PC Based Number Plate Recognition System

Charl Coetzee, *Member, IEEE*, Charl Botha and David Weber

Abstract— **A PC based number plate recognition system is presented. Digital gray-level images of cars are thresholded using the Niblack algorithm, which was found to outperform all binarization techniques previously used in similar systems. A simple yet highly effective rule-based algorithm detects the position and size of number plates. Characters are segmented from the thresholded plate using blob-colouring, and passed as 15x15 pixel bitmaps to a neural network based optical character recognition (OCR) system. A novel dimension reduction technique reduces the neural network inputs from 225 to 50 features. Six small networks in parallel are used, each recognising six characters.**

The system can recognize single and double line plates under varying lighting conditions and slight rotation. Successful recognition of complete registration plates is about 86.1%.

Keywords— **number plate recognition, optical character recognition, neural networks, image processing, thresholding, dimension reduction**

I. Introduction

The automatic detection and recognition of car number plates has become an important application of artificial vision systems. The object is to develop a system whereby cars passing a certain point are digitally photographed, and then identified electronically by locating the number plate in the image, segmenting the characters from the located plate and then recognizing them.

Some applications for a number plate recognition system are:

- Traffic flow measurement and planning (origin / destination surveys)
- Tracking stolen vehicles
- Control and security at tolling areas, e.g. parking garages
- Traffic law enforcement (automatically identifying speeders, illegal parking, etc.)

The system could also be adapted for use in reading e.g.:

- Warehouse box stencil codes
- Train rolling stock codes
- Aircraft tailcodes

This system consists of two high-level stages: In the first stage, the number plate is detected and segmented from a digital image of the car being examined. This is documented in section II of this paper. The plate location and size are passed to the optical character recognition (OCR) subsystem, documented in section III. Experimental results are shown in section IV.

Images were acquired under varying lighting conditions — no special lighting was used.

The authors are with the Department of Electrical and Electronic Engineering, University of Stellenbosch, Private Bag X1, Matieland 7602, South Africa. Email: *weber@espresso.ee.sun.ac.za*

II. License Plate Detection and Segmentation (PlateFinder)

Figure 1 shows a high level block diagram of the platefinder algorithm's working. In the following sections we document each of the functional blocks in this figure.

A. Image Pre-processing

The first step is to threshold the image to discount the gray-level information. Thresholding results in a binary image which facilitates rule-based pixel-oriented image processing.

The Niblack Binarization algorithm [1] was selected as it provided robust thresholding in the presence of shadows and other image defects. It also provides and excellent choice of input for the OCR stage (section III).

The algorithm calculates a local binarization threshold by calculating the local standard deviation and mean and then adding the mean to the product of a predefined weight constant and the standard deviation:

$$T(x,y) = w \times \sigma(x,y) + \mu(x,y),$$

where T is the threshold at pixel (x,y), w is the weight, and $\sigma(x,y)$ and $\mu(x,y)$ are the standard deviation and mean of the local neighbourhood of pixel (x,y) respectively. This makes for a threshold that intelligently adapts to its pixel neighbourhood. The neighbourhood used was 15 by 15 pixels.

B. Digit location

Entities that could possibly be alphanumerics are located by this stage. The logic iterates through all the pixels in an image and checks at every iteration, using a set of rules, whether there's a candidate digit at the current position. This consists of the two following steps:

B.1 Adaptive size bounding box searching

Previous work [2] used expected alphanumeric size as a criterium for a candidate digit. This is a fair assumption, if we take into account that the distance between camera and car stays fairly constant and assume that the true size of the digits is also more or less constant, or within a set range. This assumption is used by the platefinder algorithm.

The "adaptive bounding box searching" can be further divided into three sub-stages (figure 2):

1. An upside-down "L"-shaped construct of predefined dimensions is moved through the image. At every pixel position the following checks are done:

(a) If there are pixels on the vertical or horizontal lines of the upside-down "L", there is no digit at this position.

(b) If there are no "on" pixels directly adjacent to the top bar *and* the left bar of the "L", there is no digit at this position.

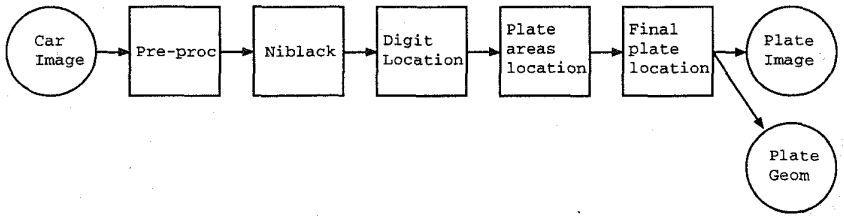

Fig. 1. High level block-diagram of the platefinder algorithm

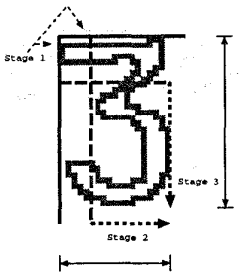

Fig. 2. Illustration of adaptive size bounding box

If a position satisfies these conditions, it may indicate the upper-left corner of a background-bounded object.

2. A second vertical bar is moved, pixel for pixel, to the right, starting at the position to the immediate right of the left vertical line discussed above. This second bar is moved until there are no "on" pixels anywhere on its vertical extent. In this way the width of the detected entity is tentatively calculated and if this width falls outside of a certain range (defined as percentages of the expected width), the position is disqualified. However, this means that a large entity with a narrow extension on its top would still be flagged as a potential digit.

3. In this stage, a second horizontal bar, as wide as the width determined in the above stage, is moved downwards until there are no "on" pixels anywhere on its horizontal extent. If the height determined in this way is within a certain range, again predefined as a percentage variance of the expected digit height, this position is allowed through to the next step.

B.2 Pixel coverage checking

"On" pixels in the calculated area are counted and a coverage percentage is calculated. If this percentage is above 15%, the position is classified as a potential alpha-numeric. If not, it's disqualified.

This coverage rule is based on the assumption that a correctly edge-detected or binarized digit will have a minimum pixel population of the area bounded by its extremities. In this case, an "L" would be expected to have the least percentage coverage.[1] The predefined coverage minimum is set with this in mind—a wide safety margin is built-in.

[1]Note that an "I" would have very good coverage, seeing that the bounding box would size so as to be completely filled by the "I".

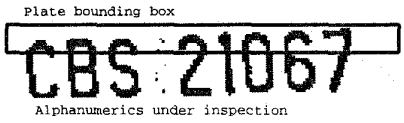

Fig. 4. Illustration of working of plate bounding box

C. Plate areas location

The logic of this stage uses the positions of the digit-like entities detected in the previous stage to attempt determining the geometries (i.e. location and size) of all candidate plate areas (figure 3).

A bounding box of the width of an average digit-width wider than the widest expected plate size and height of a few pixels is used in this step. Only the upper-left corner of a digit has to fit inside the digit-box for this digit to be tagged as part of the plate area. It is then logical that the height of the plate bounding box will determine the rotation invariance of this detection stage. Figure 4 illustrates this concept. If the height is chosen too great, too much vertical variance is allowed, and invalid plate areas will easily slip through. If the height is chosen too small, plates that are somewhat rotated due to a imperfect camera setup or poor capture will not be detected. Experimentation favoured a rule-of-thumb value of half the height of the expected digit[2]. Also, because only the upper-left "corners" of the digit have to be bounded by the box, more width-variance than immediately thought is budgeted for.

The algorithm then starts with the first detected digit as rendered by the previous stage, and places the bounding box's upper-left corner on that digit's upper-left corner position. The bounding box is horizontally shrunk if too close to the far right edge of the image. All other digits in the detected digit list are then checked for inclusion in the region bounded by the plate box. If there's only the one digit in the bounding box, it is obviously disqualified. A candidate area with only two digits is *not* disqualified, as this might constitute part of a two-line number plate. The true width of this candidate plate is then calculated by subtracting the coordinate of the leftmost digit from the coordinate of the rightmost digit and adding the width of the rightmost digit. The maximum reasonable width of the candidate plate is then calculated by adding all the widths

[2]This implies that if a plate's rotation causes a height difference between the start and end of the candidate plate of less than half the expected digit height, the plate will still be detected

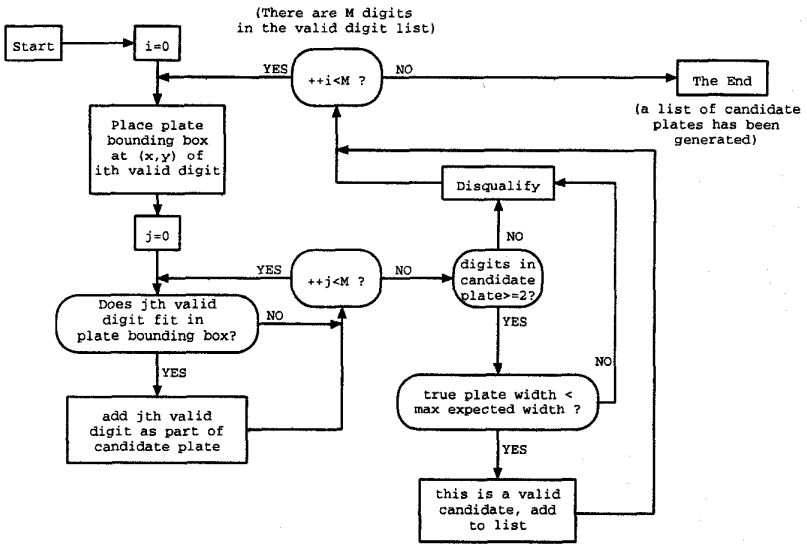

Fig. 3. Flow chart of plate areas location stage (first stage)

of the constituent digits and multiplying by a preset constant. If the true width of the plate area is more than the maximum reasonable width, the candidate is also disqualified. The preset constant used to calculate the maximum reasonable width of a candidate plate has been chosen as two, which yields good results. Choosing this value too small disqualifies valid candidates with digits that are further apart than the norm, or candidates with digits that weren't detected by the digit location logic. Choosing it too high simply lets too many invalid plate candidates through.

The process in the paragraph above is repeated for every digit in the located digits list, and all the candidate plates' coordinates and dimensions are stored in another list.

D. Final plate location

This stage iterates through all the candidate plate areas and eliminates all children, i.e. any plates that are spatially totally contained by other plates in the list.

We find that after this stage that there is often only one plate candidate left, in which case this is obviously returned as the true plate location and dimensions.

In the case of more than one candidate plate area, the plate with the most digits is chosen as a the dominant candidate, a step that was justified by experimental data. The other plate areas are then examined for possibly being an extra number plate line above or below the dominant candidate. If a candidate falls within the horizontal range of the dominant plate's width, and is close enough to the bottom or top of the dominant plate, it is "fused" with the plate, i.e. logic calculates new coordinates and dimensions for a two-line number plate. This new width is obviously the true width of the widest number plate line, the new column coordinate is the column coordinate of the leftmost digit in the new "fused" set, and the new row coordinate is the row coordinate of the topmost digit in the "fused" set.

If fusion takes place, the fused plate is deduced to be the true number plate (figure 5). If no fusion takes place, and the dominant plate candidate has more than two digits, it's seen to be the true plate, else it's disqualified and the logic deduces that there are no detectable plates present in the image.

Fig. 5. Example of segmented two-line number plate

90% of plates are located successfully (97.2% if pathological cases, i.e. obscured or missing plates, are discarded).

III. Optical Character Recognition (OCR)

Once the plate has been located, it is subjected to separate processing, as illustrated in figure 6. The thresholded plate is extracted and inverted if necessary: In all cases white letters on a black background are presented to the segmentation algorithm.

The individual characters are then extracted from the image using a fast implementation of the blob colouring algorithm. [3]

Each individual character bitmap is passed to the neural network classifier to be identified. A measure of classification confidence is also generated.

A. Thresholding

The thresholding done on the full car image (to detect the plate) is not necessarily sufficient for good OCR. Hence the OCR research used only the coordinates of the plate, started with the gray-scale plate image, and determined suitable thresholding.

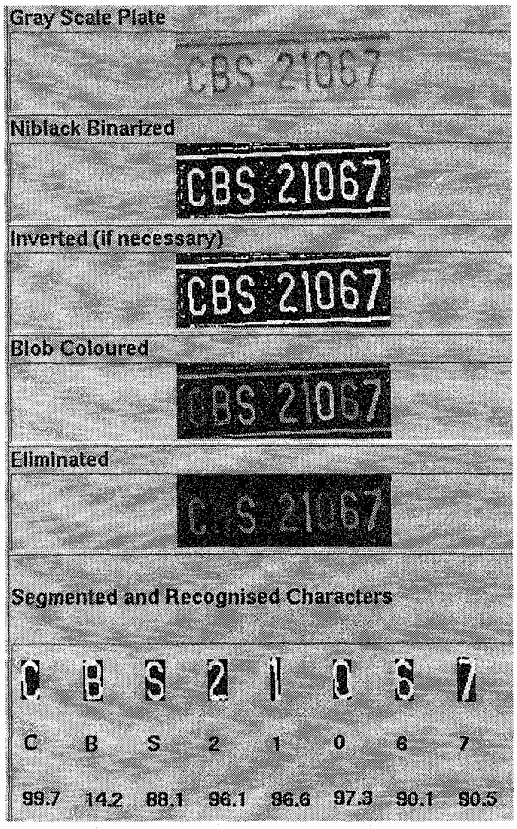

Fig. 6. OCR processing stages

The thresholding procedure should introduce as little noise as possible, since subsequent processing steps may be adversely affected by this. Also, because the lighting conditions vary widely over a plate, locally adaptive thresholding is required. The area over which the algorithm adapts is chosen to be small, yet large enough always to contain both plate background and part of a character.

Eleven adaptive thresholding algorithms from popular literature were tested using [1]. The Niblack algorithm gave the best overall performance.

Fortunately, the this algorithm also functions the best for the plate-detection part of the system. The thresholded result from the plate detector can be used directly for the OCR.

B. Character Extraction (Segmentation)

To extract the individual characters from the plate, various segmentation algorithms were tested. The traditional peak-valley method (0-degree radon transform) performed unsatisfactorily — no precise method for determining breaks between characters could be found.

Our approach was rather to analyse contiguous regions. Number plates use only uppercase letters and numbers, all of which are contiguous (unlike the letter "i"). In general these remain contiguous after thresholding.

Blob Colouring [3] is used. This is a region-growing method which operates on binary images. It labels pixels which form 8-connected contiguous regions ("blobs"), each region receiving a unique label. It thus "colours" the "blobs".

To discard artifacts, bounding boxes for each blob are calculated and a set of heuristics applied. Too small or large blobs are discarded. This eliminates borders, noise blobs, hyphens and the Gauteng registration logo.

Our method has the following advantages:

- Image noise in the vicinity of the character is not extracted with the character.
- Extracted noise blocks can be discarded.
- It automatically processes two-line plates, and plates at slight rotation.

Disadvantages are:

- A character must be continuously connected to be extracted.
- If different characters are connected in some way they will be extracted together.

Each character bitmap is rescaled to 15×15 pixels, maintaining aspect ratio. This is done using split, merge and zero-pad [4]. This helps to make the OCR translation and scale invariant. The OCR networks are in turn trained to be rotation invariant.

C. Classifier

The OCR classifier is neural network based. Unfortunately, the 15x15 bitmap size implies an input space of 225 features. The resulting network is too large for successful training [5][6].

Our classifier solves this problem in two ways: The 225 input dimensions are reduced to 50, and a special network structure is used.

C.1 Binarization Linear Transform (BLT)

We have developed a novel dimension reduction technique for our system. This method reduces the input dimensions by more than is possible with the vanilla Karhunen-Loeve Transform (KLT) (see Table I), yet yields similar performance. Furthermore, no information is discarded.

Since each input pixel is constrained to either "on" or "off" (1 or 0), there are only 2^{225} possible input patterns, which form a set of 2^{225} discrete points in 225-dimensional space.

If sets of the input pixels are grouped (e.g. 0 0 0 1 1 1 0 0), this group can be seen as the binary number 28 by weighting each pixel by the respective power of 2. This entire group of 8 pixels can thus be represented by 1 feature, with a one-to-one correspondence between each original 8-dimensional vector and a unique 1-D number.

Figure 7 demonstrates the concept for an 8 x 8 bitmap. The "1" in the figure would transform to the feature vector [28 60 124 28 28 28 28 28].

It is thus possible to transform the 225-dimensional space into a 50-dimensional space with the input patterns still forming a set of 2^{225} discrete points. These vectors are no longer orthogonal or linearly independent, but since all points are discrete, classes are still completely separable.

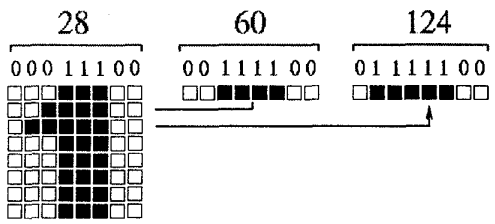

Fig. 7. Example BLT for 8x8 bitmap

This can be implemented by a simple matrix multiplication.

We used the following transform to obtain 50 features: write the 15x15 = 225 pixels as a binary row vector $\mathbf{x}$, then multiply by the 225 × 50 matrix $\mathbf{B}$:

$$\mathbf{T} = [\ x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{225}\ ] \times \mathbf{B}$$

with

$$x_i = \begin{cases} 0 & \text{if pixel is off,} \\ 1 & \text{if pixel is on.} \end{cases}$$

and

$$\mathbf{B} = \begin{bmatrix} 8/15 & 0 & \ldots & 0 & 0 \\ 4/15 & 0 & \ldots & 0 & 0 \\ 2/15 & 0 & \ldots & 0 & 0 \\ 1/15 & 0 & \ldots & 0 & 0 \\ 0 & 16/31 & \ldots & 0 & 0 \\ 0 & 8/31 & \ldots & 0 & 0 \\ 0 & 4/31 & \ldots & 0 & 0 \\ 0 & 2/31 & \ldots & 0 & 0 \\ 0 & 1/31 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 8/15 & 0 \\ 0 & 0 & \ldots & 4/15 & 0 \\ 0 & 0 & \ldots & 2/15 & 0 \\ 0 & 0 & \ldots & 1/15 & 0 \\ 0 & 0 & \ldots & 0 & 16/31 \\ 0 & 0 & \ldots & 0 & 8/31 \\ 0 & 0 & \ldots & 0 & 4/31 \\ 0 & 0 & \ldots & 0 & 2/31 \\ 0 & 0 & \ldots & 0 & 1/31 \end{bmatrix}$$

This transform has the effect of alternately grouping 4 and 5 pixels per feature. This alternate grouping helps to spread any numerical inaccuracies which may arise during the subsequent floating point calculations of the neural network.

Table I shows various experimental recognition rates obtained using the KLT and BLT on small databases. Fifty output features were chosen, since for these database sizes, the KLT will retain 90% of training information. However, as characters are added to the database, this percentage declines rapidly, as does the performance of the KLT.

Since the BLT transformation matrix $\mathbf{B}$ is independent of the training data, this technique is also well-suited to systems utilizing on-line training. By comparison, use of the KLT would entail a recalculation of the transformation matrix for each added training pattern.

TABLE I

BLT VS. KLT FOR REDUCTION OF 225 INPUTS TO 50 FEATURES

No. of characters tested	KLT Correct	BLT Correct
332	87.3 %	88.9 %
378	81.5 %	80.4 %
437	83.1 %	86.3 %
647	83.2 %	83.3 %

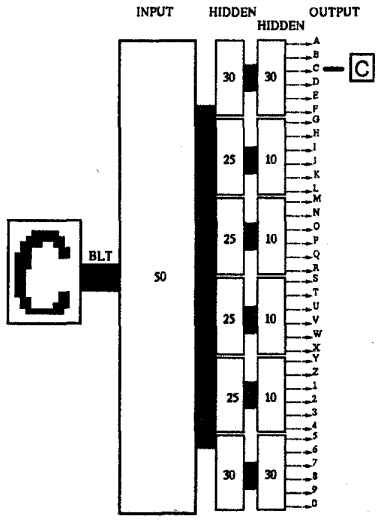

Fig. 8. OCR neural network structure

C.2 Neural Network Structure

The traditional route has been to create a hierarchical multilayer perceptron (HMLP)[7]. However, this network is not easily optimised while experimenting, since this typically entails cumbersome manual changes to the network structure. Since network training is largely guided by trial and error, this is undesirable.

Our approach is to use 6 smaller MLPs, each with 6 outputs, to recognise 6 characters. Each network is trained to produce zero outputs for classes it should not recognise, hence effectively increasing training data 6 times for each network.

The full network structure (figure 8) thus consists of these six subnetworks in parallel (sharing the input layer), it is thus a special case of an HMLP network. Each subnetwork is separately optimised for its own group of characters.

The largest of all 36 outputs is chosen as the classification — i.e. "winner-takes-all" is used. Classification confidence is estimated by taking the difference between the two largest outputs.

IV. Experimental Results

The system was tested with a set of images not used during design and training. The size of the plates in the images were about 190×56 pixels.

The system completed full plate recognition (including location and segmentation) in an average of 1.5 seconds. It is foreseen that this time can be reduced to less than a second if the code is optimized (which was not the case during these tests).

If pathological cases (i.e. plates with overhangs, tow hooks and very bad lighting) were discarded, as in [2] and its references, 86.1% of plates were located and read correctly.

Performance for individual system subsections were as follows:

- 90% of plates were located successfully (97.2% if pathological cases were discarded).
- Characters were segmented correctly from 92.5% of manually extracted plates, 96.7% of these characters were correctly classified and complete manually extracted plates (pathological cases included) were segmented and read with 82.5% accuracy.

V. Conclusions

In practice, system performance is aided by supplying the system with large plate images. This can be achieved by ensuring correct framing and placement of the camera. It may be possible to take multiple images of the vehicle and correlate the system results.

The greatest weakness of the system is the inability of the segmentation-algorithms (both plate location and character extraction) to process correctly characters which are connected to each other or to the border. Nevertheless, the system gives good performance under widely varying lighting conditions.

Results compare very favourably with [2], which lists the most advanced systems known to the authors. These achieve on average 83% recognition of the complete plate, rejecting plates which are subjectively poorly lit. The system in [4] achieves a 97% plate recognition rate by enforcing ideal uniform lighting and framing conditions.

References

[1] D. Trier, "Software and documentation: Xite - binarize," tech. rep., University of Oslo, 1997. ftp://ftp.ifi.uio.no/pub/blab/xite/.

[2] S. Draghici, "A neural network based artificial vision system for license plate recognition," *International Journal of Neural Systems*, vol. 8, pp. 113–12, February 1997.

[3] D. Ballard and C. Brown, *Computer Vision*, pp. 151–152. Prentice Hall, 1982.

[4] C.-M. Hwang and S.-Y. Shu et al, "A pc-based car licence plate reader," *Proc. SPIE*, vol. 1823, pp. 272–283, 1992.

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, pp. 138–229, 363–370. Macmillan, 1994.

[6] D. Hush and B. Horne, "Progress in supervised neural networks," *IEEE Signal Processing Magazine*, vol. 10, pp. 8–39, January 1993.

[7] J. Chang and N. C. Griswold, "A hierarchical multilayer perceptron neural network for the recognition of the automobile licence plate number," *Proc. SPIE*, vol. 2664, pp. 138–144, 1996.

[8] SABS, "Retro-reflective registration plates, South African Standard SABS 1116-1 to 1116-4," tech. rep., SABS, 1996.

[9] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.