Journal of Applied Intelligence,12(1-2):95-114 (2000)
© 2000 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Recognition of Handwritten ZIP Codes in a Real – World Non-Standard-Letter Sorting System

M. PFISTER
Siemens AG

S. BEHNKE
Freie Universität Berlin

R. ROJAS
Freie Universität Berlin

Received Juli 1, 1998; Revised October 11, 1998

Editors: S. Devadas and P. Michel

Abstract. In this article, we describe the OCR and image processing algorithms used to read destination addresses from non-standard letters (flats) by Siemens postal automation system currently in use by the *Deutsche Post AG*[1].

We first describe the sorting machine, its OCR hardware and the sequence of image processing and pattern recognition algorithms needed to solve the difficult task of reading mail addresses, especially handwritten ones. The article concentrates mainly on the two classifiers used to recognize handprinted digits. One of them is a complex *time delayed neural network (TDNN)* used to classify scaled digit-features. The other classifier extracts the structure of each digit and matches it to a number of prototypes. Different digits represented by the same graph are then discriminated by classifiying some of the features of the digit-graph with small neural networks.

We also describe some approaches for the segmentation of the digits in the ZIP code, so that the resulting parts can be processed and evaluated by the classifiers.

Keywords: Postal automation, address reading, neural networks, handprinted digit recognition.

1. Introduction

Optical character recognition (OCR) has reached such a high quality level that reading postal addresses (especially ZIP codes) quickly became one of the first industrial applications in this area. The automatic sorting of so-called 'standard letters' (with envelopes smaller than 11.5×23 cm in Germany) is a problem on which companies like AEG ElectroCom (now Siemens ElectroCom) have been working since 1970. The work done by J. Schürmann from the Daimler-Benz Research Center in Ulm, Germany, has been reported in a number of publications, the most important one beeing [23]. Also N. Srihari from the CEDAR Institute in Buffalo has concentrated on these kinds of problems (see e.g. the CEDAR homepage www.cedar.buffalo.edu or [10, 19]).

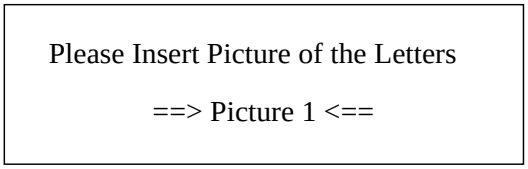

Fig. 1. Some examples of non-standard letters (flats).

However, so-called *non-standard letters* (in this paper shortly called *flats*), constitute a large fraction of the daily postal items to deliver and sort. In Germany flats are larger than standard letters but smaller than $35.3 \times 25.3 \times 2$ cm. Sorting flats is slightly more difficult than sorting standard letters, mainly because of two reasons. Firstly, the adress block is not located in a specific region (like in standard letters) and, secondly, the variety of handwriting that we find is certainly larger than for standard letters. Writers have much more room to be 'creative', so the shape and also the size of writing may differ markedly. Figure 1 shows some examples.

As the result of a worldwide competition in 1994, the German *Deutsche Post AG (DPAG)* awarded Siemens AG a contract to install a prototype of a flat sorting machine (in German: *Großbrief-Sortieranlage*, abreviated in the sequel as *GSA*[2]). This was an international premiere: it was the first time, a country considered automating the entire flat sorting process. At the time of this writing 150 GSAs have been delivered and installed in about 80 sorting centers all over Germany. They sort millions of flats daily with a GSA throughput up to 20,000 flats per hour. More than 85% of the adresses are found and read correctly by the machines with an error rate of less than 1%. Those letters rejected by the GSA-OCR system are sent to so called *Video Coding Places (VCPs)* to be classified by human experts using a numeric keyboard. Assuming that about 15% of the flats contain handwritten addresses, this means that the recognition rates are 78% for handwritten, and far over 90% for typewritten adresses.

Figure 2 shows a picture of a GSA postal sorting machine. The envelopes are fed into the GSA and are separated by four *feeders* (1). They are transported on a *conveyor belt* (2) running at about 2m/s. They go below the *linescanning camera* (3), which captures a greyscale image of each letter. While the letter is passed to the *sorter* (6), the GSA-OCR (4) starts processing the image to recognize the destination address. If no valid address is found, the image is delivered to video coding places VCPs (5), where postal workers handle it. If the ZIP code information can be automatically recognized or is entered by a human operator, the letter is dropped in one of 200 mailboxes (7), each one covering different ZIP code ranges.

In order to solve the difficult problem of automatically reading flat mail adresses with such high accuracy and at an average speed of 6 flats/second, several algorithms out of the image processing, pattern recognition and neural networks field had to be used. Also new algorithms were developed and special hardware was used. In this article we discuss several of the GSA algorithms, some of them roughly, and others, such as the handprinted ZIP code classifiers, more in detail. Together they constitute a real world application of 'applied intelligent algorithms'.

This article is organized as follows: In the rest of this section, we give an overview on the OCR-hardware used in the GSA and we then go

Please Insert Picture of the GSA

==> Picture 2 <==

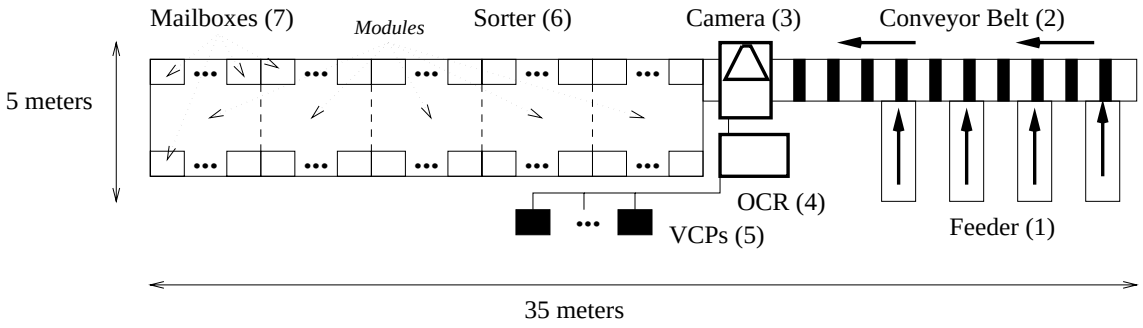

Fig. 2. Siemens flat sorting machine GSA installed in a DPAG sorting center.

through the sequence of subproblems that have to be solved to complete the address identification task. In section 2 we deal briefly with the the problem of finding the addresses on a flat (especially when the envelope contains much detail) and then we explain how this address is segmented into lines and words (section 3). Section 4 contains the main part of this article. We first discuss the different methods used to classify handprinted digits implemented in the GSA. Then we address the more difficult problem of segmenting connected handprinted ZIP codes in single (potential) digits, we introduce our approach and discuss how these algorithms interact with the digit classifiers. Finally, in section 5 we describe how the ZIP codes which we have read can be double-checked and verified.

Nota bene: Developing the GSA and its OCR was a Siemens project involving many researchers and developers from many Siemens divisions. The development of some methods was done in colaboration with the Freie Universität Berlin, namely the ones described in detail in this article.

1.1. Hardware of the GSA-OCR system

Together with the hardware components used to control the GSA (the Siemens S5 Program-

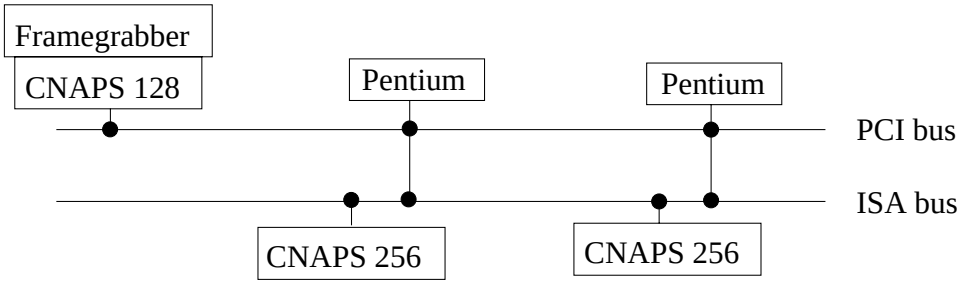

Fig. 3. Basic structure of the GSA-OCR computer.

able Controler) or the computers used for manual coding of the rejected letters, the 'heart' of the GSA is the GSA-OCR system running on the so-called *GSA-OCR computer*. This GSA-OCR computer is basically a conventional dual Pentium computer with some additional hardware, mainly three multiprocessor boards (Adaptive Solutions *CNAPS* [1, 20]) and a special framegrabber board. All the algorithms described below are run by this system. Figure 3 shows the basic structure of the GSA-OCR computer and figure 4 that of the CNAPS.

The CNAPS boards consists basically of 256 (or 128) very simple processor nodes (PNs) running in SIMD mode. The PNs have a shared global memory (where the executable program is stored) and also small 4K local memories [1, 20]. Due to the SIMD structure and their simple PNs, the CNAPS boards are used to process the time-consuming image-processing or classification tasks.

1.2. Solving the GSA-OCR subproblems

The problem of automatic postal sorting non-standard-letters, as done by the GSA, is very complex. It principally consists of the two main subproblems *finding* the address on the letter and then *reading* it. Both of these tasks include of course many other subtasks.

Since we can not be sure that a region of the letter surface that looks like an address region actually contains the address. These "potential address regions" will further be called *Areas Of Interest, AOIs*. The following list shows the sequence of subtasks that have to be solved to analize AOIs to finally get the destination ZIP code from the letter.

1. AOI determination,
2. AOI binarization,
3. line segmentation,
4. word segmentation,
5. character segmentation,
6. character classification,
7. interpretation of the classification results,

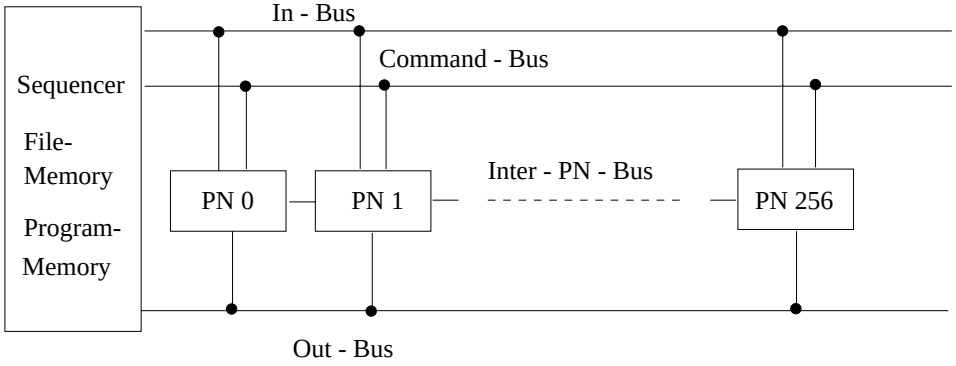

Fig. 4. Basic structure of the Adaptive Solutions CNAPS SIMD computer.

8. eventual alternative handwritten ZIP code processing,
9. address verification.

Note that the task sequence is not necessarily *strict*. If it is noticed, in any step, that the AOI examined by the GSA-OCR contains no valid address, another can be checked. Also if, for example, binarisation fails, it may be repeated with a slightly modified algorithm and so on.

This article concentrates on the recognition of handwritten ZIP codes. Anyway these can not be processed or verified without solving the other tasks of the AOI analysis as listet above. So it is interesting, and also important, to understand how these work and interact. Therefore they will be outlined in the next two chapters, although they are rather technical. Unfortunately the methods used to solve the basic AOI analysis can only be outlined very roughly. They all, but especially the line segmentation algorithm, are protected by patents and very strict copyright restrictions, so they can not be published by the authors.

All algorithms discussed below work using the greyscale image of the letter, which is scanned by a line camera at a resolution of about 150 dpi and which is encoded using 255 grey-levels (1 byte/pixel) .

2. Finding the address candidates on the letter

This is the first hard problem to be solved. If you have e.g. one of your monthly computer journals at hand, or look t the picture of the catalouge in figure 1, you can see that finding the address, contained probably in a label placed somewhere on the front page, is not so easy to do at first sight. Contrary to standard letters, where the address is almost always 'in the lower right corner', it can be anywhere on a flat, even hidden in some other text-blocks, like for example headlines. The GSA-OCR finds these addresses using neural networks and geometrical (statistical) information about flat envelopes, as briefly described in the following.

During the scanning process, that is in real time, the pixels of the incoming image are grouped in square regions called *superpixels*. These superpixels (also some extracted spatial frequency features) are now classified using small neural networks. It is decided, if the superpixel belongs to a region containing, for example, background, unstructured noise, typewriting, handwriting or corners of a label. Some of these classes, like 'typewriting' or 'handwriting' describe foreground features, others like 'background' or 'noise' describe backround ones.

In the second step, when all these features have been computed, superpixels of the same foreground type are clustered together by grouping them into 'address shaped rectangles'. These rectangles, the potential AOIs, are then ranked by taking into account some geometrical information. So for example a handwritten address will be more likely located in the lower right corner but not on the upper left and so on. One (or more) AOIs are selected to be processed further according to this ranking. With the method outlined above (parts of it, using different features, are described in more detail in [29, 30]), more than 95% of the right addresses are the highest ranking AOI.

3. Locating the ZIP code in the address

Having located the address (rsp. an AOI) on the letter, we have to make our way to the ZIP code in order to finally classify the main clue for the destination of the letter. This is basically done by the GSA-OCR as it is done by humans: the address is divided first into lines and then into words.

The first two steps are rather technical and are handled by basic image processing algorithms. First the background of the AOI is removed and the foreground is set to 'black' (i.e. the image is *binarized*); then connected foreground pixels are grouped in so-called *connected components*. On the basis of this information, the writing style (handwritten or machine printed) is determined and further segmentation of the AOI into lines and words is performed.

3.1. Binarisation of the address block

The first step consists of converting the greyscale image into a binarized one, i.e. an image containing only 'black' pixels (the foreground) and 'white'

pixels (the background). The motivation is that since the foreground (i.e. the writing) usually contains all the information we need to read the letter, we can have a much more compact description of the picture.

Two assumptions are made to binarize the image. The first one is that the interesting foreground is darker than the unimportant background, i.e. the image pixels of the foreground have higher values than those of the background. The second asumption is that the picture (especially the background) has been more or less homogeneously illuminated, i.e. the variance of the background pixels values is low. Under these assumptions, binarisation of the image reduces to choosing a threshold θ and setting all pixels $p(x, y)$ of the image to

$$p(x,y) := \begin{cases} 1 & if \quad p(x,y) > \theta \\ 0 & else. \end{cases}$$

There are many ways to estimate such a threshold θ and some popular methods are described in [19]. We can also avoid choosing a *global* threshold θ as suggested above, selecting instead a *local* $\theta(x, y)$ for each pixel $p(x, y)$, depending on the values of the neighboring pixels. Local methods have the advantage that they can deal with inhomogeneously illuminated surfaces, but they are computationally expensive and are often sensitive to some parameters (e.g. the estimated strokewidth of the characters in the image) [19]. Since for the application described the disadvantages overcome the advantages, a global method was choosen. The threshold estimation method implemented in the GSA is based on the *Otsu method* described in detail in [18, 17]. The pixels of the image are separated into two classes (foreground and background) using *linear discriminant analysis methods*.

After having binarized the image a very compact *runlength representation* is used. The idea is not to represent each pixel of the image by its value, but to group horizontal lines of black pixels (the *runs*) and represent these runs by their starting- and ending position. This is a very common way to store binarized images. Choosing this representation, it is very easy to find connecting parts of foreground regions, the *connected components*. Their connectedness give clues about where on the AOI there could be text, characters, or maybe gaps between them. Figure 5 shows one handwritten and one machine printed AOI after binarisation. The bounding boxes of its connected components are also shown.

3.2. Segmenting the ZIP code into lines of text

This is surely the most difficult problem in the basic AOI analysis. The problem is that it cannot be assumed that lines are strictly horizontal. The task becomes even more challanging in the case of handwriting, were the lines may not be clearly separated by white spaces. The algorithm used to solve the line separation task is very robust, even for AOIs rotated up to ± 10 degrees. It is an iterative approach, where in the first steps certain connected components are regarded as initial lines. To these, the rest of the components are grouped until all of them are distributed to the lines in the AOI. Figure 5 shows the final line segmentation result of two examples.

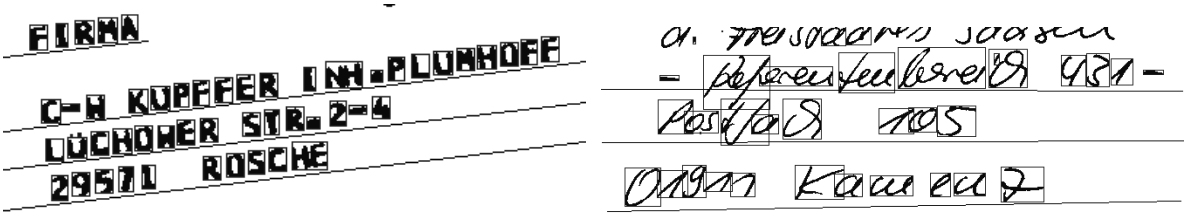

Fig. 5. Final result of the line segmentation for two binarised AOIs.

3.3. Determination of the AOIs writing style

The determination of the AOIs writing style (handwritten or machine printed) is again done on the basis of connected components information and the information additionally available about lines in the AOI. The discrimination is done to be able to choose e.g. different types of character classifiers, since a 'general digit classifier' would be much more complex and less reliable than a special one for different writing styles.

The algorithm itself is very fast and highly reliable. The basic idea is, that obviously machine printing is more 'regular' than handwriting. This regularity is measured in different features, possible ones are described in [11]. They use connected component features like histograms showing the heigth or width distribution of the components. These features are then evaluated by a small classifier to get a descision. For the AOIs shown in figure 5 we get e.g. for the height histogram:

	Num. of Heights	Average
Machine AOI	5	3.7
Hand AOI	18	8.0

so the differences of this feature and the rsp. descision are obvious.

3.4. Segmentation of the address lines into words

The word segmentation step is again done to be able to choose special classifiers or methods for each type of word. So principally special digit classifiers or methods can be used to read ZIP the code and special character classifiers to read the city name. The latter problem is of course more difficult for handwriting. Here completely different methods may have to be used, as described below.

The decision is in this case rather simple, since it is not attempted to handle the *general* word segmentation problem. The main interest is to find a ZIP code in the AOI (rsp. in a line), and since a line in a valid German address in general contains only two words, the task can be reformulated as: *Try to separate the line into two words, so that the left one may be a ZIP code.* Figure 6 shows how the lower line of the AOIs in figure 5 were separated into ZIP code and cityname candidates.

The idea is that words are separated by gaps which are larger than the gaps separating characters within a word. Also the word left of the potential word segmenting gap must have certain features (e.g. a certain width/height ratio) to be a ZIP code at all. So the algorithm is simply based on the computation of some features for each gap between two connected components (starting with the largest one), like size of the gap, height/width ratio of the resulting left word, the (potential) ZIP code or height/width ratio of the resulting right word, the (potential) city name.

4. Interpretation of handwritten ZIP codes

This is the main section of this article. We now assume to be faced with the isolated ZIP code block, i.e. in Germany a text block containing five digits (see figure 14). We now want to segment this block into its single digits and classify them. Since the task is rather trivial for machine printing (as classification and segmentation of this kind of writing is) we concentrate exclusively on the handwriting problem in this section.

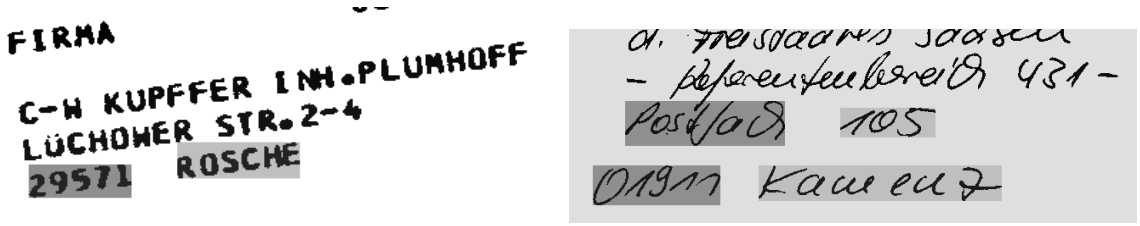

Fig. 6. Final result of the word segmentation (lower line) for two AOIs.

We first describe the approaches we have developed to classify isolated handprinted digits, and then describe two approaches for the much more complex task of handprinted ZIP code segmentation.

4.1. Classifying isolated handprinted digits

The heart of any OCR processing system is a high performance character classification system, since this is the place where the unstructured pixel patterns get their 'meaning', i.e. they are identified as a '6' or an '*A*'. J. Schürmann calls this the "borderline between the subsymbolic and symbolic world. The task of pattern classification is throwing the bridge beween both worlds – generating symbols from subsymbolic observations" [23].

The performance of a classification system can be evaluated using criteria such as the size of the alphabet which it recognizes, its reliability or writer-independency. Since the methods described up to here were designed to read handprinted ZIP codes, the problem reduces basically to the recognition of the digits 0 to 9. On the other hand, very high reliability and writer-independency is required for this application. The system has to deal with widely different sizes and slants, with different shapes and width of the strokes.

In the past many approaches have been suggested to solve the problem of classifying single, isolated digits. Some use as input for powerful neural, polynomial or statistical classifiers the (in some sense) normalized pixel image of the digit or some more abstract features. Other classifierss preprocess the image in order to obtain the structure of the digit and base their decision on these features [13, 14].

No single approach is able to solve the problem perfectly, all methods have their particular stengths and weaknesses. Pixel oriented methods, for example, are able to tolerate structural defects much better than structural methods, as long as the main shape of the digit is retained. Structural methods, on the other hand, perform better on deformed digits having a typical structure.

To produce a system as writer independent and reliable as possible, we decided to combine two methods: a fast but reliable method which analyzes the structure of the digit, and in a second step, a powerful but more computationally intensive pixel oriented neural classifier. Both methods are described in the following sections together with remarks on the combination of the two classifiers.

4.1.1. The TDNN classifier

The TDNN (*Time Delayed Neural Network*) classifier is a high performance neural classifier, which

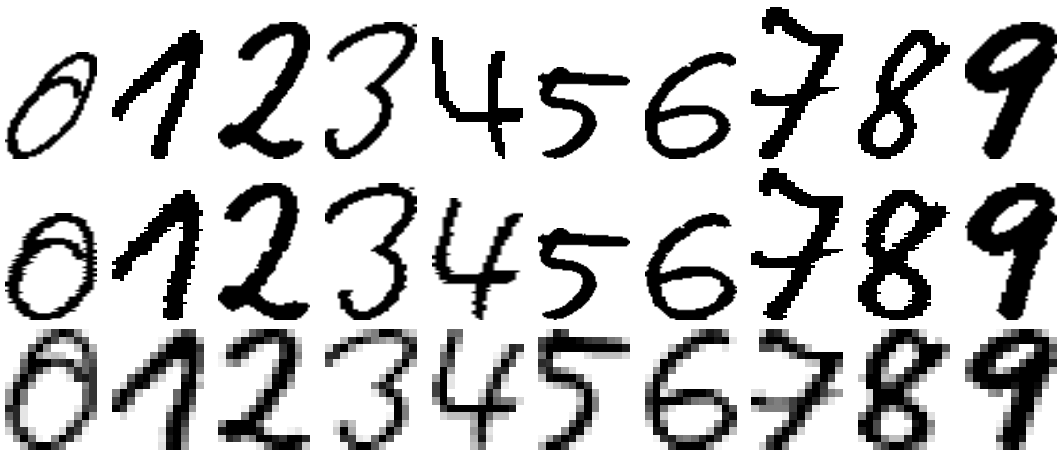

Fig. 7. Normalization of some digits slant and size.

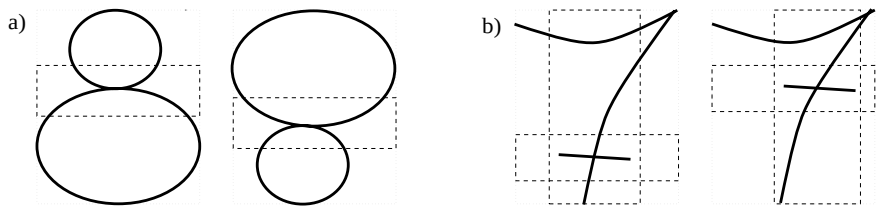

Fig. 8. Locations of characteristica of the same digit depend on the writer.

takes the image of a binarized digit, scaled to a fixed size, as its input. In a preprocessing step, some of the digit's variance is removed. The most important ones are the slant and size, as shown in figure 7. For better visualization, the digits were all scaled to the same width.

The principal axis of the digit is estimated and then the digit is sheared, so that his axis is vertical after the transformation. Then the resulting digit is scaled to a fixed size of 16 pixels height and 12 pixels width. After this process, the digit is not binarized anymore, so-called *pseudo-grey values* occur.

Beside these variance, to be removed by preprocessors, the system has to detect those characteristic features of the digit, which also help us humans to discriminate and 'classify' it. These features may be in different locations of the pixel-image, due to nonlinear deformations, as suggested in figure 8.

These characteristic features may be shifted in horizontal (figure 8 a), vertical, or both directions (figure 8 b), depending on the writer, the digit and the preprocessing of the digit.

TDNNs have shown to be successful in the general problem of detecting distinct patterns independently of their localisation in longer signals, although they were originally introduced to solve a specific speech recognition problem, namely the recognition of distinct vowels in long speech signals [27]. Good results have been also obtained in biochemical problems [22]. Because of these attractive and typical features of TDNNs, produced by their weight sharing receptive fields and the inclusion of unit's activations of previous time-steps into the current computations, we decided to use a TDNN to scan the normalized pixel image of the digit in order to perform a shift-invariant recognition.

The general architecture of a TDNNs is shown in figure 9 (for more details see e.g. [22]). Each group of input nodes (called the *receptive fields*

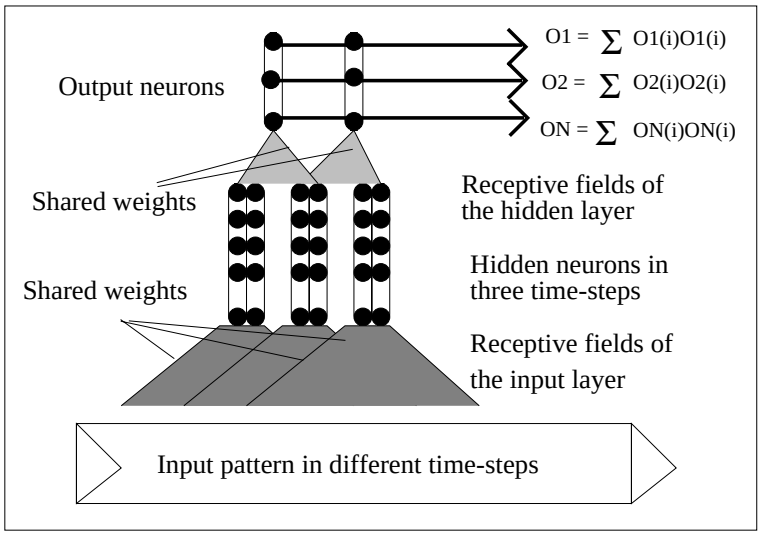

Fig. 9. Receptive fields with shared weights in different layers of the TDNN.

with shared weights) 'sees' only a small window of the input stream, which 'marches' through the windows one position in each time step. The output of the hidden layer is also covered with receptive windows using shared weights. The network's output consists of the sum of squares of the different time steps of the output neurons. This has the advantage, that small individual outputs tend to become less important [22].

The *input* of the OCR-TDNN consists of the binarized image of an isolated digit, scaled to a fixed height P_H=16 pixels and a fixed width P_W=12 pixels. There are R_1 receptive fields, each one 'sees' R_0 columns of the picture. These input windows are sharing theire weights and are shifted by a one column of the pixel-image. During the recognition process the columns are moved from left to right through the receptive fields, i.e. a horizontal scanning of the digit is performed. Best results have been obtained with field-sizes $R_0 \approx P_W$, e.g. $R_0 = 11$ or $R_0 = 13$. The number of time-states should be limited to $R_1 \leq 5$ because of the computational complexity. Anyway best results were obtained with $R_0 + R_1 > P_W$, so the total input window should be larger than the digits width. During the scanning, the image is never moved out of the total input window, and the digit is (virtually) enlarged with white columns left and right to have a well defined input for each node of all receptive fields.

The *hidden layer* consists of N_H hidden nodes in R_1 time-states. This is realized by connecting each group of N_H hidden nodes with the corresponding input window.

The *output layer* of the network consists of 10 nodes, each one representing one class of the digits '0' to '9', which are fully connected to *all* hidden nodes. The output layer thus works without receptive fields. This modification of the standard TDNN is motivated by two ideas. First of all, it accelerates and simplifies the learning algorithm of the TDNN. Secondly, working in this manner, the output layer gets a 'full view' over all time states of all hidden nodes, which also significantly improves the networks performance. During the discrete steps of the scanning process, the output of the output neurons is monitored. The most confident output is regarded as the final recognition, where results obtained from a more centered position of the digit get a little higher scores.

Also experiments with vertical scannings were performed, but with less impressing results. The recognition rates improve a little, if the digit is scanned in both directions, but not enough to justify the increasing computational complexity in the very time critical letter sorting process [3].

The OCR-TDNN is trained using a simple online gradient descent algorithm, similar to backpropagation [20, 22]. The only difference is, that during the weight-update step the corrections for the different receptive fields have to be averaged to guarantee the identity of the weights of the different receptive fields. The TDNN is fully self-trained using a backpropagation-like algorithm, so the feature-extractors and the classification layer are automatically adapted to each other and to the problem. Thus no 'expert knowledge' about the classification problem is needed and no extensive feature detecting preprocessing has to be performed.

The performance of the TDNN classifier was evaluated on the NIST special database [12, 3]. The TDNN was trained using about 120.000 digits and the tested on an independent validation set. On this set, the TDNN reached maximum recognition rates of up to 99.1% when substituting the other 0.9% and rejecting none. The substitution rate could be lowered to 0.1% when about 4.8% of the digits were rejected.

4.1.2. Classifying digits using structural information

The second classifier implemented in the GSA-OCR uses structural information for the recognition of isolated hand-printed digits. This hybrid classifier is described in more detail in [2]. Structural information and quantitative features are extracted in a multi-stage process from the digits pixel image. The goal is to preserve the information essential for recognition and to discard unnecessary details. Figure 10 shows the stages of the recognition process.

The preprocessing consists of a vectorization of the digit. This produces a line-drawing which is analyzed to construct a structural graph representation. The two-stage decision process matches first the structure of the digit to a structural

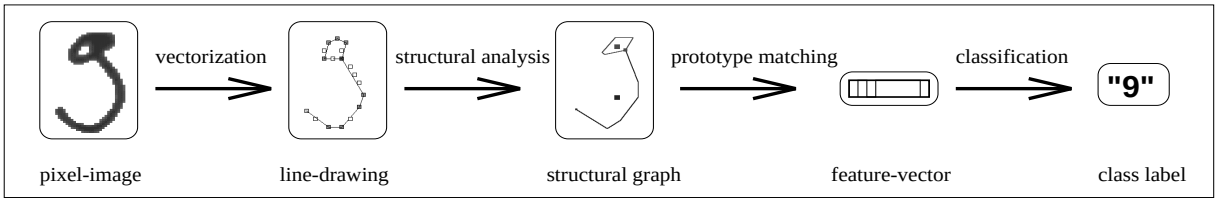

Fig. 10. Stages of the Structural Digit Recognition.

prototype whose associated neural classifier has been trained to distinguish digits that have the same structure based on extracted quantitative features.

Vectorization: The grey-level pixel-images of isolated digits constitute the input to the vectorization routine. Vectorization is done in two steps as illustrated in figure 11: i) the pixel image is preprocessed and a skeletonization operator is applied, ii) nodes are positioned and connected.

First a binarization threshold is used to remove the background, preserving the grey-levels in the foreground. The size of the image is scaled by factors of two to fix them to a certain range (*not* size). The resulting image should have width and height between 30 and 70 Pixels. Then the line-width (pen-thickness) of the digit is roughly estimated. Since too wide lines worsen the skeletonization process, the image is again scaled down, if the line-width is too large. Finally a low-pass filter is applied to the image to ensure that pixels are the darker, the more central they are in the lines and that each line cross-section has one maximum only.

Skeletonization is used to reduce the line width to approximately one pixel. Unlike morphological methods which iteratively erode the lines, the operator used here directly finds a skeleton in the middle of a line. The idea is to regard the stroke as a 'mountain chain' and the respective skeleton as the 'ridge'. The operator observes 3×3 pixel regions to decide if the central pixel belongs to the skeleton, by deciding if it has enough lighter ('lower') neighbors, i.e. if it belongs to the 'ridge'. Pixel having grey-level zero (white) do not belong to the skeleton, but to the background. For all other pixels, the number $c(x, y)$ of neighboring pixels (8-neighborhood) having an equal or higher grey-level is computed and if this number is less than three the central pixel is added to the skeleton. The resulting skeletons can be seen in figure 11(b).

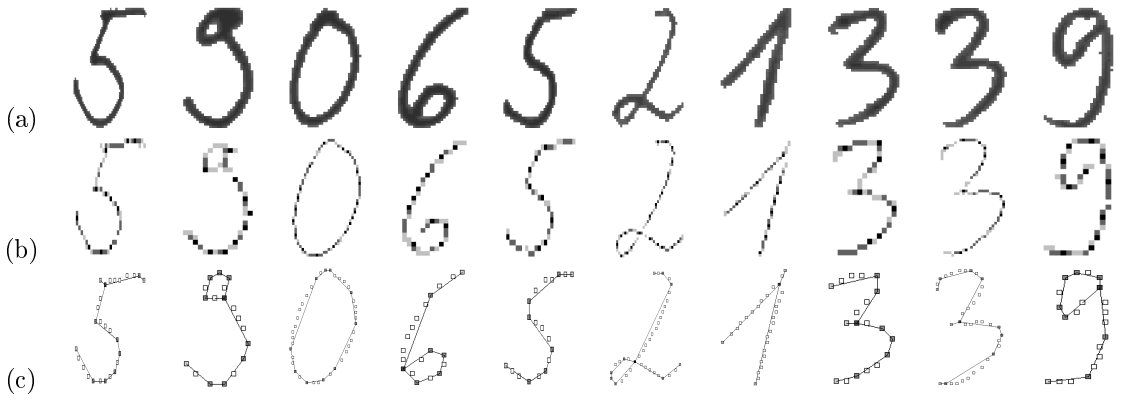

Fig. 11. Some digits: (a) background removed, (b) skeleton, (c) line-drawings.

Now a vectorial representation is constructed. Nodes are placed starting at peaks of the skeleton ($c(x,y) = 0$). Then nodes are placed at pixels belonging to ridges ($c(x,y) = 1$ and $c(x,y) = 2$), but with a minimum distance of two between them. The nodes need to be connected to represent strokes. First, the connection structure of the skeleton is reconstructed by inserting connections where 3×3 regions of nodes overlap or touch on the skeleton.

In order to insert the few remaining connections necessary to recover the original strokes, more global information is needed. Connections are inserted according to the principles of Gestalt psychology. The goal is to get lines exhibiting good continuity, closure and simplicity. To achieve this, candidates for new connections are determined and are evaluated using a measure based on the distance of the nodes to be connected, angles between connections, the grey-level of the pixels between the nodes, and topological information of the connection graph. New connections are inserted, ordered according to this measure, until a topology dependent threshold is reached. After all connections have been inserted, the line drawing is simplified. The lines are smoothed and nodes are taken out at locations of low curvature. Short lines ending in junctions are eliminated and connected junctions that are close together are merged to form a crossing. The resulting line-drawings are shown in figure 11(c).

Structural analysis: The next step derives a more abstract digit representation consisting of strokes which are merged to form larger curves. In order to reduce the variability of the input the vertical principal axis and the size of the digits are normalized.

A *stroke* is formed by several lines connected by joints (nodes of degree two) which have a common rotation direction and do not form sharp angles. A stroke has an initial and an end node, such that from the perspective of the initial node, the lines rotate to the right only. Straight strokes run from down to top. Starting from the nodes having a degree other than two, a topological structure is built by following the connecting lines. The length of the segments and the rotation angle are accumulated for each stroke. The strokes found touch each other only at the initial or end nodes. The contact points may represent junctions, crossings or changes of rotation direction.

A set of strokes can be merged to curves to reconstruct the way the digit was drawn. Two strokes are connected and reduced to a *curve* only if the rotation direction is preserved and the second constitutes a good continuation of the first. In this step we try to find long curves and the formation of loops is forced. This is done by testing for each common node of two strokes if the two strokes can be merged into a single curve. If this is the case, the candidate is evaluated using the local rotation angle and the total length of the curve. The mergers are performed starting with the best candidates.

Sometimes, digitalization defects, noise or small loops and embellishments of the handwritten digits produce short curves which must be eliminated before proceeding to recognize the digit. Using some topological information and the length of the curves it is decided whether it is beneficial to eliminate them from the graph or not. This step simplifies the structural description. Figure 12 shows some simplified curve representations.

The set of curves found in the previous steps is described now using a bipartite graph as can be seen in figure 13(a). Each curve is represented by a node in the left layer of the graph. Nodes in the right layer represent characteristic points such as curve ends, junctions, crossings and turning points. The graphs edges are derived from the

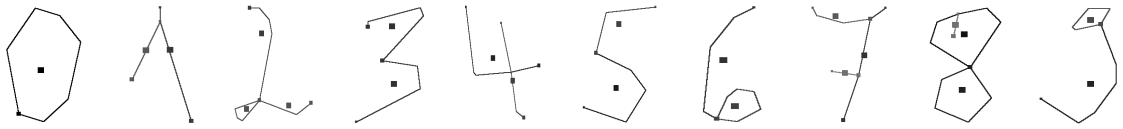

Fig. 12. Curve representation of some digits. Large squares are located at the center of gravity of the curves. Curves run from the middle-sized squares to the small squares.

curve representation. Each curve is connected to its characteristic points in the same order in which they appear when following the curve. Each node contains attributes, which summarize quantitative information about the curves and points. The curve nodes store the xy-coordinates of the center of gravity of the curve, the accumulated rotation angle, the length, and the distance of end and initial point relative to the total length of the curve. The shape of the curve is summarized by the xy-coordinates of six points distributed uniformly on the curve. The point nodes are described by their xy-coordinates.

Prototype matching: The attributed structural graph describes the essential features of the digit to be recognized. Recognition is done in two steps: i) the structural graph is matched to prototypes that have been extracted from the training set, ii) for each prototype there is a neural classifier which is used to distinguish digits having the same structure based on the extracted quantitative features.

Two structural graphs are called *isomorph*, if there exists a bijective mapping from the nodes and edges of one graph to the ones of the other graph. Curve nodes can only be assigned to curve nodes and the order of the edges must be preserved. Curves that are almost straight can also be mapped in inverted direction. The training set is partitioned into maximal sets that have isomorphic structural graphs. Each partition corresponds to a structural prototype for the matching step, if the partition contains a significant number of examples. These extracted typical structures represent not only perfect digits, but frequent structural deviations as well.

To test for isomorphism first a necessary condition is checked quickly. The number of curve nodes and the number of point nodes of each degree must match. If this holds, the curves of the first graph are permuted and inverted, and the resulting descriptions are matched with the description of the second graph. Sometimes more than one match between the graphs is possible. In this case all matches are used when partitioning the training set, but only the first match is used for recall. If the structural graph does not match any prototype, it is simplified by taking out the shortest curve and is matched again. If there is still no match, the digit is rejected.

Classification: In some cases prototype matching constitutes already a classification decision. There are prototypes that correspond almost only to examples from a single class, e.g. perfect zeros or eights. Other prototypes represent digits from more than one class, e.g. sixes and nines, fives and nines, and fours and sevens (see figure 13(b)). The extracted quantitative features are used to discriminate the digits that have the same structure, but belong to different classes. Depending on the complexity of the structure the feature vector that is presented to the classifier has a length ranging from 19 to 128. For each structure a specialized neural classifier is trained.

We use Cascade-Correlation [6] networks, since they are able to adapt their architecture to the difficulty of the problem. The sizes of the input and output layers are determined by the length of the feature vector and the number of classes. Training starts with no hidden units. As training proceeds a cascade of hidden units is created.

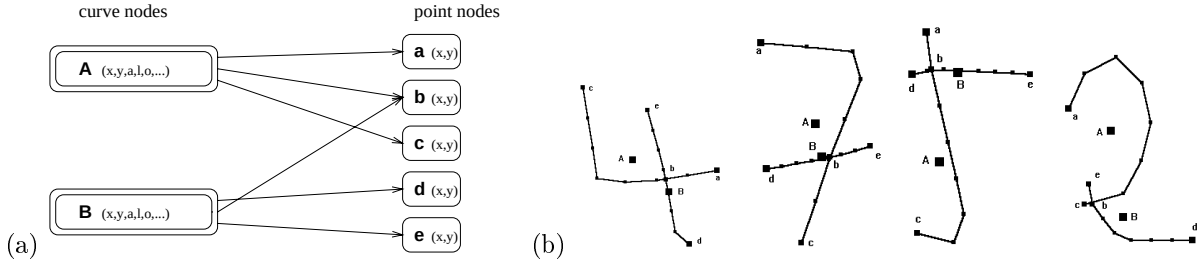

Fig. 13. Attributed structural graph (a) and some assigned digits (b). The first two are typical, the others aren't.

Training stops when the performance on a test set does not improve any more. A number of trials is performed and a reject criterion is varied to find a good network.

Experimental Results: To validate the performance of the described structural digit classification system, again the well known NIST special databases 1 and 3 have been used [12, 2, 3]. Unfortunately, the digits of this database have been binarized, which makes intensive low-pass filtering necessary to prepare the images for the skeletonization operator.

About 500 structures have been extracted from the training set, but only about 300 were frequent enough to be used as prototypes. The recognition results show that there is a tradeoff between reliability and recognition rate. A useful choice of the reject criterion could be such that rejection and substitution rates are equal. In this case the structural classifier has recognition rates of about 97.5% on the test set and about 96.8% on the validation set.

These recognition rates by itself would not justify the employment of the structural digit recognition, but the combination with the TDNN makes the hybrid system more reliable. Its distinctive features are its speed, its ability to recognize deformed digits and its high reliability for higher reject rates. The throughput of the entire classification is about 500 characters/second on a Pentium-II/266 system. It is able to classify deformed digits as long as the typical structure is retained. Most substitutions occur due to structural defects of the digits and can be avoided when allowing the classifier to reject ambiguous digits. For the NIST data set a substitution rate of only 0.19% is observed when rejecting 11.55% of the digits.

4.1.3. Combining classifiers

Now that we have two 'digit classifying experts', we have the problem of combining their (eventually) conflicting decisions. There are several ways to deal with such classifier combinations [3, 23]. The two main alternatives are *parallel* and *sequential* combination.

For the parallel combination, both classifiers are run and their results are merged, usually by some kind of voting mechanism or a small third classifier [3, 23]. This kind of combination usually yields very low error rates, but has the disadvantage that *both* classifiers have to be run, which is a very time consuming process and not neccessary for 'easy to recognize' digits.

For the sequential combination, the simplest and thus lesser time consuming classifier C_0 is run first. If it recognizes the digit with high confidence, we are done without having to ask the second classifier C_1. If it C_0 does not recognize the digit, the more powerfull classifier C_1 is run and the results of both are merged. The disadvantage of this method is of course that any missclassification done by C_0 cannot be overruled by C_1, therefore we must ensure that C_0 yields very low error rates.

In our case, we decided to run the structural classifier first. It is very fast and it also yields very low error rates. It also has the advantage that, due to the structural analysis of its input pattern, it is also able to tell if a pattern is 'far away from being a digit' (e.g. segmentation alternatives, see below). In these cases we can also avoid to run the TDNN.

Combining the two classifiers in such a way, we were able to obtain recognition rates of about 97.5% with less than 0.1% substitution rate on the NIST handprinted digits dataset, or a recognition rate of 99.5% with 0.5% substitutions [12, 3]. This is a significant improvement over the recognition rates yield by the two classifiers alone.

In practice, where also 'non-digit patterns' are present, resulting from segmentation trials or wrong AOIs, the TDNN is run and this happens only in one fifth of all classifier calls. This ensures significant speed with high reliability.

4.2. Separating connected digits in a ZIP block

The main problem of interpreting handwritten ZIP codes is, of course, that in practice the image extracted from a scanned letter will not appear as five separated digits. This is due to the limited resolution of 150 dpi or is caused to binarisation problems, or because the writer did not separate the digits at all, as shown in figure 14.

In the subtasks solved so far, e.g. line- and word segmentation, segmentation decisions of different kinds had to be made. But in these cases, there

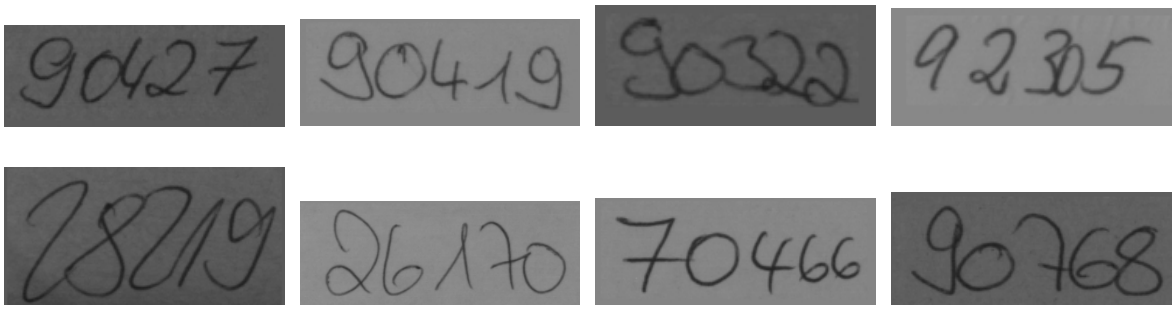

Fig. 14. Some examples of ZIP codes extracted from flat mail pieces.

were only few doubts, e.g. very often there is only one possible segmentation gap between a ZIP code block and the cityname. Only in very few ambigous cases (assuming the AOI handled actually contains an address) we have to revise our first alternative and check a second one.

The character segmentation task is much more complex. If we observe ourselves when reading a highly connected digit- or character string we will notice that our 'segmentation decision' is always doublechecked. We think of some strokes as belonging to a certain character and if this strokes then really form a valid character, and so do the strokes right and left of the recognized one, we will accept our decision. Otherwise, if something seems odd, we revise our decision and 're-group' the stroke. This is nothing but a permanent iteration between a classifying and segmentation state. Nothing else is done by the GSA-OCR system. The only difference is that the generation and evaluation of the segmentation alternatives is not neccessarily done in iterations, as we will see below.

As for the classifiers, we have again two tools for ZIP code segmentation. The first one is analyzing the binarized ZIP code block taking into account the connected component- and runlength information. Possible segmentation alternatives are found and organized in a segmentation tree. The resulting alternatives are then evaluated by the classifying system and the best path is taken as the result.

The second method converts the ZIP code block into a set of vectors as described in section 4.1.2. Then the vectors are grouped into clusters forming the potential digits, where ambigous strokes may belong to more than one cluster. The alternatives are again evaluated by the classifier and the best combination is taken as the result. Both approaches will be described in the following sections. Other approaches to solve this problem can e.g. be found in [5, 25].

4.2.1. Building and evaluating a segmentation tree

The first problem in building the segmentation tree is the estimation of the possible segmentation points. These are more or less found by analyzing the outlines of the different connected components [5, 7, 8, 9].

The first assumption made, is that a single digit does not consist of more than one connected component. Exceptions are e.g. '5' or '4', which sometimes consist of two unconnected strokes. These are then grouped using special heuristics, which are also able to handle special binarisation problems.

The determination of the ZIP code blocks possible segmentation points is now done by analysing the components shape as shown in figure 15 and explained in [5, 7, 8, 9].

The different components are regarded as a priori separated, so we are only interested in further splitting each of the two components. Starting at the top (the bottom) we follow the contour and search for paths which go as much down (up) through the digit block as possible. This means, we are looking for local maxima and minima in the component contour. In our example these are the

Fig. 15. The different possible segmentation paths for an example ZIP code block.

top-minima A for the first, and B, C and D for the second component rsp. the bottom-maxima A for the first, and B and C for the second component.

Now these different maxima and minima are combined to potential segmentation points by merging those top- and bottom-paths which 'almost meet each other'. This is called "Hit-and-Deflection Strategy [5]". The constraint is, that not all combinations are regarded as valid segmentation paths, they have to have certain properties. So more vertical paths are preferred, or paths cutting through as less as possible black pixels and so on.

In our example we have (from top-minima to bottom-maxima) the following possible 'legal' cuts:

1. Component:
   (a) $A \leftrightarrow A$.
2. Component:
   (a) $B \leftrightarrow B$ and
   (b) $D \leftrightarrow C$.

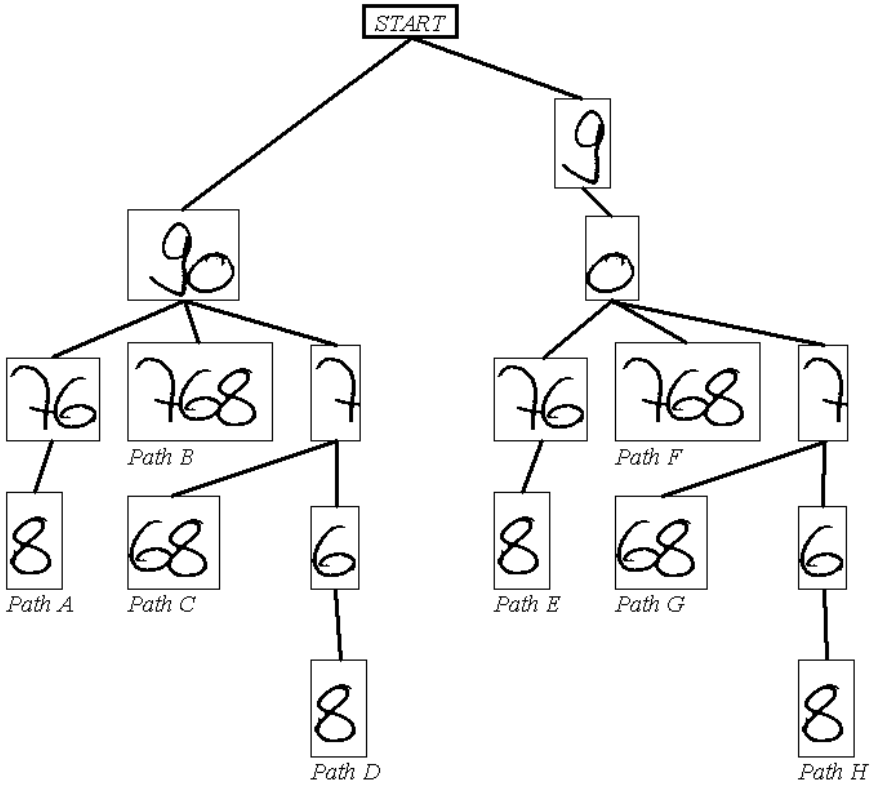

Fig. 16. All possible segmentation paths for an example ZIP code block.

So we have one possible segmentation point for the first and two for the second component. Since we (at this state) cannot know, which segmentation points are right or wrong or sufficient and which maybe even split digits in half, we have to examine all eight possible segmentation alternatives. This means building a tree as shown in figure 16.

These alternative segmentation paths are now evaluated by the digit classifier which has to examine all nine resulting 'snippets' (potential digits). The total score of one of the segmentation paths is the product of all the single classifier scores of its member snippets. So paths containing snippets which the classifier rejects by giving it a very low confidence get very small total scores. It keeps the path containing as much as possible 'recognizable snippets' (i.e. well-segmented digits), in our example *Path H*.

Whether the path with the maximum score is finally believed to be a valid ZIP code or not depends (besides on its score) on certain other considerations which will be explained later in section 5.

4.2.2. Separating digits using structural information

The second system used to separate the digits of a given ZIP-code works using structural information. It consists of the stages: i) preprocessing, ii) structural segmentation, iii) classification of isolated digits, and iv) combination of the single digit results into a ZIP-code. The system starts with the leftmost digit and uses the classification results to determine which segmentation alternative is evaluated next. Therefore, the segmentation process is recognition oriented.

Preprocessing: The preprocessing of the pixel-images that contain scanned ZIP-codes is very similar to the preprocessing of single digits for structural recognition described in section 4.1.2. After the background has been removed from the ZIP-code image (see figure 17(a)), it is scaled, smoothed, and skeletonized (b). A line-drawing (c) is produced that is analyzed to construct a stroke-representation (d).

The normalization now differs from the digit-normalization. Here the image is sheared vertically in such a way that the horizontal principal axis becomes exactly horizontal. Also size normalization uses a different box for the scaling. The simplification of the stroke-representation takes into account that some of the ZIP-codes have been underlined. It removes long strokes at the bottom of the digit-block.

In addition to the horizontal slope normalization we also apply a vertical slant correction, shown in figure 17(e). Since the digits have not been isolated yet, we can not estimate the vertical axis of the single digits. Therefore, we estimate the slant of the digit-block by computing the length of its parallel projections to the horizontal axis for different projection angles that vary around the vertical. If the projection angle matches the digit slant, the length of the projection (shadow) is minimal. This gives us a slant-estimate that is used to put the digits upright with a horizontal shear.

Preprocessing produces a cleaned representation of the digit-block. Most of the noise has been removed, some connected digits have been separated and most broken lines have been completed. If adjacent digits touch, they do so only at the ends of strokes.

Structural segmentation: Now the strokes have to be grouped as digits. We take into account that German ZIP-codes are composed of exactly five digits. The idea is to run an iterative clustering process that assigns each stroke to one of five centroids. Clustering is based on the weighted LBG-method [16]. Here, we use a customized distance function between a stroke s_i and a centroid d_j:

$$\delta(s_i, d_j) := |\mathsf{center\_x}(s_i) - \zeta_j| + \eta_1 \, \varsigma_j + \eta_2 \, \mathsf{different\_component}(s_i, d_j),$$

where $\mathsf{center\_x}(s_i)$ is the x-coordinate of the center of gravity of s_i, ζ_j is the x-coordinate of d_j, ς_j is the sum of the length of all strokes that have been assigned to d_j, and $\mathsf{different\_component}(s_i, d_j)$ indicates with a value of one that s_i is not member of the connected component of strokes that has been assigned to d_j. The distance is based on the x-differences between strokes and centroids. The parameters η_1 and η_2 control the influence of the fairness component and the connectivity component, respectively.

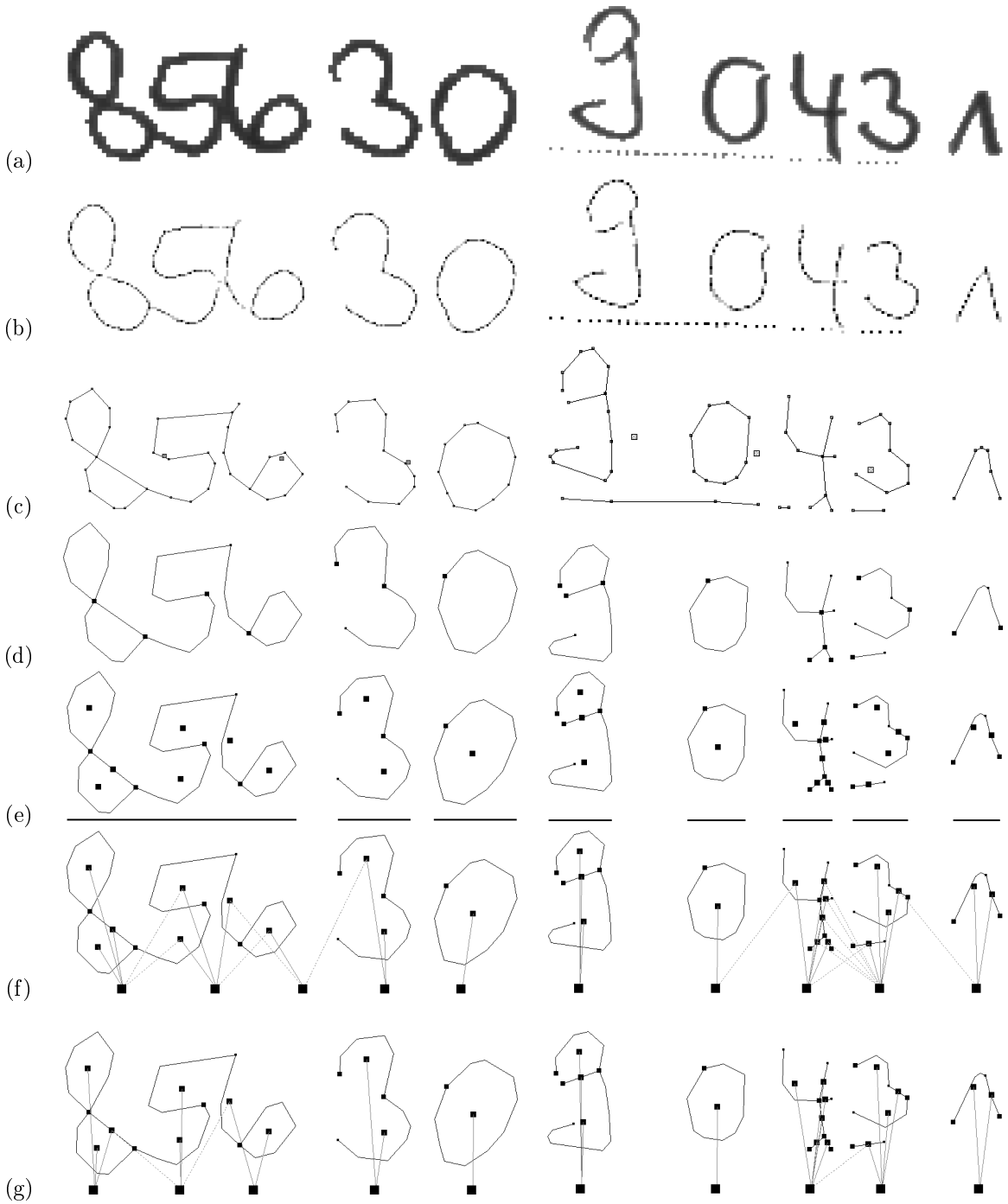

Fig. 17. Preprocessing and segmentation of digit-blocks: (a) background removed, (b) skeleton, (c) line-drawing with estimated horizontal axis, (d) stroke-representation, (e) upright with shadow, (f) clustering initialized, (g) final assignment of strokes to centroids.

The projection of the digit-block to the horizontal axis is used to initialize the centroid positions, as can be seen in figure 17(f). If the shadow is composed of exactly five components (the optimal case), one centroid is placed in the middle of each component. If there are only four components (two digits touch), two centroids are placed in the largest component.For other numbers of components, the centroids are distributed regularly with slightly larger distances on the left side of the block, since people tend to write the first digits of a ZIP-code larger than the last ones.

The following clustering process works iteratively in EM-manner. In each step the positions of the centroids are estimated based on the assignment of strokes and the assignment of strokes is updated based on the positions of the centroids. The clustering is terminated, if the centroid positions do not change any more or a maximal number of iterations has been exceeded.

The assignment of strokes uses the distance function $\delta(s_i, d_j)$. Each stroke is assigned to the centroid that has the minimum distance, as indicated by solid lines in figure 17(e, f). If the second best centroid is not farther away than twice the minimal distance, a secondary assignment is done (indicated by dotted lines). The existence of secondary assignments signals the uncertainty of the algorithm about the correct segmentation.

The new centroid positions are a weighted average of the x-coordinates of the assigned strokes centers. The weighting factor is the stroke length. The update of the centroid positions is damped to avoid oscillations. It can happen during clustering that a centroid does not get any strokes assigned. In such a case the centroid with the largest assigned stroke length is split by placing the empty centroid next to it.

Digit classification: The results of the structural segmentation are the primary and secondary stroke assignments, as indicated in figure 17(g) and figure 18. Now the digits need to be cut out of the digit-block to be presented to a digit classifier. Up to three separation proposals are produced for each centroid:

- *Primary separation:* All strokes that are primarily assigned to the centroid.
- *Secondary separation:* All strokes that are primarily or secondarily assigned to the centroid.
- *Third separation:* All strokes that are primarily assigned to the centroid and that have no secondary assignment to some other centroid.

Note that not all three proposals need to exist and that some proposals might be identical.

The separation proposals are generated in this order, starting with the leftmost digit, and are presented to the digit classifier. If this classifier accepts the digit, then the process moves on to the next digit. If the classifier rejects a separation proposal, the next separation proposal is generated and is again presented to the classifier until it accepts the digit or all three proposals have been generated. If more than one separation was tried, the classifier outputs are combined to achieve a consistent interpretation.

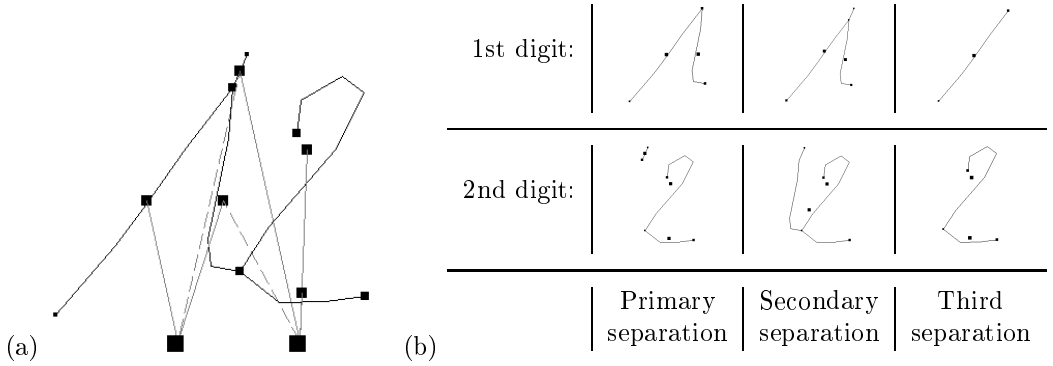

Fig. 18. Separation proposals: (a) Assignment of strokes to centroids. Primary assignments are shown as solid grey lines, secondary assignments are shown as dashed lines. (b) All possible separation proposals.

For the digit recognition we use the combined structural and TDNN classifier system that has been described in section 4.1. The preprocessing for the structural classifier is reduced to a slant and size normalization of the digit and the merging of strokes to curves. For the TDNN-preprocessing, the strokes are traversed with a pen of the estimated line width. This produces a mask which is used to cut the digit out of the pixel image, as illustrated in figure 19.

Combination of the classifier outputs: The digit recognition system produces for each digit the two best classes with assigned confidence values. Furthermore, a reject suggestion is produced for each digit. Now we combine these results to get the ZIP-code. For the first proposal the best classes from the five digits are simply concatenated. For the second ZIP-code proposal we replace the best class with the second best class at the position where the largest ambiguity about the correct classification of the digit exists. This is indicated by the largest confidence for a second best digit class. The confidence for the ZIP-code is the product of all used digits confidences. The ZIP-code is rejected, if this confidence falls below a threshold or some digits are rejected.

Experimental results: The performance of the structural ZIP-code recognition has been evaluated using a database of 5137 isolated German ZIP-codes. One indication for the effectiveness of the structural segmentation is the fact that the segmentation is unambiguous for about 70% of the blocks. No secondary stroke assignments exist in these cases and exactly five primary digit cuts are presented to the classifier. In those cases in which secondary stroke assignments indicate ambiguity about the correct segmentation, additional separations need to be considered only, if the primary cut is rejected. The average number of 5.12 cuts per block indicates that this is frequently not the case. In most cases the digit recognition is done using only the structural classifier. Only for about 1.33 digits per block the TDNN-classifier is consulted.

The structural segmentation method is quite fast, since it does the preprocessing for the structural digit recognition. The throughput of the entire recognition system is about 30 blocks/second on a Pentium-II/266. The most time consuming tasks are slant normalization of the blocks and the classification with a TDNN.

4.2.3. Combining the different segmentation states

We can combine different classifier outputs (as shown in section 4.1.3 and we can now as well combine the two segmentation results. This is also done in a sequential manner, for performance reasons.

First of all, the segmentation tree is built and evaluated as explained in section 4.2.1. If a valid and acceptable result is found, we take it and are finished. If not, the structural segmentation is run. If we get a valid and acceptable result this time, we take it and are finished now.

If both segmentation approaches fail, we try to combine their results. This is done for those cases where the results suggest, that the problem was rather the classification of the snippets than the segmentation itself. Due to the different preprocessing and classifier combination in the different segmentation tools, we might get different result which complement each other. So a combination

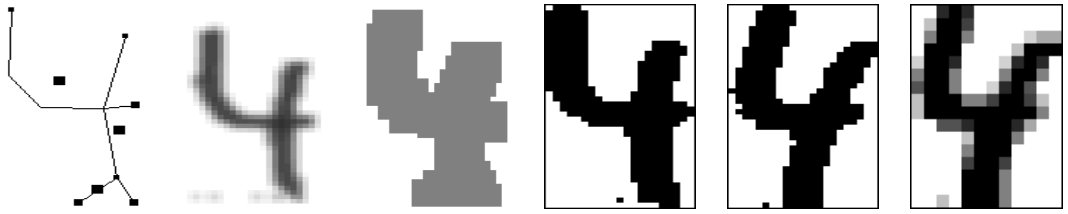

Fig. 19. Cutting digits from the pixel-image and preprocessing for the TDNN: (a) stroke-representation, (b) corresponding smoothed pixel-image, (c) mask, (d) cut and binarized digit, (e) slant corrected digit, (f) scaled digit.

of the results is only performed, if none of the segmentation tools completely refuses to segment the image into five digits and both results do not differ in more than two digits. Then a combined result is produced by choosing those digits which had, in the different results, the highest confidence. Two examples are given below

	Result	Confidences
Seg. Tree	90700	1.0;1.0;1.0;0.1;0.2;
Structural Seg.	90768	0.9;0.5;0.2;1.0;1.0;
Combination	90768	1.0;1.0;1.0;1.0;1.0;
Seg. Tree	90700	1.0;1.0;1.0;0.1;0.2;
Structural Seg.	00768	1.0;1.0;1.0;1.0;1.0;
Combination	REJECT	-.-;-.-;-.-;-.-;-.-;

The restriction of not fixing more than two uncertain digits is done to avoid to 'interprete' ZIP codes into e.g. non digit blocks, only because of possible individual classifier weaknesses and so get a failsort.

5. ZIP code verification

In the previous sections we have shown how, from a given AOI, the digit block containing the ZIP code can be extracted and how finally this block can be interpreted. The final question is, how other information on the letter can be used to make sure that the 'ZIP copde read is the ZIP code ment', so we have to verify the result. We have basically three kinds of information to answer this question.

The first information is the score or confidence with which the segmentation tool has recognized the ZIP block and its single digits. The lower this confidence is, the more likely we have made an error and should reject the letter, unless we can exploit more detailed information (see below).

The second information is, if the zipcode recognized actually exists or not. In germany, out of the possible 100.000 zip codes (five digits) only about 49.000 do exist at all, due to the method the regions of the country and states were mapped onto postal regions.

The third, most important information about the destination address (besides the ZIP code) is of course the cityname written on the letter. How it can be read using special approaches is briefly described below. The name of the city has to fit with the ZIP code (i.e. the pair must exist in a special postal database), otherwise we have possibly made an error processing the AOI. In some cases (especially an if few digits were only weakly recogniced) we can 'fix' the total result by combining cityname and ZIP code results.

5.1. Reading handwritten citynames using Hidden Markov Models

The problem of reading handwritten city names is highly complex, and the 'segmentation and classification approaches' described above for the ZIP code often fail for this task. The number of possible segmentation possibilities explodes and isolated characters, or potential ones, are often too ambigous when taken just for themselves. Therefore other, probably segmentation-free approaches have to be used, which take into account contextual information. Usually *Hidden Markov Models (HMMs)*, known from speech processing, are used. For a general description of HMMs see [21].

The use of HMMs for the handwriting recognition problem has been proposed by several authors [28, 24, 4]. The basic idea is always similar. Over the normalized (i.e. binarized, slant corrected, etc.) image of the city name a window of usually a fixed number of columns is displaced and a feature vector f_i is extracted.

During the *HMM training*, the possible feature vectors are first reduced by clustering them using a vector quantization approach (LVQ). Each cluster is represented by a prototype, which is stored in a codebook. Then HMMs have to be trained, either for parts of the city name (characters, sylables, trigrams) or for the whole word. For the actual version of the GSA-OCR, models of single characters are trained, using the *Baum-Welch algorithm* [21, 24].

To simplify the actual *HMM recognition step*, information already available about the ZIP code is exploited. It has been already classified, with possible some uncertain digits in it. A ZIP code with one uncertain digit has a maximum of ten alternatives, which lead to a list of maximum ten citynames $cn_0, \ldots, cn_9$. The problem thus reduces

from a 'free recognition' of the city name to matching it to one of the cn_i names in the list. From the citynames cn_i, corresponding HMMs $HMM(cn_i)$ are produced as a chain of the models of the character in cn_i. Then also a feature vector f_i are extracted from the cityname to recognize as described for the training. These are than mapped to the codebook to obtain the prototype c_i representing f_i. Then all the resulting city name models $HMM(cn_i)$ are evaluated with the sequence of these codebook vectors c_i using the *Viterbi algorithm* [21, 24]. The one which matches best (above a certain threshold) is the result.

6. Conclusion

In this article we have described the OCR system implemented in the Siemens flat reading system *SICALIS FSS - C200*, GSA for short[3]. The GSA is a real world application of neural and image processing algorithms; the machine sorts millions of non-standard letters every week in about 80 sorting centers all over Germany with significant speed and recognition rates.

We gave a brief description of how addresses are found on the envelopes and how these addresses are analyzed to finally get to the ZIP code block. We especially concentrated on the problem of reading and segmenting handwritten ZIP codes. Two methods were described for classifying handprinted digits, and two methods to segment the ZIP code block into its single digits.

The final classifier is a combination of a *pixel-oriented method*, the neural TDNN classifier, and a *structure analyzing method*. Both offer some advantages and disadvantages. With the neural approach, very high recognition rates can be obtained using very little prior knowledge. After the usual preconditioning (binarization, uprighting and scaling to fixed size), feature extraction and classification is done automatically by the TDNN learning algorithm. This requires, on the other hand, more computational effort than the structural approach.

The structural approach is based on extensive preprocessing of the digit to be recognized. The image is first skeletonized and then converted to a graph of strokes, which is then matched against a set of prototype graps extracted from a training set. Different digits resulting in the same graph are then discriminated using quantitative features of the graph used as input to small neural networks. This structural approach exploits of course extensive prior knowledge about digits and the way they 'could have been' written. This leads to significant recognition speed and very low error rates, since only digits exhibiting a typical structure are recognized. The drawback is, of course, the much larger effort involved in building and training the classifier. The recognition rates are also sligthly lower than those of the TDNN.

The same hybrid approach was used for the segmentation algorithms. We have a pixel oriented method, which analyzes binarized connected components and a second structural approach used as backup. In both tasks (classification and segmentation) the two used orthogonal approaches complement each other strongly. The advantage is obvious for the classifiers: The bulk of the digits which are carefully written and thus possess a typical structure are fast and reliable recognized by the structural method. When in doubt, the digits are rejected and are recognized by the powerful TDNN, which relies more on the 'general appearance' of the digits image, not on structural similarities. When used in combination, more than 99% of the NIST digits can be recognized correctly.

Also segmentation states complement each other. If the 'free' segmentation (i.e. the number of digits to separate is undetermined), done by analysis of the components outlines, fails, the structural segmentation is called. Again its orthogonal vectorization approach and also its more strict segmentation goal (strictly dividing the ZIP code into five clusters) as well as the eventual prior knowledge will very likely complete the task.

This paper has thus shown that a complex classification task can be better solved by using a hybrid approach: when two or more classifiers base their decision on different sets of features, they can be combined to produce a more reliable system. This hybrid approach is followed consistently in the non-standard letter sorting machine GSA designed and built by Siemens.

Notes

1. The system described was ordered, developed and installed three years before the aquisition of ElectroCom GmBH (SEC) by Siemens. It is not identical with the SEC OCR.
2. The official name is *SICALIS FSS-C200*, see also note above
3. See also note 1 above.

References

1. Adaptive Solutions Inc. Beaverton OR., 1992. *CNAPS Manual.*
2. S. Behnke, M. Pfister and R. Rojas. *Recognition of Handwritten digits using Structural Information.* Proceedings of the ICNN '97, Houston/Texas, 1997.
3. S. Behnke, M. Pfister and R. Rojas. *A Study on the Combination of Classifiers for Handwritten Digits Recognition.* Proceedings of the rd International Workshop on Neural Networks in Applications NN '97, Magdeburg/Germany, 1998.
4. H. Bunke, M. Roth and E. G. Schukat-Talamazzini. *Offline Cursive Handwriting Recognition using Hidden Markov Models.* Pattern Recognition, Vol. 28, No. 9, 1995, pp. 1399-1412.
5. R. G. Casey and E. Lecolinet. *A Survey of Methods and Strategies in Character Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 7, pp. 690–706, 1996.
6. S. Fahlmann and C. Lebiere, *The Cascade Correlation Learning Algorithm.* Technical Report CMU-CS-90-100, Carnegie Mellon University, 1990.
7. R. Fenrich. *Segmentation of automatically located handwritten words*, Proceedings of International Workshop on Frontiers of Handwriting Recognition, Bonas, France, September 1991.
8. R. Fenrich. *Segmentation of Diverse Quality Handwritten Digit Strings*, Sixth Annual University at Buffalo Graduate Conference on Computer Science, March 18, 1991, pp. 1-9.
9. Fenrich, R. *Segmentation of Automatically Located Handwritten Numeric Strings*, From Pixels to Features III, Sebastiano Impedovo and J.C. Simon (Editors), 1992, pp. 47-60.
10. R. Fenrich, S. Lam and S.N. Srihari. *Optical character readers* Encyclopedia of Computer Science and Engineering, Third Edition, A. Ralston, ed., Van Nostrand, 1992, pp. 993-1000.
11. J. Franke and M. Oberländer. *Writing style detection by Statistical Cobmination of Classifiers in Form Reader Applications.* Proceedings of the ICDAR '93, Tsukuba Science City/Japan, pp. 581–584, 1993.
12. M. D. Garris *et al NIST Form-Based Handprint recognition System.* NIST Internal Report 5959, 1994.
13. P. J. Grother and G. T. Candela. *Comparison of Handprinted Digit Classifiers.* Technical Report NISTIR 5209, NIST, 1993.
14. L. Holmström, P. Koistinen and E. Oja. *Comparison of Neural and Statistical Classifiers – Theory and Practice.* Research Report A13, University of Helsinki, Finnland, 1996.
15. A. Kaltenmeier *et al. Sophisticated Topology of Hidden Markov Models for Cursive Script Recognition.* Proceedings of the ICDAR '93, Tsukuba, 1993.
16. Y. Linde, A. Buzo, and R. M. Gray, *An Algorithm for Vector Quantizer Design.* IEEE Transactions on Communications, vol. 28, no. 1, pp. 84–95, 1980.
17. H. Niemann. *Klassifikation von Mustern.* Springer, New York, 1983.
18. N. Otsu. *A Threshold Selection Method from Greylevel Histograms.* IEEE Transactions on Man, Systems and Cybernetics, Vol. 9, No. 1, 1997, pp. 62–66.
19. P. W. Palumbo, P. Swaminathan, S. N. Srihari. *Document Image Binarisation: An evaluation of Algorithms.* Proceedings of the SPIE Symposium on Applications of Digital Image Processing IX. Vol 697,pp. 278–295, 1986.
20. M. Pfister. *Learning Algorithms for Feedforward Neural Networks – Design, Combination and Analysis.* PhD Thesis, FU Berlin, 1995.
21. L. A. Rabiner. *A Tutorial on Hidden Markov Models with selected Applications in Speech Recognition.* Proceedings of the IEEE. Vol 77, No. 2, 1989.
22. R. Rojas. *Neural Networks.* Springer, New York, 1996.
23. J. Schürmann. *Pattern Classification – A Unified View of Statistical and Neural Approaches.* Wiley-Interscience, New York, 1996.
24. M. Schüßler and H. Niemann. *A System for Reading Handwritten Addresses.* Accepted at the 6th International Workshop on the Froniers of Handwriting Recognition, Taejon City/Corea, 1998.
25. L. Shastry and T. Fontane. *Recognizing Handwritten digit Strings using Modular Spatiotemporal Connectionist Networks.* Connection Science. Vol 7, No. 3, 1995.
26. P. Simard, Y. LeCun and J. Drucker. *Improving Performance in Neural Networks using a boosting Algorithm.* Advances in Neural Information Processing Systems. Vol. 5, pp. 42–49, 1993
27. A. Waibel *et al. Phoneme Recognition using time-delay Neural Networks.* IEEE Transactions on Acoustics, Speech and Signal Processing. Vol 37, No. 3, pp. 328–339, 1989.
28. A. Williams Senior. *Offline Cursive Handwriting Recognition using Recurrent Neural Networks.* PhD Thesis, Trinity Hall, Cambridge/England, 1994.

29. M. Wolf, H. Niemann and W. Schmidt. *Fast Address Block Location on Handwritten and Machine Printed Mail-piece Images.* Proceedings of the ICDAR '97, Ulm/Germany, pp. 753–757, 1997.

30. M. Wolf and H. Niemann. *Form-Based Localization of the Destination Address Block on Complex Envelopes.* Proceedings of the ICDAR '97, Ulm/Germany, pp. 908–913, 1997.