Identity verification using computer vision for automatic garage door opening

Identity Verification using Computer Vision for Automatic Garage Door Opening

Rob G. J. Wijnhoven, *Member, IEEE* and Peter H. N. de With, *Fellow*, IEEE

Abstract — *We present a novel system for automatic identification of vehicles as part of an intelligent access control system for a garage entrance. Using a camera in the door, cars are detected and matched to the database of authenticated cars. Once a car is detected, License Plate Recognition (LPR) is applied using character detection and recognition. The found license plate number is matched with the database of authenticated plates. If the car is allowed access, the door will open automatically. The recognition of both cars and characters (LPR) is performed using state-of-the-art shape descriptors and a linear classifier. Experiments have revealed that 90% of all cars are correctly authenticated from a single image only. Analysis of the computational complexity shows that an embedded implementation allows user authentication within approximately 300ms, which is well within the application constraints.*

Index Terms — Garage door opening, computer vision, license plate recognition, object detection, histogram of oriented gradients.

I. INTRODUCTION

Automatic garage door opening is typically implemented by use of a radio receiver and transmitter. When arriving in the neighborhood of the garage, the user presses a button on the transmitter and a radio signal is sent to the receiver inside the garage. The receiver then verifies the signal and opens the garage door. Multiple persons can make use of this system by configuring multiple radio transmitters.

The use of such a system poses a security issue, which evolves from many possible situations that finally lead to unauthorized access. For example, the radio transmitter can be stolen or lost, the radio code can be captured and reproduced with a specialized receiver, or codes can be tested within a certain neighborhood to find matching locations. In all cases, the total system needs to be reconfigured. Any automatic access control system that is currently on the market uses only a radio signal for verification of the user. Visual checking of the identity of the car is lacking in these systems.

In this paper, we propose a system that works completely without a radiographic signal and can be used as a complementary verification stage to the existing access control systems. The additional part that we propose uses computer vision technology for the verification of the user's vehicle and is operating as follows. A camera is installed in the garage, having a clear view of the driveway. Object recognition technology is used to detect and verify the user's car. First, a car detector is used to detect the presence of any car in front of the door. As soon as a car is detected, a description of the car is generated and this description is matched to the database of allowed vehicles for verification. In addition, a second verification stage is applied. The car license plate is recognized and matched to the set of allowed plates. If both the car and the license plate are authenticated, the garage door will be opened. For additional security, an alarm may be generated as soon as someone with a non-authorized car, but with the same plates appears.

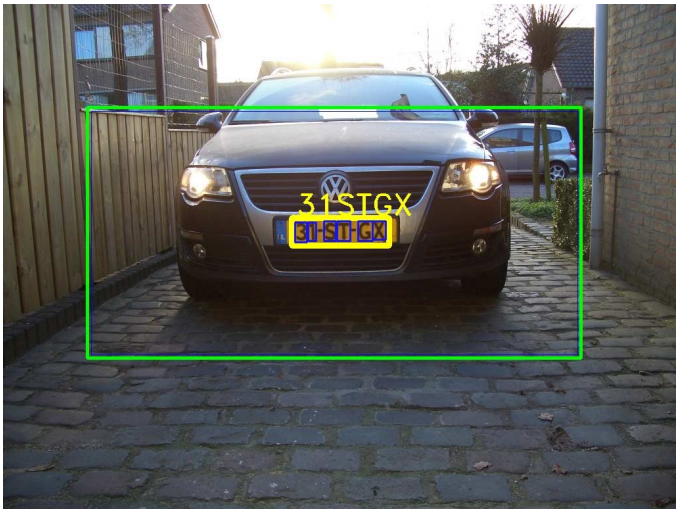


Fig. 1. The proposed system applies visual verification of the user's car.

The above described system has a basis in the generic detection and recognition of visual objects. We show that it is possible to use the same algorithms for both the detection of cars and the recognition of characters. Although the character recognition is not optimized for license plates and can recognize any font or size, the recognition performance for plates is excellent. Our approach does not require a special License Plate Recognition (LPR) camera, although this can be advantageous in cases with extreme glare from the sun. In contrast to typical LPR systems, we embed a car detector to avoid authentication when someone just makes a copy of the user's license plate and holds this for the camera. Moreover, the car should match the description as stored in the system for additional security.

R. G. J. Wijnhoven is with ViNotion, Eindhoven, 5600 CH, The Netherlands (email: rob.wijnhoven@vinotion.nl)

P. H. N. de With is with the University of Technology Eindhoven, EE Dept., 5600MB Eindhoven, The Netherlands

Contributed Paper
Manuscript received 04/15/11
Current version published 06/27/11
Electronic version published 06/27/11.

Because the recognition algorithm should be implemented as an embedded system, computational complexity is an important design criterion. Recognition should be applied to images captured by a low-cost camera. Moreover, because the system should also function reliably during low-light conditions (at night), color information cannot be used. The system should detect and authenticate the car upon entering the driveway and approaching the door. This results in a time-requirement of only a few seconds for the recognition process. In addition to constraints related to cost and processing time, the system should be user friendly, and require minimal user interaction. The training of the system only involves the placement of the user car in front of the camera and triggering the system to take an image of the car. A description of the car and the license plate are automatically generated and stored in the database, which is sufficient for authentication during normal operation. If the car does not have a license plate at the front (USA), the camera can be installed at the start of the driveway.

II. Related Work

Commercial systems for License Plate Recognition (LPR) typically use infrared cameras and filter the specific frequency band in which the license plate has strong reflections (infrared). While this significantly simplifies the analysis process, it results in a higher system cost. To address this concern, we aim at the use of an inexpensive consumer camera that captures visible light only.

Most algorithms in literature deal with the recognition problem in two separate stages: license plate detection for finding the location of the plate, and license plate recognition for extracting and recognizing the individual characters. An extensive algorithmic overview of both processing stages is discussed in [1].

Detection of the license plate is typically realized using edge detection or color segmentation, with additional geometrical filtering [2]. Other approaches scan the image with a classifier that responds strongly to license plate regions. A covariance descriptor in combination with a neural network is proposed in [3] and a similar approach using Haar-like features is used in [4]. Another approach uses the MSER region detector to detect both license plates regions and individual characters [5].

When a plate is detected, it is typically normalized to orientation/skew, size and brightness/contrast. Next, characters are segmented and recognized. Finally, some (country-specific) syntactical rules are applied. In our application, plate distortion is limited because cars always drive in a perpendicular direction towards the camera so that normalization is not required.

In order to recognize characters, the plate image is binarized and connected components are extracted [1]. Classification of each individual character is performed using a neural network. In [6], segmented characters are size-normalized and depending on the number of holes, a different classification is performed, using a Self-Organizing Map (SOM). Pre-segmented characters are described using a Gabor filter bank [7] and recognized by comparing them with a clustered set of template characters.

The use of a separate plate detector requires the license plate to be detected prior to recognition. The placement of the camera can cause part of the plate to be occluded, resulting in a missed plate detection. In contrast to a separate plate detector, we detect license plates indirectly, as a result of the detection of individual text characters. Because we do not encode specific features of license plates, other text strings in the image will also be detected. However, since characters are only located within the detected car regions, the only detected text will be from license plates.

The aim of the system design is therefore to detect cars appropriately, such that a further plate recognition is enabled. This implies the use of two recognition systems, where the efficiency and cost constraint of an embedded system lead to the consideration of reusing the same detection system for two purposes: car and text recognition. This is one of the important research questions that we investigate in this paper. Another question to be addressed is the robust detection and recognition of characters such that the system can be used in different countries, despite the international differences in plate layouts and character fonts.

In Section III, we describe the complete system in the form of a block diagram. Section IV discusses the recognition stages, addressing first the car detection and then the character detection and recognition. Section 0 presents experimental results for both stages. Section V concludes the paper.

III. System Overview

An overview of the system is depicted in Fig. 2. The traditional radio system is shown at the bottom of the figure (red), while the proposed additional computer vision system involves the major upper part of the figure (green). The system works as follows. First, video is captured by the "camera" in the garage door. "Car detection" is applied in each video frame. Once a car is detected, a description of the car is extracted and sent to the "car verification" block. In conjunction, the location of the car is sent to the "plate recognition" block. This block localizes and recognizes the license plate of the car. Once a plate is found, the characters are sent to the "plate verification" block.

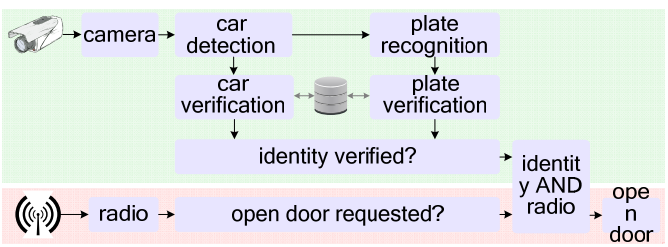


Fig. 2. Overview of the processing blocks in the system.

Once both verification modules send out a signal of verification, the "identity verified?" block checks if the car matches the plate. The user has now been fully verified and if a radio request has been made by the user, the garage door will be opened.

IV. Detection and Recognition

The recognition part of the system consists of two different sub-systems that both apply *detection*, and *recognition* for verification. *Detection* comprises finding any object part of an object class, where *recognition* consists of deciding which specific object has been detected. This recognized object is then matched to the database of known objects to support a possible granted access. This process is called *verification*.

Multiple cars might be authorized by the system, for example, when the system is deployed in a private parking, where multiple cars are parked. Therefore, the system should detect all objects in the category *car* and not be limited to the detection of the specific user car.

For the application considered in this paper, we detect cars and should verify the car type by comparing it to the database without classifying the brand and color, etc. In this paper, we discuss the detection of cars and license plates. Additionally, we design a plate verification step that recognizes the text on the plate. However, we will not describe the verification stage of the car and leave that for future work. In literature, for car verification, a technique is proposed in [4].

When a car is detected, text is detected within the region of the car. A car detector will detect any type of car, which can be the user's car or any other car. As soon as a car is detected, a description of the car is extracted and this description is verified with the database of authorized cars. Second, for the category "text", detection will return any object in the image that looks like a text character ([0-9][A-Z]). Recognition is then applied to classify each found character (compare with the alphabet). In a second stage, groups of characters are extracted and the groups matching the constraints of license plates are used for verification.

Let us first discuss the object detection system and visual features used for detection and recognition.

A. Car Detection

1) Object Detection using Shape Features

Models for object detection are based on describing appearance and spatial information of the objects. Bag-of-Words models only use appearance information [8] and model an image as a histogram of the appearance of *visual words*. A visual word is a description of a small part of the image (an image patch). These models do not consider the location of the found words and are therefore robust to deformation of the object. Implicit Shape Models (ISM) [9] also use a dictionary of visual features and store the position of the feature with respect to the center of the object as additional spatial information. Detection of objects is applied by finding the independent visual words and casting a vote for the center of the object. The object votes for all visual words are merged and each maximum represents a detected object. Sliding window classifiers [10] encode appearance information on a regular grid, thereby implicitly modeling spatial information. In contrast to bag-of-words models or ISMs, no dictionary of visual features is used, but the fixed-size search window is treated as a single, high-dimensional feature.

We consider the detection system as proposed by Dalal and Triggs [10] because of its simplicity and good performance. In this approach, a window is shifted over the image and each position in the image is classified into object or background. Because the size of the object to detect is not known in advance, the image is scanned at different scales by resizing the image to different resolutions. A visual interpretation of the detection process is shown in Fig. 3.

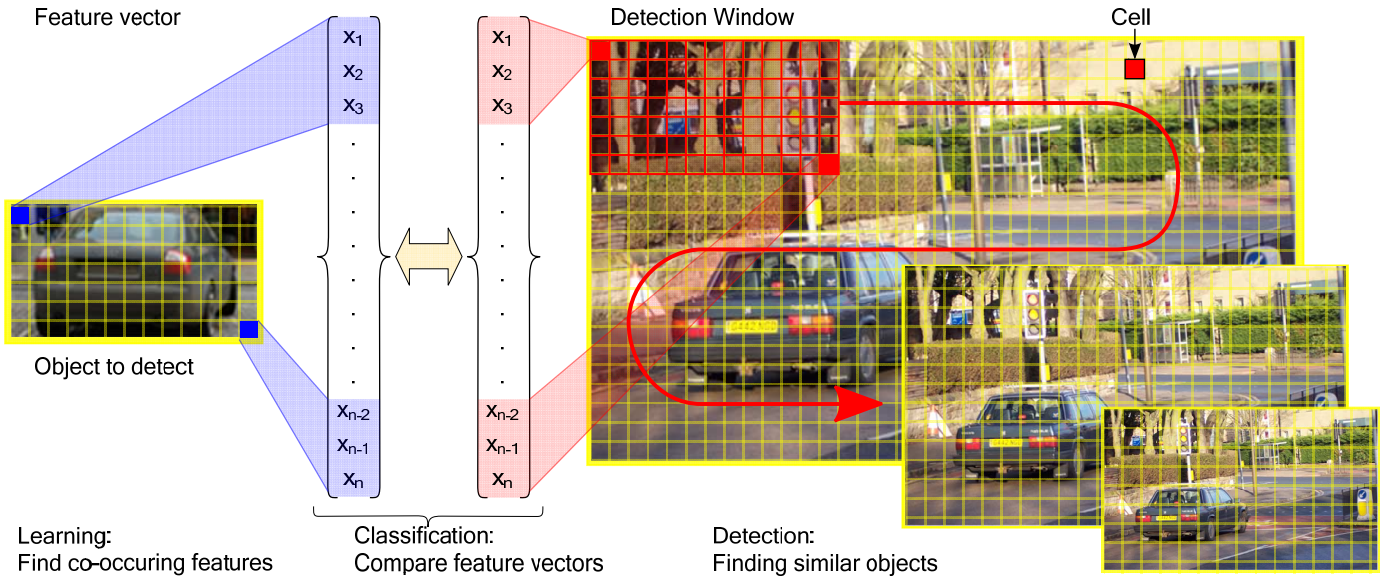


Fig. 3. Object detection using a sliding classification window at different scales.

Fig. 4. Histogram of Oriented Gradients (HOG) feature calculation.

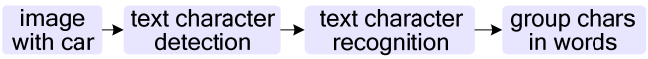


Fig. 5. Text character detection and recognition block diagram.

We will now discuss the HOG technique in more detail. The overall block diagram is shown in
Fig. 4. First, the image is processed with a gradient filter. A simple 1×3 filter with coefficients [1,0,-1] is applied in both horizontal and vertical direction. The gradient direction is then computed by the ratio-strength of both dimensions. The gradient orientation is quantized into bins, and the corresponding orientation bin is updated with the gradient magnitude. Input pixels are spatially quantized into cells of n×n pixels, where n is the cell size. Each cell results in one orientation histogram. To allow for small spatial and orientation shifts, linear interpolation is used in both the two spatial and the gradient orientation dimension. Invariance to the absolute luminance level is obtained by use of the gradient operator. To be invariant to contrast variations, the orientation histograms are normalized. Dalal and Triggs introduced the concept of overlapping blocks. Each block is a group of b×b cells and a cell is part of multiple blocks (typically, b=2, number of blocks B=b×b). Each cell is specifically normalized for each of the block configurations that it is part of. The description of a cell is the concatenation of all B normalized versions of the cell's orientation histogram. The feature vector describing the total image window is the concatenation of all cells' descriptions.

1) Classification

Object detection is obtained by sliding a window over the image and classifying the local description for each position into object or background. To detect objects of different size, the detection process is repeated for scaled versions of the input image. For each position in the (scaled) image, the extracted shape descriptors are compared with the object's shape *template* and the class of the current sliding window position is determined (object/background). For object detection, this comparison is called *classification*.

There are two main options for classifying unknown data. One is a direct comparison of a test sample with the training samples (non-parametric). The other is generating a model from the training data and comparing each test sample with the model. Nearest neighbor classification is an example of a classification method of the first type. The algorithm finds the training sample with smallest distance to the test sample and returns the class to which that sample belongs (object/background). However, for training an object detection system, we typically train the samples of the object class against all other visual patterns: the number of negative training samples is infinite. Therefore, the method of classification by comparing against all training samples is impractical and a classification model should be designed.

Several classification models exist, of which the simplest model is linear classification. Initially, the purpose of classification is to distinguish objects from background samples, where the decision is binary and the exact position with respect to the decision boundary is irrelevant. Since the classifier has to be applied to a very high number of image positions in the detection process, a simple (e.g. linear) classifier is preferred for its computational simplicity. For multi-dimensional signals, linear classification represents a hyperplane and the classification function is a simple dot-product between the test feature vector and the normal of the hyperplane.

More complex classification methods such as kernel machines can be used to give higher classification performance, but come at a significantly higher computational complexity. These kernel methods are an intermediate solution that store the most relevant training samples and use these to implicitly define the decision boundary. The Support Vector Machine (SVM) [11] algorithm is a popular kernel method, for which the computational complexity for evaluation of a feature vector (classification) is linear to the number of support vectors, which involves for a typical classification problem in the order of hundreds to thousands vectors. In contrast, only one comparison is made when using a linear classifier. For this reason, the linear classifier will be used in our experiments, as we have a strong constraint on the computational complexity.

2) Training the classifier

Training a classifier is not straightforward, because we have to deal with a high number of samples, where each sample also has a high dimensionality. During training, we want to find and position the decision boundary between all object samples (positive samples) and all background samples (negative samples). Object samples are clearly defined as annotated training samples (e.g. images of cars). Negative samples are basically anything, except cars. Since this number is infinite by definition, typically a random sampling from images without objects of interest is applied. The feature vectors describing the object and background samples are typically highly dimensional.

The object detection problem is typically highly asymmetric (also in this application), because of the limited variation in object appearance (few samples), compared to the background as this can be anything (many samples). To train a linear classifier for a highly unbalanced detection problem, a Support Vector Machine (SVM) [11] is often used, as it can handle high-dimensional feature vectors effectively. For our experiments, we have used the Stochastic Gradient Descent (SGD) algorithm for SVM training, because it is fast and has high performance [12].

B. Text Detection and Recognition

When a car is detected, the image is forwarded to the LPR module, that extracts the license plate. The algorithm works in three stages. First, individual text characters are detected. Second, the detected characters are recognized: each character is classified into the set of known characters from an alphabet (OCR). Third, the individual characters are merged into words. The found words are sent to the verification module to perform the actual authentication. The system overview is depicted in Fig. 5. In the sequel, these three steps are further detailed.

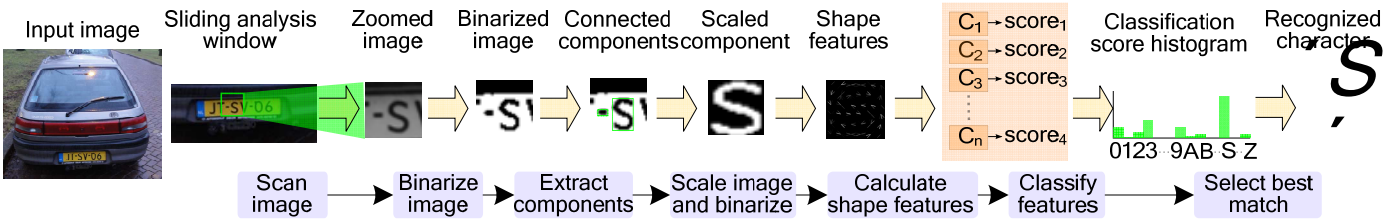


Fig. 6. Text detection and recognition process.

1) Text character detection

In order to detect text characters, we employ three assumptions about the characters to be detected.

– Text and background color are assumed to be uniform
– Text characters are typewritten
– Text characters are spatially separated

The detection process is as follows. We scan the entire image with an overlapping detection window of $n \times n$ pixels. In each position of the window, we make the assumption that there is text inside the window. The image is binarized to obtain a pixel-true segmentation of the character(s) inside the window. Although many binarization techniques are available, we have selected Otzu's method [13] for its high performance. Connected components are extracted from this binary image and when they do not touch the detection window borders and have significant height, their bounding boxes are supplied to the recognition module. Because different international plates have different character sizes, we scan the image at multiple resolutions and merge the associated detections.

2) Text character recognition

When characters have been detected separately, each character is processed by the recognition module. In order to recognize a character, we create a shape description of the binarized character and apply a classifier for each character in the alphabet. Although we use the same shape descriptor as employed for the car detection, we do not need to apply the sliding window detection and apply classification. The class of the character is assigned to the classifier that has the highest score. In the general case of unconstrained text detection with a Roman alphabet, there are 62 classifiers in total: 10 digits, 26 lower-case characters and 26 upper-case characters. For the application of LPR, we are only interested in digits and upper-case characters and train 36 classifiers.

A fundamental difference between text and cars is the *aspect ratio*. Cars have a fixed aspect ratio with only little deviation between different car models (ignoring the viewpoint). This means that cars are very suitable to be described by a shape descriptor with a fixed spatial grid. In contrast, text characters on different plate styles and fonts have a rather varying aspect ratio, which does not enable the selection of a fixed window size. The ratio varies *locally* between different characters, like the characters 'i' and 'w', and *globally* because of the different international styles. Although there are different solutions like scaling, we restrict ourselves here to the most appropriate concepts. An interesting option is to normalize the aspect ratio of each independent character to support independent classification. Because binarization is used in the detection process, each individual character is already segmented. To normalize the aspect ratio, we scale each binarized character to a fixed size, similar to the size of the descriptor window. After scaling, the image is binarized again to obtain a pixel-true segmentation mask at the resolution of the shape descriptor window. The resulting binary image is used as input to generate the shape descriptor. As with the car detection, we use the Histogram of Oriented Gradients (HOG) features [10] to describe the shape of the normalized character. The process of normalizing each character to a fixed size results in an *aspect-ratio invariant* description for recognition.

3) Training Text Recognition

To recognize a detected character, we apply a shape classifier for each known character in the alphabet. To the knowledge of the authors, the only publicly available OCR database that considers text in natural environments is the *Chars74k* dataset [14]. This dataset contains 20k images of English characters, which involves on the average about 320 samples per character. Although this number is sufficient to train a classifier, we have found it impractical for our application because the images contain too much variation in appearance and lack spatial and rotational alignment. Therefore, we create a new dataset of characters without the above limitations in appearance. The annotation of different characters from images is a very time-consuming process. Therefore, we automatically generate training samples from *font files* that are freely available from the Internet. Each font file automatically provides a single training sample for each character in the alphabet. To realize a sufficient amount of variation in appearance, we use 100 different fonts to create our dataset. For enforcing a fixed aspect ratio, each character is normalized to a fixed size of $C \times C$ pixels. A visual representation of the training set used for the experiments is shown in Fig. 7.

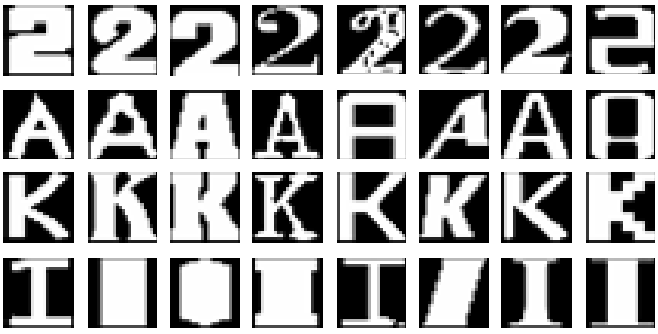


Fig. 7. Example training samples generated from font files for characters '2', 'A', 'K' and 'I'.

After generating the training samples, we calculate the shape features and train a classifier for each character class. The HOG descriptions are generated and a Stochastic Gradient Descent (SGD) algorithm is used to train a linear

classifier. To train each classifier, we use the set of positive training samples (characters) and a set of randomly extracted samples from images without characters.

To classify an unknown character, the character is classified by each classifier. The recognized character class is assigned to the classifier with highest score. Note that there are characters with similar shape, like the '0' (digit zero) and the 'O' (character), and the digit '8' having similarities to '9' and 'B'. To exploit the similarities during the merging of multiple detections at different image resolutions, the scores for each classifier are stored in a *score histogram*.

4) *Text character grouping*

For a certain region in the image, many characters may be detected and recognized. Individual characters are then merged into words, where merging is implemented in two stages. First, characters with similar size and strong spatial overlap are merged together (e.g. two P's are detected in neighboring scales at the same position) and their score histograms are accumulated. Second, we merge multiple detections inside the same character with differences in scale. For example, when the resolution of the image is high, two zeros are detected inside every '8'. Simple merging of these zeros with the '8' can result in erroneous classification after merging. If the smaller character has similarities to the characters '0' (zero) or 'O', it is not further used and only the larger character is evaluated for the recognition. Therefore, small characters inside larger ones are combined using heuristic rules that exploit knowledge of these characters. Let us now briefly discuss how those heuristic rules are found. First, we identify smaller characters that potentially can occur in larger characters. Second, we check the location of the smaller character w.r.t. the larger character. For example, a smaller character can be located at the top of the larger character, like the characters '9', 'g', 'p', 'q', 'P', '8' or 'B'. Similarly, smaller characters can be located at the bottom of the larger character, like the characters '6', 'b', 'd', '8' or 'B' In both cases, the smaller character is combined with the larger character. If the smaller character has similarity to the characters '4' or 'A', they are combined when the larger character also has similarity to the characters '4' or 'A', respectively. After merging the overlapping characters, the peak in the final score histogram denotes the recognized character class.

After merging the overlapping characters, the remaining characters are merged into words. Two characters are merged together, when they are close in both *x*- and *y*-direction, and their aspect ratios are similar. For each merged word, the corresponding text string is stored and forwarded to the verification module for the actual authentication of the license plate.

EXPERIMENTS AND RESULTS

Although several public datasets with cars are available, most sets have only limited resolution of the license plate, which restricts the use for evaluation of the LPR algorithm. Furthermore, no public datasets for the evaluation of license plates recognition are available. In our case, we aim at a recognition experiment in an unconstrained environment (driveways). Therefore, we have created a novel dataset by taking still pictures from a car parking during daytime, with varying light and weather conditions. Cars have been photographed from the front and rear from a distance of 3 to 4 meters using a 10 MPixel consumer photo camera[1]. Each car has been photographed three times, fully from the front/back, and twice under an angle (left and right of car center) of 20 to 30 degrees, measured in the ground plane. The capturing position of the camera is 1.70 meters above ground level. The dataset contains 360 images of cars, each containing a single license plate. Prior to processing with our algorithm, the images are downscaled four times. Examples of the images are shown in Fig. 8. Examples of the total recognition results are shown in Fig. 12.

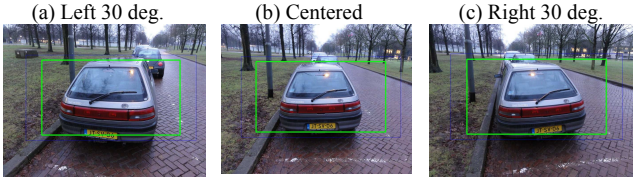
C. *Car Detection*

To train the car detector, we use the Histogram of Oriented Gradients (HOG) [10] algorithm. The following parameters are used: cells of 8×8 pixels, 1×1 block normalizations, 18 orientation bins using the sign, L2 feature normalization and a detector size of 128×64 pixels. The dimensionality of the final feature vector is 2,304.

Our algorithm is trained on the public multi-view car dataset from Kuo and Nevatia [15]. The 2,462 positive car samples are mirrored to obtain 4,924 samples. We apply bootstrapping to obtain negatives from the PASCAL 2007 non-car samples (8,427 images). Stochastic Gradient Descent (SGD) is used to train a linear detector in 10 epochs, using λ=0.001 (a similar training process for a different dataset is described in more detail in [12]).

We have applied the trained car detector to the images of our LPR dataset. Detection is applied by downsampling each image in steps of 1.05. We start at scaling factor 3, because cars have to be close to the camera for successful text recognition and thus have a large pixel size (higher than 192px). Window-level detections are merged using a mean-shift procedure. For evaluation, we use the PASCAL VOC evaluation criteria [16]. Because the cars are close to the camera, the perspective distortion is large and the car aspect ratio changes, causing detections to focus either on the car top or the car bottom. Since we are only interested in the actual detection of the car, we require a 30% overlap in the evaluation of the detection performance (compared to the 50% overlap that is typically used). The recall-precision curve is shown in Fig. 9. Recall represents the percentage of cars in the dataset that are detected and precision represents the number of correct detections. The area under the curve (AUC) for the dataset is 98%.

(a) Left 30 deg. (b) Centered (c) Right 30 deg.



(d) Left 30 deg. (e) Centered (f) Right 30 deg.

Fig. 8. Example car detections on the LPR dataset. All car detection are correct.

[1] Consumer photo camera DSC-HX5V

Our detector localizes all cars with only 10% false detections. Since we use the car detection as a preprocessing stage for our character extraction and recognition, false detections are allowed. Some car detections are depicted in Fig. 8, showing that the car detector finds cars, although there is a variation in both viewpoint and object appearance.

D. License Plate Recognition (LPR)

When a car is detected in the image, characters are extracted. This is performed using a sliding detection window of 32×32 pixels that samples the image with steps of 16 pixels horizontally and 8 pixels vertically. The vertical step size is lower to avoid binarization effects around the borders of the license plates. The image is scaled down in steps of 1.5 to detect the larger characters. For recognition, we again use the Histogram of Oriented Gradients (HOG) descriptor [10] with the following configuration. We use a description/detection window of 24×24 pixels (n=24), cells of 4×4 pixels, blocks of 2×2 cells, 12 orientations including sign and L2 normalization. Because the images have a very low resolution (see Fig. 7), smoothing the image during gradient calculation is desired. Therefore, we applied a 3×3 Sobel filter instead of the more simple 1×3 filter used for the cars. The linear classifier is trained using SGD with 20 epochs and λ=1e-4. For each classifier, we have trained based on the 100 positive training samples, generated from the font files. To generate the negative samples, we have randomly extracted 100 samples from each of 1,000 images without characters, resulting in 100k negative samples. The recognition scores of the characters on the plates are as follows. In total, we have obtained perfect detection and recognition of all characters on the plate for 80% of all plates/images. Some detections are shown in Fig. 10.

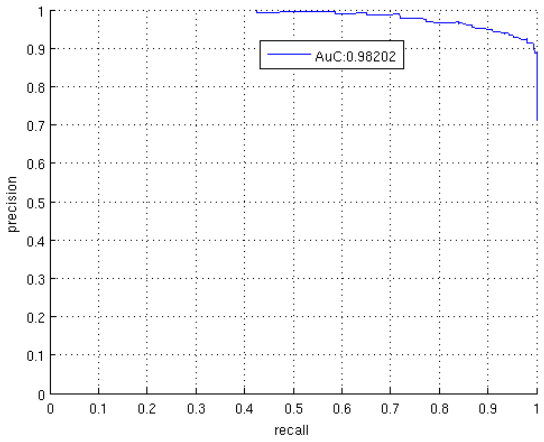


Fig. 9. Recall-precision scores for car detection on the LPR dataset.

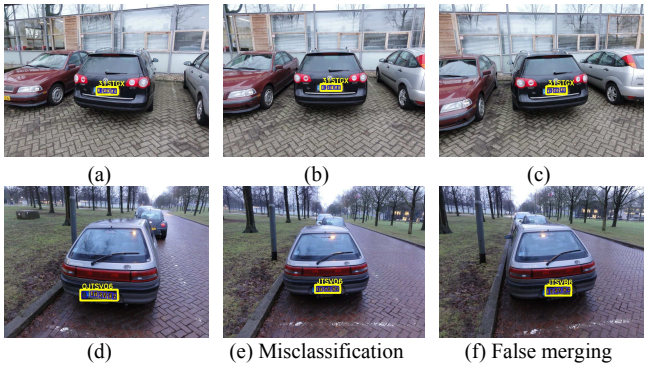


Fig. 10. Example character recognition on the LPR dataset. Detections in (a-c,e) are correct, (d) misclassified character, (f) false merge.

Some extra characters (false positives) are detected on the left/right side of the plate (for an example, see Fig. 10 (d)). These false detections are merged with the correct plate characters in 10% of the images. Although this number is high, the false merges are not a problem for the considered application. For verification, we are only interested if the string of the user's plate is a substring of the detected string. When approving these plates as correct recognitions, we obtain a 90% correct recognition rate. A wrong classification (OCR) is made for 15 characters (out of the total of 2k characters). A total of 48 characters are missed, causing the verification to fail. Only for two plates, additional characters are detected in between the real characters. A histogram representation of the errors made over the total dataset is shown in Fig. 11.

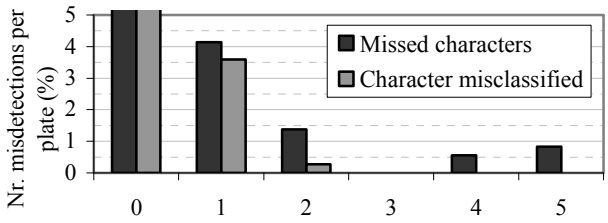


Fig. 11. Histogram of number of character errors per license plate (in percent of number of plates). Black bars represent missed characters, grey bars represent misclassified characters. Values at bin 0 (correct detections) are 90% for both error types.

Most misdetections of separate characters are caused by binarization, leading to incorrect segmentation. Denser scanning of the image improves the binarization performance, but comes at a higher computational cost. A further analysis of the errors has revealed that errors mainly occur in very challenging conditions when the plate is very dirty. For typical images, the detection and recognition performance of the system is almost flawless, which shows the feasibility of the approach for the considered application.

E. False Decisions for Authentication

A critical aspect of any verification system is the analysis of the number of false decisions. *False positives* represent the cars that are incorrectly authenticated, *false negatives* represent cars that have access, but are not authenticated by the system. In the context of the considered application, false positives cause other people to enter the user's garage, and false negatives restrict the user to enter his own garage.

False positives in the car detection are not a problem, because the license plate recognition system still has to identify the plate. If no plate is found, the user will not be verified. *False negatives* (missed car detections) restrict authentication, but as shown in Fig. 9, all cars are detected.

For the character recognition, false positives (extra detected characters) cause false authentication only when they are detected between the actual plate characters, which occurs only in two plates (0.6%). Falsely classified characters cause verification errors, which occurs in 14 plates (3.9%). Characters are missed in 25

plates (6.9%), also causing incorrect verification. In total, 90% of the car plates were correctly authenticated.

These performance measures are obtained with single images only. The real application would benefit from using multiple frames for the analysis. Furthermore, if the computer vision system works as an additional verification stage with the radio-based system, false visual authentication is not critical because a radio signal is also required.

F. Computational Performance

Since we aim at employing this application as a low-cost embedded processing system, the computational complexity of the system is important. We have evaluated the execution times of both the car detection and the character recognition components. On our test system[2], car detection required 42ms and the text recognition needed 22ms executing time. We estimate that the runtime on an embedded processor[3] (using CPU benchmarks) will be approximately five times lower. On the embedded platform, this results in a total processing time for authentication of approximately 300ms, which is well within the application constraints.

V. Conclusions

In this paper, we have investigated the use of computer vision techniques for the design of an automatic garage door opening system. We have proposed a novel technique for license plate recognition, using a single object detection framework with shape features for both the detection of cars and the recognition of individual characters.

The system has been evaluated on a new dataset of 360 cars, captured at a car parking. The car detector is trained with a public dataset and character recognition is trained using artificial character samples, generated from font files. Our system shows that all cars are detected with only a few false detections, which does not influence the authentication performance. License plate recognition has been applied to the image region of the car detections. Experiments have revealed that 90% of all cars are correctly authenticated from a single image only. Main causes for incorrect recognition are very dirty plates.

(a) Example image with car on driveway.

(b) Example image from LPR dataset, used for performance evaluation.

Fig. 12. Correct detections of car box (green rectangle) and recognized text.on the license plate (yellow).

Analysis of the computational complexity shows that an embedded implementation allows user authentication within approximately 300ms, which is well within the application constraints. The high efficiency of the system is explained by the elegant reuse of the feature-based object detection algorithm for both the detection and recognition of cars and text.

[2] Benchmark: Intel Core i7 2600 (3.4GHz), using single thread

[3] Embedded processor: Intel Atom N270 (1.6 GHz)

References

[1] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos and E. Kayafas, "A License Plate-Recognition Algorithm For Intelligent Transportation System Applications", *IEEE Trans. on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 377-392, Sept. 2006

[2] C. D. Nguyen, M. Ardabilian and L. Chen, "Robust Car License Plate Localization Using A Novel Texture Descriptor", *Proc. IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS)*, pp. 523-528, Genova, Italy, Sept. 2009

[3] F. Porikli and T. Kocak, "Robust License Plate Detection Using Covariance Descriptor In A Neural Network Framework", *Proc. IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, Nov. 2006

[4] L. Dlagnekov and S. Belongie, "Recognizing Cars", Technical Report CS2005-083, UCSD CSE, 2005

[5] M. Donoser, C. Arth and H. Bischof, "Detecting, Tracking And Recognizing License Plates", *Lecture Notes in Computer Science, Asian Conference on Computer Vision (ACCV)*, vol. 4844, pp. 447-456, Tokyo, Japan, Nov. 2007

[6] S. Chang, L. Chen, Y. Chung and S. Chen, "Automatic License Plate Recognition", *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 1, pp. 42-53, Mar. 2004

[7] J. J. Weinman, E. Learned-Miller and A. R. Hanson, "Scene Text Recognition Using Similarity And A Lexicon With Sparse Belief Propagation", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 10, pp. 1733-1746, Oct. 2009

[8] G. Csurka, C. R. Dance, L. Fan, J. Willamowski and C. Bray, "Visual Categorization With Bags Of Keypoints", *Proc. European Conference on Computer Vision (ECCV)*, Prague, Czech Republic, May 2004

[9] B. Leibe, A. Leonardis and B. Schiele, "Robust Object Detection With Interleaved Categorization And Segmentation", *International Journal of Computer Vision (IJCV)*, vol. 77, no. 1, pp. 259-289, May 2008

[10] N. Dalal and B. Triggs, "Histogram Of Oriented Gradients For Human Detection", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886-893, June 2005
[11] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, vol. 20, no. 3, pp. 273-297, Sept. 1995
[12] R. G. J. Wijnhoven and P. H. N. de With, "Fast Training Of Object Detection Using Stochastic Gradient Descent", *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, pp. 424-427, Istanbul, Turkey, Aug. 2010
[13] N. Otsu, "A Threshold Selection Method From Gray-Level Histograms", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62-66, Jan. 1979
[14] T. E. de Campos, B. R. Babu and M. Varma, "Character Recognition In Natural Images", *Proc. International. Conference on Computer Vision Theory and Application (VISAPP)*, pp. 273-280, Lisboa, Portugal, Feb. 2009
[15] C. Kuo and R. Nevatia, "Robust Multi-View Car Detection Using Unsupervised Sub-Categorization", *Workshop on Applications of Computer Vision (WACV)*, Snowbird, UT, USA, Dec. 2009
[16] M. Everingham, L. V. Gool, C. K. I. Williams, John Winn and A. Zisserman, The PASCAL Visual Object Classes (VOC) Challenge, *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303-308, Sept. 2010

BIOGRAPHIES

Rob G. J. Wijnhoven (M'06) Rob Wijnhoven graduated in Electrical Engineering from the University of Technology in Eindhoven in 2004. From 2004 to 2009 he worked on object categorization for video surveillance at Bosch Security Systems, Eindhoven, The Netherlands. In 2009, he joined ViNotion, Eindhoven, and is working on object detection in various application fields. Since 2004, he has been active in several related international projects. His interests include pattern recognition and machine learning for computer vision applications and he is currently working towards a PhD degree.

Peter H. N. de With (F'06) graduated in Electrical Engineering from the University of Technology in Eindhoven and received his Ph.D. degree from the University of Technology Delft, The Netherlands in 1992. He joined Philips Research Labs Eindhoven in 1984, where he was until 1993 involved in several European projects on SDTV and HDTV recording. In this period, he contributed as a principal coding expert to the DV standardization for digital camcording. Between 1994-1997, he was leading the design of advanced programmable video architectures as a senior TV systems architect. In 1997, he was appointed as full professor at the University of Mannheim, Germany, at the faculty Computer Engineering, where he was heading the chair on Digital Circuitry and Simulation. Between 2000 and 2007, he was with LogicaCMG (now Logica) in Eindhoven as a principal consultant. In 2008, he joined CycloMedia Technology, The Netherlands, as vice-president for video technology. Since 2000, he is professor at the University of Technology Eindhoven, at the faculty of Electrical Engineering and leading a research group on 3D video and video analysis. He has written and co-authored over 300 papers on video coding, architectures and their realization. He has received several awards for IEEE CES Transactions papers, SPIE papers and company inventions. Mr. de With is a Fellow of the IEEE, program/technical committee member of the IEEE CES, ICIP and VCIP, a regular scientific board member or advisor to various companies, and of the Dutch Imaging school ASCII, and board member of various working groups.