Morphological design of Discrete-Time Cellular Neural Networks

Brugge, Mark Harm ter

Chapter 7 Application of the Theory in Car License Plate Recognition

In this chapter the morphological DT−CNNdesign method is demonstrated in a practical application: a system for the automatic recognition of car license plates. An overview of the recognition system is given and the modules that are most suitable for a fast parallel implemention by DT−CNNsare identified. For these modules morphological solutions are defined and transformed into DT−CNN templates. No extensive simulations are required, since the designed DT−CNN systems are correct by construction. The modules that are less suitable for implementation with DT−CNNsare presented superficially, just for the sake of completeness. The aim of this chapter is not to show that these modules perform better or worse than similar systems found in literature and the examples given in this chapter are merely illustrative.

This chapter is based on [14][15][16][19][81][82][83][84].

7.1 Introduction

The automatic identification of vehicles by the content of their license plate is an important issue in private transport applications such as travel−time measurement [56], parking lot traffic management [37], toll collection [30], speed−limit enforcement [93], and identification of stolen cars [55]. The use of a Car License Plate Recognition (CLPR) system in a real−time/life environment frequently puts high demands on the contents of the image and the throughput of the system. For example, the use as a law enforcement system requires that the car and a reasonable amount of environment is visible in the image. This implies that the license plate occupies only a small part of the overall image (see Fig. 7.1). In low−end markets like the parking market where low−cost cameras are the only option, low resolution images and difficult lighting conditions will make the task of automatic license plate recognition a real challenge.

The limited resolution of the recorded characters together with dirt (dust, flies), screws and bolts (used to attach the plate to the car), overhanging car parts (tow bar mounted too high), etc. make the development of a reliable optical character recognizer (OCR) very complicated. Many of the

approaches common in conventional OCR systems [77] turn out to be useless for CLPR. It is therefore necessary to build a system that is capable of exploiting both rules defined by the license plate registration regulation like syntactical, geometrical and font descriptions (explicit knowledge) [101] and features present in the measurement of the OCR task at hand (implicit knowledge) [1].

Besides obtaining high recognition rates for the complex situations described above, system throughput is also a decisive factor in the successfulness of a commercial CLPR system. Many applications demand response times well below 0.3 seconds per image. This time requirement applies not only to the standard CCTV image streams, but also to the larger 3000×2000 law–enforcement type images. Meeting these timing requirements poses a real challenge to the design of a CLPR system.

Figure 7.1: Some input examples for the CLPR system. The images are recorded by a police–operated monitoring system. Each image contains 768x567 pixels with 8–bit gray scale information per pixel.

The CLPR system described in this chapter has come a long way since the initial attempts in 1994. In the first release of the system we have focused on the exploitation of fuzzy logic to reason from the registration regulations [83]. In [84] the character recognition is pushed onwards. In [19] we identify the most time–critical parts of the system and use the theory developed in [17] to replace them by formally derived DT–CNN realizations. In [15][16] the system is extended with mixed expert systems and the recognition performance obtained through extensive testing is compared to the recognition performances of alternative systems found in literature. This chapter presents an overview of the above work and some of the work that has been done by DACOLIAN B.V.

(of which the author of this dissertation is one of the founders). The company has invested over 20 man–years to develop one of the world's best libraries for the high–yield recognition of vehicle license plates.

Fig. 7.2 gives a schematic overview of the CLPR system. It consists of four main units: a plate segmenter, a character isolator, a character recognizer and a syntactical/geometrical analyzer. The segmenter determines the location of the license plate based on structural features and some weak size constraints. The large amount of data that has to be inspected (the whole image) and the space invariant character of the problem makes this unit a primary candidate for realization by DT–CNNs. The extracted plate is passed to the character isolator, which determines the location of the characters on the plate. The character isolator is largely realized by DT–CNNs. The isolated characters are passed to the character recognizer, which uses a mixture of different multi–layer perceptrons (MLPs) and a template matcher to read the character. Finally, the syntactical/geometrical analyzer checks whether the candidate characters returned by the recognizer satisfy the syntactic and geometrical rules that exist in a certain country or state. If these rules are not satisfied, or one of the characters is unrecognizable, the image is rejected.

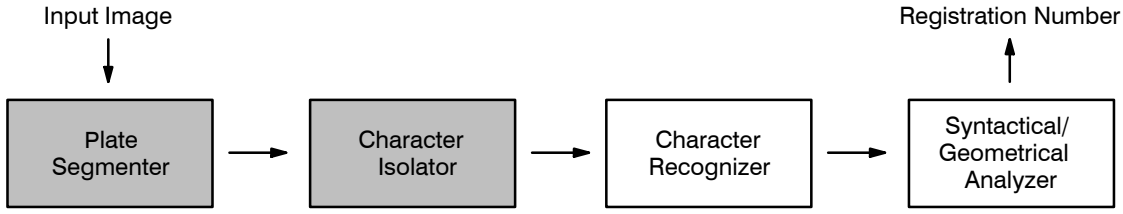

Figure 7.2: Outline of the Car License Plate Recognition System. The gray shaded modules use DT–CNNs.

The organization of this chapter is as follows. Section 7.2 discusses license plate segmentation. Section 7.3 describes the character isolator. Then we focus our attention on feature–based classifiers in Section 7.4, followed by Section 7.5, in which the syntactical/geometrical analyzer is described.

7.2 Segmentation

One may conclude from the images in Fig. 7.1 that a fixed location for the license plate within the overall image cannot be assumed. Virtually each pixel may belong to a license plate and needs therefore to be processed. In order to reduce processing time it makes sense to reduce the area of interest as soon as possible. Only those parts of the input image that fulfill a set of "license plate properties" need to be considered for further inspection.

Several authors have proposed solutions for the license plate segmentation problem [4][55][84][90]. All of these methods incorporate a number of features of the license plate like for example: "a license plate is a rectangular area on the car, which contains a number of dark characters". Inspired by this kind of explicit knowledge, two features (grayness and texture), have been appointed to each pixel in the image. In [84] we use fuzzy membership values between 0 and 1 for both features. The large amount of pixels that has to be processed in this segmentation phase make it very beneficial to use algorithms and techniques that are potential suitable for a parallel implementation. This observation is for example subscribed by [2] in which two–dimen-

sional cellular automata are used to calculate horizontal gradients and to cluster "plate" pixels. Other researches also use spatial invariant operations, but do not focus on their parallel implementation possibilities. For example in [9] and [40] morphological operations are used to eliminate noise from the input images. More sophisticated use of morphological techniques that extend beyond noise elimination are described in [53] and [105].

To obtain a real increase in throughput we have to parallelize the complete segmentation phase and not only a part of it. The research described in this thesis has shown that morphological specifications can be realized by fast parallel DT–CNNs. With this in mind and timing requirements as a driving quality attribute, we have designed a segmentation algorithm that is almost completely based on morphological operations.

The first step of the segmentation unit is based on the work we describe in [84]. The use of DT–CNNs forces us to represent grayness and texture by crisp values -1 and 1 (see [19]). A grayness of 1 indicates that the pixel has a gray–value that corresponds to the color of a license plate. A grayness of -1 indicates that the color of the pixel is outside the range of gray–values that is common for license plates.

The appropriate ranges for grayness and texture are derived statistically. For the grayness feature, the gray–values of pixels taken from a large number of exemplary license plates are used to construct a frequency table (the number of occurrences of each gray–value in each license plate). Based on this table, the value range is derived. For our particular system set–up, the gray–value of most license plate pixels turned out to be in the range $[0.10, 0.57]$. For the texture feature the histogram is constructed by applying a 3×3 Sobel operator to each license plate pixel. Most license plate pixels turned out to have an absolute Sobel–value larger than 0.73.

The range for grayness has an upper bound and a lower bound. This requires a two step evaluation. In the first step all pixels of the input image (Fig. 7.3a) with a grayness value larger than 0.1 are extracted (Fig. 7.3b). In the second step all pixels with a grayness value larger than 0.57 are deleted from the output (Fig. 7.3c). The template for grayness evaluation is given by:

$$T_1 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -0.1 > \qquad T_2 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -0.43 >$$

Note that not all pixels on the plate are black in Fig. 7.3c. Particularly, this is the case for pixels that are part of a character. Such pixels are not selected since their grayness value is too low. One way to solve this problem is lowering the lower bound of the grayness range. However, this makes the grayness feature less specific or even worthless. Another way to solve the problem is to make all white pixels black that are close to one or more black pixels. Morphologically, such an operation is described by a dilation. We have chosen a 7×7 block structuring element to dilate the grayness image (Fig. 7.3d). The choice of the SE is based on the assumption that each character consists of a set of narrow lines that have a width of less than 7 pixels. According to Section 3.6 this dilation is implemented by a DT–CNN with a 7×7 template. To avoid the use of templates larger than 3×3, the 7×7 SE is decomposed into three successive dilations with a 3×3 block SE before a mapping onto DT–CNNs is constructed (see Chapter 5). After decomposition in the morphological domain, the following additional DT–CNNs steps are obtained for the evaluation of the grayness feature:

$$T_{3..5} = < \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 8 >$$

For texture evaluation the Sobel operator is mapped onto DT–CNNs. Section 3.6 shows that morphological operators can be mapped onto DT–CNNs. For traditional filter operations (possibly combined with traditional set operations) a similar mapping can be constructed. For example, the combination of a filter operation with a mask M and a threshold operation with threshold T is implemented by a DT–CNN with a control template that is equivalent to the transpose of M, a zero feedback, and a bias that is equivalent to $-T$. The proof is left as an exercise for the reader. This rule is used to derive the template for the first step, in which all pixels with a Sobel–value larger than 0.73 (dark/light transitions) are extracted. In the second step all pixels with a Sobel–value smaller than -0.73 (light/dark transitions) are added (Fig. 7.3e), resulting in the following DT–CNN for extracting texture pixels.

$$T_1 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, -0.73 > \qquad T_2 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, -0.73 >$$

Notice that for edge/texture detection we have constructed a two–layer Sobel DT–CNN instead of using the traditional single–layer Laplace DT–CNN. In practice Sobel is often preferred to Laplace since Laplace is a second–derivative operator, which makes it more sensitive to noise.

Similar to the initial grayness image, the texture image also needs to be dilated. To this purpose we dilate with a 15×9 block SE, which can be decomposed into a sequential dilation with four 3×3 block SEs and three 3×1 block SEs. A DT–CNN implementation of this operator sequence (except for the last dilation) is given by:

$$T_{3..5} = < \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 8 > \qquad T_{6..8} = < \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 2 >$$

The reason for not mapping the last dilation is that it can be combined with the following operation. Remember that a part of the image is very likely to be a license plate if it has the appropriate grayness and the appropriate texture. The combined feature is obtained by computing the intersection of the grayness and the texture image (Fig. 7.3g). To this purpose the grayness image is stored on the input of the second DT–CNN. The next step this DT–CNN has to perform is dilating the output (the postponed dilation) and computing its intersection with the input image. Substituting the 3×1 block SE into the generic template definition of $U \cap (V \oplus A)$ (see Section 4.7) gives:

$$T_9 = < \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -1 >$$

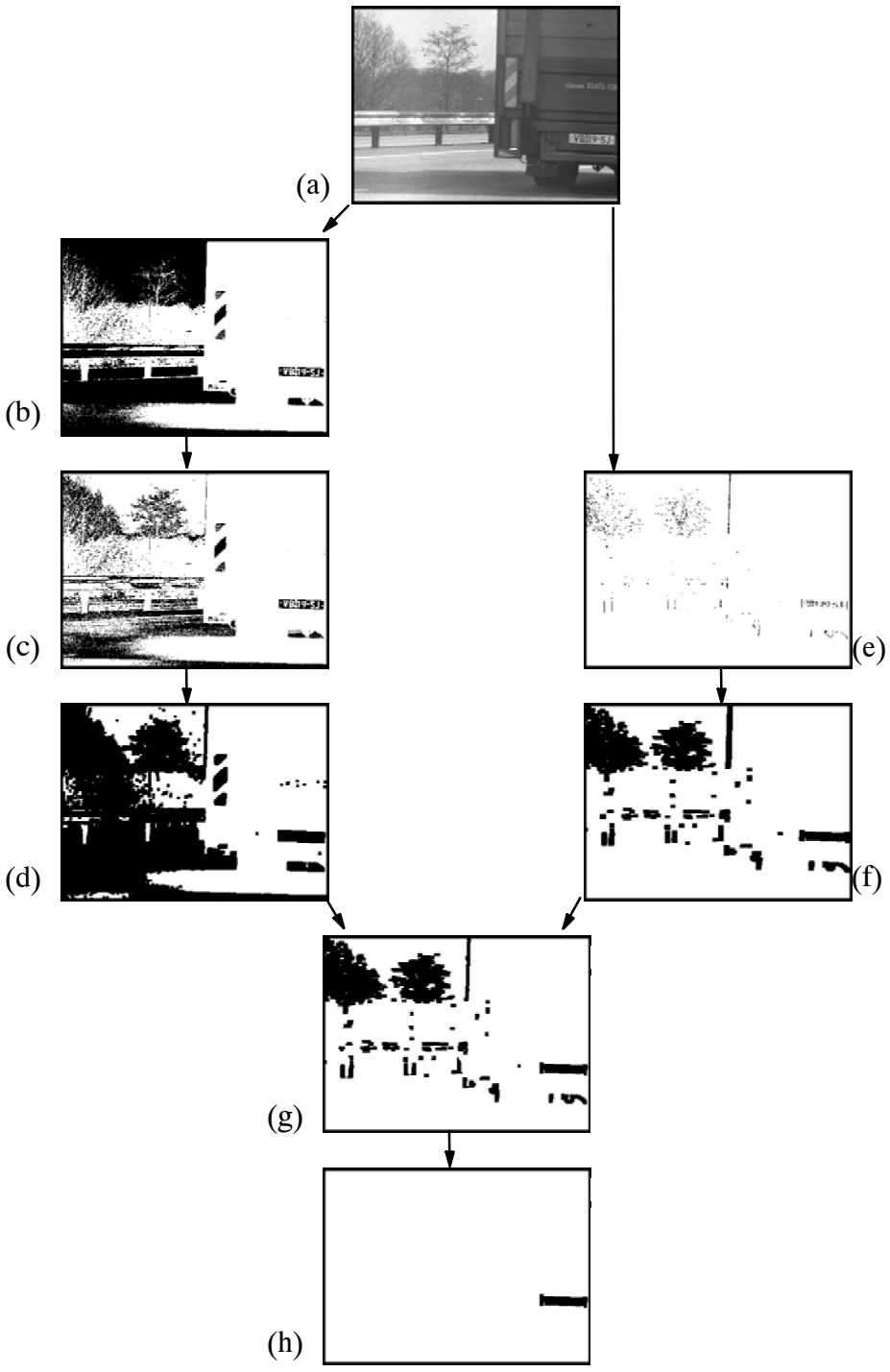

Figure 7.3: Extraction of potential license plates: (a) input image, (b) pixels that are bright enough, (c) grayness feature, (d) dilated grayness feature, (e) texture feature, (f) dilated texture feature, (g) combination of texture and grayness, (h) plates after deletion of undersized and oversized components.

Notice that Fig. 7.3g still contains a number of objects, that are not license plates. Apparently, objects like trees and parts of the crash barrier have the same characteristics as license plates. To

further reduce the number of non–plate components, we use a number of weak size constraints as described below. After applying these constraints the image in Fig. 7.3h is obtained.

The four weak size constraints that we use to reduce the set of potential license plates are the minimum/maximum height and the minimum/maximum width. In our case, the width of a plate is at least 91 pixels while the height is at least 13 pixels. The rejection of smaller plates at this stage turns out to have no effect on the overall performance of the system, since these plates are unreadable anyway due to the limited character resolution. The maximum width of a plate is 171 pixels and the maximum height is 61. The size constraints can be computed in parallel by four DT–CNNs. First a selector image is computed for each size constraint (Fig. 7.4a–d). Such an image contains pixels at positions where a horizontal/vertical bar of a certain length fits in the black parts of the image given in Fig. 7.3g. For example, the minimum height selector contains a pixel everywhere a vertical bar of height 13 fits in the image. Morphologically, a size selector is found by means of the erosion operator with the appropriate SE. The minimum height selector is found by an erosion with a 1×13 block SE. Similar to dilation, erosion with a 1×13 block SE is decomposed into six erosions with a 1×3 block SE. Using the generic template for erosion the following six–step DT–CNN is found:

$$T_1 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, -2 > \qquad T_{2..6} = < \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -2 >$$

The first step dilates the image that is stored on the input (Fig. 7.3g), while the following five steps perform the additional dilations on the output. After computing the selection, all objects need to be restored (Fig. 7.4e). To this purpose each object in the output is enlarged (dilated with a 3×3 cross SE) step by step. In order to avoid adding pixels that are not part of the object in the original image (that is still on the input), an intersection with the input is required after each step. Substituting the 3×3 cross SE into the generic template that evaluates $(V \oplus A) \cap U$ (U is the input image, V is the current output image output, A is the SE), the following restoration template sequence is obtained:

$$T_{7..k} = < \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -1 >$$

where k is the time step at which the network has converged. The three other selectors are restored in an identical way (Fig. 7.4f–h). Finally restoration results need to be combined. The set of objects that are large enough to be a license plate is found by computing the intersection of Fig. 7.4e and Fig. 7.4g. Then, all too large objects are deleted by subtracting image Fig. 7.4f and Fig. 7.4h from this result. The set subtraction ($F(U, V) = V \setminus U$) to obtain the objects that have the appropriate width and the one needed to extract objects with the appropriate height are merged through the union ($V \cup U$) in a three–layer DT–CNN.

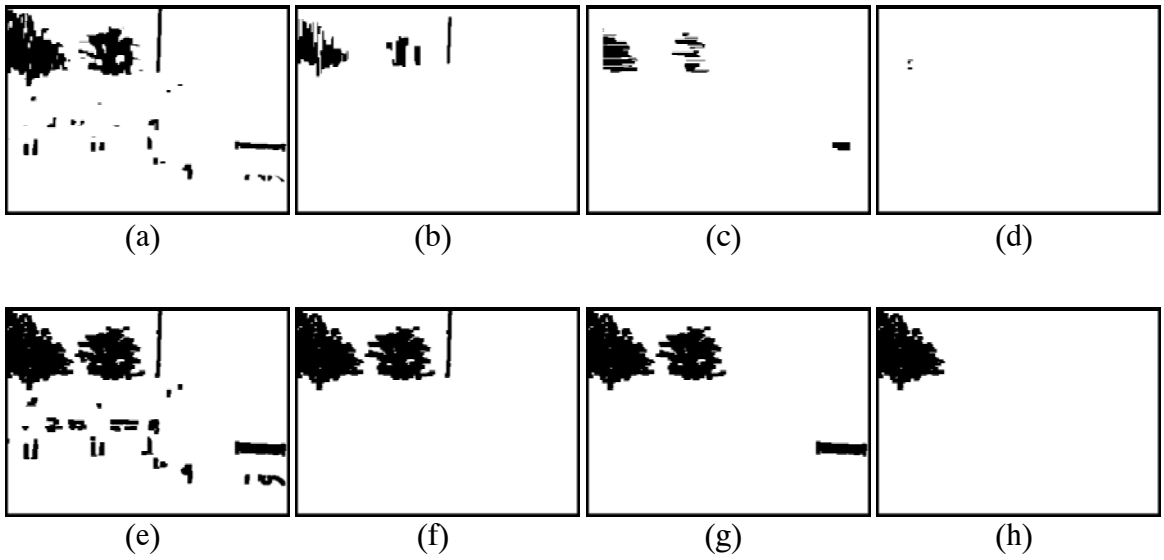

Figure 7.4: Processing size constraints: (a) selector for objects that satisfy the minimum height condition, (b) selector for objects that do not satisfy the maximum height condition, (c) minimum width selector, (d) maximum width selector, (e)−(h) corresponding selections.

7.3 Character Isolation

The second module of the CLPR system is responsible for the isolation of the individual characters on the license plate. This module is implemented by a single DT−CNN with a time varying template, and a very small part of traditional programming. A morphological parallelizable approach to solve this task is given in [109]. The authors propose the use of grayscale morphology to isolate the characters on the plate. Inspired by these ideas, we transformed the grayscale algorithms to a similar binary version. As a consequence, we need an explicit binarization step to transform the input image to an initial binary image. To this purpose two classes of adaptive binarization methods are apparent from literature [104]. *Global adaptive binarization* calculates a single threshold value for the entire image. Pixels having a gray level darker than the threshold value are labeled black, otherwise white. In contrast, *local adaptive binarization* methods compute a threshold for each pixel in the image on the basis of information contained in a small neighborhood of the pixel. Especially for images with heavy luminance variations, local adaptive methods perform better than global methods. In [15] we have confirmed this observation by giving some comparative examples. Here we use global adaptive thresholding since its spatial invariant character makes it more suitable for implementation by DT−CNNs.

Thresholding is performed by a single−layer DT−CNN, whose input is the gray−scale image of the cut−out plate (Fig. 7.5a) and whose bias is the threshold value t as computed by the global adaptive binarization method:

$$T_1 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -t >$$

The result after this first step is shown in Fig. 7.5b. Before performing a connected component search, the black area around the plate is removed. To this purpose we first determine what this area is (Fig 7.5e). Since characters are thin objects, characters can be deleted from the image by a so–called morphological opening. This is an erosion with a certain SE followed by a dilation with the same SE. Again, the choice of the SE, a 7×7 block SE, depends on the thickness of the characters. After decomposition and DT–CNN transformation the following six additional DT–CNN steps are obtained:

$$T_{2..4} = < \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -8 > \qquad T_{5..7} = < \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 8 >$$

The network output after time step 2 and time step 4 are shown in Fig. 7.5c and Fig. 7.5d respectively. In time step 8, the binarized input image is stored on the network input and the components on the plate are extracted (Fig. 7.5f) by subtracting the current output ($F(U,V) = U \setminus V$). The template for this operation is given by $T_8 = < \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -1 > \cdot$

After that, a rectangular hull extraction is performed (Fig. 7.5g). During this iterated process, white pixels that have at least two black 4–neighbors are painted black each step, until the network converges. According to [44], this template is given by $T_{9..k} = < \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 3 > \cdot$

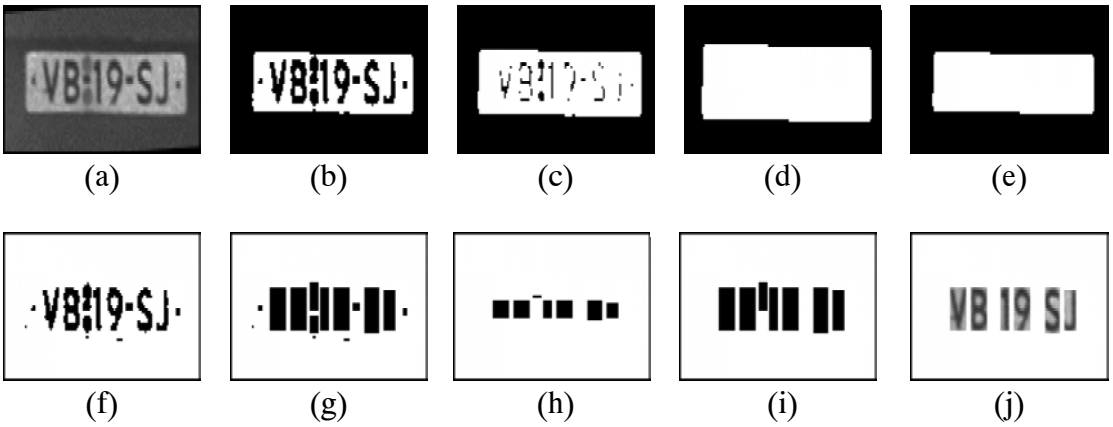

Figure 7.5: Isolation of characters: (a) the input image, (b) the binarized image, (c) the slightly eroded plate, (d) the plate after three erosions, (e) the result after three dilations, (f) the intersection of (b) and the complement of (e), (g) the convex hull of the components, (h) the selector of components that are large enough, (i) the set of components that are large enough, (j) the extracted characters.

Finally a minimum height criterion is applied similar to one of the criteria used for license plate segmentation. Since the minimum character height is 11 pixels, a height selector image is evaluated by five successive erosions with a 1×3 block structuring element (Fig. 7.5h). Restoration of the objects is much easier than the restoration operation that is described in the previous sec-

tion. Since all objects are rectangles, restoration is done by five dilations with a 1×3 block structuring element (Fig. 7.5i). Removal of boxes with a height less than 11 pixels is therefore performed by the following ten time steps:

$$T_{k+1..k+5} = < \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, -2 > \qquad T_{k+6..k+10} = < \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, 2 >$$

The last step before characters are cut out and passed to the recognizer is a selection of the objects that are centered around a (possibly slightly rotated) line with almost equal height. Since it is very hard (and probably impossible) to do this by means of DT–CNNs, this last step is performed using traditional programming. Finally, the remaining set of objects is used to cut out the characters from the original plate (Fig.7.5j).

7.4 Character Recognition

To recognize the characters, the isolated character images are transformed into a representation such that a recognizer can see the difference more easily. Presenting the image directly to a classifier makes it hard to handle distorted images (as resulting from rotation, sizing and/or translation). Therefore, it is necessary to derive a set of features that is less sensitive to such irrelevant variances. For character recognition a wealth of feature extraction methods can be found in literature. We have tested some of them on our particular data sets and found that the following three methods gave satisfactory results:

- moments
- principal components analysis
- sideway projection

Given a feature vector produced by one of the above methods, the actual classification is done by Multi–Layer Perceptrons (MLPs). The last classifier that we use is template matching. The four classification techniques will be described in more detail in the next subsections. Finally, we use a classifier combining scheme to combine the answers of the four individual classifiers (see Fig. 7.6).

The reason that we use a model with multiple classifiers is that none of the above classification strategies will achieve a 0% error at an acceptable recognition rate. This is for one reason caused by the imperfection of the limited feature sets. The other reason is the balance between false acceptance and correct acceptance. When a classifier is not permitted to make false accepts, the amount of rejects increases considerably. The motivation for using a combination of classifiers is the observation that different classifiers potentially offer complementary recognition information. Though the type of classifier models (MLPs) that are used in the first three classifiers are identical, the potential differences in classifier behavior are implied by the differences in the way the feature sets are constructed. Obviously, template matching potentially has a different classification behavior due to its different underlying evaluation model.

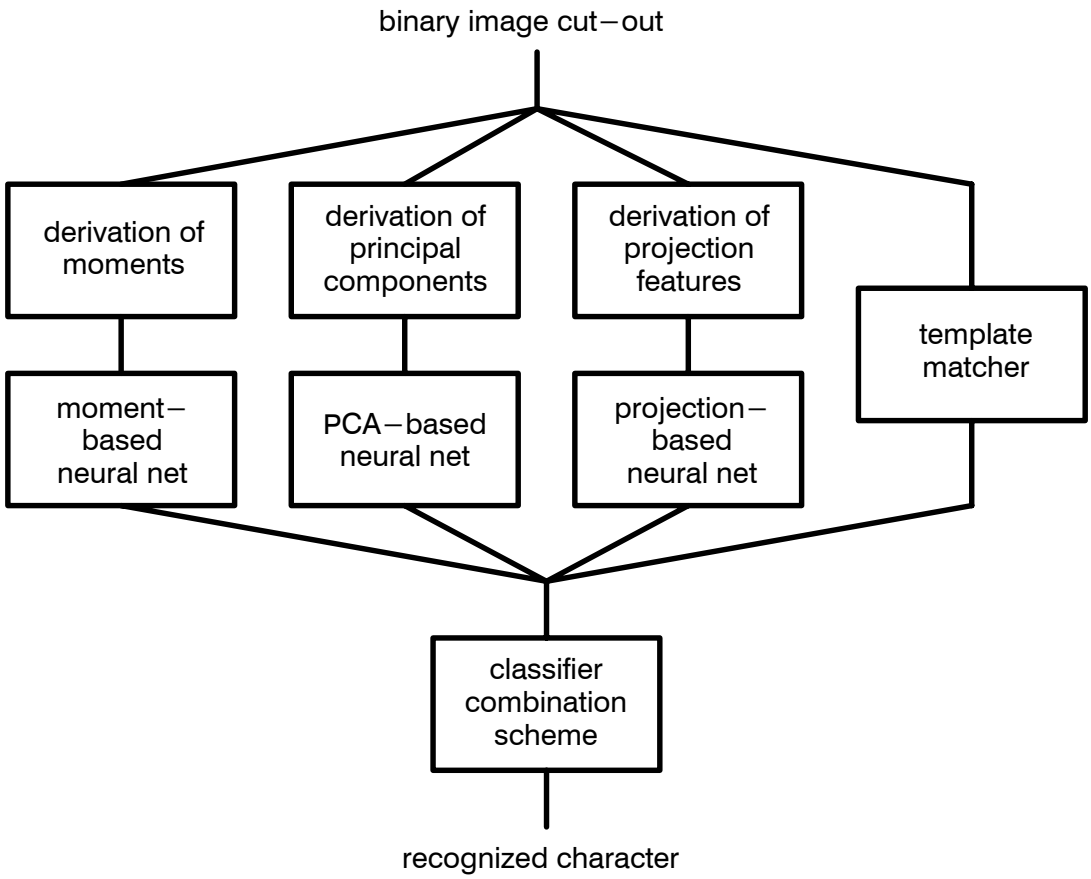

Figure 7.6: Schematic overview of the recognition module. Multiple neural network classifiers (each operating on a different set of input features) and a template matcher, are used in parallel. The sum−based classifier combination rule [60] produces the final classification result.

There are numerous possible strategies of combining the results of the individual classifiers. For a thorough overview the reader is referred to [60]. Based on experimental comparative tests the authors of this article conclude that the sum rule outperforms all other combination schemes. Since classifier combining is not the main issue here, we base our particular choice of the combination rule on their findings. As an example, for the recognition of Swedish characters, the performance curves for the (combination of) individual classifiers is shown in Fig. 7.7. In this graph the percentage of correctly classified patterns is plotted against the percentage of incorrectly classified patterns. It is clear that the combination of classifiers is significantly better than the individual classifier. Finally, the performance curve of the combined classifier allows us to make a trade–off between false acceptance and correct acceptance. The actual recognition rate and error rate are found by picking a point on this curve.

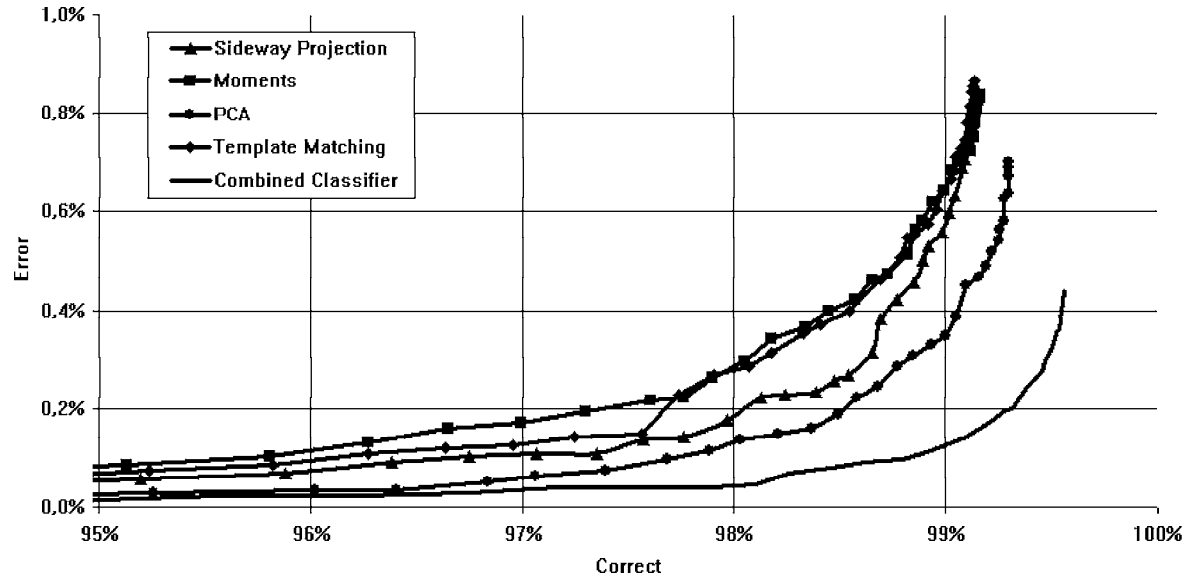

Figure 7.7: Performance curves of the four individual classifiers and the combined classifier. The combined classifier performs better than each of the individual classifiers.

Moments

Moments are numeric characteristics of a distributed set of objects, measured relative to a point of reference. They appear in various areas of research, such as in stochastics and mechanics. In a grayscale image, moments are based on n pixels p_i with a gray level m_i and a distance r_i between p_i and a certain point of reference:

$$M = \sum_{i=1}^{i=n} (m_i \times r_i) \tag{7–1}$$

In other words, a moment gives a measure of the internal structure. If the point of reference is chosen at the heart of the image, the measure is called the central moment. The central moment is invariant to the rotation and sizing of the segmented part. The impact of the different character parts on the moment can be differentiated by raising the mass and distance contributions within the formula to the powers p and q respectively, giving the definition of the (p,q)–order moment:

$$M_{p,q} = \sum_{i=1}^{i=n} (m_i^p \times r_i^q) \tag{7–2}$$

A large range (p,q)–order moments can be generated and therefore it is of interest to find which combination will actually provide compound moments that support the recognition of the different characters [54]. Thorough analysis of different combinations lead us to an optimal set of 16 (p,q)–order moments [15]. This particular set is a selection of the (p,q)–order variants proposed in [91] and [103].

Principal Component Analysis

Principal Component Analysis (PCA) [57][67] is a statistical method that determines an optimal linear transformation $y = Wx$ for a given input–vector x of a stationary stochastic process and a specified dimension m of the output–vector y where W is the desired transformation matrix. In pattern recognition and communication theory, the PCA is known as the Karhunen–Loéve or Hotelling transform. The intention is to compress the information contained in the input data (i.e. $m < n$) by exploiting statistical regularities. In fact, PCA transforms correlated input data into a set of m statistically decorrelated features (or components), usually ordered according in decreasing information content.

The first principal component y_1 is the normalized linear combination of the components of the input data vector. It has on average the largest output variance. Similarly, the second principal component y_2 has on an average the largest variance among the directions orthogonal to the direction of y_1. Then, the third principal component y_3 is taken in the maximum variance direction in the subspace perpendicular to the first two eigenvectors, and so on. Though seemingly computationally elegant, the standard approach that uses a precomputed autocorrelation matrix is impractical for large m. We use an adaptive learning algorithm to derive the 30 first principal components from the input data.

Sideway Projection

A third way to quantify character images is by sideway projection [41]. A straightforward way to accomplish this is by one–dimensional counting of black pixels. This can be viewed as related to texture analysis, where next to this simple pixel–based information one soon takes recourse to runs of similar pixels. An alternative can therefore be the distribution of runs of a specific length in the character image. Most popular is the Connected Component Count, where one counts the number of white/black traversion during the single line scan over the image. We use four different features: horizontal projection, vertical projection, horizontal connected component counts and vertical connected component counts. The features are based on the normalized version of the binary character image. The characters are proportionally scaled to a height of 20 pixels using the nearest neighbor resampling method. Each feature is resampled to a vector of 8 values giving a 32 dimensional feature vector.

Template Matching

Template matching is a technique that evaluates the similarity between an image and a set of reference images called templates. To this purpose, the normalized correlation is often used as the similarity measure. For binary template matching, this measure is identical to the relative number of pixel locations in which the image and the template are identical. The actual classification is mostly done by the k–nearest neighbor method [29].

Template matching has two major problems to tackle. First, template matching is sensitive to image distortions. Though license plates contain printed characters of a well–defined font, such distortions are always present for example due to variations in the viewing angle, diffused light, bent plates, screws and bolts that distort the view, and a limited image resolution. In other words, the actual characters are not only translated, scaled and rotated in the view, but also distorted and of low precision. One way to deal with such variations is not to use only the ideal templates, but also a large number of distorted versions. Such templates are either obtained by generating the

distorted images from the ideal ones by means of distortion models or extracting the templates from real–life data. We have chosen for the latter option, using approximately 100 samples of each character (some of them are shown in Fig 7.8). The samples are generated by subsequently applying the plate segmenter and character isolator to a set of example input images and storing the extracted binary character images.

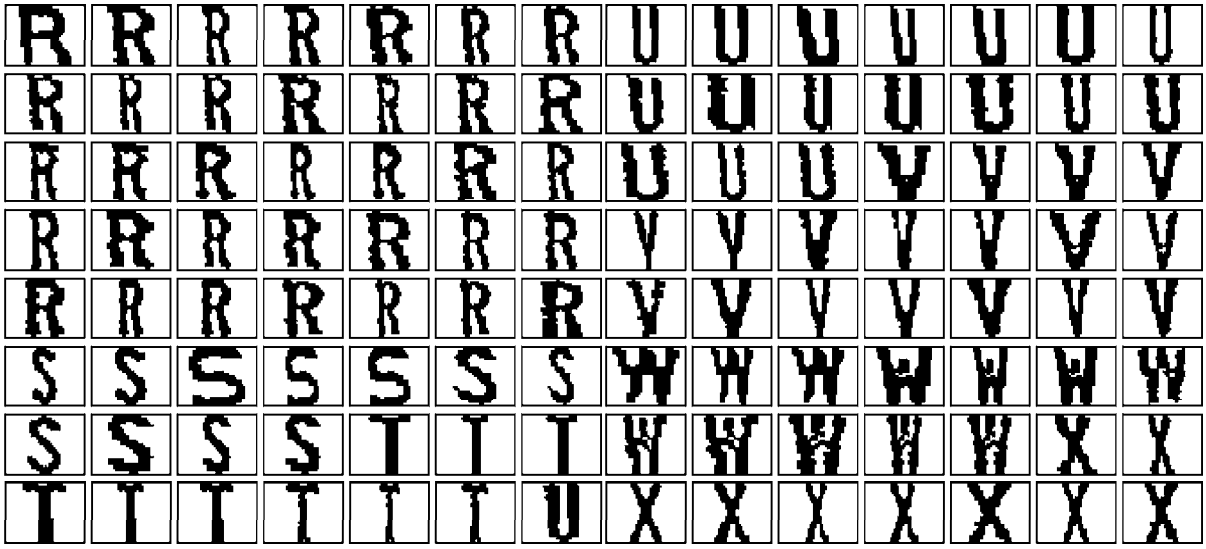

Figure 7.8: Some examples of templates taken from real–life data.

This evidently implies a second problem. To recognize an image, it needs to be correlated with all templates in the reference set. The computational costs are high, since each of such correlations requires a number of bit comparisons that is equivalent to the number of pixels in the image/template. We solve this problem by an incremental evaluation strategy. The strategy is based on the observation that strongly correlated templates will produce almost identical match scores. For a pair of templates and an image to be matched, the difference in match score is determined completely by the image pixels in which both templates differ. By an off–line analysis of the template set (which is static), an optimal incremental derivation scheme can be determined. This scheme is used at run–time to determine whether the match score for a certain template should be evaluated from scratch (traditional correlation evaluation) or incrementally (use the match score of an already evaluated template and update this score by investigating the image positions in which both templates differ). After evaluating all match scores, the actual classification is done by selecting the character with the highest occurrence in the 9 best matching templates (k–NN, with k=9).

7.5 Syntactical and Geometrical Analysis

In the last step of the CLPR system the results of the character recognition module are checked versus some syntactical and geometrical rules. The syntactical rules detect illegal combinations of numerical and alpha–numerical characters. For some countries the syntactical rules are very simple. For example, Brazilian license plates start with three letters followed by three or four digits. The letter group and digit group is separated by a horizontal bar. Other countries like for exam-

ple Germany have very complicated syntactical rules. This is the reason why syntactical analysis is performed by a generic syntactical engine that operates on a so–called syntax specification file. For Canada, an example of such a file is shown Fig. 7.9.

```
CDN_PLATES    >   [ROOT][COUNTRY=CDN]{provinces};
provinces     >   {Ontario};

Ontario       >   {O_passenger} | {O_commercial} | {O_trailer};
O_passenger   >   {O_p_1997_cur} | {O_p_1986_1997} | {O_pre_1986};
O_p_1997_cur  >   A {L}{L}{L}{crown}{D}{D}{D} |
                  B {L}{L}{L}{crown}{D}{D}{D};
O_p_1986_1997 >   {D}{D}{D}{crown}{L}{L}{L};
O_pre_1986    >   {L}{L}{L}{crown}{D}{D}{D};

O_commercial  >   {L}{L}{D}{crown}{D}{D}{D} |
                                {D}{D}{D}{crown}{D}{L}{L};
O_trailer     >   {L}{D}{D}{crown}{D}{D}{D} |
                  {D}{D}{D}{crown}{D}{D}{L};

L             >   A|B|C|D|E|F|G|H|J|K|L|M|N|O|P|Q|R|S|T|V|W|X|Y|Z;
D             >   0|1|2|3|4|5|6|7|8|9;
crown         >   [SEPARATOR];
```

Figure 7.9: Syntax specification file for Canada.

Besides the syntactical rules, each candidate license number should also satisfy a set of geometrical rules in order to be marked as recognized. The geometrical rules take into account the spatial requirements of the characters as defined by the official guidelines. For example, for Dutch license plates a specific left and right margin is defined for each character and the separating bar. The space between a pair of consecutive characters on the plate is identical to the sum of the right margin of the first character and left margin of the second character. Other countries like for example France have less strict rules and allow variable spacing between characters.

7.6 Conclusions

In this chapter the theory introduced in the previous chapters has been applied to the development of a Car License Plate Recognition (CLPR) system. We have shown that DT–CNNs can be successfully used within the CLPR. Their parallel processing capabilities present the means for a very fast image data reduction so that more time can be spent on the close inspection of image regions that actually contain useful information. The system parts that are implemented by DT–CNNs are the parts in the beginning of the recognition chain (segmenter an isolator). The spatially invariant character and the large amount of information that has to be analyzed make these parts the obvious candidates for the introduction of parallelism. The system contains several DT–CNNs whose templates are partly constructed by combining the appropriate morphological operations and traditional filter techniques. This approach of designing DT–CNN templates by algebraic construction rather than by solving large sets of linear equations, allows high–level image recognition tasks like 'finding a license plate' to be easily expressed in terms of DT–CNNs.

The material presented in this chapter is a summary of the research that has been previously published by the author of this dissertation supplemented by the work that has been done by DACO-

LIAN B.V. (of which the author is one of the founders). The company has invested over 20 man–years of research to develop the Intrada ALPR library. Currently, this library is capable of reading license plates from more than 30 countries under extreme external imaging conditions. Independent tests have identified Intrada ALPR as one of the best available license plate recognition engines in the world (see for example [10]). Typical recognition rates are 95% with an error rate of less than 1%. Nowadays, instances of the library are used world–wide in application areas like law enforcement, free–flow tolling and parking applications.