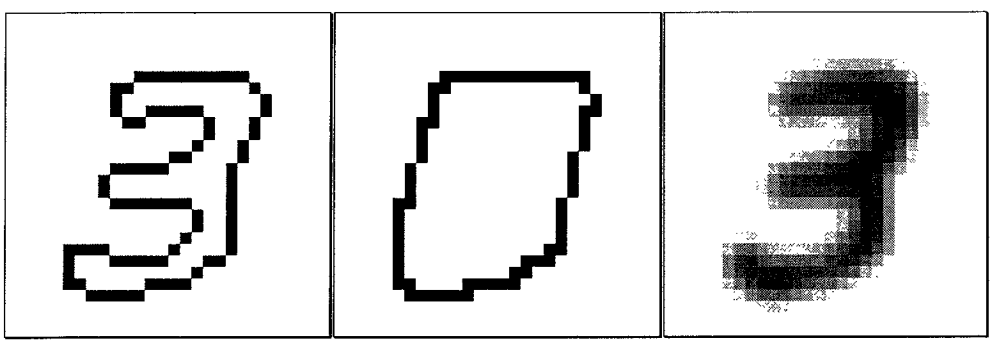
- Convex hull
- Smoothing

Binarization is required to eliminate low-level intensities from gray-scale images [43]. It removes all pixels within a threshold. The threshold is set equal to 255/9 in order to eliminate the least color intensities. A binary image obtained by this procedure is shown in Figure 8 b).

Denoising procedure is applied to binary images for noise reduction [43]. The procedure removes the convex pixels and fills the concave pixels of a binary image. In a 3x3 neighbourhood, if the number of surrounding black pixels is less than 5, then it removes the black pixels. It places a black pixel if the number of surrounding white points is less than 5. The result of this procedure is given in Figure 8 c).



a) Original image b) Binary image c) Denoised image



d) Contour image e) Convex hull image f) Smoothed image

Figure 8. Image processing results

Image contours are obtained by *contour tracing* (boundary extraction) procedures [18]. The procedure checks whether the color continuity is preserved or not for the last consecutive two pixels of each row and column of a binary image. When there is a switch, it records it as a contour pixel. A contour image is displayed in Figure 8 d). For the contour images obtained by this procedure, chain coding can be applied. Chain coding is needed to label the contour pixel with Freeman direction codes. It examines each of the contour pixels for the direction of neighbouring pixels (for any contour pixel it is assumed that there exists at least two neighbouring contour pixels). Then, it sets the direction code with respect to the line between the pixel and the first met pixel in a clockwise direction starting from the previous direction at its 3x3 neighbourhood.

The result of the next image processing procedure is *convex hull* [18]. It can be computed from the contour image by making use of Graham's scan method. Each time it examines two pixels in order to estimate the set of break points. The break points are the corner points of the convex hull. Using the line equation from two points, the method examines all continuous points of the line. It eliminates all points between the first and last points. Then, it adds the first and last points to the set. Thereafter, it considers the last point as the first point of the next line equation. The second point is obtained from the chain of contour points. In case of the contour image, it would be the next available point right after the point that was considered during the last iteration. It repeats these steps until the starting point is met. When the complete set of break points is obtained, it connects all neighbouring pairs of the break points by the line equation from them. The resulted image will become a convex hull image and an example of it is shown in Figure 8 e).

The last image processing procedure to be specified is a *smoothing* procedure [18]. We produce a smoothed image by applying a Gaussian filter of 5x5 size to the binary image for three times. An example of the smoothed image is shown in Figure 8 f).

CHAPTER 3
SUPPORT VECTOR CLASSIFICATION

Support vector classification is discussed in this chapter. The classifier considered in this system is Support Vector Machines (SVMs). SVM is a powerful learning technique based on Statistical Theory [10], [51].

The discussions regarding to SVM technique and its implementation issues are presented in section 3.2 and section 3.3, respectively.

Before the discussions about SVM, standardization of the raw GSC features (section 2.1) is given in section 3.1.

3.1 Standardization

There are two standardization techniques considered: normalization and scaling. Normalization is a well-known technique widely used in learning methods [3]. It increases the efficiency of the features in discriminating the patterns. Scaling of the feature vectors was advocated for use in SVM by several researchers [11], [12]. It is mainly used to avoid dominance features in greater ranges. Additionally, it helps to avoid numerical difficulties during the computation because very large numbers may result in an "overflow" in computers.

3.1.1 Normalization

According to [3], *normalization* of the feature vectors is a useful procedure that normalizes the distribution of the feature vectors. It normalizes the set of feature vectors with respect to the mean and variance feature vectors. First, the mean vector is computed by averaging all feature vectors. Then, the variance for each feature is calculated using the mean vector. Finally, normalization is applied using both the mean and variance feature vectors. The process of normalization can also be expressed as follows:

$$
\begin{aligned}
x'_{i,j} &= \frac{x_{i,j} - x_i^m}{\sigma_i}, \\
\sigma_i^2 &= \frac{1}{n-1}\sum_{j=1}^{l}\left(x_{i,j} - x_i^m\right)^2, \\
x_i^m &= \frac{1}{n}\sum_{j=1}^{l} x_{i,j},
\end{aligned}
\qquad (4)
$$

where j indicates the index of the feature vector, i indicates the index of the vector dimension, and n denotes the number of the feature vectors. Moreover, x_i^m and σ_i stand

for the *i*-th value of the mean and variance vectors. Finally, $x_{i,j}$ denotes the previous feature vector, whereas $x'_{i,j}$ denotes the resulted normalized feature vector.

3.1.2 Scaling

In Pattern Recognition, similarity measurements are assessed by the *feature space metrics* [26]. A measuring rule $d(x,y)$ for the similarity distance between feature vectors x and y is considered as a metric if it has the following properties:

$$\begin{aligned} &d(x,y) \geq 0, \\ &d(x,y) = 0, \quad \textit{if and only if } x = y, \\ &d(x,y) = d(y,x), \\ &d(x,y) \leq d(x,z) + d(z,y), \end{aligned} \tag{5-1}$$

This metric is called a *norm* and denoted as $d(x,y) = \|x - y\|$ when the property below is satisfied:

$$d(\alpha x, \alpha y) = |\alpha| d(x,y), \quad \alpha \in \Re \tag{5-2}$$

We used a Euclidian norm, one of the most common norms, in this study. It is also referred to as the *Euclidian distance* and expressed as follows:

$$\|x - y\| = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}\,, \tag{6}$$

where m indicates the dimension of the feature space.

Now, let us describe scaling. In this study, scaling is applied to reduce the similarity measurements so that the Euclidian distance between any two feature vectors lies within the interval of 0 and 1 [11]. Scaling of a single feature vector can be achieved by the multiplication of the feature vector with a constant value. This constant value is

called the *scaling factor*. It has to be estimated in advance by making use of a certain algorithm. When the scaling factor is determined, then the scaling of feature vectors is straightforward. For the estimation of the scaling factor and for the scaling of feature vectors, the following algorithm was developed:

1. Scaling factor λ is set to 1.
2. For all vector pairs of the data the next condition is considered: If the scaled Euclidian distance of feature vectors x and y is $\|\lambda x - \lambda y\| > 1$, then scaling factor λ is reduced by small steps until the distance becomes smaller or equal to 1.
3. All feature vectors are scaled as $x' = \lambda x$, where x' and x denote the scaled and original feature vectors, respectively.

There are *n(n-1)/2* vector pairs of the data that need to be considered for *n* feature vectors. It is quite exhaustive. However, the computation time can be reduced substantially when data clustering is applied. First, we use a simple *k-means* clustering algorithm, k $(k << n)$ cluster centers can be determined. Then, there are just kn $(kn << n(n-1)/2)$ pairs required for step 2 of the above algorithm. This algorithm helps us estimate an approximate scaling value.

3.2 Support Vector Classification

Statistical Classification, Neural Networks, and Support Vector Machines are common classification techniques used in Pattern Recognition [3], [26]. In this study, *Support Vector Machines* (SVMs), the most recent classification technique, is considered. Due to its ability to provide a good generalization, it has gained much attention from the community of the machine learning researchers over the last few years. SVM was first introduced by V. Vapnik [51]. The nature of SVM is two-fold. It was originally designed to use a nonlinear mapping to transfer the input space into a high dimensional feature space. The main idea of SVM is to find a decision boundary in the feature space that maximizes the margin between the classes (the sum of the closest perpendicular distances of the classes to the boundary) with a certain tolerance towards the training errors. This decision boundary is also called an optimal hyperplane, and is defined by a small subset of training samples, called support vectors. SVM is also different for its metric, which is dot product [38]. In practice, nonlinear mappings in SVM are substituted by kernel functions [30]. According to the different learning concepts used in SVM, it can be referred to as a hyperplane classifier, large margin classifier, as well as kernel methods. Moreover, depending on the task, it can also be referred to as Support Vector Classification (SVC) or Support Vector Regression (SVR), respectively. So far, SVM has been successfully applied to face recognition, web page classification, HWR, etc [5]. For more details about SVM, one can refer to [4], [9], [10], [30], [36], [38], and [51]. First, we give a brief introduction to hyperplane classifier. Second, kernel functions are overviewed. Lastly, the use of a new SVM algorithm, called ν-SVM, is justified.

3.2.1 Hyperplane classifier

Let us consider a binary classification problem. Let us assume that a linearly separable binary problem is given by (x_i, y_i), $x_i \in X$, $y_i \in \{+1,-1\}$, $i = 1,2,...,l$, where x denotes the training vectors and y indicates the positive and negative class labels. Let us also assume that *dot product space* H, where the patterns are represented by the mapping $\Phi : \mathrm{X} \to \mathrm{H}$, is given and the similarity measurement of the following form $\langle x, x' \rangle := \sum_{i=1}^{l} [x]_i [x']_i$ for the dot product space is established.

Consider that there exists such a hyperplane that separates the positive and negative classes and expressed as follows:

$$\langle w, x \rangle - b = 0,\ w \in \mathrm{H},\ b \in R, \tag{7}$$

where w is normal to the hyperplane and $-|b|/\|w\|$* is the perpendicular distance from the optimal hyperplane to the origin.

This hyperplane is called an *optimal hyperplane* if it yields the maximum margin between the classes. Consequently, the optimal hyperplane corresponds to a decision function:

$$f(x) = \mathrm{sgn}(\langle w, x \rangle + b), \tag{8}$$

The sign of the decision function determines which class example x belongs to.

* $\|w\|$ stands for Euclidian norm of w.

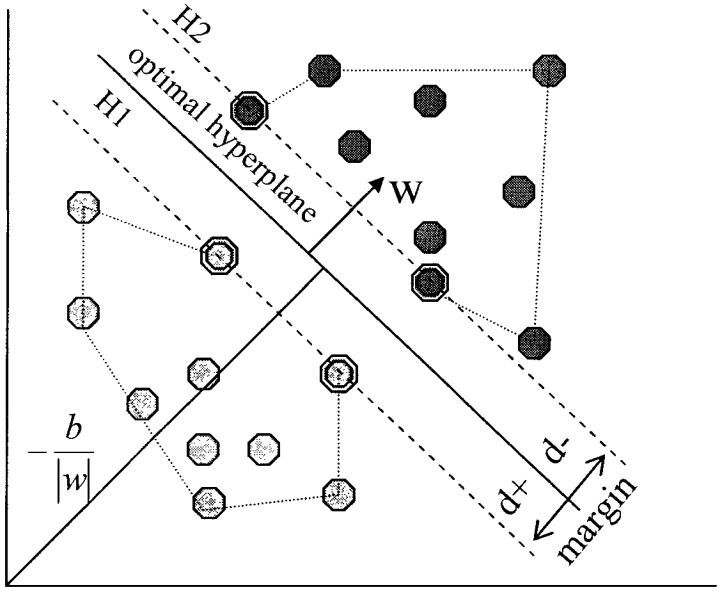


Figure 9. Geometrical interpretation of SVM

Now, let us explain how the maximal margin can be obtained. This is also illustrated in Figure 9. If we assume that the parameters w and b are scaled so that the closest points to the hyperplane assure $|\langle w, x_i \rangle + b| = 1$[**] then the following constraints become true:

$$\begin{aligned} &\langle w, x_i \rangle + b \geq +1 \quad for\ y_i = +1, \\ &\langle w, x_i \rangle + b \leq -1 \quad for\ y_i = -1. \end{aligned} \tag{9}$$

By combining these inequalities the following set of inequalities can be obtained:

$$y_i \cdot (\langle w, x_i \rangle + b) \geq 1. \tag{10}$$

If we separately consider the points for the equalities in equation (9) we can derive hyperplane equations $H_1 : \langle w, x_i \rangle + b = +1$ and $H_2 : \langle w, x_i \rangle + b = -1$ for the positive and negative classes with the perpendicular distances $|1 - b|/\|w\|$ and $|-1 - b|/\|w\|$,

[**] Here and below, $i = 1,2,...,l$.

respectively. Additionally, let the margin between the classes be the sum of d_+ and d_-, which denote the shortest distances from the hyperplane to the positive and negative classes, accordingly. By summing the perpendicular distances, we get $d_+ = d_- = 1/\|w\|$. So, the margin equals to $2/\|w\|$.

Thus, an optimal hyperplane with the maximal margin between the classes can be constructed by solving the following *primal problem*:

$$\min \frac{1}{2}\|w\|^2 \tag{11}$$

subject to $y_i(\langle w, x_i \rangle + b) \geq 1$.

This type of problem is called a quadratic programming problem, which can be addressed using optimization theory.

In general, it is more convenient to deal with a dual form of the problem in (11). It is derived by introducing the Lagrange multipliers $\alpha_i > 0$. For more detailed information about how the dual form was obtained, one can refer to [4], [8], and [37]. Here, we present the "final version" of the dual form. In accordance with Lagrange derivatives, the equalities $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $w = \sum_{i=1}^{l} \alpha_i y_i x_i$ are obtained. Then, the Lagrange multipliers $\{\alpha_i\}$ are determined by solving the following *dual problem*:

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{12}$$

subject to $\sum_{i=1}^{l} \alpha_i y_i = 0, \quad \alpha_i \geq 0$.

The points for which coefficients $\alpha_i > 0$ are called *support vectors* (SVs) and lie on one of the marginal hyperplanes H_1 and H_2.

3.2.2 Kernel functions

The above learning method can be extended for nonlinear problems by defining a *nonlinear mapping* $\Phi : \mathrm{X} \to \mathrm{H}$, which transforms the data from input space X into feature space H, which is called dot product space in our case. However, we do not need to define explicitly the nonlinear mapping. In practice, *kernel functions* can be used to perform the similarity measure in the feature space instead of nonlinear transformations. By introducing the kernel function $k(x,x') = \langle \phi(x), \phi(x') \rangle$, the feature space is implicitly specified. An illustration of the kernel function is presented in Figure 10.

Making use of the kernel functions, the problem in (12) can be rewritten as follows:

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \qquad (13)$$

$$\text{subject to } \sum_{i=1}^{l} \alpha_i y_i = 0, \quad \alpha_i \geq 0.$$

There are several kernels available in the literature: Gaussian, polynomial, and sigmoid kernel functions [36], [38]. Kernels with different properties can solve different types of problems. Particularly, the *Gaussian radial basis function* (RBF) kernel, given by $k(x,x') = f(d(x,x'))$, where $d(x,x') = \|x - x'\|$ is the metric on the input space X and f is a function on R_0^+. Having the properties of unitary and translation invariance [38]

makes RBF kernel convenient for its use in learning methods. The most common form of the Gaussian RBF kernel is given as follows:

$$k(x,x') = \exp\left(-\|x-x'\|^2/\sigma\right), \sigma > 0, \quad (14)$$

where σ (sigma) is its radius parameter.

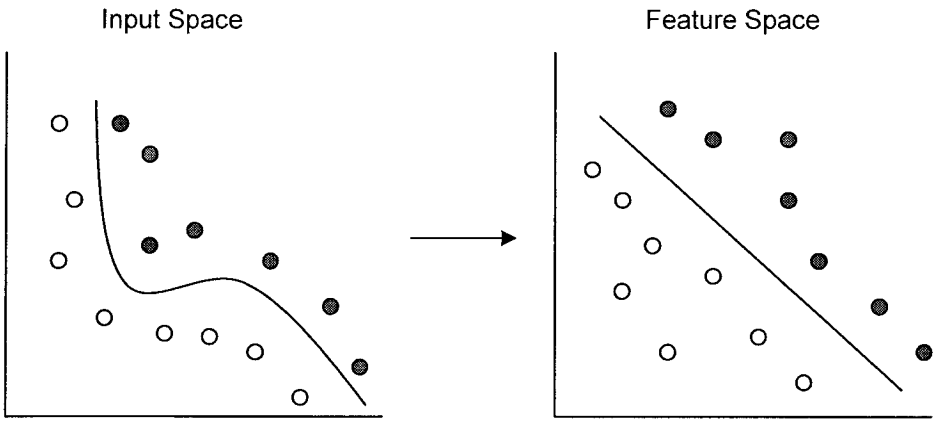


Figure 10. Example of kernel mapping

3.2.3 ν-SVC versus C-SVC

Designing SVM for nonlinear cases is an important issue in classification. In the case of linear SVMs, even a single mislabelled example in the training set may cause failures. So, it is crucial to design SVM with a certain tolerance to errors. The idea behind it is to enhance the learning algorithm so that it becomes capable of both dealing with the training errors and maximizing the margin at the same time.

C-SVC: Cortes and Vapnik [10] introduced *slack variables* $\xi_i \geq 0$ to tolerate a certain fraction of outliers in the training process. Using these variables violates the constraint of the primal problem in expression (11). Slack variables can be used to relax the constraints as follows:

$$y_i(x_i w + b) \geq 1 - \xi_i. \quad (15)$$

In addition to this constraint relaxation, the objective function has to be changed. It is another way to assign an extra cost. A sum of slack variables is added to the existing function. Finally, we come up with the following problem:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \tag{16}$$

$$\text{subject to } y_i(x_i w + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

where parameter C is a constant assigned by a user to penalize the training errors. The larger C permits more errors in the training process. This technique is referred to as C-SV classifier (C-SVC) in the literature. In terms of Lagrange multipliers and kernel functions, the problem in (16) becomes:

$$\max \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \tag{17}$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad i = 1,2,...,l, \quad \sum_{i=1}^{l}\alpha_i y_i = 0.$$

ν-SVC: In C-SVC, constant C is a user-set constant determining the trade-off between the training errors and performance. As a result, we have two conflicting goals: the training errors should be minimized and the margin between the classes should be maximized at the same time. Unfortunately, this technique leaves us no way to select the constant C, though it is an intuitive parameter. When $C \geq 0$, it is unclear how to select an optimal C. However, a new modification of this technique proposed by Scholkopf et al. [37] can be useful. It replaces C with an intuitive parameter $0 \leq \nu \leq 1$, which enables control of the number of margin errors and support vectors. A primal problem of this approach, termed ν-SV classifier (ν-SVC), is given as follows:

$$\min \frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \tag{18}$$

subject to $y_i(x_i w + b) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \rho \geq 0$.

There are two parameters ν and ρ, instead of C, that need to be optimized. Parameter ρ can be derived from the constraint of (18). The setting $\xi = 0$ states that two classes are separated by the margin $2\rho/\|w\|$.

Its dual form is expressed as follows:

$$\max -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j k(x_i, x_j) \tag{19}$$

subject to $0 \leq \alpha_i \leq \frac{1}{l}, \quad \sum_{i=1}^{l}\alpha_i y_i = 0, \quad \sum_{i=1}^{l}\alpha_i > \nu.$

This algorithm is not fundamentally different from C-SVC. It seems that for certain parameter settings, the results of C-SVC and ν-SVC coincide. Using parameter ν makes it more convenient, than parameter C, as it has an intuitive interpretation to specify the fraction of outliers. The significance of ν is specified by propositions 1 and 2 from [37]. Particularly, proposition 1 postulates that parameter ν is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors. Proposition 2 states that if ν-SVC leads to $\rho > 0$, then C-SVC, with C set a prior $1/lp$, leads to the same decision function. This is a connection between C-SVC and ν-SVC.

Experiments for choosing an optimal ν for different noise models were carried out by [6]. The curve of the risk (generalization error) and ν are flat and smooth

indicating that there is an optimal area rather than a sharp end. The authors advocate that well-behaved parameter ν is easy to use in practice. Moreover, varying ν changes the number of support vectors and this is useful when the number of support vectors needs to be controlled. For more detailed information about this method, one can refer to [6], [8], [9], and [37].

3.3 SVM Implementation Issues

Theoretically, SVM is a clearly defined learning technique. However, it is costly to implement. SVM depends on the optimization method used to solve the quadratic programming problem. Moreover, it requires a large number of data to be allocated in the memory during the training. For these reasons, the early implementations of SVM algorithm were too slow. The problem of speed is still a major question to be investigated, though much research has been done to speed up the algorithm. The discussions about efficient SVM algorithm are given in subsection 3.3.1.

Originally, SVM was designed for binary classification. The methods of extending SVM for multiclass classification are still being investigated. In subsection 3.3.2, we discuss different SVM approaches that deal with multiclass classification.

3.3.1 Decomposition method

Recently, a combination of a decomposition method [36] and sequential minimum optimization (SMO) [33] algorithms have enabled the design of fast SVM algorithms [37], [7]. The basic idea of the *decomposition method* is that the optimization problem (19) can be iteratively solved by partitioning the data into two sets and solving one set as a sub-problem while keeping another set fixed. For more details about the decomposition

method, one can refer to [7], [8], [37]. The convergence of the decomposition method was recently given by Chang et al. [7]. SMO provides an analytical solution for the optimization of a two-variable sub-problem. It is worth noting that another approach was proposed by Dong et. al. [12] to speed up the SVM algorithm, which permits dealing with large scale data.

In this study, we consider the decomposition method for ν-SVM problem in (19). The decomposition method is an iterative process. In each iteration, the index set of vectors is partitioned into two sets B and N, where B is a *working set*. During the process the vectors corresponding to set N remain fixed while a sub-problem of working set B is optimized. According to [9], algorithmically, the method can be stated as follows:

1. Given a number $q \le l$ as the size of the working set. Find α^1 as an initial feasible solution of equation (19). Set $k = 1$.
2. If α^k is an optimal solution of (19), then stop. Otherwise, find a working set $B \subset \{1,...,l\}$ whose size is q. Define $N \equiv \{1,...,l\} / B$, α_B^k and α_N^k to be sub-vectors of α^k corresponding to B and N, respectively.
3. Solve the following sub-problem with the variable α_B:

$$\min_{\alpha_B \in R^q} \frac{1}{2} \sum_{i \in B} \sum_{j \in B} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_{i \in B} \sum_{j \in B} \alpha_i \alpha_j^k y_i y_j k(x_i, x_j) \qquad (20)$$

subject to

$$0 \le \alpha_i \le \frac{1}{l}, \quad i \in B,$$

$$\sum_{i \in B} \alpha_i y_i = -\sum_{i \in N} \alpha_i^k y_i,$$

$$\sum_{i \in B} \alpha_i = \nu - \sum_{i \in N} \alpha_i^k.$$

4. Set α_B^{k+1} to be the optimal solution of the constraint $0 \le \alpha_i \le \frac{1}{l}$, $i \in B$, and $\alpha_N^{k+1} \equiv \alpha_N^k$. Set $k \leftarrow k+1$ and go to step 2.

Working set selection is an important part of the decomposition method. The selection strategy is based on the violation of the following condition, termed as Karush-Kuhn-Tacker (KKT) [36]:

$$\alpha_i \cdot \left[y_i \left(\sum_{j=1}^{l} \alpha_j k(x_i, x_j) + b \right) - \rho + \xi_i \right] = 0 \tag{21}$$

The problem lasts as long as the condition (21) is violated and feasible set of $\{\alpha_i\}$ is not obtained. Using the condition $0 \le \alpha_i \le \frac{1}{l}$ of (19) and the property $y_i = \pm 1$, in terms of dual form, the condition (21) can be rewritten as follows:

$$\begin{cases} \max\limits_{i \in I_{up}^1(\alpha)} \nabla W(\alpha)_i \le \rho - b \le \min\limits_{i \in I_{up}^1(\alpha)} \nabla W(\alpha)_i, \\ \max\limits_{i \in I_{low}^{-1}(\alpha)} \nabla W(\alpha)_i \le \rho + b \le \min\limits_{i \in I_{low}^{-1}(\alpha)} \nabla W(\alpha)_i, \end{cases} \tag{22}$$

where

$$\nabla W(\alpha)_i = \sum_{j=1}^{l} \alpha_j y_i y_j k(x_i, x_j),$$
$$I_{up}^1(\alpha) := \{i \mid \alpha_i > 0,\ y_i = 1\},$$
$$I_{low}^1(\alpha) := \{i \mid \alpha_i < 1/l,\ y_i = 1\},$$
$$I_{up}^{-1}(\alpha) := \{i \mid \alpha_i < 1/l,\ y_i = -1\},$$
$$I_{low}^{-1}(\alpha) := \{i \mid \alpha_i > 0,\ y_i = -1\}.$$

This method is also called SMO-type decomposition method for the size of the working set, where it considers the sub-problem consisting of a pair of feature vectors in working set *B*. The selection of the working set is performed by calling a pair of vectors *i*

and j such that $(i,j) \in I_{up}^{1}(\alpha) \times I_{low}^{1}(\alpha)$ *or* $I_{up}^{-1}(\alpha) \times I_{up}^{-1}(\alpha)$ and violates (22). While a set of $\{\alpha_t\}$ is not optimal, the algorithm keeps looking for any pair of vectors, which does not satisfy (22).

The *stopping condition* of the decomposition method is derived from the optimality conditions in (22). The decomposition method stops when the following condition is satisfied:

$$\max\left(\left(\max_{i \in I_{up}^{1}(\alpha)} \nabla W(\alpha)_i - \min_{i \in I_{up}^{1}(\alpha)} \nabla W(\alpha)_i\right), \left(\max_{i \in I_{low}^{-1}(\alpha)} \nabla W(\alpha)_i - \min_{i \in I_{low}^{-1}(\alpha)} \nabla W(\alpha)_i\right)\right) < \varepsilon, \qquad (23)$$

where $\varepsilon > 0$ is a user-set stopping tolerance.

This algorithm has been chosen as the basis of our algorithm. However, some modifications have been made to tailor the algorithm for the problem specifications considered in this study. The core parts of the algorithm such as optimization, working set selection and stopping condition, are kept the same. However, the other procedures such as "kernel caching", "digest strategy" and others were implemented. Kernel caching is a programming trick that ensures saving the temporary kernel values in the memory storage. Digest strategy is a rule used to update kernel values. Moreover, a new file indexing scheme was developed that uses the saved feature vector addresses in the data file to speed up file accessing. Overall, the performances of these algorithms are the same, but my algorithm is a bit slow.

3.3.2 Multiclass SVM

Currently, there are two types of approaches for multiclass SVM: constructing a combination of several binary classifiers, and considering directly all classes together ("all-together") in one optimization problem. According to a comparison between binary classification-based and "all-together" support vector classifiers considered in [20], the second approach is slow. Solving "all-together" multiclass classification is computationally expensive, though the performance is similar to the binary classification-based strategies. There are three binary-classification based strategies: "one-against-all", "one-against-one" and directed acyclic graph SVM (DAGSVM).

"One-against-all" is probably the first strategy to extend SVM for multiclass classification [36], [12]. It constructs k SVM classifiers, where k is the number of classes. For each class it builds a two-class classifier with positive and negative class labels, where the samples of the considered class are labeled as positive while all samples of the remaining classes are labeled as negative. Then, it obtains k decision functions by solving k binary problems. The right class is determined by the largest positive value of the decision functions.

Another binary classification-based strategy is *"one-against-one"*. It is a common learning strategy, which was used in SVM for the first time in [36]. In this strategy, $k(k-1)/2$ two-class SVM problems are constructed from each pair of classes for the given k-class problem. The same number of classifiers is trained on each of the two-class data by labeling two counterpart classes as positive and negative, respectively. For the testing stage, the voting scheme suggested in [14] is commonly used. This voting scheme is also called "Max Wins" algorithm. When a pattern x is given to the decision function of the

pair of classes i and j, the algorithm adds a vote either for class i if the decision function returns a positive value, or for class j if the decision function returns a negative value. Similarly, it goes through the rest of $k(k-1)/2-1$ classifiers giving a vote for the winning classes. Then, the class with the maximum number of votes is assigned to the pattern. In case several classes have the same amount of maximum votes, it simply assigns the one with the smallest index.

The last binary classification-based strategy is DAGSVM [35]. Its training stage is the same as "one-against-one". It also constructs and solves $k(k-1)/2$ problems for the given k-class problem. However, in the testing stage, it uses a rooted binary directed acyclic graph with $k(k-1)/2$ internal nodes and k leaves. Each node is a binary classifier. Starting from the root node, the nodes of the graph are traversed down to a leaf by evaluating SVMs at each node. With respect to the output of SVM at a node it moves to either right or left until it reaches one of the leaves. The true class is determined by the leaf where the graph ends its path for a given pattern x. For a graphical illustration of DAGSVM please refer to Figure 11.

Here we present a brief overview of all three strategies with respect to their training times, testing times and recognition performance. According to [20], the recognition performance is more or less the same for all of the strategies. However, their speeds in training and testing stages vary a lot. Even though "one-against-all" has fewer SVM problems rather than "one-against-one" and DAGSVM, it is still slow. This is due to the large amount of data given for the training of a single problem. We would also like to point out that the strategies "one-against-one" and DAGSVM have the same training speed. Regarding the testing time, Hsu et al. [20] postulates that in general the testing

time is proportional to the number of support vectors, which means that the more support vectors obtained during the training stage, the longer the testing lasts. Among all, DAGSVM has the fastest testing speed. It is mainly due to the acyclic directed graph used in the testing, which avoids unnecessary computations by going through just *k-1* nodes out of *k(k-1)/2* nodes. However, DAGSVM returns a single hypothesis about the problem, which is ineffective when multiple hypotheses are required. For information about how to make multiple hypothesis using SVMs, one may refer to [27]. Overall, except for the training time all other factors are similar for all three strategies. For practical uses, "one-against-one" and DAGSVM strategies are more suitable [20].

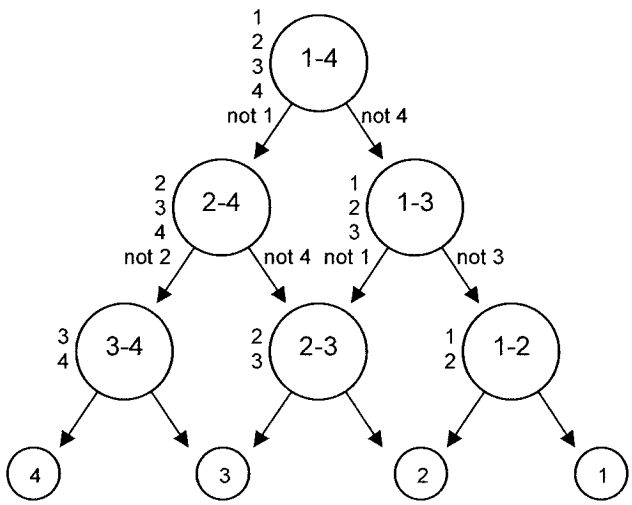


Figure 11. Explanation of DAGSVM

CHAPTER 4
SYNTHETIC DATA GENERATION

In this chapter, synthetic data generation is discussed. It is based on morphing transformation and other ad hoc techniques designed to support the optimal use of morphing. The theoretical and practical issues of morphing are considered.

In section 4.1, morphing transformation is discussed. Graphical objects such as curves, regions, and images, are defined. Warping filters along with amplitude filters, the components of the morphing, and morphing itself are described. Then, warping by normal and convex curve evolutions is specified. This section also discuses the principles of good morphing.

In section 4.2, practical algorithms for morphing transformation of handwritten numerals are described. An algorithm of warping and morphing for generating intermediate images between two numerals is described.

In section 4.3, the process of the data synthesis is described. First, a selection strategy for handwritten numeral pairs, source and target samples, is described. Then, the synthetic data evaluation method is described.

In section 4.4, a comparison of this synthetic data generation method with other methods is given. Advantages as well as limitations of each method are discussed.

4.1 Morphing Transformation

In Computer Graphics, morphing transformation is a widely used technique for animation, visualization and blending of graphical objects. Morphing is also known as warping in the case of planar shapes. Several morphing techniques such as parameter-based, feature-based, free-form based and physics-based techniques available in the literature [2], [41], [42]. Additionally, morphing techniques are studied by Gomes et al. and presented as a book and a chapter in [17] and [16], respectively.

4.1.1 Basic definitions

In Computer Graphics, the concept of a graphical object encompasses various subjects such as curves, surfaces, images, solids (volumetric objects) and others. According to [16], a *graphical object* is a subset in the Euclidean space $U \subset R^n$, and a function $f : U \to R^p$. The set U and the function f of the object describe its shape and attribute properties, respectively. The shape describes the object's topology and geometrical form, whereas the attribute characterizes its content such as color, texture, material and so on.

In this study, the images of handwritten numerals and their transformations are considered. The shapes of the handwritten numeral images can also be characterized as planar curves as well as regions. Therefore, the above graphical objects, planar curves, regions, and images, are defined.

Curve is a basic graphical object. It can be defined as a 1D graphical object (The dimension of the graphical object is derived from the dimension of its shape U). It is called a *planar curve* when the shape of the curve is a subset of the plane. In other cases,

we may refer to it as a *topological curve*. The planar curve is called a *Jordan curve* when it is homeomorphic to a circle. In other words, it is a closed planar curve without self-intersections.

The Jordan curve divides the plane into two regions, bounded and unbounded. An object comprised of a number of bounded Jordan curves describes the *planar regions*. The planar regions represent 2D graphical objects on the plane.

Another important graphical object is an *image*. It is a graphical object with the shape consisting of a subset $U \subset R^2$, and the function $f : U \to R^n$. Normally, its shape is a rectangle on the plane. The attribute of the image is defined by a function $f(p)$ for each point $p \in U$. The most common attribute of the image is its color.

4.1.2 Warping and amplitude filters, and morphing

Transformations between graphical objects are basic operations in computing with graphics. There are three techniques that describe these transformations. The first technique is called a *warping filter*, which includes all transformations that are designed to deform the shapes of graphical objects. The second technique describes the attribute changes in graphical objects, and is called an *amplitude filter*. In general, it is used to deal with color intensities. The third technique describes the process of transition between the graphical objects. It is called *morphing*. Warping filters along with amplitude filters are used to define the morphing technique. Morphing involves a continuous series of transformations in both the shape and the attribute properties of a graphical object. By repeatedly applying these transformations, one can obtain a transition of one object towards another. The first object, where the transition starts, is called the *source*. The

second object, at the end of the transition, is called the *target*. Intermediate objects that occur during the transition will be referred to them as *frames*. Furthermore, the role of warping and amplitude filters in designing morphing, and the morphing itself will be discussed.

Warping filters play a key role in morphing. They are used to specify the mapping of the source's shape to shape of the target. The mapping made by a warping filter is called forward mapping for the warping of the source towards the target. It is referred to as inverse mapping when the warping is performed in the opposite direction. In order to properly describe the warping technique it is necessary to make a few assumptions about the mapping. Let us assume that the mapping is both bijective and continuous. In other words, both the forward mapping and the inverse mapping are permitted (admitted) and continuous. When these properties of the mapping are in place, then the correct specifications of warping are possible. Warping techniques can be classified into several strategies with respect to their specifications:

- Parameter-based
- Feature-based
- Free-form based
- Physics-based

Parameter-based warping can be controlled by a few parameters, e.g. scaling, skewing. Feature-based warping is designed to preserve the feature correspondence between the graphical objects [2]. Free-form based warping is specified by the free-form curves such as B-splines, Bezier, and others [41]. Physics-based warping is computed from the

simulations related to the physical world. The method in [40] considers warping of two wires by bending and stretching their curvature. (For the warping method used in this study, refer to section 4.1.3.)

Amplitude filter is used to create color transitions between images. The most common technique that performs a linear interpolation of the colors between two images is called *cross-dissolving*. For the given source f and target g, cross-dissolving is performed as follows:

$$h_\lambda(x,y) = (1-\lambda)f(x,y) + \lambda g(x,y), \quad 0 \le \lambda \le 1 \tag{24}$$

where λ is a transition parameter. At the initial stage of the transition $\lambda = 0$, the color of the resulting image remains the same as the source image. At the final stage of the transition $\lambda = 1$, the resulting image becomes the target image. For the rest of the stages, the fraction of colors from the source and target images is regularized by (24).

According to [17], morphing can be described as continuous families of transformations. Let us assume that a subset $V \subset R^n$, an *n-parameter family of transformations*, is given as a map $h: V \subset R^n \to \Phi$, where Φ is a space of transformations and V is a *parameter space*. Then, each of the $(k_1, k_2, ..., k_n) \in V \subset R^n$ corresponds to a single transformation in the family, which can be denoted as $h_{(k_1,k_2,...,k_n)}$. An example of a one-parameter family was given in (24) for the cross-dissolving between images, where the transition between images was described by parameter λ. As it varies from 0 to 1, a continuous transition between images is obtained. Such a continuous sequence of images indicates an *animation*.

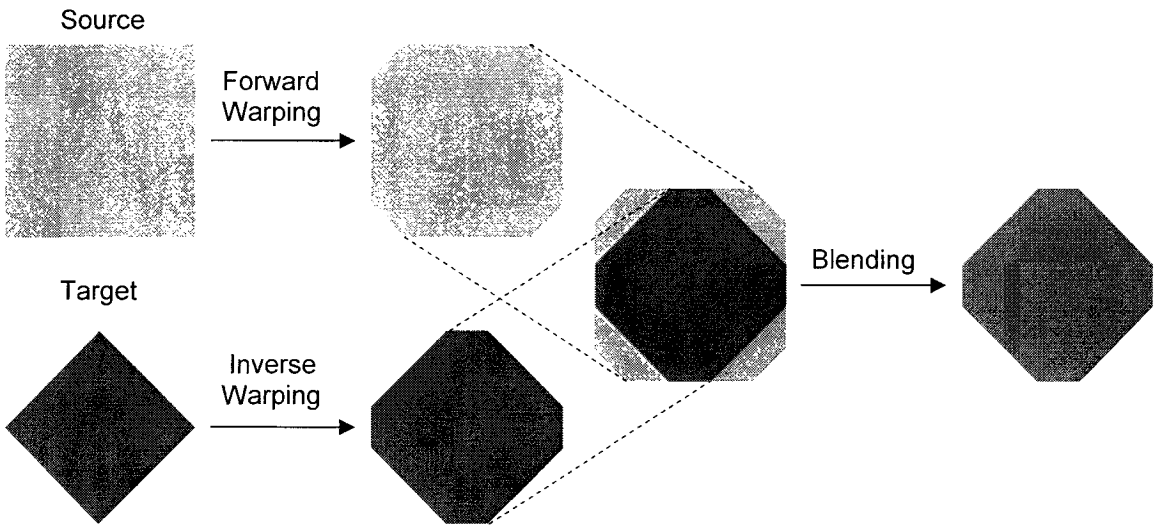


Figure 12. Morphing strategy

In general, morphing is performed in accordance with feature preservation, which maintains the proper correspondence between the distinguished topology elements of the graphical objects. Therefore, it is necessary to use local transformations of the shape in order to obtain a feature alignment. Let $h_\lambda, 0 \le \lambda \le 1$, be a morphing sequence between the graphical objects f and g. Then, for each λ, a graphical object h_λ, which has an intermediate shape and attribute, can be obtained. That is, the resulting object is a *blending* of the source f and the target g. In fact, the blending is weighted so that when λ is close to 0, the blended object resembles the source f, and when λ is close to 1, it resembles the target g.

A common strategy for morphing is given as follows:

1. Perform forward warping.
2. Perform inverse warping.
3. Blend the warped objects in 1 and 2.

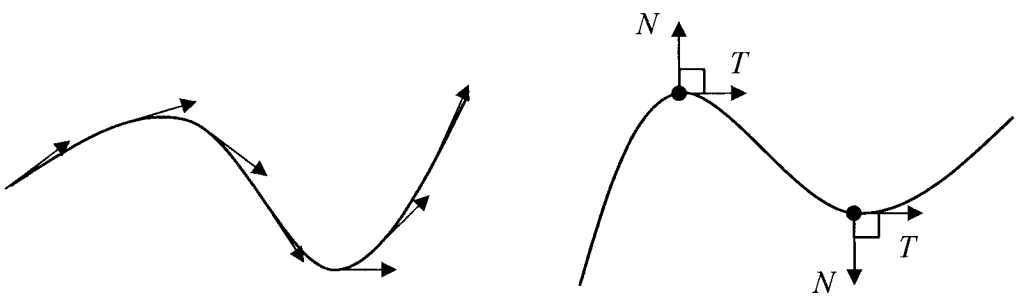
The forward and inverse warpings bring the source and the target closer together so that it is easier to devise good techniques to blend them. This is also illustrated in Figure 12. In short, this strategy indicates that during the morphing process the objects undergo a geometrical alignment before the shape and attribute of the intermediate object are computed. In the literature, this is summarized symbolically as follows:

$$\textbf{Morphing} = \textbf{(Warping)}^2 + \textbf{Blending}, \tag{25}$$

which indicates that the blending of the results forward and inverse warpings defines morphing.

4.1.3 Warping by curve evolution

In this study, warping by curve evolution, one technique of defining morphing, is used. This technique can be interpreted as an evolution of the interface between two regions on the plane. Physical properties related to each of the regions can be used to obtain the interface evolution. Curve evolution is based on the idea of inflation of a balloon until it becomes a circular shape.



a) Velocity vector field b) Tangent and normal vectors

Figure 13. Planar curve properties

Let us assume that a Jordan planar curve φ is given and parametrically specified as $\varphi(t)=(x(t),y(t))$, where t is a parameter of the curve. As a geometric nature, this type of curve has properties such as tangent and normal vectors, and curvature. For other properties, one may refer to [18]. The derivative $\varphi'(t)$ of the curve $\varphi(t)$ defines the *velocity vector field* of the curve. For any point t_0 on the curve, where $\varphi'(t_0)\neq 0$, this vector defines a *tangent* to the curve at the point $\varphi(t_0)$. A line equation of the tangent is given by $T(s)=\varphi(t_0)+s\varphi'(t_0)$, $s\in R$. Furthermore, let us assume that the velocity vector field is unitary and the curve φ is parameterized by arc length. In other words, geometrically, the parameterization function preserves the length of the intervals where the parameter t is defined. Then, the tangent vector $T=\varphi'(t)=(x'(t),y'(t))$ is unitary and a unit *normal* vector N is defined by $N(t)=(-y'(t),x(t))$. That is, the normal vector N is obtained by a rotation of 90^0 of the tangent vector T in the counterclockwise direction. For the illustrations of the velocity vector field, as well as tangent and normal vectors, refer to Figure 13. Since the unit tangent vector T has a constant length, it follows that $\langle T,T\rangle=1$. Deriving this equation, we obtain $\langle T',T\rangle=0$. That is, the derivative T' of the tangent vector is orthogonal to T. It also follows that there exists a scalar $k(t)$ such that $T'(t)=k(t)N(t)$. The function k is called the *curvature*. The curvature of the planar curve $\varphi(t)=(x(t),y(t))$ can be computed as follows:

$$k(t)=\frac{x'(t)y''(t)-x''(t)y'(t)}{\left(x'(t)^2+y'(t)^2\right)\sqrt{x'(t)^2+y'(t)^2}}, \tag{26}$$

where x', y', x'', and y'' denote first and second derivatives.

Now, let us define *curve evolution*. It is a member of the one-parameter warping family of curves. An evolution of the curve $\varphi(t)$ is a family of curves $\varphi(t,s), s \in [0,\infty)$, where the parameter s is an evolution parameter, provided that $\varphi(t,0) = \varphi(t)$. For each s, there is a transformation $h_s : \varphi \to R^2$, which defines a curve $\varphi(t,s)$. For each point $p = \varphi(t_0)$ on the curve, $\varphi(t_0,s)$ defines the animation path of the point p along the evolution. The velocity vector $\partial\varphi(t_0,s)/\partial s$ is called the *evolution velocity vector*.

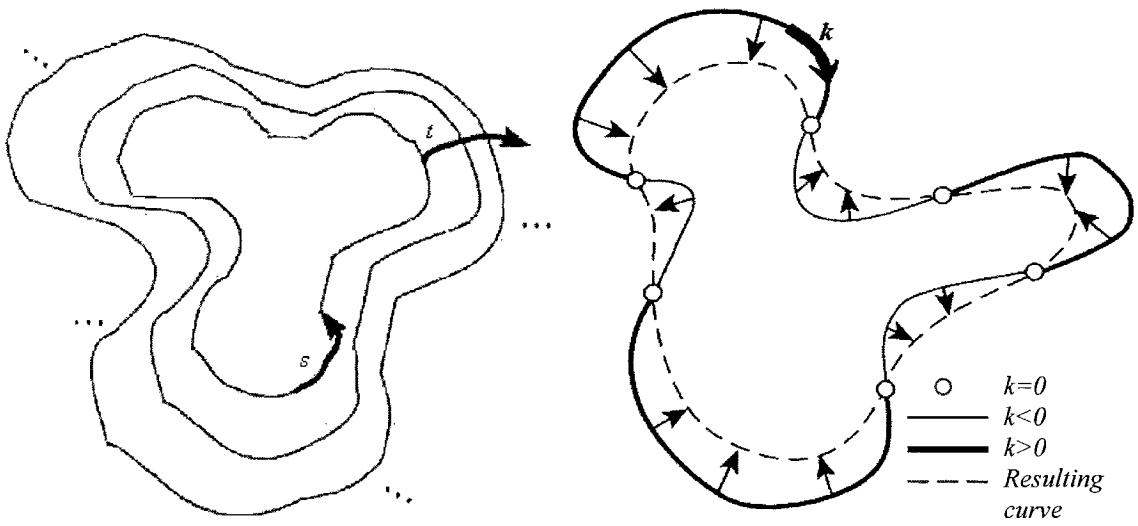
The curve evolution is called *normal evolution* when the evolution velocity vector is orthogonal, as shown in Figure 14 a). If N is the unit normal vector to the curve $\varphi(t)$, then the equation of the normal evolution can be given by

$$\frac{\partial\varphi(t,s)}{\partial s} = F(k(\varphi(s)))N(\varphi(t)), \tag{27}$$

where the scalar function F is the speed function of the evolution. In the case where the speed of the evolution is proportional to the curvature, it is called *convex evolution*. Its speed function F is given as follows:

$$F(\varphi(s)) = \alpha k(\varphi(s)), \quad \alpha = const. \tag{28}$$

Figure 14 b) illustrates the evolution of a curve with $\alpha < 0$. Note that the inflection points $(k = 0)$ remain fixed. Convex points $(k > 0)$ move inward, and concave points $(k < 0)$ move outward. In other words, convex sides shrink inside, whereas concave side expand outside. The distant points of both convex and concave sides move faster than the closest points.



a) Normal evolution b) Convex evolution

Figure 14. Curve evolution methods [from Gomes et al. 1999]

Curve evolution is designed to deal with a single planar curve, which evolves until it becomes a circular shape. In normal evolution, the shape evolves along normals. It expands until it becomes a circular shape at the end. In contrast, in convex evolution the shape evolves with respect to its curvature. Making use of the curvature information, the curve is divided into convex and concavity sides. When evolution begins, the convex sides start to shrink, whereas the concavity sides start to expand. This also results in a circular shape at the end. For more studies about convex evolution, one can refer to [19].

In practice, convex evolution can be performed without curvature information. Particularly, two overlapping similar shapes can replace the curvature information. Making use of the overlapping shapes we can determine convex sides, concavity sides and inflection points of the shape. Let us take a closer look at Figure 14 b), where the process of convex evolution is displayed. The bold solid curves indicate the convex sides of the curve, whereas the thin solid curves indicate the concavity sides of the curve. The dashed curve indicates the resulting curve of the evolution at this stage. Now, let us

assume that the solid curve is the source shape and the dashed curve is the target shape. Then, it is obvious that warping of the source towards the target by convex evolution is feasible. Similarly, this concept can be extended to the warping of handwritten numerals.

4.1.4 Principles of good morphing

In principle, it is difficult to precisely define an "optimal" morphing because it is application dependant. According to [18], morphing involves perceptual issues in medical image analysis, and the special effects industry, whereas the mathematical meaning of optimal morphing can be obtained in physics-based applications.

In this study, some of the guidelines of "good" morphing from [18] are used. We found it useful to maintain the following principles of good morphing:

- Topology preservation
- Smoothness
- Nonlinearity

For the homeomorphic shapes, the topology preservation principle is required. Smoothness is useful to avoid tears and rips during the morphing process. The continuity of warping preserves smoothness. Lastly, we should avoid using linear transformations. They do not perform well in the objects that have complex topologies. For example, rotational transformations cannot be performed by linear interpolation. Therefore, better morphing results are obtained by using nonlinear transformations.

4.2 Algorithms for the Morphing of Handwritten Numerals

In this study, morphing is used to generate synthetic samples of handwritten numerals. Morphing is specified by convex evolution warping and cross-dissolving. The handwritten numerals are characterized as images, planar curves and regions. For warping by convex evolution, we used contour images from two handwritten numerals. Each contour image serves as either discrete planar curve or a region of the numeral. When two overlapping discrete planar curves of numeral shapes are given, they can be smoothly evolved by curve evolution to each other. The numeral images are used for cross-dissolving. In cross-dissolving, color intensities between two images are linearly interpolated using equation (24) in order to determine the color of the intermediate images.

4.2.1 Algorithm for warping by convex evolution

A simple iterative algorithm based on contours of the numeral characters was developed for the warping by convex evolution. At each iteration, the algorithm performs the following four steps for the given two images of the handwritten numerals, the source and the target:

1. Locate the target over the source so that the Hamming distance (pictorial difference) between them is minimal.
2. Compute the contour of the source.
3. Identify the convex and concave sides of the source making use of the normals estimated from two neighbouring contour directions, and determine the moving directions for each point of the contour.

4. Calculate the moving priority for each contour point of the source, and move the top three high priority points along their moving directions.

This algorithm repeats the above iteration until the source coincides with the target. Note that after each iteration the source will be assigned to the newly generated frame and the target remains the same throughout the evolution.

For a demonstration of this algorithm, the most confusing pair of handwritten numerals, the numerals "4" and "9", were chosen as an extreme case, as shown in Figure 15. Furthermore, each of the steps will be described.

At the initial stage, the target image is relocated over the source so that the Hamming distance between the images is the least, as in Figure 15 a). In other words, we find the best location for the target, where it finely fits the source.

The contour of the source is then extracted making use of a simple contour tracing algorithm, as in Figure 15 b). It is also chain coded with respect to Freeman directions. The contour information of the curve can be interpreted as the discrete velocity vector field.

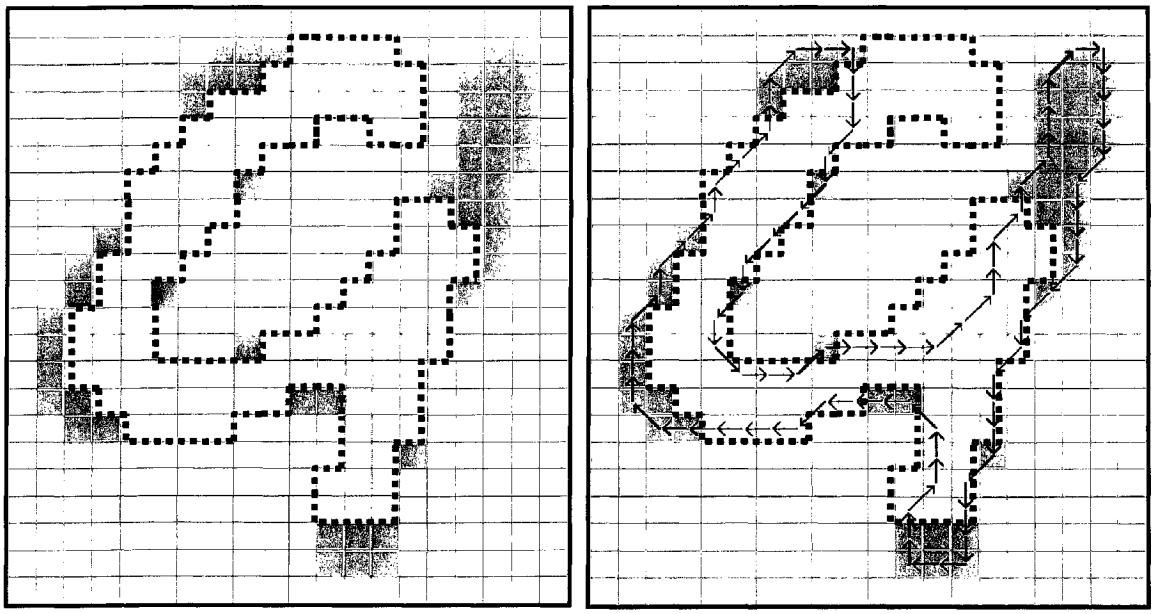
In the next step, the source shape will be prepared for convex evolution. First, the source curve is divided into convex sides, concavity sides, and inflection points, which are determined from the contours of the source and target shapes. These convex sides, concavity sides and inflection points are shown in Figure 15 c) with inward arrows, outward arrows and rectangles, respectively. The contour points of the source that are located outside of the target are classified as the convex points. The contour points of the source that are located inside of the target image are classified as the concavity points. The inflection points are determined from the contour points of the source and target

shapes that coincide with each other. Second, the normals at each point of the source curve are estimated from the two neighbouring contour directions. Then, the moving directions are determined from these normals with respect to convex and concavity sides. The normals of the convex side are reversed, whereas the normals of the concavity side remain unchanged. Note that there are no moving directions for the inflection points.

In the final step, the moving priority of the source is computed by measuring the distance of the contour points to the corresponding target points. For each of the contour points located at the convex sides, we calculate the distance to the closest target point that is located inside of the source. Similarly, for each of the contour points located at the concavity sides, we calculate the distance to the closest target point that is located outside of the source. To the farthest points the highest moving priority values are assigned. Thereafter, the top three points with the highest priorities are selected and moved along their moving directions.

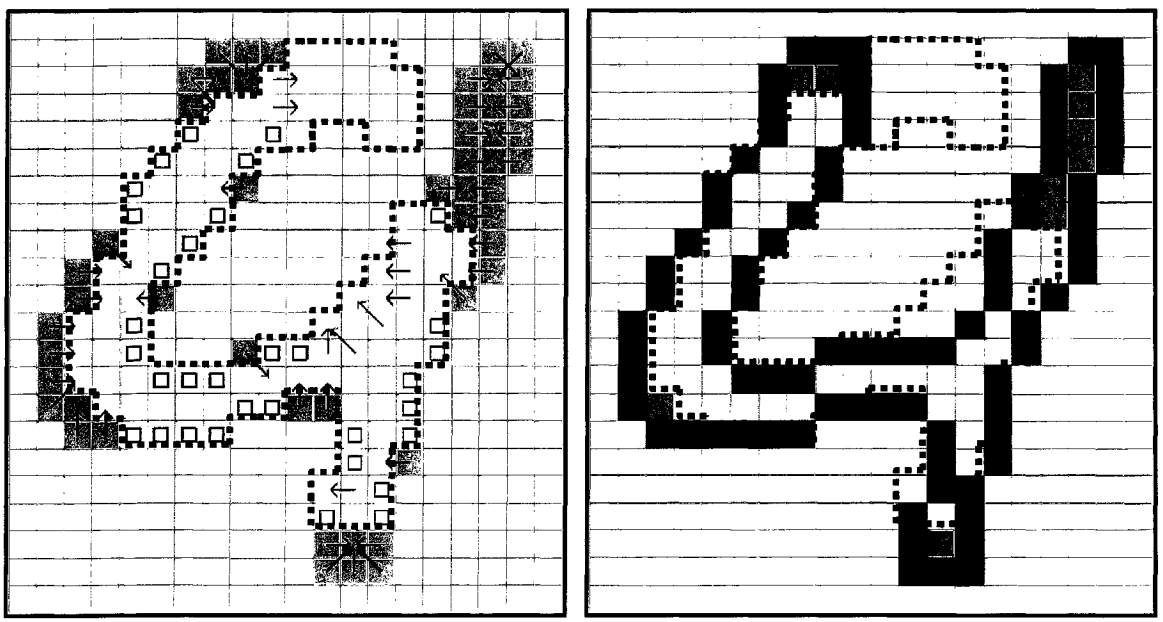
This algorithm helps us to maintain the principles of good warping (morphing) such as topology preservation, smoothness, and nonlinearity. Topology limitations can be avoided by choosing numerals from the same category for warping. The numerals of the same category are assumed to have the same topology or at least they are very similar. Moving three or less points at a time can generate smooth transition between the curves of numerals. The above iterative algorithm also permits us to preserve the nonlinearity of the warping process because the high priority points are randomly selected at each iteration, which indicates that there is no linearity.

The results of this algorithm for the warping of two different variations of the numeral "1" are illustrated in Figure 16.



a) Initial state

b) Contour information

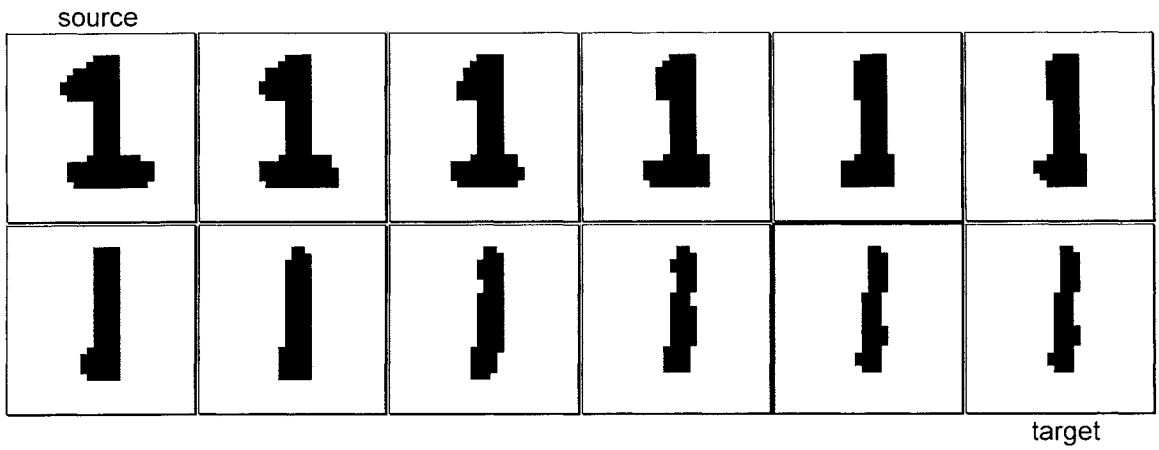


c) Moving directions

d) Moving priority

In these figures, the numeral "4" is shown with a light blue color and the numeral "9" is shown with a yellow color and bold dashed boundary. The points of numerals that coincide with each other are colored green. In the figure d), the moving priority for the contour points of the source is shown. The lighter color indicates a higher moving priority.

Figure 15. Warping by convex evolution



In this figure, the images on the top left and bottom right corners are the original images, the source and the target, respectively. The growth of the transition between them is shown in increasing order from left to right on the top row and continued along the bottom row from left to right.

Figure 16. An example of warping by convex evolution

4.2.2 Morphing strategy

For the description of morphing, an amplitude filter should be specified. We used a cross-dissolving amplitude filter to specify the color change between the images of handwritten numerals. MNIST, the dataset used in this study, contains samples of gray-scaled isolated numerals. Therefore, it is important to specify the proper color change. In cross-dissolving, the color change between the numeral images is regularized as in equation (24). However, the transition parameter λ needs to be determined. It can be defined by making use of the Hamming distance (pictorial difference) between the binary images. Let us mark the maximal pictorial difference as $d_{\max}$, and the current pictorial difference as d. Note that at the beginning of the transition the pictorial difference has the maximum value. Thus, the transition parameter can be computed as follows:

$$\lambda = 1 - \frac{d}{d_{\max}}. \tag{29}$$

In general, morphing transformation consists of forward warping, inverse warping and blending. The same strategy is used for the morphing of handwritten numerals. For the given two numeral shapes, the source and the target, we perform forward and inverse warpings, and blend them. Algorithmically, this looks as follows:

1. Forward warping is performed by evolving the source towards the target.
2. Inverse warping is performed in reverse direction.
3. Blending is computed by cross-dissolving the resulting images of forward and inverse warpings.

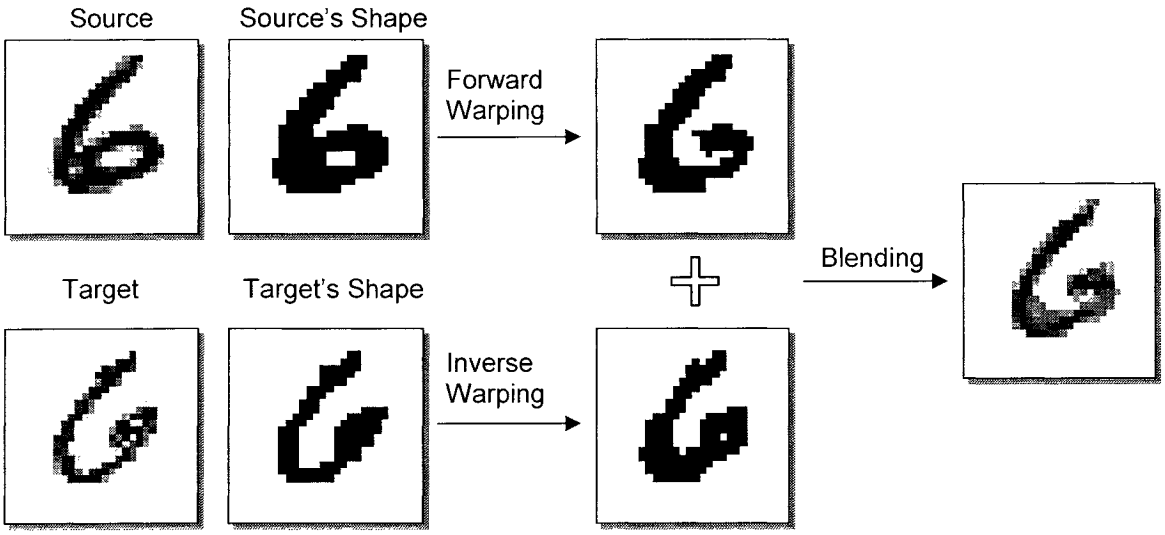
However, we should point out that the handwritten numeral images should be binarized at first. Afterwards, the above sequence of the transformations should be performed.

For the illustration of morphing transformation, we considered two different variations of the handwritten numeral "6", as in Figure 17. Firstly, we performed binarization for each of the source and target images. Then, we followed the above algorithm. Consequently, forward warping, inverse warping and blending of the resulting images were performed. As a result, we obtained a finely morphed sample, which represents an intermediate image between the source and the target.

In this study, the morphing transformation is designed to produce two intermediate samples for each pair of source and target samples. It generates the first intermediate sample by morphing the source towards the target. The second intermediate sample is produced by morphing the target towards the source. In other words, we change direction of morphing so that it morphs the target towards the source. In order to optimize

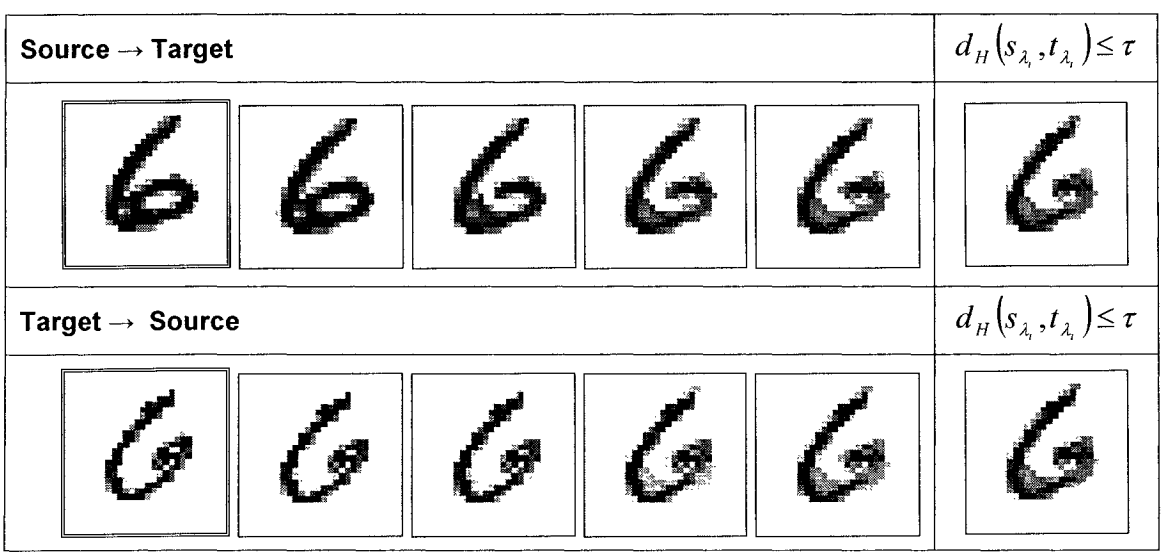
these operations, we developed an algorithm, which simultaneously performs these two morphing transformations. Thus, the source and target samples are transformed concurrently.

Let us assume that the source and the target are given, and labelled as s_0 and t_0 at the initial stage. Let us denote the morphing from the source towards the target as $h_0 : s_0 \to t_0$, and the morphing in reverse direction as $h_0^{-1} : t_0 \to s_0$. Then, as the transition λ grows from 0 to 1 so the source and the target are morphed. The relationship between *i+1*-th and *i*-th iterations (*i=1,2,...*) are given as $s_{\lambda_{i+1}} = h_{\lambda_i} : s_{\lambda_i} \to t_{\lambda_i}$ and $t_{\lambda_{i+1}} = h_{\lambda_i}^{-1} : t_{\lambda_i} \to s_{\lambda_i}$ for the morphing transformations from the source towards the target and in reverse direction, respectively. This algorithm lasts while the Hamming distance between the source and the target at *i*-th iteration $d_H\left(s_{\lambda_i}, t_{\lambda_i}\right) > \tau$. When the condition $d_H\left(s_{\lambda_i}, t_{\lambda_i}\right) \le \tau$ is satisfied, the algorithm stops and outputs the last images of these two transitions to be used as the synthetic data. In the stopping condition, τ denotes the tolerable pictorial difference between the source and the target. In this study, it was set equal to $d_{\max}/2$, where $d_{\max}$ denotes the maximal pictorial difference. An illustration of this algorithm is shown in Figure 18.



In this figure, the original images of numeral characters are shown under the notes "source" and "target". Right next to them, their corresponding shapes in binary forms are shown. The results from the forward and inverse warpings are shown after the arrows. The blending result of these images is placed on the far right. Note that the arrows in the figure indicate transformations and the "+" sign denotes the blending of the source and target images.

Figure 17. Morphing transformation



In this figure, the images of the source and the target at the initial stage of the transition are shown with double borders. The two transitions from the source towards the target as well as from the target to the source are shown under the notes "Source→Target" and "Target→Source", respectively. The last frames of these two transitions are the outputs of the morphing to be used as synthetic data.

Figure 18. Morphing transitions

4.3 Data Synthesis

The process of data synthesis consists of three major procedures: selection of source and target pairs (Selection Procedure), morphing transformation, and synthetic data evaluation (Evaluation Procedure). The data flow in data synthesis, which also includes the relationship between the procedures of data synthesis, is described as follows (see also Figure 2, in section 1.3.2):

1. A number of source and target pairs of handwritten numeral samples are identified by the selection procedure and passed to the morphing transformation procedure.
2. Morphing transformation generates two intermediate samples between source and target pairs and passes them to the synthetic data evaluation procedure.
3. The evaluation procedure examines whether these samples belong to the same category (class) of the source and target or not. If the samples belong to the same category, then they are qualified to be the synthetic samples.

In this section, we describe only the selection of the source and target pairs and synthetic data evaluation procedures because the morphing transformation procedure has already been discussed in sections 4.1-4.2.

4.3.1 Selection of source and target pairs

The selection procedure performs the selection of the source and the target pairs. A certain set of sources, defined by a user, should be given in advance. For example, the set of support vector samples can be used as the set of sources. Alternatively, the sources can be randomly selected from the training set. In the selection procedure, the sources are

fixed. For each source, the selection procedure assigns a number of target candidates and selects the best candidate for the source from these candidates. The selection is based on two conditions: Hamming and Euclidian distances between the source and the target candidates.

Hamming distance determines the pictorial difference between the source and the target candidate. It is expressed as follows:

$$d_H(s,t) = \sum_{i=1}^{h}\sum_{j=1}^{w}\left|s_{i,j} - t_{i,j}\right|, \tag{30}$$

where s and t denote the source and the target candidate, and h and w denote the height and width of the source and target images.

Euclidian distance is computed from the two GSC feature vectors (section 2.1) extracted from the source and the target candidates. A mathematical expression of the Euclidian distance is given in equation (6) in section 3.1.2. Here, we just denote the distance as d_E in order not to confuse it with the Hamming distance.

The selection procedure looks for the best target candidate that satisfies the following conditions:

$$\begin{cases} \min\limits_{t_i} d_H(s,t_i) d_E(s,t_i) \\ d_H(s,t_i) > \mu \end{cases} \quad i = 1,2,...,n, \tag{31}$$

where μ is a user-set constant and n is the number of target candidates.

The second condition in (31) is necessary to avoid the cases where the source and the target are identical in terms of the Hamming distance. It can also be interpreted as the acceptable pictorial difference between the source and the target candidate.

4.3.2 Synthetic data evaluation

The synthetic data evaluation procedure examines whether each of the morphing outputs belongs to the same category (class) of the source and the target or not (Also refer to Figure 2, in section 1.3.2). This is a useful technique that avoids accidentally generated erroneous samples during the morphing. The evaluation procedure performs exactly the same function as the testing module of the recognition system (Figure 1). First, it extracts the GSC features (section 2.1) from each of the outputs (frames) of the morphing transformation. Second, it standardizes these features and passes them to the synthetic data evaluation component of the data synthesis module. This component performs classification by making use of the support vector classifier derived from the original training. If the result of classification shows that the output is in the category (class) of its source and target with a high rank, then this output is qualified as a synthetic sample. All of these synthetic samples are accumulated into a set, which can be further referred to as the set of synthetic samples.

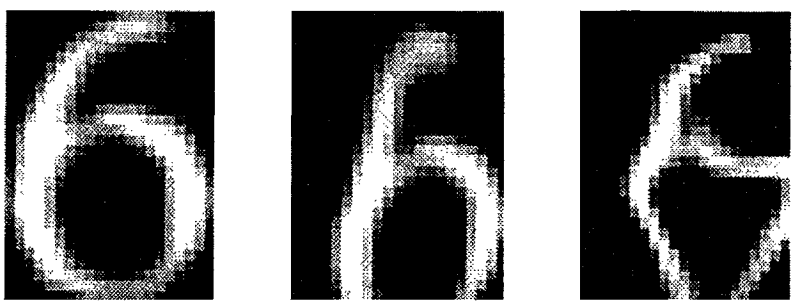
4.4 Comparison of the Outputs of Different Methods

In section 1.2, we mentioned affine and elastic distortions. In this section, comparison of morphing transformation to these distortions is given. The description of each method and its advantages and limitations are described.

Affine distortion (AD) generates the synthetic samples of the handwritten numerals by making use of affine transformations such as translation, rotation, and line thickness. For any given sample, AD generates a number of variations. Acquiring the translated, rotated, thickened and thinned versions of the given sample helps to increase

the performance. However, the number of transformations that it can perform is limited. Therefore, the samples generated by AD cannot cover all possible variations of the numeral samples.

Elastic distortion (ED) generates the synthetic handwritten numeral samples by making use of a random displacement and a Gaussian filter. First, ED randomly moves each pixel of the image within a pixel width to one of the four directions: left, right, up and down. Then, it is smoothed by the Gaussian filter. In Figure 19, the outputs of this technique are illustrated. The results of the elastic distortion, applied for the original image in Figure 19 a), are presented in Figures 19 b) and c). In contrast to affine distortion, elastic distortion is able to generate many variations of the numeral sample. The outputs of the elastic distortion are very diverse and some of the outputs may look unnatural as shown in Figure 19 c). Therefore, these kinds of synthetic samples may not always be useful for the training.



a) Original Image b) Distorted Image 1 c) Distorted Image 2

Figure 19. The results of elastic distortion [from Simard et al. 2003]

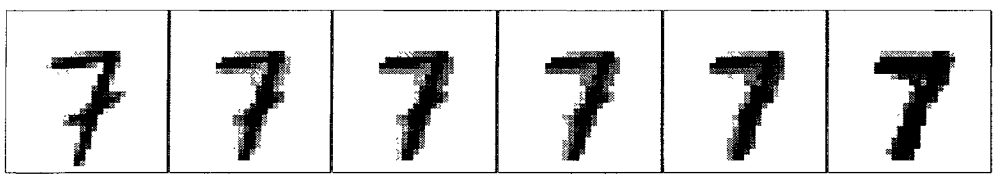


Figure 20. The results of morphing transformation

Morphing transformation generates synthetic examples between two handwritten numeral images, the source and the target. It creates a transition from one image to another. One example of such transitions is demonstrated in Figure 20, for two variations of the numeral "7": the numeral "7" with a cross and the thick numeral "7". Unlike affine and elastic distortions, morphing transformation creates a transition between two different samples. This is useful technique that balances the distribution between variations of a single class. If we assume that each of the variations of the numerals is a separate cluster, then morphing can help to fill the gaps between the clusters. Morphing transformation generates examples that cannot be obtained by the other data synthesis methods. However, morphing transformation may not solely replace the benefits of all other methods. Incorporation between them is still preferred.

CHAPTER 5
EXPERIMENTS

In this chapter, the experiments conducted are discussed. The experiment settings are described in section 5.1, where the database, the extracted features and the strategies of support vector classification and standardization are described. Afterwards, the two types of the conducted experiments, the experiments with and without synthetic samples generated by morphing transformation, are described in sections 5.2 and 5.3. The discussion about these experiments is given in section 5.4.

5.1 Experiment Settings

5.1.1 Database

In this study, experiments for handwritten numeral recognition were conducted on the MNIST dataset (LeCun [23]) consisting of 60,000 training and 10,000 testing samples of handwritten numerals. According to LeCun, the original binary numeral images were first size normalized into the 20x20 size preserving their aspect ratios. Then, these normalized images were centered with respect to their centers of mass and placed on the larger images with sizes of 28x28 pixels.

Additionally, the training set of MNIST was divided into two sets in order to estimate the SVC parameters. The first set is used for the training, and it contains 50,000 samples. The second set is used as the validation set, and it consists of 10,000 samples.

5.1.2 Extracted features

In total, 300 GSC features were extracted. Specifically, 200 gradient, 40 structural, and 60 concavity features were extracted from the smoothed and binarized images of the MNIST numerals, as described in chapter 2. The size of GSC feature vectors remained the same throughout the experiments.

5.1.3 Standardization and SVC strategies

For the standardization of the GSC features, normalization and scaling were applied (see section 3.1). In the normalization, feature vectors were normalized with respect to the mean and variance feature vectors. In the scaling, feature vectors were scaled so that the Euclidian distance between any pair of the feature vectors would lie in the interval of 0 and 1.

For the support vector classification (sections 3.2-3.3), ν-SVC with a "one-against-one" multiclass strategy was employed. For the testing part of this classifier, "Max Wins" algorithm was used. However, for the parameter estimation purposes, DAGSVM algorithm was also used to test the performance of the SVC. A Gaussian RBF was used as a kernel function of the SVC. The SVC parameters ν (nu) and σ (sigma) were determined by making use of a grid search during the experiments.

5.2 Initial Experiments

The performance of the handwritten numeral recognition system was evaluated by two different strategies with respect to the size of their training set. The training of the system without adding synthetic samples to the original training set was called *initial training*, where the only original training set with a size of 60,000 samples was used. The performance of the system that was trained on the support vectors only was called *support vector training*. For the support vector training, 4,278 support vector samples obtained by the initial training were separated from the original training set and used as a training set for the support vector training.

In this section, we discuss the initial training and the support vector training experiments. For both of these experiments, we first performed a grid search for the estimation of the SVC parameters. Then, making use of the estimated parameters their performances were tested on the MNIST testing set.

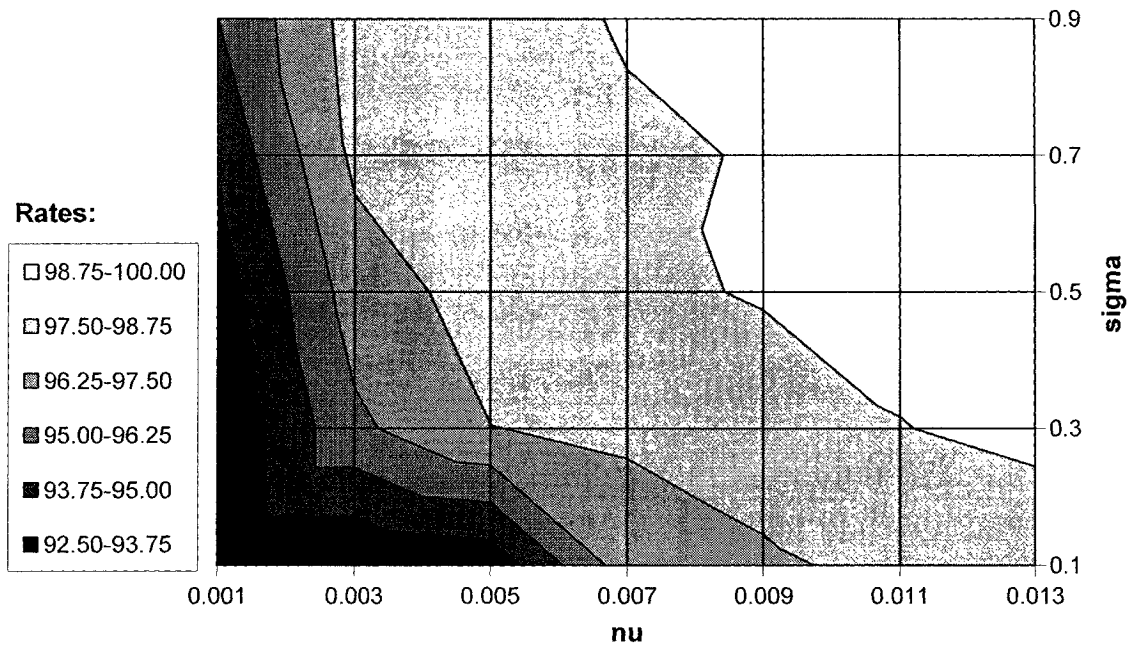
5.2.1 Initial training

In the initial training, a grid search was performed first in order to estimate the optimal area of the SVC parameters ν (nu) and σ (sigma). We ran the parameter ν from 0.001 to 0.011 by the step 0.002 and the parameter σ from 0.1 to 0.9 by the step 0.2. For each combination of these parameters, SVC trained on 50,000 training samples and its performance was tested on the validation set. (The stopping tolerance ε of the training, which was described in subsection 3.3.1, was set to 0.1 to speed up the training of support vector classifiers). The results of this search are given in Table 1, and graphically illustrated in Figure 21, where the six levels of the recognition performance can be seen from the top as an elevation from the lowest to the highest recognition levels. Additionally, the performance of the training with fixed $\sigma = 0.5$ and ν varying from 0.011 to 0.015 by the step of 0.002 is illustrated separately in Figure 22.

We selected the optimal parameters $\nu = 0.011$ and $\sigma = 0.5$ for the initial training. We set $\nu = 0.011$ because after that no more major improvements made. This point of the search ($\sigma = 0.5$ and $\nu = 0.011$ with the recognition rate of 98.96%) belonged to the white area of optimal parameters, as shown in Figure 21.

Table 1. The results of grid search for the initial training

Rec. Rates		nu						
		0.001	**0.003**	**0.005**	**0.007**	**0.009**	**0.011**	**0.013**
sigma	**0.1**	84.77	88.93	92.90	96.87	97.33	97.79	98.31
	0.3	89.91	95.99	97.48	97.68	98.09	98.73	98.92
	0.5	93.04	96.86	98.02	98.50	98.85	98.96	98.98
	0.7	93.98	97.76	98.23	98.63	98.80	98.89	98.96
	0.9	94.99	98.01	98.40	98.82	98.88	98.94	98.98



In this figure, the six recognition levels are shown by six color transitions from black to white. The white area indicates the area with the highest recognition rates.

Figure 21. Contours of the different recognition levels of the grid search for the initial training

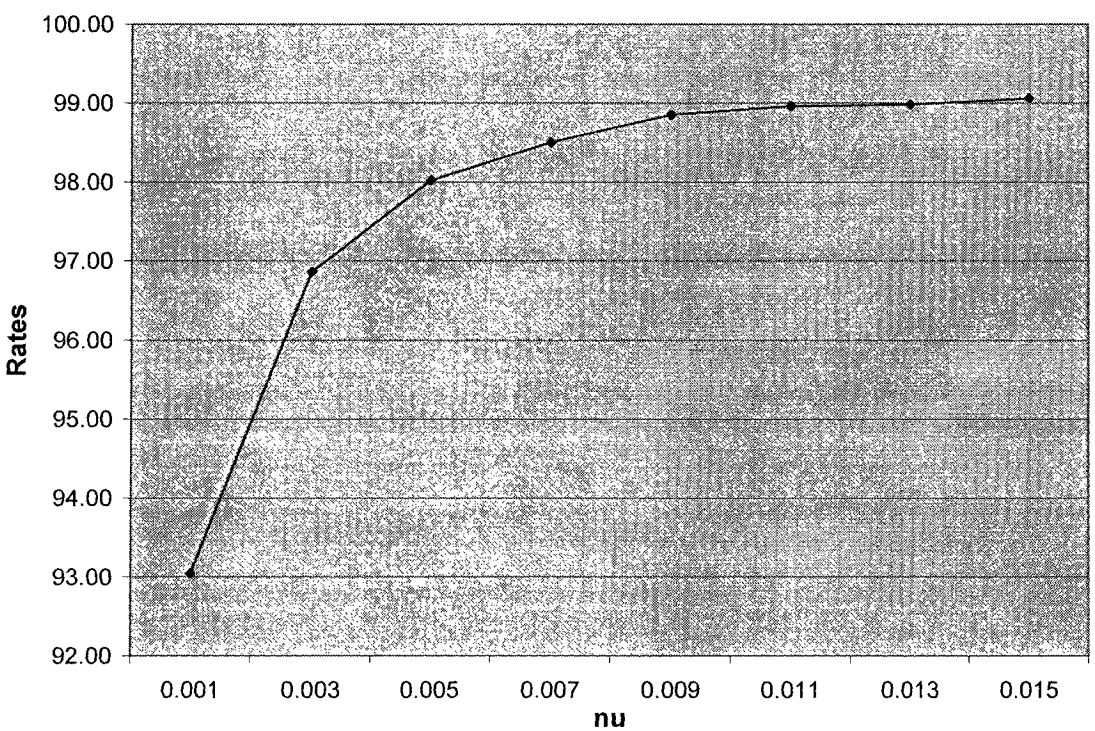


Figure 22. The performance of the search for the initial training with fixed sigma at 0.5

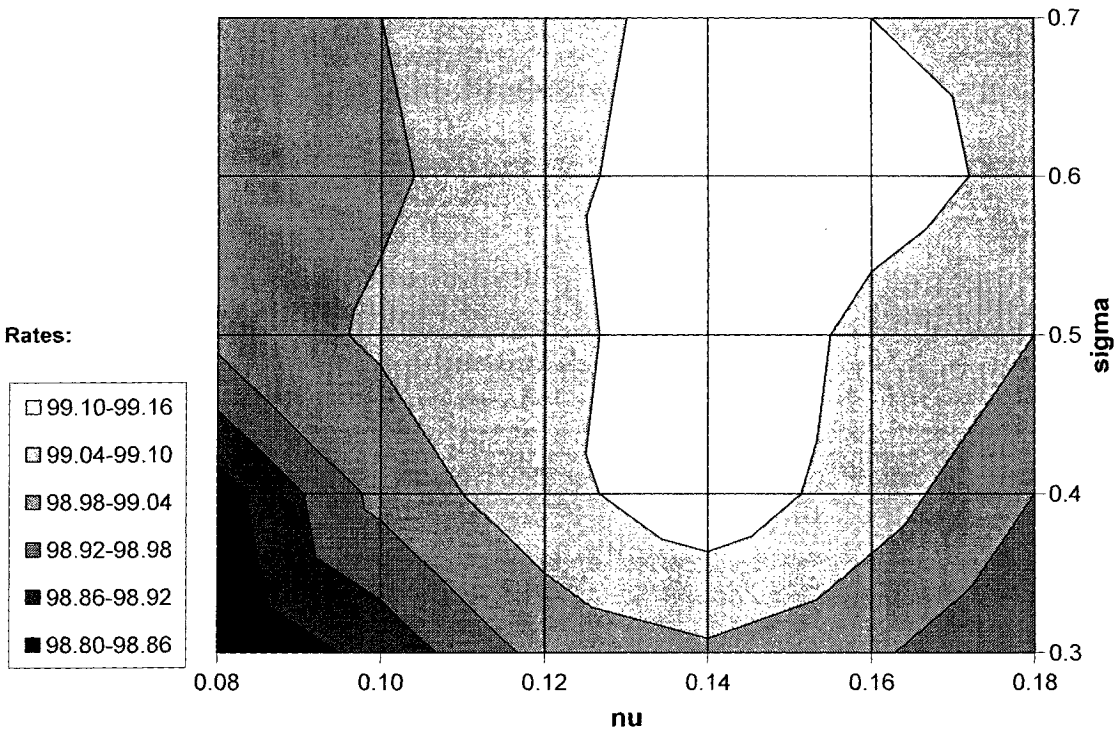
The initial training with the SVC parameters $\nu = 0.011$ and $\sigma = 0.5$ was performed on the full training set of 60,000 samples and tested on the testing set. (In this training, the stopping tolerance ε of the training was set to 0.01.) It reached 99.07% recognition rate.

In total, 4,278 support vectors were obtained by this initial training. It was used in another experiment, which is discussed in the next subsection.

5.2.2 Support vector training

In the support vector (SV) training, we also performed a grid search for the parameters ν (nu) and σ (sigma). However, this time the more detailed grid search was performed because the small size of the training set permits to perform fast SVC training. So, we ran the parameter ν from 0.08 to 0.18 by the step 0.02 and the parameter σ from 0.3 to 0.7 by the step 0.1. During the search, the SVC was trained on 4,278 support vector samples obtained by the initial training. Its performance was tested on the MNIST testing set. (The stopping tolerance ε of the training was set to 0.01.) The search results are graphically illustrated in Figure 23. Similar to the initial training, the performance of the training with fixed $\sigma = 0.5$ and ν varying from 0.01 to 0.20 by the step of 0.02 was demonstrated separately in Figure 24.

The best case of SV training reached the recognition rate of 99.12% with the parameters $\nu = 0.14$ and $\sigma = 0.5$.



In this figure, the six recognition levels are shown by six color transitions from black to white. The white area indicates the area with the highest recognition rates.

Figure 23. Contours of the recognition levels of the grid search for the SV training

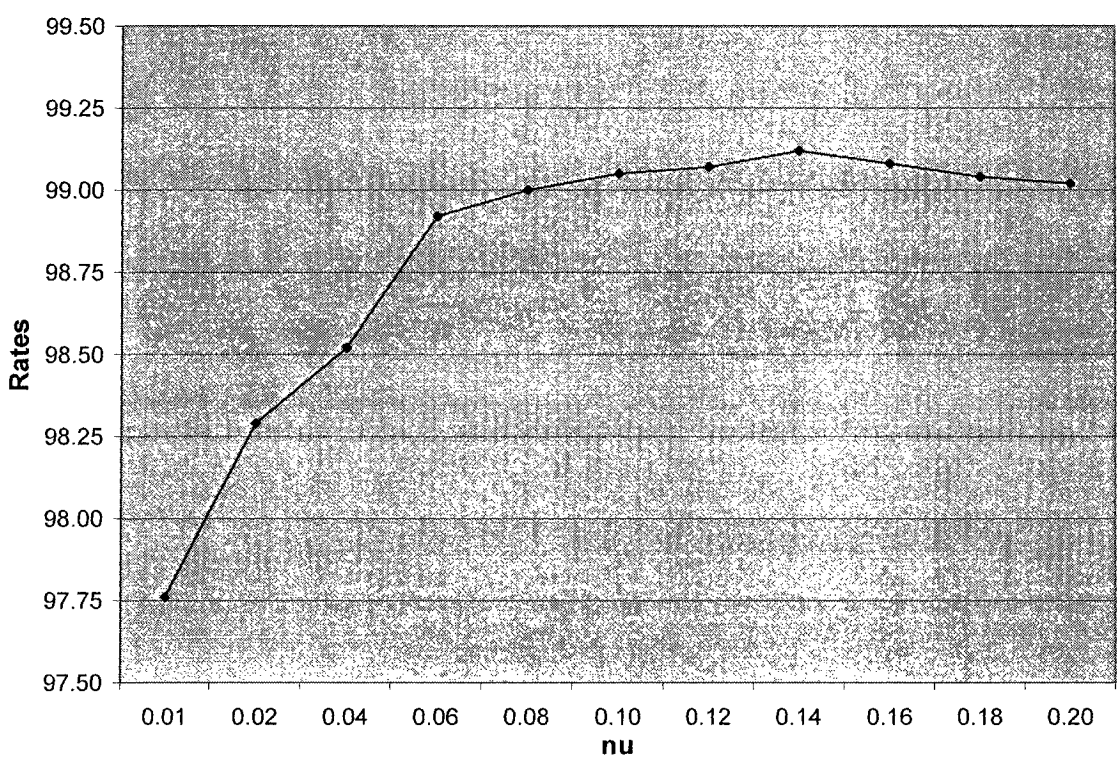


Figure 24. The performance of the search for the SV training with fixed sigma at 0.5

5.3 Experiments with Synthetic Samples

The performance of the extended training dataset with the synthetic samples (generated by morphing transformation) was investigated on both initial training and support vector training strategies. An experiment about the initial training with the training set extended by adding the synthetic samples (generated by morphing transformation based on clustering information) was called *extended training*. Another experiment about the support vector training with the training set extended by adding the synthetic samples (also generated by morphing transformation) was called *extended support vector training*. The optimal parameters obtained from the initial training experiment were used in the extended training. The optimal parameters of the support vector training also kept the same for the extended support vector training.

5.3.1 Extended training

In the extended training, the selection procedure based on clustering information was used for the generation of the synthetic samples. Using the clustering algorithm we aimed to balance the original data distribution by increasing the number of rare samples. In this algorithm, it is assumed that the examples that are closer to the cluster centers are better representatives of the cluster than the distant examples. Thus, the selection of the target candidates from the "good" representatives can be useful to generate good quality synthetic samples.

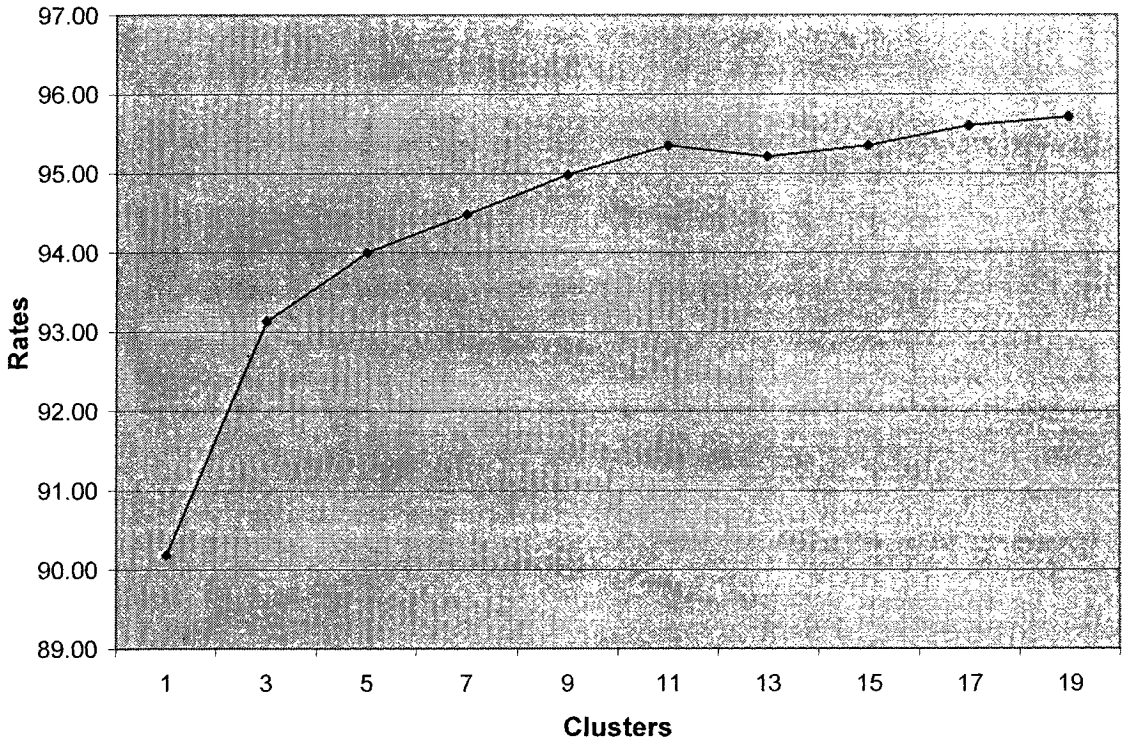


Figure 25. Clustering results

In order to estimate the "good" target candidates, the following steps performed. First, the training set was clustered by making use of the *k*-means clustering algorithm [16]. Second, the examples at each cluster were sorted in ascending order with respect to the Euclidian distance between the example and its corresponding cluster center. Then, we selected the ¾ of the sorted list as the fraction of the good representatives, discarding the remaining ¼.

The study about finding the sufficient number of clusters for the selection procedure was also conducted. First, the MNIST training set was divided into two sets with sizes of 50,000 and 10,000 samples, respectively. The first set was used as the training set, whereas the second set was used as the validation set. Second, the GSC features (section 2.1) from each of the sets were extracted and standardized. Then, *k*-

means clustering was performed on the training set and its performance was tested on the validation set by making use of nearest-neighbours classification [27]. For each of the odd k numbers from 1 through 19, the clustering was performed and tested. The results of this study indicate that after $k = 11$, the performance of the clustering is more or less the same, as shown in Figure 25. Thus, using 10 clusters per class should be sufficient for the selection procedure of the synthetic data generation method (described in section 4.3).

In order to increase the number of sloppy (rare) samples, the support vector samples obtained by the initial training were used. In total, 4,278 support vector samples were selected as the sources. For each source, the selection procedure found the best three targets from the number of target candidates with respect to the conditions based on Hamming and Euclidian distances described in section 4.3. Then, synthetic examples were generated between each source and its targets. In total, 25,021 synthetic samples were generated. Then, these samples were added to the MNIST training set of 60,000 samples. Thus, a new training set of 85,021 samples were created. The extended training was trained on this extended training set. The extended training making use of the parameters $\nu = 0.011$ and $\sigma = 0.5$ (obtained by the initial training) was performed and tested on the MNIST testing set. The performance of this extended training experiment reached a 99.06% recognition rate.

5.3.2 Extended support vector training

For the extended support vector training, the training set consisting of 4,278 support vector samples was increased by adding the synthetic samples, which were generated by morphing transformation. For each source of the 4,278 support vector samples, the selection procedure of the synthetic data generation technique (section 4.3)

paired it with the best target from the rest of the support vector samples. (The data syntheses for both of the extended training and the extended support vector training experiments were almost the same. The only difference was that the set of target candidates for the extended training experiment was determined by the clustering algorithm, while the set of target candidates for the extended support vector training experiment was taken from the set of support vectors) Then, for both experiments the synthetic examples were generated by morphing transformation between each of the source and target pairs. In total, 6,062 synthetic samples were generated by making use of this selection procedure. For the extended support vector training, we used the parameters $\nu = 0.14$ and $\sigma = 0.5$, obtained from the support vector training.

In order to determine the classes that would benefit from the morphing, for each class, the ten extended SV training experiments were performed on the training set of 4,278 support vector samples and the synthetic samples of each considered class (see Table 2). We took the best performance of the SV training, which had a recognition rate of 99.12%, as a base line in order to compare it with the results of these extended SV trainings. The differences in errors were computed by subtracting the recognition rate of the corresponding extended support vector training from the recognition rate of the SV training. The results of these experiments are presented in Table 2.

According to the results, the morphing was useful for the classes "3" and "6". The morphing also resulted in a major increase in errors for the classes "1", "7", "8" and "9". The rest of the classes had minor changes. The error differences of each class are illustrated in Figure 26.

Table 2. The difference between the SV training and the extended SV training of each class

Experiments	Training samples			Base line	Rec. rate	Diff. in errors
	Original	Synthetic	Total	A	B	A-B
Class "0"	4,278	433	4,711	99.12	99.10	0.02
Class "1"	4,278	669	4,947	99.12	99.02	0.10
Class "2"	4,278	492	4,770	99.12	99.10	0.02
Class "3"	4,278	549	4,827	99.12	99.14	-0.02
Class "4"	4,278	611	4,889	99.12	99.08	0.04
Class "5"	4,278	485	4,763	99.12	99.11	0.01
Class "6"	4,278	560	4,838	99.12	99.16	-0.04
Class "7"	4,278	655	4,933	99.12	98.99	0.13
Class "8"	4,278	787	5,065	99.12	99.04	0.08
Class "9"	4,278	821	5,099	99.12	99.01	0.11

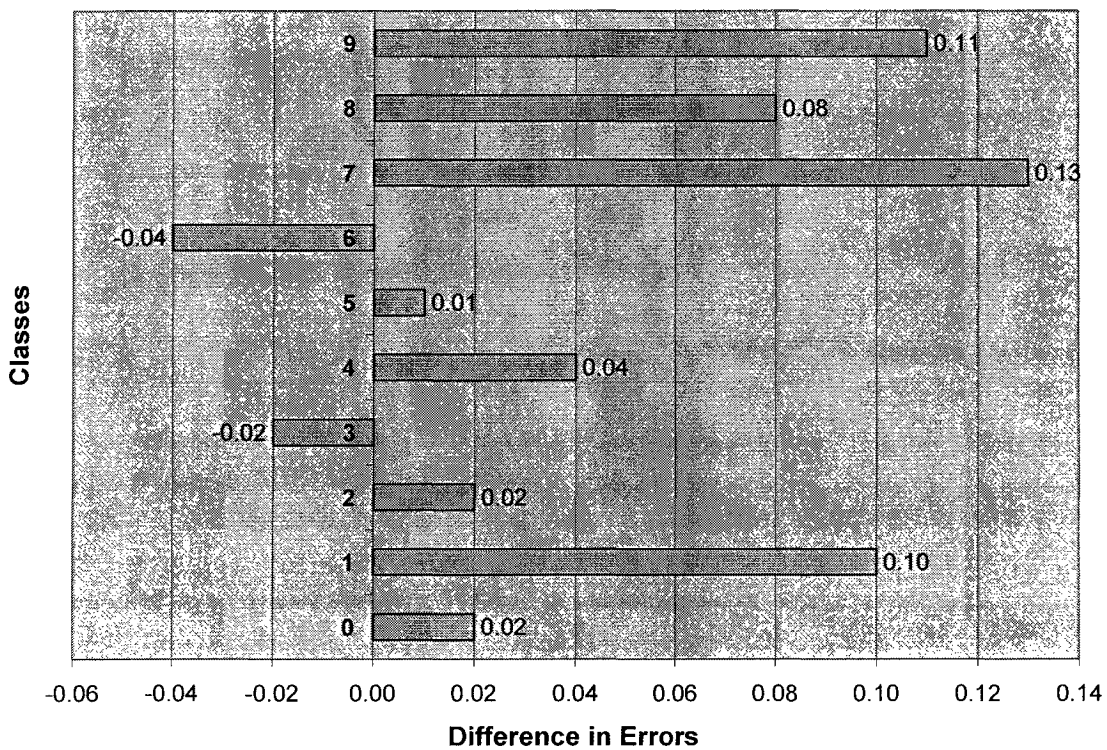


Figure 26. The difference between the SV training and the extended SV training of each class

Furthermore, we did experiments with the combinations of the best 2, best 3, best 5 classes and all classes combined. For the best 2 classes, we took the synthetic samples of the classes "3" and "6". For the best 3 classes, we considered using the synthetic samples of the classes "3", "5" and "6". For the best 5 classes, the synthetic samples of the classes "0", "2", "3", "5" and "6" were added to the training set. Lastly, we added all of the generated samples to the training set. Amongst them, the best result was attained from the best 2 classes of "3" and "6". A list of all 81 misrecognized samples of this experiment is presented in the appendix on page 92. The performances of these combinations of the classes are presented in Table 3. The number of errors per class was also given in this table.

Table 3. The performance with different combinations

Experiments	Training samples	Rec. rate	Number of errors per class										Total errors
			0	1	2	3	4	5	6	7	8	9	
SV Training	4,278	99.12	7	5	7	8	11	9	10	8	12	11	88
Best 2 {"3","6"}	5,387	99.19	7	6	6	7	8	8	6	11	11	11	81
Best 3 {"3","5","6"}	5,872	99.14	7	5	5	9	10	8	10	9	13	10	86
Best 5 {"0","2","3","5", and "6"}	6,797	99.13	5	3	7	10	11	8	10	10	13	10	87
All Classes Combined	10,340	99.11	6	3	9	8	8	8	13	6	14	14	89

5.4 Discussion

In this study, four experiments were conducted. They included the following: initial training, extended initial training, support vector (SV) training, and extended support vector (ESV) training. The sizes of their training sets and their parameters are summarized in Table 4.

Table 4. The training set sizes and the experiment parameters

Experiments	Training samples			SVC parameters	
	Original	Synthetic	Total	nu	sigma
Initial Training	60,000	0	60,000	0.011	0.5
Extended Training	60,000	25,021	85,021	0.011	0.5
SV Training	4,278	0	4,278	0.14	0.5
ESV Training	4,278	1,109	5,387	0.14	0.5

Table 5. The recognition rates and number of errors

Experiments	Rec. rate	Number of errors per class										Total errors
		0	1	2	3	4	5	6	7	8	9	
Initial Training	99.07	4	3	8	7	9	12	10	7	16	17	93
Extended Training	99.06	5	2	9	9	9	12	9	8	15	16	94
SV Training	99.12	7	5	7	8	11	9	10	8	12	11	88
ESV Training	99.19	7	6	6	7	8	8	6	11	11	11	81

For the experiment results, the rejection rates were not provided because the support vector classifier used in this study only returns ranking information, which is not enough to give good rejections. According to Platt [34], SVC requires additional trainings in order to obtain confidence values for the outputs of SVC. However, providing probabilities for the SVM adds additional complexity to the SVM algorithm. Therefore, only the recognition and error rates were provided. For the conducted experiments, the recognition rates and the number of errors per class are presented in Table 5.

As shown in Table 5, the extended training (with synthetic samples) did not improve the results of its initial training (without synthetic samples), but it also did not decrease much. This indicates that either 25,021 synthetic samples did not provide new information for the 60,000 training data or this amount of synthetic samples was not sufficient. It was time consuming to consider a large number of synthetic data in training with SVC because the training time of the SVC was proportional to the size of the training data. Therefore, increasing the size of the training data by adding a large number of synthetic data may result in this kind of time limitation for the SVC training.

The results of these experiments show that the ESV training outperforms the SV training. For the ESV training, the performance of morphing transformation could have been better if a larger number of support vectors would have been used instead of using the optimal set of support vectors.

The intermediate frames between the source and target were used as synthetic samples in both of the extended training and extended support vector training experiments in order to increase their training sets. Interesting results may have been obtained if the frames closer to the source and closer to the target were used.

In order to show the difference between our method and other similar methods, we provide a list of error rates for them in Table 6. The first three methods are SVM-based methods. In these methods, the data transformation was performed in the feature space. The next three NN-based methods used distortions. The distortions were performed in the input space. The last four experiments were conducted in this study. The results of the data transformation used in this study, morphing transformation, are presented in 8-th and 10-th rows of the table. Making use of morphing transformation we were able to decrease the error rate from 0.93% to 0.81%.

Table 6. Comparison with other methods

Data Space	Methods	Error Rate	Reference
Feature Space	1. Affine distortions with SVM	1.4%	[11]
	2. Virtual SVM	0.6%	[11]
	3. SVM Kernel jittering	3.3%	[37]
Input Space	4. NN without distortions	1.6%	[45]
	5. Affine distortions with NN	0.6%	[45]
	6. Elastic distortions with NN	0.4%	[45]
	7. Initial Training	0.93%	This thesis
	8. Extended Training (Morphing Transformation)	0.94%	This thesis
	9. SV Training	0.88%	This thesis
	10. ESV Training (Morphing Transformation)	0.81%	This thesis

CHAPTER 6
CONCLUSION

In this study, we investigated the effect of augmenting a training set by using morphing transformation to generate synthetic examples of handwritten numerals. Making use of this technique, we aimed to balance the data distribution. Normally, the training data has a poor distribution due to the data redundancy and sparseness caused by frequent and rare samples, respectively. In terms of data clusters, some clusters were small, some were large and filling the gap between these clusters with artificial data helped to smooth the clusters. Using the SVM, the rare samples (support vectors) were determined. Then, the number of rare samples was increased by adding the synthetic samples generated by morphing transformation.

The performance of this technique was tested in two ways. First, we increased the size of the training set by adding the generated synthetic samples using morphing. This method is similar to the previously reported methods such as elastic and affine distortions with NN classifier [46]. Second, as in virtual SVM [12], we performed transformations with the support vector samples only. The difference between the virtual SVM method and this experiment was the data representation space. In virtual SVM, the transformations were performed on support vector samples in the feature space where the data was mapped by the kernel function. In our experiment, the data transformations occurred in the input space.

Our results show that increasing the size of the training set by adding the synthetic samples generated by morphing transformation did not improve its initial performance and did not decrease it much. Therefore, adding the synthetic samples here did not make much difference. This technique was time consuming because the training speed of the SVC (used in this study) is proportional to the size of the training set. We think that this can be avoided when NN is used.

Generating synthetic versions of the support vector samples was useful for the ESV training. We generated synthetic samples between similar support vector samples. Then, they were added to the training set of support vector samples. Using this extended training set, we reached the recognition performance of 99.19% for the best case of ESV training. Moreover, the speed of the training was fast because of the smaller size of the training set.

Additionally, some feature extraction techniques were used for the first time in Handwriting Recognition. In an effort to provide a more informative description for the numeral characters, the double contour information and the pie-like zoning techniques were used. The double-contour information provided stable information for the sloppy numeral characters, whereas the pie-like zoning technique was very useful to extract background information from the numeral images.

Furthermore, one of the newest algorithms in SVM, so-called ν-SVM, was used in this study. It was helpful in controlling the number of support vectors and the number of training errors. This SVM method with a "one-against-one" multiclass strategy and a Gaussian RBF kernel function demonstrated a stable performance for HWR. For the testing part, we had a choice of using "Max Wins" and Directed Acyclic Graph Support

Vector Machine (DAGSVM) algorithms. "Max Wins" performed a little better than DAGSVM and provided ranking information. However, DAGSVM was much faster, and was used during the parameter estimation.

Future Work: Morphing transformation was used for the generation of synthetic samples within a single class. The source and target of the transition were taken from the same class. In the future, morphing transformation may also be useful in building strong verifiers by training the synthetic samples generated between the confusing pairs of classes because it can provide more information to the verifier on a confusing pair of classes by generating synthetic samples from the source of one class and the target of another one. For the future work, this technique could therefore be worthwhile. In addition to that, we advocate that using morphing transformation in combination with affine and elastic distortions may further increase the performance of the HWR. Morphing transformation generates synthetic samples that cannot be obtained by other methods. When it is used in combination with other methods, the synthetic samples in various shapes can be obtained. Consequently, this may improve the performance of the system trained on these samples.

BIBLIOGRAPHY

[1] Bahlamann C., Burkhardt H., "The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 3, 2004, pp. 299-310.

[2] Beier T., Neely S, "Feature-based image morphing," *Computer Graphics*, vol. 26, no. 2, 1992, pp. 35-42.

[3] Bishops C., "Neural Network for Pattern Recognition," Clarendon Press, Oxford, (1995).

[4] Burges C. J. C., "A tutorial on support vector machines for pattern recognition," *Data Mining Knowledge Discovery*, vol. 2, no. 2, 1998, pp. 121-167.

[5] Byun H., Lee S.-W., "A survey on pattern recognition applications of support vector machines," *International Journal of Pattern Recognition*, v. 17, no. 3, 2003, pp. 459-486.

[6] Chalimourda A., Scholkopf B., Smola A.J., "Experimentally optimal ν in support vector regression for different noise models and parameter settings," *Neural Networks*, vol. 17, 2004, pp. 127-141.

[7] Chang C.-C., Hsu C.-W., Lin C.-J., "The analysis of decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, 2000, pp. 1003-1008.

[8] Chang C.-C., Lin C.-J., "Training nu-support vector classifiers: theory and algorithms," *Neural Computation*, vol. 13, no. 9, 2001, pp. 2119-2147.

[9] Chen P.-H., Lin C.-J., Scholkopf B., "A tutorial on nu-support vector machines," 2003, [online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/nusvmtutorial.pdf.

[10] Cortes C., Vapnik V., "Support vector networks," *Machine learning*, v. 20, 1995, pp. 273-297.

[11] Decoste D., Scholkopf B., "Training invariant support vector machines," *Machine Learning Journal*, vol. 46, no. 1-3, 2002, pp. 161-190.

[12] Dong J. X., Suen C. Y., Krzyzak A., "A fast SVM training algorithm," *International Journal of Pattern recognition and Artificial Intelligence*, vol. 17, no. 3, 2003, pp. 367-384.

[13] Favata J. T. and Srikantan G., "A multiple feature/resolution approach to handprinted digit and character recognition," *International Journal of Imaging Systems and Technology*, v. 7, 1996, pp. 304–311.

[14] Friedman J. "Another approach to polychotomous classification," 1996, [online]. Available: http://www-stat.stanford.edu.reports/friedman/poly.ps.z.

[15] Friedman M., Kandel A., "Introduction to pattern recognition," World Scientific, Singapore, (1999).

[16] Gomes J., Darsa L., Costa B., Velho L., "Warping and morphing of graphical objects," Morgan Kaufmann Publishers, San Francisco, California, U.S.A., (1999).

[17] Gomes J., Velho L., "Image processing for computer graphics," Springer, New York, U.S.A., (1997).

[18] Gonzalez R.C., Woods R.E., "Digital image processing," Prentice Hall, Upper Saddle River, New Jersey, U.S.A., (2002).

[19] Grayson M. "The heat equation shinks embedded plane curves to round points," *Journal of Differential Geometry*, vol. 26, 1987, pp. 285-314.

[20] Hsu C.-W., Lin C.-J, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, 2002, pp. 415-425.

[21] Iga C., Wakahara T., "Character image reconstruction from a feature space using shape morphing and genetic algorithms," *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition*, Kokubunji, Tokyo, Japan, 2004, pp. 341-346.

[22] LeCun Y., Bottou L., Bengio Y., Haffner P., "Gradient-based learning applied to document recognition," *Proceedings of IEEE*, vol. 86, 1998, pp. 2278-2324.

[23] LeCun Y., "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist.

[24] Liu C.-L., Nakashima K., Sako H., Fujisawa H., "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern recognition*, vol. 36, 2003, pp. 2271-2285.

[25] Liu C.-L., Nakashima K., Sako H., Fujisawa H., "Handwritten digit recognition: investigation of normalization and feature extraction techniques," *Pattern Recognition*, vol. 37, 2004, pp. 265-279.

[26] Marques de Sa J. P., "Pattern Recognition," Springer, Berlin, (2001).

[27] Maruyama K.-I., Maruyama M., Miyao H., Nakano Y., "A method to make multiple hypotheses with high cumulative recognition rate using SVMs," *Pattern Recognition*, vol.37, 2004, pp. 241-251.

[28] Mitoma H., Uchida S., Sakoe H., "Online character recognition using eigen-deformations," *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition*, Kokubunji, Tokyo, Japan, 2004, pp. 3-8.

[29] Mori S., Suen C. Y., Yamamoto K., "Historical review of OCR research and development," *Proceedings of the IEEE*, vol. 80, no. 7, 1992, pp. 1029-1055.

[30] Muller K.-R., Mika S., Ratsch G., Tsuda K., Scholkopf B., "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, v. 12, n. 2, 2001, pp. 181-202.

[31] Oliveira L. S., Sabourin R., Bortolozzi F., Suen C. Y., "Automatic recognition of handwritten numeral strings: A recognition and verification strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, 2002, pp. 1438-1454.

[32] Pavlidis I., Singh R., Papanikolopoulos N. P., "On-line handwriting recognition using physics-based shape metamorphosis," *Pattern Recognition*, vol. 31, no. 11, 1998, pp. 1589-1600.

[33] Platt J., "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, B. Scholkopf, C. J. C. Burges, A. Smola Eds., The MIT Press, Cambridge, MA, (1999).

[34] Platt J., "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Barlett, B. Scholkopf, D. Schuurmans, Eds., Cambridge, MA: MIT Press, 2000, pp. 61-74.

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

[36] Scholkopf B., Burges C. J. C., Smola A. J., "Advances in Kernel Methods – Support Vector Learning," MIT Press, Cambridge, MA, (1999).

[37] Scholkopf B., Smola A. J., Williamson R. C., Barlett P. L., "New support vector algorithms," Neural Computation, 12, 2000, pp. 1207-1245.

[38] Scholkopf B., Smola A., "Learning with kernels," MIT Press, Cambridge, MA, (2002).

[39] Scholkopf B., Williamson R.C., Smola A., Shawe-Taylor J., "SV estimation of a distribution's support," Microsoft Technical report MSR-TR-99-87, 1999.

[40] Sederberg T.W., Greenwood E., "A physically based approach to 2D shape blending," *Computer Graphics*, vol. 26, no. 2, 1992, pp. 25-34.

[41] Sederberg T., Parry S., "Free-form deformation of solid geometric models," *Computer Graphics*, vol. 20, no. 4, 1986, pp. 151-160.

[42] Sethian J. A., "Level set methods: Evolving interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences," New York, Cambridge University Press, (1996).

[43] Seul M., O'Gorman L., Sammon M.J., "Practical algorithms for image analysis," Cambridge, Cambridge University Press, (2000).

[44] Shi M., Fujisawa Y., Wakabayashi T., Kimura F., "Hanwritten numeral recognition using gradient and curvature of gray scale image," *Pattern Recognition*, vol. 35, 2002, pp. 2051-2059.

[45] Simard P. Y., LeCun Y., Denker J. S., Victorri B., "Transformation invariance in pattern recognition – Tangent distance and tangent propagation," *International Journal of Imaging System and Technology*, vol. 11, Issue 3, 2001, pp. 181-194.

[46] Simard P. Y., Steinkraus D., Platt J. C., "Best practices for convolutional neural networks applied to visual document analysis," *Proceedings of 7th International Conference on Document Analysis and Recognition*, vol. 2, Edinburgh, Scotland, 2003, pp. 958-962.

[47] Singh R., Papanikolopoulos N., "Planar shape recognition by shape morphing," *Pattern Recognition*, vol. 33, 2000, pp. 1683-1699.

[48] Suen C. Y., Tan J., "Analysis of errors of handwritten digits made by a multitude of classifiers," *Pattern Recognition Letters*, v. 26, 2005, pp. 369-379.

[49] Suen C. Y., Shillman R. J., "Low error rate optical character recognition of unconstrained handprinted letters based on a model of human perception," *IEEE Transactions on Sytems, Man and Cybernetics*, 1977, pp. 491-495.

[50] Trier O. D., Jains A. K., Taxt T., "Feature extraction methods for character recognition – A survey," *Pattern Recognition*, v. 29, no. 4, 1996, pp. 641-662.

[51] Vapnik V., "The Nature of Statistical Learning Theory," Springer-Verlag, New York, (1995).

APPENDIX

The list of 81 misrecognized numeral samples of the MNIST testing set with the sequence number (ID), the true label (TL) and the estimated label (EL):

Misrecognized samples

#	Sample	ID: TL→EL	#	Sample	ID: TL→EL
1.		248: 4→6	21.		1902: 9→4
2.		446: 6→0	22.		2019: 1→7
3.		583: 8→2	23.		2110: 3→7
4.		584: 2→7	24.		2131: 4→9
5.		675: 5→3	25.		2136: 6→1
6.		685: 7→3	26.		2226: 8→9
7.		948: 8→9	27.		2294: 9→4
8.		1015: 6→5	28.		2319: 0→6
9.		1034: 8→1	29.		2327: 0→8
10.		1040: 7→3	30.		2388: 9→1
11.		1113: 4→6	31.		2415: 9→4
12.		1227: 7→2	32.		2455: 6→5
13.		1243: 4→9	33.		2463: 2→8
14.		1261: 7→1	34.		2598: 5→3
15.		1501: 7→3	35.		2608: 7→4
16.		1523: 7→1	36.		2655: 6→1
17.		1531: 8→7	37.		2679: 4→9
18.		1682: 3→7	38.		2824: 7→3
19.		1791: 2→7	39.		2928: 3→7
20.		1879: 8→3	40.		2940: 9→5

Misrecognized samples (continued)

#	Sample	ID: TL→EL
41.		2953: 3→5
42.		3024: 8→5
43.		3074: 1→2
44.		3504: 9→1
45.		3521: 6→4
46.		3559: 5→8
47.		3768: 7→2
48.		4079: 9→3
49.		4177: 2→7
50.		4239: 7→3
51.		4266: 4→3
52.		4307: 3→7
53.		4498: 8→7
54.		4501: 9→1
55.		4508: 1→2
56.		4762: 9→4
57.		4861: 4→9
58.		4880: 8→6
59.		5451: 0→5
60.		5458: 1→2
61.		5938: 5→3
62.		5956: 3→8
63.		5973: 5→3
64.		5974: 3→8
65.		5983: 5→3
66.		5998: 5→3
67.		6561: 9→5
68.		6572: 9→7
69.		6573: 1→3
70.		6577: 7→1
71.		6598: 0→7
72.		6626: 8→3
73.		6652: 0→8
74.		6756: 8→9
75.		6784: 1→6
76.		7217: 0→6
77.		8095: 2→8
78.		9635: 0→8
79.		9730: 5→6
80.		9793: 4→9
81.		9840: 2→7

GLOSSARY

AD	Affine Distortion
ED	Elastic Distortion
ESV	Extended Support Vector
DAGSVM	Directed Acyclic Graph Support Vector Machine
GSC	Gradient, Structural and Concavity
HWR	HandWriting Recognition
MNIST	A Modified benchmark dataset of handwritten numerals originally built by the National Institute of Standards and Technology, Gaithersburg, US
NN	Neural Networks
OCR	Optical Character Recognition
RBF	Radial Basis Function
SV	Support Vector
SVM	Support Vector Machine
SVC	Support Vector Classifier