Automatic license plate recognition on microprocessors and custom computing platforms: A review

Princewill Akpojotor Ph.D¹ | Adebayo Adetunmbi¹ | Boniface Alese¹ | Ayodeji Oluwatope²

¹ Department of Computer Science, Federal University of Technology, Akure, Nigeria
² Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria (Email: adejnwale@gmail.com)

Abstract
Automatic license plate recognition (ALPR) is the process of extracting and recognizing character information within a localized license plate region. Typically, ALPR involves three steps: image capture, image procession and plate recognition. The performance of an ALPR is largely dependent on the quality of the captured image, which is determined by factors such as environmental variation, camera quality and occlusion. Image procession and plate recognition step involves image processing techniques that extract and recognizes license plate and characters, respectively. ALPR systems could be realized on microprocessors (software-based) or custom computing platforms (hardware-based). Drawbacks such as portability, power consumption and computational speed limit software-based ALPR for real-time deployment. Custom platforms for ALPR consume less power and achieve high processing speed for real-time capability. However, limited computing resources available within a custom chip make it difficult to implement State-of-the-Art computationally intensive algorithms. Thus, very few literatures discussed ALPR techniques on custom computing platforms. This paper presents a comprehensive review of algorithms and architectures of ALPR on microprocessors and custom computing platforms. Design approaches, performance, gaps, suggestions and trends are discussed.

1 INTRODUCTION
Automatic license plate recognition (ALPR) also known as automatic number plate recognition (ANPR) is a well-proven technology with real-life applications in access control, traffic monitoring and toll payment systems. ALPR remains the most important infrastructure in vehicle identification and autonomous intelligent transportation systems. Despite challenges of alteration, variation and occlusion, license plate remains the primary and principal vehicle identifier [1], and can be further exploited as a means for other intelligent infrastructure. This makes ALPR the most deployed vehicle identification system in many intelligent transportation applications. Figure 1 illustrates different vehicle identification systems with their common applications.

Typically, the ALPR system receives an image or sequence of images from a camera, and detects the presence of license plates, extracts the license plate regions and recognizes information within these regions. ALPR algorithm is composed of three processing modules: license plate localization (LPL), also known as number plate identifier or license plate extraction; plate character segmentation (PCS) and segmented character recognition, commonly known as optical character recognition (OCR). The first two stages involve the field of image processing and the final stage applies pattern recognition techniques. A complete ALPR system should contain all three stages and process information in real-time. That is, fast enough operating speed lesser than or equal to 50 ms [2] in order not to miss a scene. Figure 2 depicts ALPR processing stages.

During the design process, factors such as environmental conditions, vehicle speed, camera distance, and type are being prioritized depending on the target application. As an example, applications for secure access control is designed with a focus on higher recognition accuracy instead of recognition speed. This is because any misclassification would result in system failure. Hence, tailoring design to the expected application may be necessary to improve system performance. Also, in evaluating the performance of an ALPR system, the quality and nature of dataset is important. The quality and number of test sets used in assessing the performance of an ALPR system directly affects
the evaluation of that system. For optimum results, choice dataset should be carefully chosen depending on the nature of target application. Typically, datasets are classified based on five criteria; image resolution, camera distance, environmental illumination, capture angle and background complexity. Depending on target application and ALPR design, dataset class can be referred to as "normal" or "difficult" sample. Figure 3 shows sample sets of an ALPR system with an estimated size of the license plate on the image.

ALPR data are either captured with static or moving cameras, which inform the nature of the captured set. Datasets captured with moving cameras often have few variation in image resolution or camera distance but much variation in background complexity, and environmental illumination. While datasets captured with static cameras have less background or illumination variation but with much varying license plate sizes.

Existing ALPR systems are realized on either microprocessor (software-based) or custom computing platform (hardware-based). Software-based ALPR systems could be computed on generic processors or specialized processors, such as DSPs. Most hardware-based ALPR systems are embedded and are implemented using custom computing platforms, such as FPGAs. Implementing ALPR on microprocessors come with the limitation of processing instructions and data sequentially. In addition, the use of microprocessors require ALPR designs to run on an operating system. Although, this approach provides enough computational resources for complex image processing algorithms, meeting real-time demands still remains a challenge. To meet real-time demand, custom platforms are considered best alternative for processing ALPR algorithms. Their inherit parallel computing architecture makes it possible to exploit true parallelism. However, limited computing resources and storage size within the chip make it difficult to explore novel ALPR algorithms on custom platforms.

Since image processing techniques are considered a software domain task, many ALPR algorithms and techniques have been implemented on microprocessors [3], and only a few on custom computing platforms (fixed-function hardware). Also, the use of microprocessors for ALPR designs have been very appealing because of the vast availability of computational resources and data representation flexibility. However, the sequential execution of computationally intensive ALPR algorithms has negative effects on the processing speed of software-based ALPR systems. This limitation is a major drawback in meeting real-time demand. To meet real-time demands on microprocessor, high-performance microprocessors and specialized processors (such as DSPs) are used to emulate parallelism by rapidly switching computational tasks [4,5,6]. Although, this approach provides better computing power necessary to process ALPR algorithms faster, monopolized computational resources and increased power consumption remain a challenge [8].

To meet real-time demand with less power consumption, custom hardware such as the Field Programmable Gate Arrays (FPGAs) are being considered as alternative platform for ALPR designs [7]. The need for low-cost, power-efficient ALPR systems motivated researchers to consider FPGAs as a suitable computing platform for intelligent and embedded camera application. Power-efficient ALPR within a chip makes it possible to achieve portable, standalone systems with minimal communication overhead for both mobile and real-time applications. This custom hardware is inherently parallel and can be used to exploit parallelized image processing operations. But image processing techniques are thought of as a software domain task and most of its processes are designed with dependent sequential operations. Thus, if algorithms are not properly chosen and optimized with proper resource management, the limited hardware resources available within an FPGA chip might not suffice. To bridge this gap, chosen algorithms must be thought of in terms of the underlying FPGA hardware architecture and its available computational resources. Implementing choice algorithms on FPGAs is not a simple task due to data representation constraints. Hence, choice algorithm, image processing techniques and computing platforms are criteria that could affect the performance of an ALPR system.
The purpose of this paper is to provide researchers a comprehensive review of existing ALPR systems on microprocessors and FPGA platforms, with details on methods and architectural approaches. This paper serves as a guide for further review of specific processing techniques by discussing the pros and cons of methods, comparing their performances.

The remainder of this paper is organized as follows. In Section 2, ALPR systems on microprocessor and specialized processors are reviewed; processing techniques, algorithms and methods are discussed. Section 3 discusses ALPR on FPGA platforms; feature extraction techniques and design approaches are discussed. Section 4 further presents the architectural description of embedded ALPR. The paper concludes with Section 5, which summarizes the literature.

2 ALPR ON MICROPROCESSOR PLATFORM

ALPR on microprocessors are also termed software-based approach because the design is developed on either generic or specialized processors that run an operating system. Software-based ALPR comes with certain design conveniences and has become the most common implementation approach. However, these design conveniences comes with operational limitations, which have a significant impact on system performance. Microprocessor provides huge computational resources and data representation flexibility, making it possible to exploit different image and pattern processing techniques. However, the sequential execution of computationally intensive ALPR algorithms remains a limitation to achieving real-time processes. To meet real-time demand, high-performance processor are used to emulate parallelism by rapidly switching between computational tasks. The drawback to this is excessive power consumption caused by high clocking speed and increased overhead. A generic solution to this is to reduce the computational complexity of an ALPR design and this comes at the expense of its performance.

ALPR on specialized processor is another software-based approach. Unlike generic processors, specialized processors are specially designed to process complex image processing algorithms. An example of this processor is a Digital Signal Processor (DSP). Its architecture contains dedicated logic blocks for processing complex, resource-unfriendly operations. DSPs can contain segmented blocks for parallel processing of resource-unfriendly ALPR operations. This feature is good for resource usage optimization. Specialized processors are attractive for resource-unfriendly operations because they have many dedicated multipliers for good computational speed. However, unlike other signal processing applications that require more multipliers, image processing applications needs more internal memory (RAM) for buffering. This requirement makes the DSP unattractive for processing memory-demanding ALPR algorithms. To offset this limitation, DSPs are often coupled or integrated with other computing platforms. A hybrid approach requires ALPR algorithms to be classified into memory-friendly and memory-unfriendly blocks that should be implemented on a DSP and its loosely-coupled co-processor, respectively. The additional size, power consumption and increased communication overhead makes ALPR on specialized processors unattractive for portable embedded ALPR applications. Thus, minimizing the trade-off between complexity and performance remains the focus of researches on ALPR on microprocessors and specialized processors. The following subsections review license plate localization, segmentation, and recognition techniques on microprocessors and specialized platforms.

2.1 License plate localization techniques

License plate localization techniques extract license plate features in order to identify the presence of a plate. Feature extraction methods can be classified into seven categories. These categories are based on their image processing technique.

2.1.1 Localization by spatial filtering

Spatial filtering is the process of using a sub-image (mask, window or filter) to produce a desired effect on an input image. Feature extraction for license plate detection using spatial filters is best used in identifying edges or boundaries. The concept is based on the fact that a license plate region contains more vertical lines than any other region in a vehicle image due to the presence of characters. Edges are a local set of connected pixels, different from boundaries which are a global set of connected pixels. Edges in an image are identified using first-order derivatives. Since first-order derivatives are sensitive to noise,
preprocessing is required before edge extraction. Thus, morphological operations and other filtering techniques are combined with traditional edge identifiers for optimum performance.

A microprocessor-based ALPR system uses vertical edge detection and statistical analysis to filter non-edge region by combining points to form lines and lines to form rectangles, based on neighborhood threshold of pixels [9]. If the plate is not detected an optional morphological dilation is used to localize degraded plate. The work depicted a high localization rate instead of localization accuracy. High false positive depicted is a reflection of its sensitivity to noise from environmental complexity. Another hybrid technique combined edge statistics and morphology. By assigning weights to scanned vertical edges, the candidate region is selected using a density function [10]. Morphological operation is used to eliminate unwanted lines generated from the spatial mask.

Vertical edge is detected using spatial filters that detect black-white and white-black transitions [11]. Unwanted lines are eliminated using a line removal algorithm that eliminates oriented pixels. This approach reduces noise to a good degree and could be very effective in eliminating blobs during segmentation. Also, the vertical edge detection algorithm is robust to inclination.

[12] enhanced the work of [11] for low-resolution images. Vertical edge detection algorithm (VEDA) is used to extract vertical edges based on binary level transition and neighborhood pixel value conversion. AND and NAND gates are used to highlight desired details by combining the input image with the VEDA output. An advantage of this approach is that it distinguishes the plate region, particularly the beginning and end of each character. However, the approach comes with several drawbacks and limitations. Firstly, the vertical edge detection technique results in doubled character boundary, causing character overlapping which could make segmentation difficult.

That is, using VEDA outputs a 1 to 2 or 2 to 1 vertical edge thickness, depending on the foreground and background intensity. It should be noted that the candidate region extraction concept makes the system restricted to a controlled environment, because it uses the horizontal distance of a character width as criteria for extraction. This means that the distance of a camera has to be within a limited range. Besides, binarization technique adopted at the preprocessing stage of the work consumed much of the entire license plate detection system. This limits the approach to images lesser than 352 × 288 resolutions.

Although, VEDA is nine times faster than the traditional Sobel filter, it needs prominent edges for accurate detection. To speed up processing time [13], Sobel filter was combined with morphological operation to extract license plate region. Achieving this comes at the expense of localization accuracy.

[14] used Sobel filters to extract edges on Chinese plates. Candidate regions are verified by identifying the presence of characters. By combining Extremal Region and Adaboost classifier for verification of candidate plate, the work demonstrated an efficient way of using classifiers on fine extraction of plate instead of the coarse approach.

The aforementioned approaches involve complex image processing operations that take significant computation time. The work of [4] is a comparative analysis for four edge detection techniques applied to license plate detection on specialized DSP. The work compares Prewitt, Canny [22], Haar-wavelet and Sobel. Report shows that Haar-wavelet is more efficient on DSPs than other edge detection techniques. However, image sharpening processes are required when adopting Haar, to enhance flat or distorted edges.

### 2.1.2 Localization by sliding concentric window

Sliding concentric window (SCW) is a segmentation technique for identifying irregularity in image pixels using their statistical values. Concentric windows (mask) slide on an input image, computing their statistical values to segment the image using a predefined threshold. A predefined threshold value is chosen empirically after a trial and error procedure.

In [1], sliding concentric window is used to segment region of interest, and connected component is used for candidate region extraction. In another work, a license plate detection model is developed based on the natural properties of the plate region by finding vertical and horizontal edges to obtain its rectangular boundaries. Sliding concentric window (SCW) is used to identify vertical and horizontal edges of a grayscale image. The result is a binarized image with filtered non-edges obtained by the ratio of the SCWs, based on a threshold value [15]. An OR operation is further used to mask both edges (vertical and horizontal) to produce regions with rectangular shape. This approach is susceptible to image angle of view and environmental variation. One major reason for its poor localization accuracy is unfiltered edge from background complexity. Also, sliding window technique used for binarization is not adaptive, leading to a high level of noise in the binarized image.

[16] used adaptive segmentation technique based on sliding concentric window to detect ROI. Adaptive thresholding binarizes the ROI image. Better localization accuracy is obtained than [15], even with low-resolution images. To improve localization accuracy using SCW, one work used sliding window to extract corner features, computing their corner density within the window to determine candidate region [17]. Though, corner feature extraction is not computationally intensive, it is a complex task and susceptible to the effects of camera distance.

### 2.1.3 Localization by region labeling

Region labeling technique for license plate localization involves grouping pixels into regions with similar properties. Major ways of achieving this could be via connected component analysis or mean-shift feature extraction. Mean-shift seeks to find the main mode amongst sets of data points. Its output is a filtered image with range of value of convergence points. Convergence points are local mode associated with each pixel. On the other hand, connected component is a well-known technique in image processing that scans an image and labels its pixels into components based on connectivity (similar intensity). Extremal region (threshold-separable region) is the process of
decomposing object of interest into smaller regions using connected component obtained by applying thresholds at certain levels. Feature extraction using the concept of decomposing object of interest into extremal regions are usually scale invariant.

In [18], the object of interest is decomposed into extremal regions and is ordered by enumeration. The described extremal region is filtered by a Category Specific Extremal Region (CSER) detector and is computed using incremental computable approach that merges two regions that are computable, to find a new central moment for a new region. The result shows a localization accuracy of 95% even with a critical angular view. However, the processing speed reported shows the system is not suitable for real-time application.

A region-based license plate localization model is developed using mean-shift [19]. The algorithm first generates candidate region using mean-shift segmentation, then validates candidate region of interest by extracting features that should be classified using Mahalanobis classification. To generate candidate regions, a 2D space image is digitzed to a 2D lattice image. The space of the lattice is known as spatial domain and its gray level is known as range domain. Data points are obtained by concatenating spatial and range which the mean-shift acts upon to segment regions. In another work, mean-shift is applied for segmenting candidate regions [20]. It uses mean-shift on data points in the joint spatial range domain to extract points of convergence. The technique is variant to camera distance and susceptible to license plate background colour similarity. Also, mean-shift segmentation technique consumed too much of the processing time (over 5s). Though not suitable for real-time application, it achieves good localization accuracy.

### 2.1.4 Localization by morphological operations

Morphological operations are based on set theory, and its two primitive operations are dilation and erosion. One of the simplest applications of dilation is for restoring broken pixels while erosion is used basically for unwanted pixel elimination. From these primitive operations are other operations (open, close etc.) derived. In license plate localization systems, morphological operations are best used by combining them with other localization techniques [9,12,21,22], but basically for eliminating unwanted lines and identifying degraded license plate boundaries. Only very few works used morphological operations solely for license plate localization and this is because morphological operations alone are not as effective for vertical line detection compared with other vertical line identification methods.

In [12], morphological operation is combined with vertical edge detection algorithm to eliminate horizontal and inclined lines. In [22], morphological dilation is combined with Canny edge detection and Radon transform to eliminate noise from binarized image. Morphological operations are used to extract plate features, highlight plate regions, and group connected pixels for license plate localization [23]. The work demonstrated the possibility of uncombined morphological operations for license plate localization.

The most commonly used method for license plate detection is certainly the combination of edge detection and mathematical morphology. Edge detection is used to extract edge info and morphological operations, fuse, pixels or eliminate unwanted lines. However, this approach increases computational cost especially in embedded application, because edge detection is based on matrix multiplication.

### 2.1.5 Localization by classification

Classifiers like SVM and neural network are the most commonly used classifiers for license plate recognition. An SVM is a supervised learning technique that uses statistical learning theory and structural risk minimization as its optimal object to realize best generalization. Its two main approaches are “One to All” and “One to One” with the “One to All” a better approach for license plate localization. Common issues with license plate extraction with classifiers are computational complexity, huge learning samples to achieve fairly accurate results and long training time required to learn samples. It is best to use two-stage detectors that require a region proposal to extract candidate regions for a classifier to classify license plates [24].

This approach uses networks or structural information to propose candidate regions coarsely before license plate classification [25–29]. The limitation to this is increased computational demand, training sets, and processing speed. Recent advances in object detection uses deep multi-task networks for single-stage detection [30–33]. This approach mitigates the limitations of two-stage networks but demonstrates lower license plate detection performance with datasets having complex or varying backgrounds.

In [20], Mahalanobis classifier is used to verify true license plate after a mean-shift candidate region segmentation. The classifier takes in three feature samples and trains by computing their mean vector and covariance matrix based on the Mahalanobis distance. To reduce computational complexity, the classifier eliminates some candidate regions based on selected criteria such as angularity, aspect ratio and edge density. Although angularity criteria make the system robust to license plate size, processing time greater than 5 s limits the system for real-time application.

Adaboost classifiers have been a successful approach for face detection, minimizing computational time with a high detection rate. It has also been applied to detecting text in a natural scene [37]. In [38], Adaboost combines weak classifiers at the first stage for character detection. Detected character windows are passed to an SVM through a SIFT extracted key points. Localization speed consumed over 5 s because 36 classifiers were used to detect license plate. Also, its experiment was not based on natural text but artificially generated character training sets. A gentle Adaboost is implemented with Haar-like features on license plates edges [39]. A fast localization speed is achieved at the expense of a high false positive rate of 1.0. The system was also susceptible to license plate rotation. Hence, there is a claim that the set of features used for face detection might not be suitable for detecting text [38].
To gain significant processing speed, Adaboost for license plate detection is exploited on specialized DSPs [5, 6]. In [5], a generic object detection model based on Viola-Jones detector module is used to detect the presence of vehicles by identifying license plates. Compared with other boosting algorithms, the work demonstrated Adaboost to be the best choice for DSPs. However, the algorithm which is originally for generic object detection is not optimized for license plate localization. Thus, localization speed on the dedicated DSP did not meet real-time requirement of (50 ms). In addition, the exhaustive search technique of Viola-Jones detection leads to multiple detections for a single license plate. To reduce computational resource utilization and processing time on the DSP, [6] focused on the architecture and implementation of a complete license plate detection system on DSP platforms. Non-maximum suppression is used to consider confidence values of detection so as to merge multiple detections, and Kalman tracker is integrated into the system to limit the search space of the detector to certain areas of the input image. The update and prediction operation of the tracker reduced the computational time as well as improved character classification stage. Localization accuracy of 99% at (36.9 ms) localization speed was reported. Huge computational resource consumed at the detector module depicts that localization speed and computational demand is not always linear due to other factors that may affect speed.

Fuzzy logic is used to solve uncertainty in varying environmental conditions for colour recognition of license plate on a specialized DSP processor [79]. Colour features are first extracted by identifying candidate license region in images using edge texture information, histogram and geometric correction. Candidate region’s colours are then extracted using reverse colour identification, where background information of region is first obtained and compared to other pixel information within the region. The final step in feature extraction is colour space conversion that involves converting the standard RGB to HSV space. At this stage, Fuzzy logic is introduced to improve recognition performance in the presence of uncertainty. Colour is recognized by dividing license plates into sub-regions of same sizes. Localization accuracy of 95.83% is achieved with a very huge test dataset at 444 ms localization speed on a (600 MHz) DSP. The system was reported to demonstrate capabilities of adaptability to the effects of varying environmental conditions. However, computational demand and processing speed on the specialized processor limits it. Such processing speed further depicts that the technique is unrealistic on generic processors. License plate detection on DSP has also been exploited using Hough or Wavelet transform. In [4], Wavelet transform is used to post-process localized license plate region. There has been no complete license plate extraction on specialized processor that uses Hough or Wavelet transform.

Currently, no approach achieves a negligible false positive rate (FPR) in complex environments while maintaining an extremely high true positive rate (TPR). To achieve a high TPR, statistical mechanism like spatial features and temporal information has been adopted for license plate detection [40]. The work proposed a cascade framework for license plate recognition, with a balanced trade-off between computational cost and accuracy. Balancing is introduced because of the trade-off between discrimination of features and computational cost. To reduce computational cost the cascade framework made up of rejection mechanism is used to develop a real-time statistical plate recognition system based on two coarse-to-fine computational designs.

Textual images captured from natural scenes in real-time are difficult to segment due to background complexity. To improve the performance on poor quality license plate images in a real-time application, SVM-based multi-class classifier is used in a “One to All” approach [41]. “One to All” approach reduces the multi-class problem to a set of binary after normalization and feature extraction of the license plate image. The work achieved localization accuracy varying between 89.86% and 97.8% depending on image resolution. In [42], SVM is used to classify and localize license plate after features have been extracted using Histogram of Gradient and a variable sliding window that selects candidate regions from a wide range. In [43], a classifier-based license plate detection method robust to illumination variation is proposed using a two-stage detection model that consists of three processing blocks. The two-stage model is designed using cascade Adaboost and CCA. The work uses global threshold for the first stage and adaptive thresholds for the second stage if the license plate is undetected at the first stage. The first stage uses a global threshold technique to enhance illumination issues at a reasonable processing speed. Haar features are extracted and trained using Adaboost as a classifier. The second stage entails the same process as the first, except that the threshold is adaptive to mitigate severe illumination cases missed at the first stage. The work reported reasonable localization accuracy with several missed detection resulting from severe illumination from headlamps. Applying adaptive thresholds was a good approach to improving localization accuracy. However, this comes at the expense of processing speed about three times slower than global threshold.

To speed up localization speed using classifiers [44], trained weak classifiers with image features and weighted samples. Combining these weak classifiers a strong classifier is achieved, which is applied to a whole image to find sub-regions of a license plate. In another work [45], an inductive learning method combined with SVM is used for fast classification of license plate. A multi-layer classifier that hierarchically combines an inductive learning-based method for coarse classification and SVM for fine classification is designed. A computationally intensive system with a localization accuracy of 82.3% at real-time speed is reported.

[46], highlighted the advantages of using monochrome cameras over traditional colour camera sensors for ALPR systems. The work proceeded in using SVM for fine classification after a coarse candidate license plate have been extracted using Random Sampling Consensus algorithm. In time past, the use of character presence as a form of validating candidate plate became increasingly adopted. An effective approach is using traditional filters such as Sobel or light-weighted filters such as Top-hat to extract edge details. Region of interests are extracted as coarse license plates and fine extraction is obtained using classifiers. The reason SVM is preferred for fine extraction
over other classifiers is because it consumes lesser processing time.

In recent works, license plate processing using convolutional neural network (CNN) classifiers has gained significant interest due to higher recognition accuracy with text processing. To achieve processing time compared to those achieved with SVM, network complexity is reduced by adopting end-to-end processing techniques. End-to-end approach detects license plates and recognizes plate characters without explicitly segmenting the characters from the license plate [24–35]. This is achieved by jointly learning the segmentation stage with recognition stage for direct character recognition on plates. Although, this approach doesn’t perform well on low-quality images but has the advantage of reducing error propagation from an explicitly processed segmentation stage.

Based on R-CNN framework for object detection [25], the work of [28] adopted a two-stage detector for license plate detection on images with complex background. The work uses a region proposal network to generate candidate plates. ROI are subsequently classified using a 3-layer R-CNN with inception blocks to regress license plate’s four corner points. Regressed points are used to rectify extracted plate orientation before recognition. The author argued that using corner points for boundary extension and rectification is more effective than bounding boxes. To perform recognition, unsupervised spatial transform networks implicitly segment character features that are learned with shared weights on CNN.

[29] also applied a two-stage detector approach using a region proposal network for license plate detection. The work unified the learning process for both the detection and recognition network. The author reported improvement in accuracy with unified learning approach compared with when the same network is trained separately. Although, CNN often suffer drawback in recognition accuracy when segmentation network is jointly learned with recognition for low-quality images.

One-stage detectors using YOLO (You Only Look Once) framework [31] have also been exploited for license plate detection [33], divided processing tasks into car detection, plate detection and character recognition. The work adopted YOLO for recognizing license plate characters but error propagated from previous stages limited its performance [36]. applied YOLO2 [32] directly for license plate detection without performing vehicle detection. By changing the YOLO grid and anchor boxes the work achieved significant improvement in detection accuracy.

[24] proposed an end-to-end single-stage license plate recognition system that mitigates the impact of error propagation between sub-tasks. By using CNN with a new loss function for extending bounding boxes localization results were improved. At the recognition stage, emphasis was on the training approach which adopted a joint learning representation for segmentation and recognition stage of a multi-task deep convolution network.

[47] used convolutional neural network (CNN) for multi-directional license plate detection. The work presented interesting results by using a YOLO framework to determine the angle of plate rotation before training. However, CNN takes lots of computational time and its better used for validating candidate plate region than it is used for fine extraction.

More recently, a light-weight CNN is proposed to reduce computational cost for license plate detection [34]. An end-to-end approach was adopted and a suitable CNN model was built based on depthwise convolution technique to meet the target application. Although the major contribution of the work was complexity reduction at the license plate detection stage, the work did not report its complexity results.

### 2.1.6 Localization by other techniques

Other techniques exploited for license plate extraction include, Genetic Algorithm, Hough Transform and Wavelet Transform. Genetic algorithm or programming is known for its computational demand. Typically, Hough transform is used to detect straight lines, which makes it suitable for identifying degraded license plate boundaries. However, this advantage comes with the challenge of sensitivity to irrelevant vertical and horizontal lines. Wavelet Transform can locate multiple plates in an image with different orientations. But remains susceptible to camera distance, either too far or too close.

In [48], an algorithm for license plate localization based on a discrete wavelet transform is proposed to reduce the effect of environmental complexity. It uses HL sub-band to search license plate features and LH sub-band for verification by scanning horizontal lines around the features. Unwanted lines are eliminated by averaging vertical or horizontal line values and using this value as a threshold for elimination. Subsequently, license plate is localized by scanning the image row by row to identify character transition that is used to estimate candidate plate coordinates. Localization was validated using prior knowledge of the number of candidate plate characters. Although this approach improves localization accuracy, its performance is regional-feature dependent.

To reduce the computational demand of Hough transform [49], combines contour algorithm with Hough transform for boundary extraction. Generic algorithm has also been exploited for license plate extraction [50]. To determine the optimal number of vehicles required on a highway [51], uses genetic algorithm to identify license plate by distinguishing license plate text from other text in an image. Scale-invariant Geometric Relationship Matrix (GRM) is used to model symbol layout for license plate identification. Although genetic algorithms are computationally intensive and slow, however [51] model requires no segmentation stage for character recognition.

### 2.2 Plate character segmentation techniques

Character segmentation stage deals with the extraction of characters from license plate images. To achieve good character segmentation results, license plate must be properly rotated and binarized at the end of the license plate localization stage.
Typically, pixel projection is the most commonly used technique because of its simplicity and low computational demand. However, this technique has difficulty in handling character inclination resulting from oriented camera view. It is possible to bypass the segmentation stage and recognize characters directly from a localized license plate image [41, 50]. Adopting this approach might result in recognition difficulties for images taken from natural scene. To overcome the problem of poor image quality, post-processing steps handling fragmented or falsely connected character blobs are required after the segmentation stage. An approach used character segmentation to extract candidate license plate regions from an input image [52]. This approach could be computationally intensive, but remains an efficient means of validating true license plate region. In this subsection, character segmentation methods are grouped into three techniques and are discussed.

2.2.1 Segmentation by pixel projection

Pixel projection is the process of projecting colour or intensity information of an image in order to compute its pixel values. Pixel projection is the most commonly used technique for license plate character segmentation and could be applied in various ways. In addition, the technique is robust enough to follow every license plate localization technique. In [53], horizontal and vertical projections are used on localized plates. Also, projection and pixel intensity (histogram) can be applied to a peak-valley analysis for character segmentation. Uncombined Peak-valley approach is used for character segmentation in one-row and two-row license plates [22]. The algorithm checks for license plate width and height ratio and test for equality to determine whether license plate is one-row or two-row before segmentation.

Vertical projection is combined with dimensional criteria and syntax estimation in terms of valid character counts for character segmentation [54, 55]. Segmentation accuracy of 95.22% shows that character height greater than 50% of the license plate height is a good criterion for character segmentation. Another work combined projection and prior knowledge of character width [49]. Segmentation and recognition accuracy of 95.7% is reported after a skew correction.

To offset the challenge of inclination and size variation, resulting from camera angle and distance, one work proposed character extraction in complicated environment [56]. Connected component is used to label binary pixels to locate the top and bottom of each character. Computed values from characters are used to draw up two horizontal lines and the average angles between the lines are used for inclination correction. Character extraction is achieved by horizontal projection and further enhanced using character sequence location based on character width, height and distance criteria. Although, width, height values with specific inter-character distance criterion makes segmentation method country-specific, the work gave a very good segmentation result.

2.2.2 Segmentation by spatial filters

Spatial filtering involves the use of spatial masks, windows, or kernels to produce the desired response from a given image. When adopted for character segmentation, sub-images are used to extract character region from a plate image by segmenting the entire image. In [1], sliding concentric window (SCW) is used to segment characters, and further validation on segmented characters is done using character orientation and height information. In another work, SCW and column standard deviation are combined to segment character [16]. Most character segmentation using SCW have their license plate localization method also based on SCW. Adopting this style reduces computational demand because a single masking operation can be used for both tasks.

To offset the effects of illumination in an out-door environment, a three-module module system is designed for character segmentation [57]. License plate image is preprocessed using block-based masking with local histogram linearization to eliminate background pixels. Adaptive thresholds and C-mean algorithms are used to segment foreground pixels and a character determination module eliminates connected components using specific dimensional criterion. Missing characters are compensated by extrapolation based on corresponding sequence information of character position. The system is susceptible to vehicle speed, achieving segmentation accuracy of 95.6%. Besides, the system outperformed commercial SeeCar when compared using the same test data.

A three-stage character segmentation algorithm that mitigates the effects of poor quality binary image in character segmentation is designed [58]. Adaptive local threshold based on Niblack technique is used for binarizing license plate image and window sliding is used to detect characters. Redundant blobs are eliminated from overlapping blobs and true blobs are selected based on position estimation. A state-of-the-art segmentation accuracy of 98.3% for 3373 test data set is achieved.

2.2.3 Segmentation by classifier

Classifiers have also been exploited for character segmentation [6, 59]. However, this technique is not very common because of drawbacks such as huge computational demand and slow processing speed.

In [59], a joint segmentation and recognition model built on two-layer Markov network is designed. Edge projection histogram and Gaussian density are used to segment characters. Subsequently, the joint segmentation and recognition is modeled as a network with three kinds of nodes; latent nodes representing segmentation and recognition variables, nodes representing low-level features, and nodes representing dummy variables. The latent nodes and low-level feature nodes are used to locate objects of interest and dummy variable nodes are used for compositional semantic learning. Belief propagation is used to estimate posterior probabilities of two types of latent variables which are applied to license plate recognition. This
approach overcomes the challenge of dependency, experienced when segmentation is done separately from recognition. However, recognition speed of one minute depicts a very slow system, unrealistic for real-time operations.

### 2.2.4 Segmentation by region labeling

To mitigate the effect of varying illumination on natural scene plates [60], applied super-pixel-based degeneracy factor to identify connected neighbouring pixels by targeting local illumination changes. Although segmentation performance is not as efficient as those obtained using pixel projection, yet it presents a new approach for character segmentation based on connected component analysis.

In another work, region-based approach is used for character segmentation on specialized processor (DSP) [6]. Region descriptors are computed incrementally and classified using support vector classification. Although character segmentation result is not reported, the DSP implemented system achieved recognition accuracy below 90% despite 99% localization accuracy. Since segmentation performance has direct impact on recognition accuracy, poor recognition result could have been influenced by the character segmentation stage. Character segmentation based on region labeling remains highly promising and open to new research.

### 2.3 Optical character recognition techniques

The character recognition stage also known as optical character recognition (OCR) scans printed text to translate them into machine-encoded text. The preceding character segmentation stage provides only relevant character information that is described and recognized by the OCR. In video surveillance, the quality of video is measured based on the usefulness of data to the OCR [61]. Although license plate OCR is less complex than OCR for non-uniform text, it still requires robust algorithm to handle random noise from environmental variation. Successful recognition of a license plate by OCR is defined as the correct recognition of all characters of a license plate. The performance of a recognition stage is largely dependent on the preceding stages. Thus, the probability of a good license plate character recognition system is computed as the probability product of every step required in the processing framework [54]. To improve performance, some systems include a syntactical module based on symbol positioning and prior knowledge to handle syntax correction and misclassification in ambiguous sets. In this subsection, OCR techniques are grouped and discussed in microprocessor-based license plate recognition systems.

#### 2.3.1 Recognition by pattern matching

Typically, pattern matching deals with representation of each class by a pattern vector (prototype), which is compared to an unknown pattern. This comparison could either be based on minimum-distance or correlation, which uses Euclidean distance or correlation functions respectively. Pattern matching is simple to implement but not robust enough in handling noisy images. In addition, it demands huge computational resources because it uses raw pixel data of templates that are stored in memory. In [52], template matching using correlation of raw pixel data is used for character recognition. Although no result is reported from the work, matching raw pixel data demands huge computational resources. To reduce computational demand using template matching, character features are extracted and pattern vectors are described using zoning density, vertical projection, block scanning and line segmentation [49]. The work reported recognition and segmentation accuracy together, with an accuracy of 95.7%.

#### 2.3.2 Recognition by statistical classifier

Due to the randomness of pattern classes, classification can also be done using probability functions such as statistical classifiers. In license plate recognition, SVM is the most adopted statistical classifier for OCR [40,45,58], and this is because of its simplicity and adaptability compared with other statistical classifiers. However, it requires a large amount of training samples and time. In [45], an inductive learning-based method for coarse classification and an SVM for a finer classification is used for the OCR. The multi-layer classifier is combined using hierarchical structure, achieving recognition accuracy of 82.3% at (60 ms). In [58], SVM classifier is used to verify true character segmentation after recognition. The work achieved recognition accuracy of 98.9% at (80 ms) recognition speed.

An SVM implemented on DSP for license plate character recognition achieved recognition accuracy of 89% despite 99% license plate localization accuracy [6]. Region-based approach is used to isolate each character and further processing is done to improve the classification result using the information provided by a Kalman tracker. The work focused on improving computational speed at the expense of recognition accuracy.

In [53], license plate character is recognized using Markov model fed with feature vectors obtained from pixel density. The work reported good performance with recognition accuracy of 97.61% for approximately 5000 test sets. Another work uses a two-layer Markov network for a joint segmentation and recognition system, achieving 94% recognition accuracy for approximately 2000 character test sets [59]. However, slow recognition speed makes this approach unrealistic for real-time operation.

#### 2.3.3 Recognition by neural networks

Neural network involves a network of multitude of organized non-linear elemental computing elements called neurons. Unlike other OCR techniques whose decision function is determined by statistical properties (assumptions) of a class, neural network determines its decision function through training of
sets of sample patterns. Using such approach, statistical assumptions will be unnecessary.

A multilayer perceptron neural network (PNN) with a topology of 108-180-36 is used for license plate character identification [1,22]. In [1], a conscience full competitive learning mechanism between the weights of the input and middle layer tracks how often the outputs win the competition with a view of equilibrating and updating the winnings. The system uses a supervised training set to develop distribution functions within the middle layer. The work reported segmentation and recognition accuracy together with an accuracy of 89.1%. The OCR processing speed of approximately 46% of the total processing speed depicts the complexity of the multilayer PNN. Another work uses perceptron neural network, achieving 97.25% recognition accuracy [22]. Character training set of 15 alphabet limits the system to Vietnam license plate alphabet style only. In [16] probabilistic neural network achieved recognition accuracy of 97.4%. Another work classified characters based on symbols, alphabets and numbers using prior knowledge of location on plate [62]. Subsequently, it fed values into a back propagation neural network, trained based on conjugate gradient. Recognition rate of 96% (symbols), 97.4% (letters), and 99.5%(numbers) is reported from the work.

To reduce the number of multiplication operation, an OCR algorithm based on feed-forward neural network is designed [63]. Resized binary character image is fed as a 1-D vector into a multilayer neural network. Different number of neurons and vector size are used to create different neural network to determine the most suitable architecture. Scaled Conjugate Gradient (SCG) algorithm is chosen for training. Recognition accuracy of 97.3% is achieved for 3700 character set at (8.4 ms) recognition speed.

In another work, the theory of event independence is used to integrate a dual LPR algorithm to improve recognition accuracy [83]. The algorithm uses neural network to recognize characters on a DSP with recognition accuracy of 91% at (600 ms).

3 ALPR ON CUSTOM COMPUTING PLATFORM

Commonly used custom platform for ALPR systems are FPGAs [64]. These ALPR systems are termed “hardware-based” and are very convenient to integrate into portable, standalone units that operate at real-time. This advantage is possible because algorithms are fine-tuned to suit an underlying architecture. Also, its architecture can implement all parallelism techniques at low clocking speed, making it power-efficient with real-time functionalities. Their major drawback is data flexibility and representation constrain, which has limited the exploitation of state-of-the-art ALPR algorithms on FPGAs. Limited computing resources is another challenge when implementing complex algorithms on custom platforms. To offset data flexibility and representation challenge, choice algorithms are replicated in suitable formats depending on the underlying computing platform's architecture. These replications produce near-similar operations as the choice algorithm but are really not exact in characteristics. Hence, research focus has been to replicate choice algorithms for an hardware, in the most suitable format while retaining a near-similar result [65]. To offset the challenge of limited computing resources, complex image processing algorithms are classified into resource-friendly and resource-unfriendly blocks. This approach requires hybrid platforms for co-processing of suitable operations. Hybrid platforms have microprocessors or dedicated processors integrated within, or loosely coupled with a custom computing hardware. The following subsections discuss techniques of license plate localization, segmentation, and recognition on FPGAs.

3.1 Localization on custom platforms

The license plate localization (LPL) stage also known as license plate identification, demands more memory and computational resources than any other stage of an ALPR system. To reduce the computational demand, the LPL stage is divided into two steps: feature extraction and candidate region selection. The feature extraction section is concerned with eliminating background noise and extracting useful information such as edges, boundaries etc. The candidate region selection deals with connecting extracted features that should be used to localize the plate area.

3.1.1 Localization using spatial filters

A number of image processing algorithms have been adopted for LPL on FPGAs. A common technique for license plate localization on FPGA is spatial filtering. An FPGA loosely coupled with a DSP is used to extract vertical edge features for detection of license plates in [3]. Median filter and background subtraction is used to get information about the captured scene geometry, which is subsequently processed by classifiers implemented on a DSP. This technique speeds up localization stage by narrowing the search space, thereby reducing the computational demand on the FPGA. In [66], Gabor filter and Connected Component Labeling are combined to detect license plate on a FPGA. Input image is binarized and edge coordinates are computed using Gabor filter. Sobel vertical edge operator combined with morphological operations is exploited for edge features extraction on FPGA [21,67]. This approach is suitable for reducing complex edge background and mitigating the adverse effects of brightness. In [68], license plate is used as a means for taxi identification using Sobel filters for vertical edge features extraction for an embedded license plate detection application.

Due to the sensitivity of edge detection filters, it is required that vertical edge filters should be combined with other functions to reduce sensitivity to noise or double edge/boundary formation. For first-order derivative filters like the Gradient and Sobel, it is ideal to combine these filters with morphological operations that should mitigate the effect of noise. Second-order derivative filters like the Laplacian are more sensitive to noise than Sobel. Hence, Laplacian filters should be combined with Gaussian function, which reduces noise by blurring an image.
In [69], an FPGA implementation uses Sliding Concentric Window (SCW) to extract texture, gradient, and colour for sets of classifiers. This is the only FPGA implementation where sliding concentric window is used for license plate feature extraction.

### 3.1.2 Localization using morphological filters

Morphological filters are the most used filters for license plate localization on custom platforms. This interest comes because of the ease of pipelining and low computational resource demand for edge extraction. Morphological operations using linear structuring element combined with Sobel [67] or Gaussian [21] have also been exploited in detecting candidate region of license plate. These works achieved localization accuracy of 99.1% [67] and 89.95% [21] on a FPGA. Also, the works demonstrate efficient computational resource utilization of morphological operations on their custom hardware. Combining morphological and spatial filters is a common approach in software-based license plate localization. But because of matrix multiplication operations associated with spatial filters, this approach is not attractive in custom platform designs. Uncombined morphological operations for license plate localization in custom platforms remain a preferred approach.

Morphological open and image subtraction is used in place of spatial filters for edge detection [70–74]. Structuring elements used for morphological operations are designed based on the size of license plate and shape of characters. This approach also known as top or bottom hat transform is a light weight alternative to traditional spatial filters. In [70], an algorithm based on Tophat transform is combined with structuring element for license plate extraction on an FPGA. To reduce hardware usage, structure element of the morphological open operation is truncated from a 3 × 3 rectangle to a diamond-shaped filter. Although adopting this technique is at the expense of lost pixels in some part of the license plate region, a close morphological operation is used to fill missing plate pixels. The work depicted good processing speed, but edge detection using morphological operations and arithmetic subtraction is typically a background elimination technique when compared to a traditional spatial filter. Thus, this approach still faces plate localization challenges that a spatial filter offsets. In [75], morphological open and close operations are used to extract license plate features by filtering out irrelevant pixels. The structure-element used is decomposed into smaller rectangles, and the open and close operations are done simultaneously for each clock cycle. Localization accuracy of 98.4% is achieved with real-time application capability.

Morphology-based license plate detection algorithm has been exploited using its application to demonstrate the capabilities of a streaming model, with automated methods and tools for generating accelerators like license plate detectors [76]. Since morphological filters are inherently parallel in operation and have a high degree of instruction and data parallelism, the work manually allocated the execution of parallelizable code on accelerators. The open and close morphological operations were implemented on an accelerator. Because the work is focused on license plate extraction in complex scenes, the morphological modules used up 50% of the total license plate recognition time.

The reason why morphological filters are seen as a lightweight alternative to spatial filters is that morphological filters function more like background eliminator on gray-scale images. The outcome of this is obvious sharp pixel transactions that are edge-like. To achieve true edge detection using morphological filters, filtering should be done on binary images because binary images already contain sharp transitions between background and foreground. Also, this approach is less computationally intensive and less susceptible to noise. Thus, making it suitable for applications on custom platforms.

### 3.1.3 Localization using region labeling

Region labeling has also been exploited on custom hardware for license plate localization. To reduce memory requirement on FPGA, one work used connected component labeling in place of contour tracing to trace out region boundaries [77]. This algorithm propagates labeling alternatively using an approach called forward and backward passes. The work achieved 87% localization accuracy with good resource utilization on the FPGA. Another work on FPGA combines connected component with Gabor filter for license plate extraction [66]. Prior to validation, connected component is performed on the coordinates computed by the Gabor filter to extract the features of interest.

### 3.1.4 Localization using pixel projection

Due to its low computational demand, pixel projection has also been exploited for license plate extraction on custom hardware [78], used vertical and horizontal projection of pixels to generate histogram data of an input image. Based on predefined threshold this data is used to determine license plate region. Although this approach comes with lots of drawbacks, it does demonstrate good localization speed and resource utilization on FPGA.

### 3.1.5 Localization using classifiers

Classifiers are rarely used for license plate detection on custom platforms. This is because classifiers demand huge computing resources and require application-level algorithms that are sequentially dependent in steps. This makes them unattractive for hardware implementation on FPGAs. In [69], an FPGA-based license plate localization using classifiers is designed as a component for system on chip (SoC). It uses independent classifiers to analyze features extracted by a sliding window. The output of all classifiers is integrated to locate potential window containing license plates, and spatial fusion is further used to select candidate window that represents a true license plate. A third-order acceleration is achieved when compared to pure software implementation. However, resource consumption remains an issue.
Table 1 summarizes the different feature extraction methods that have been adopted for license plate localization on custom platforms.

### 3.2 Segmentation on custom platforms

Character segmentation on custom platforms has exploited only pixel projection techniques. Pixel projection remains the simplest and commonly used approach amongst others, and this is as a result of its low computational demand. A common challenge in character segmentation using pixel projection is noise and inclination, thus pre-processing steps are used to mitigate the effects of noise.

#### 3.2.1 Segmentation using projection

Obviously because of its low computational requirement, projection is most appealing for embedded application [80]. In [76], horizontal projection and connected component labeling are used to select character regions. Preprocessing step is used to minimize noise between characters and plate edges by computing the horizontal projection of each column in the region and setting as background all the pixels in a column whose projection is less than a threshold. License plate segmentation is used to validate localized candidate license plate region, before summing columns to locate existence of characters in [66].

Another work combined pixel projection and morphological operations to mitigate the effect of noise in character segmentation on FPGA [81]. Two morphological operations are used at the segmentation’s preprocessing stage to minimize the impact of noise. Horizontal projection is replaced with a license plate height optimizer to reduce computational demand. Vertical projection follows an open morphological operation that eliminates unwanted blobs in license plates. Segmentation accuracy of 97.7% at (1.4 ms) is achieved. Resource utilization and power consumption on FPGA also depicts an efficient character segmentation design. Table 2 depicts a summary of license plates character segmentation methods on custom platforms.

#### 3.3 Recognition on custom platforms

In hardware-implemented ALPR, two recognition techniques commonly used are pattern matching and classifier network. In some classification approaches, features from the segmented characters are described or represented before actual classification. The recognition stage also comes with challenges such as zoom factor, character size, inclination, and thickness variation.

##### 3.3.1 Recognition using pattern matching

Pixel-wise comparison, a type of pattern matching technique is adopted for character recognition on an FPGA [76]. Features of characters are extracted from horizontal and vertical projection of both row and column of symbol prior to recognition. Pixel-wise comparison is useful for recognizing well aligned, non-broken, and fixed-size characters.

In [72], K-mean clustering is used to recognize characters extracted by partitioning character segments. Another pattern matching technique uses vector crossing and zoning to extract character features prior to template matching processes [85]. Both the feature extraction and template matching processes are first implemented on a PC to identify stages that require acceleration. Subsequently, the template matching process is implemented on FPGA and correlation values are computed and compared with a predefined threshold.

##### 3.3.2 Recognition using classifiers

The work of [63] is implemented on FPGA using the same principle in [82]. However, the classifier network achieved better recognition with larger character sizes. To compensate for resource consumption on the hardware chip, character text is scaled to a suitable size. Also, a binarization stage is introduced to reduce resource usage by making it possible to replace multipliers with multiplexers. Recognition rate is retained at 97.3% at (0.7 ms) processing speed. The resource utilization on the FPGA chip is 23%. Raw data fed into the network row by
row without any description negatively impacted the recognition result for noisy sets.

The work in [66] uses Self Organizing Map (SOM) neural network to train large amount of sample characters. Typically, a SOM has two layers; input layer and computation layer. The computational layer contains a processing unit that calculates the weight matrix during the learning phase.

In [84], the focus was to implement efficiently the activation function of Neuron on FPGA. This is achieved using piece-wise approximation and approximation error function to determine its activation function number. The function is divided into region which is used to reduce the approximation error that comes as a result of eliminating or avoiding the use of division and exponential function, required in the software implementation. Table 3 summarizes hardware-implemented OCR techniques for ALPR systems.

| Method               | Principle          | Pros                      | Cons                                         | Reference |
|----------------------|--------------------|---------------------------|----------------------------------------------|-----------|
| Template matching    | Pixel values       | Resource efficient        | Inclination, broken pixel, character size vulnerable | [76], [72] |
| Classifier network   | Feature extraction | Fast, robust              | Complex, character size dependent            | [66], [63], [83], [85] |

4 | EMBEDDED ALPR ARCHITECTURE

Hardware-accelerated platforms for ALPR consume less power and achieve high processing speed, making it possible for portable ALPR systems, suitable for smart vision applications. However, data flexibility constraint and limited resources available within the hardware makes it difficult to implement novel image processing algorithms for license plates recognition. Recent advances in hardware design, such as chip density and functionality have stirred renewed interest in embedding complex image processing operations on configurable hardware. Achieving this is not a straightforward task of merely porting software-thought algorithms on hardware, but a redesign of functions to suit the underlying hardware architecture as basics for development. This approach of carefully choosing operations and processes based on underlying hardware architecture does have a significant drawback on recognition accuracy. Thus, the focus of research on hardware-based ALPR has been to achieve recognition accuracy measurable to those achieved on microprocessors, while minimizing the trade-off on resource consumption.

Real-time computing operation and pipelining are the advantages of ALPR systems on custom platforms. The major challenge with custom hardware is limited memory resources required to process intensive ALPR algorithms. Also, target algorithm is often too complicated to be described in HDL from a software language description. Although C language description is available for solution on hardware, direct conversion does not achieve optimal performance. To offset this design challenge, it is necessary to determine the underlying architecture of a well-optimized algorithm, map the architecture to the available hardware resources before implementing the algorithm on the hardware architecture.

This section discusses various architectures that are categorized according to their processing technique. Table 4 presents an overview of existing ANPR architectures, highlighting information about the processing approach, performance, and test dataset.

4.1 | Architecture for morphological filters

Morphological operations are based on set theory, and its two primitive operations are dilation and erosion. It operates only on binary images and because of its low computational cost it has become very attractive in embedded license plate localization systems.

To exploit parallelism on an FPGA, the Structure Element (SE) used for designing the morphological operation on hardware is decomposed into smaller rectangles that could run simultaneously per clock cycle through 30 stages of pipeline [75].

To reduce complexities of describing hardware on an FPGA [67], implement its morphological operations without using multipliers and dividers. Structured Element rectangle is decomposed into two $3 \times 22$ and $3 \times 5$, with the number of pipelined stages using 22 and 5 respective stages. The Gradient Image pixel is easily formed by multiplying the output, which is a $3 \times 3$ matrix memory by Sobel $3 \times 3$ and summing the results using simple adders. The work achieved 99.1% localization accuracy for 1000 plates at 3 ms processing speed.

Most morphological filtering operations on FPGA [71–74] are built on the work of [67]. The work of [71] seeks to achieve an area-efficient architecture. The approach uses six modules for the LPL stage. A memory reader that holds RGB input image, a converter module that converts input image to an 8-bit grayscale binary image, and a morphological module where SE is decomposed. To reduce hardware usage, diamond shape SE at the filtering stage of the morphological module is replaced by a rectangular-shaped mask obtained within the diamond shape SE.

In [72], morphological filters are implemented in a pipelined manner based on row buffering. The top-hat operation that involves image subtraction from an open operation on the same image is made to execute in parallel. The row buffers for streaming input pixels are implemented on Block RAM that is adjustable.

Each accelerator was built at Register-Transfer-Level (RTL) and was not dependent on optimization tools. However, dual
| REFERENCE | PLATFORM/(mHz) | LPL | PCS | OCR | TOTAL (%) | SPEED (ms) | SIZE | POWER (mW) | USAGE (%) | TEST DATA |
|-----------|----------------|-----|-----|-----|-----------|------------|------|------------|-----------|-----------|
| [75]      | FPGA Virtex 4/(149.120) | Morphology | - | - | 98.4 | 2.7 (LPL) | 640 × 480(LPL) | 571 | 2 | 2000 plates |
| [7]       | FPGA Virtex 4/(57.6) | Morphology | Projection | ANN | 97.8 | 11.0 (ANPR) | 640 × 480(LPL) | 910 | 80 | 1000 plates |
| [67]      | FPGA Virtex 4/(114.2) | Sobel & morphology | - | - | 99.1 | 3.8 (LPL) | 640 × 480(LPL) | - | 28 | 1000 plates |
| [71]      | FPGA Virtex 4/(86) | Morphology | - | - | 97.8 | 47.0 (LPL) | 640 × 480(LPL) | 628 | 33 | 1307 plates |
| [5]       | DSP C6414/(600) & FPGA | Viola Jones | - | - | 96.0 | 141.62 (LPL) | - | - | - |
| [6]       | DSP G6414/(600) | Viola Jones | Region-based | SVM | 99.0 | 36.89 (LPL) | 120 × 40(LPL) | - | - | 520 plates |
| [77]      | FPGA Virtex 2/(72.062) | Classification | - | - | 87.0 | 9.25 (LPL) | 256 × 256(LPL) | 650 | 33 | 1000 plates |
| [66]      | FPGA Virtex 4/(NA) | Gabor | Column summing | SOM | 91.7 | 200 (LPL) &500 (ANPR) | - | 250 | - | 1436 plates |
| [4]       | DSP G64plus/(600) | Haar wavelet | - | - | 98.6 | 7.32 (LPL) | 768 × 288(LPL) | - | - | 45,032 plates |
| [81]      | FPGA Virtex 4/(74.5) | - | Projection | - | - | 97.7 | 1.4 (PCS) | 60 × 300(PCs) | 913 | 11 | 1000 plates |
| [82]      | FPGA Virtex 4/(57.6) | - | ANN | - | - | 97.3 | 0.7 (OCR) | 34 × 22(OCR) | 1523 | 23 | 3570 plates |
| [72]      | FPGA Zynq (57.8) | Morphology | - | - | 90.0 | 100 (LPL) | - | - | - |
| [73]      | FPGA Zynq7000 | Morphology | Projection | - | - | 98.0 | 16.17 (LPL) &0.59 (CS) | - | - | - |
| [85]      | FPGA Zynq7000 | - | - | Template Matching | - | - | 99.5 | 99.5 | 0.63 | 22 × 34 | - | 6 | 2790 plates |
morphological operations are duplicated, which is not an efficient resource management paradigm. The duality of morphological operations could be exploited by implementing a single architecture as hardware to perform dual functions using appropriate control logic for resource control.

The work of [73] adopted [67] top-hat technique for plate feature extraction. Unlike [67], it carefully chose processing stages that consume more processing time to accelerate them on FPGA. A video direct memory is used to keep communication between the FPGA and an ARM Soc that holds the other processing stages. Morphological operations including the top-hat transform used as a light-weight alternative for edge detection are implemented on FPGA. Implementation on FPGA remains optimized at RTL providing constraint for stream buffering in FIFO manner.

Another notable difference between the works of [73] and [67] is that original HD images are processed directly instead of converting to SD as did [67]. To effectively manage the resource demand of HD images the system is implemented on heterogeneous platforms with time-consuming blocks accelerated on FPGA chip. In addition, this work is amongst very few available literatures that demonstrates the realization of ALPR system on Intel Altera FPGA.

Morphological filters are well adopted for image processing on FPGAs because they can be implemented with basic digital logic and have high degree of data level parallelism. Another advantage is its low control overhead and how they operate on a whole frame at each invocation. In [76], parallelizable codes are manually allocated to the FPGA accelerator and sequential codes are assigned to an embedded microprocessor. This means modules such as morphological filters, differencing and binarization modules are implemented on the FPGA accelerator, while connected component labeling, confidence evaluation, which are mostly sequential are implemented on the embedded microprocessor. By merging all filters in a single stream, the work eliminates wasted bandwidth to fetch processed frames back and forth from the main memory. Also, a separate accelerator is used in the matching process to compute horizontal and vertical projection for the pixel-wise matching approach. The switching of task between accelerators and the embedded microprocessor created overhead with the LPL processing time consuming 75% of the total ALPR processing time.

License plate character segmentation is implemented on an FPGA [81]. Character segmentation architecture consist of two modules; vertical and horizontal modules. Each module had a pre-projection, morphological, and critical point localization block. Accuracy and hardware resources are gained using a license plate height optimizer to minimize the impact of noise. This is achieved at the pre-projection block by calculating the start and end address of pixels to crop images into smaller sizes. To gain processing speed, multipliers are avoided in calculating the cropping factors. Basic logic such as OR, AND are used to design the max and min functions of the morphological module. The work achieved 97.7% segmentation accuracy with a processing speed of (1.4 ms) per plate.

4.2 | Architecture for spatial filters

In [5], modular and flexible software design for object detection was implemented on a DSP. No effort was put into optimizing the algorithm, which is originally designed for generic object detection. The work of [5] was further enhanced in [6] by optimizing the Viola-Jones detector for license plate extraction. Due to the low amount of memory available on the DSP, the memory layout required a more compact representation of data (fixed point). At the license plate detection module, two classification techniques are tested, Edge Orientation Histogram (EOH) and Haar-Like Features. The EOH achieved a better result with fewer features however, it requires too much memory when implemented on the DSP. The Haar-like features are adopted and detection performance is improved by scaling down image resolution. An accuracy of 99% localization rate is achieved for 520 images. The speed of localization at (36.89 ms), reflects the complexity and resource utilization of the LPL stage. At the recognition stage, a tree-multiclass structure that requires lesser processing time is adopted.

Since it is difficult to implement products and square roots on reconfigurable hardware. In [66], Euclidian distance is implemented using Manhattan metric which reduces hardware computation cost. At the input stage, a $3\times720$ memory buffer is scanned by a $3\times3$ window repeatedly to process the coefficient of the filter. The outcome of this approach eliminates the need for a full video streaming buffer for the whole image. Total recognition accuracy for the entire ALPR system is 89.93%. Although, memory requirements are extremely low, the speed of the entire system 500 ms reflects the complexity of the algorithm implemented on the accelerated hardware.

4.3 | Architecture for pattern matching

To implement pattern matching on FPGA [85], image is streamed FIFO with a depth of 1 bit per cycle. The mean of image pixel is computed on an ARM SoC. Image sample is then compared with the mean of streamed image and their correlation is computed. Acceleration was achieved at the correlation block by using an optimized fixed-point data type. This is done by first finding the maximum bit required for integer and fractional part of floating-point parameter variables to convert it to a fixed-point data type.

In [72], feature extraction from character text is implemented on arrays of zoning units after partitioning character image into segments. Each zoning unit contains an edge matching component built from a multiplexer and an adder, which is used to update a matching register.

4.4 | Architecture for classifiers

Based on results obtained from proof-of-concept [77], decided to parallelize the first three stages of LPL stage to reduce processing time, and further communicate results once to an
embedded microprocessor. Since FPGA runs much lower clock frequency than the embedded microprocessor, direct translation of software into HDL may suffer performance problems. To achieve sufficient throughput the divided image data horizontally into 16 segments and stored each in identical memory modules in the FPGA. Computation is done simultaneously by multiple calculation units executing pipeline processing. The classification which requires large computing facilities and memory consumption is optimized by using two computing modules that alternatively work on a column of the input image data. Pixel group labeling is done by propagated Connected Component Labeling (CCL). This is achieved using an algorithm called Forward Pass and Backward Pass approach. Results from implementation show the work achieved a very fast processing speed of (9.25 ms) at the expense of localization accuracy of 87%.

A double buffer framework is adopted to allow simultaneous frame data access through system bus [69]. Two parallel operating stages, data-in and data-process are designed to fully utilize the system bus bandwidth. Since it is unlikely to develop a feature extraction engine capable of processing windows of various HD size, an eight-layer pyramid generation is implemented for down sampling the source image up to 64 times. Results show that the feature extraction stage requires further optimization having consumed more resources than the remaining two localization stages.

The work of [81] is continued in [82], which implemented an ANN on an FPGA for optical character recognition. To reduce hardware usage and complexity, the weight generated from the ANN is converted from floating-point to fixed-point arithmetic. Also, 2-to-1 multiplexers are used in the accumulator that forms part of the hidden layer design. Multiplication of weights and neurons are carried out on these multiplexers, and adders are used to integrate all their outputs. The transformation function of the tan-sigmoid block of the ANN is pre-calculated and held in a ROM, which runs in parallel with the output layer. To improve recognition speed, three off-chip memory blocks are used to process and read weights in parallel. The work achieved recognition accuracy of 97.3% at (0.7 ms) recognition speed per character. In this work, performance in terms of accuracy and speed outperforms existing OCR for ALPR on hardware platforms.

5 SUMMARY AND CONCLUSION

5.1 Summary

From the summary in Table 4, the work reported in [7] outperforms the existing hardware approach with respect to overall ALPR recognition accuracy and speed. However, when implemented on PC it achieved license plate localization accuracy of 97.9%. This result compared to results reported for a PC implemented ALPR [9] that adopts same localization method, shows the design in [7] achieved lower accuracy and speed. This drawback could be attributed to variation in the test dataset. Hence, the lack of a uniform dataset or evaluation method remains an open challenge in ALPR evaluation. Also, it is necessary to tailor the design to the expected usage environment to improve efficiency. Although significant progress has been made in ALPR design on custom platforms, lots of work remain to be done to meet state-of-the-art recognition accuracy achieved on microprocessor ALPR systems. This is because limited access to data (bandwidth), data flexibility, and available system resources are constraints when implementing on custom hardware. These challenges open up issues for future research on ALPR systems on custom platforms. To overcome these constraints, research must focus on pipelined stream processing techniques, appropriate design of memory architecture, nearsimilar light-weight processing operations and resource controlling paradigm. Achieving these would require considering hardware design principles such as caching, stacking, row buffering and resource multiplexing.

Also, working in varying environmental conditions is inevitable for license plate recognition systems. As such, the deployment of ALPR systems capable of mitigating the effects of environmental lightning is necessary. Other critical areas to consider in ALPR research are stated below.

- **Platform**: State-of-the-art ALPR designs on microprocessors could be further exploited on different computing platforms such as DSP, GPU, Xilinx or Intel FPGAs. Results such as recognition accuracy, resource utilization, power consumption, or speed may vary from one computing platform to another. However, identifying and replicating state-of-the-art ALPR algorithms on a choice computing platform to produce near-similar performance remains a contribution.

- **Database**: The need for a uniform dataset for ALPR system is paramount. A uniform set is essential to generate more results for comparison with other systems. This would bring fair evaluation when results are compared with existing models. The challenge of rebuilding existing models would be eliminated. Assessing proposed designs on custom set comes with a lot of limitations. To offset some of these limitations, existing models could be implemented for custom sets for a fair comparison with the proposed design. However, implementing an entire system for complete comparison might be tedious.

- **Evaluation**: Another important consideration is a unified performance metric. Various ALPR literature evaluate and assess system performance in different ways. This makes performance comparison difficult and unstandardized. Most ALPR systems evaluate performance using License Plate Detection rate (LDR). LDR might not be the best approach to analyze ALPR localization accuracy. This is because it focuses only on correctly detected plates (true positive). Undetected plates (false positive) and wrong detections (true negative) are not taken into account. This metric for evaluation might not suit all applications. Evaluation should be done for LDR as well as sensitivity, specificity, which take into account false positive, false negative, and true negative. Neglecting these metrics limits evaluating the system performance to specific applications. For ALPR systems on custom hardware, a common evaluation is resource utilization. Most literature presents
resource utilization results in percentage with respect to unit composition (hardware size). This is also not suitable verification approach. This is because most ALPR systems are implemented on different FPGA platforms if not vendors. A better way would be to present resource utilization results based on unit consumption for each hardware component within the unit composition.

• Application: When evaluating ALPR systems, it is necessary to classify the datasets based on its application. Datasets for evaluating performances of ALPR for traffic enforcement applications should be "difficult" sets such as day-time far, day-time shadow, and night-time images. Datasets for evaluating performances of ALPR for access control should be "normal" sets such as day-time close, day-time normal, and day-time faded. And that of road patrol application should focus on difficult sets such as day-time multiple, nigh-times and dirty plates. Evaluating ALPR based on its target application would depict its true performance.

5.2 Conclusion

This paper presented reports on ALPR on microprocessors and custom hardware by reviewing existing methods and algorithms used for the three major processing stages. Existing architectural design approach was also discussed, stating their implementation techniques and recognition results. Some comparisons of ALPR systems on specialized processors and custom platforms stated in the literature were depicted in tabular form. Also, current challenges with ALPR systems on custom platforms are stated, suggestions to mitigate these challenges are discussed and future research focus is suggested. Future research of ALPR systems on custom hardware should focus on pipelined stream processing models and parallelization techniques using appropriate architecture and resource control paradigm.

ACKNOWLEDGEMENTS

The authors would like to thank Science and Technology Education at the Post-Basic Level (STEP-B) for providing access to materials, through a World Bank grant for the establishment of Center of Excellence in Software Engineering under Grant No. 4304-UNI. Also, the authors thank the Tertiary Education Trust Fund (TETFund) for providing funds for equipping the research lab with Internet-ready computers and printers.

REFERENCES

1. Anagnostopoulos, C. E., et al.: A license plate-recognition algorithm for intelligent transportation system applications. IEEE Trans. on Intell. Transp. Syst. 7(3), 377–392 (2006)
2. Anagnostopoulos, C.E., et al.: License plate recognition from still images and video sequences: a survey. IEEE Trans. on Intell. Transp. Syst. 9(3), 377–391 (2008)
3. Du, S., et al.: Automatic license plate recognition (ALPR): a state-of-the-art review. IEEE Trans. on Circuits and Syst. for Video Technol. 23(2), 311–325 (2013)
4. Musoromy, Z., et al.: Comparison of Real-Time DSP-Based Edge Detection Techniques for License Plate Detection. In: International Conference on Information Assurance and Security, pp. 323–328. IEEE, Piscataway (2010)
5. Arth, C., Bischof, H., Leistner, C.: TRICam: An Embedded Platform for Remote Traffic Surveillance. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 125–134. IEEE, Piscataway (2006)
6. Arth, C., Limberger, F., Bischof, H.: Real-Time License Plate Recognition on Embedded DSP Platform. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 125–134. IEEE, Piscataway (2007)
7. Zhai, X., Bensaali, F., McDonald-Mater, K.: Automatic License Plate Recognition on FPGA. In: IEEE International Conference on Electronics, Circuits, and Systems (ICECS) pp. 325–328. IEEE, Piscataway (2013)
8. Haim, A., Knieser, M., Rickalla, M.: DSP/FPGA comparative study for power consumption, noise cancellation, and real time high speed applications. In: IEEE J. Softw. Eng. and Appl. 3, 391–403 (2010)
9. Hongliang, B., Changping, L.: A Hybrid License Plate Extraction Method Based on Edge Statistics and Morphology. In: International Conference on Pattern Recognition, pp. 831–834.IEEE, Piscataway (2004)
10. Le, W., Li, S.: A Hybrid License Plate Extraction Method for Complex Scenes. In: IEEE International Conference on Pattern Recognition, vol. 2, pp. 324–327.IEEE, Piscataway (2006)
11. Al-Ghaili, A. M., et al.: A New Vertical Edge Detection Algorithm and its Application. In: International Conference on Computer Engineering and Systems (ICCES), pp. 204–209.IEEE, Piscataway (2008)
12. Al-Ghaili, A. M., et al.: Vertical-edge-based car-license-plate detection method. IEEE Int. Trans. on Veh. Technol. 62(1), 26–38 (2013)
13. Saleem, N., et al.: Automatic License Plate Recognition Using Extracted Features. In: International Symposium on Computational and Business Intelligence, pp. 221–225.IEEE, Piscataway (2016)
14. Gou, C., et al.: Vehicle license plate recognition based on extremal regions and restricted hebbmann machines. IEEE Trans. Intell. Transp. Syst. 17(4), 1096–1107 (2015)
15. Deb, K., Chae, H.-U., Jo, K.-H.: Parallelogram and Histogram Based Vehicle License Plate Detection. In: International Conference on Smart Manufacturing Application, pp. 349–354.IEEE, Piscataway (2008)
16. Anagnostopoulos, C. E., et al.: A Template-Guided Approach to Vehicle Surveillance and Access Control. In: IEEE International Conference on Advanced Video and Signal-Based Surveillance, pp. 534–539.IEEE, Piscataway (2005)
17. Qin, Z., et al.: Method of License Plate Location Based on Corner Feature. In: World Congress on Intelligent Control and Automation, pp. 8645–8649.IEEE, Piscataway (2006)
18. Matas, J., Zimmermann, K.: Unconstrained License Plate and Text Localization and Recognition. In: IEEE International Conference on Intelligent Transportation Systems, pp. 572–577.IEEE, Piscataway (2005)
19. Jia, W., Zhang, H., He, X.: Region-based license plate detection. J. Netw. Comp. Appl. 30(4), 1324–1333 (2007)
20. Jia, W., et al.: Mean Shift For Accurate License Plate Localization. In: International Conference on Intelligent Transportation Systems, pp. 566–571.IEEE, Piscataway (2005)
21. Ashourian, M., et al.: Real time implementation of a license plate recognition system based on adaptive morphology. Int. J. Eng. 26(11), 1347–1356 (2013)
22. Mai, V., Mao, D., Wang, R.: Vietnam license plate recognition system based on edge detection and neural networks. J. Info. Comp. Sci. 8(1), 027–040 (2013)
23. Zhai, X., Bensaali, F., Ramalingam, S.: License Plate Localization Based On Morphological Operations. In: International Conference on Control, Automation, Robotics and Vision, pp. 1128–1132.IEEE, Piscataway (2008)
24. Goncalves, B., et al.: Real-Time Automatic License Plate Recognition on FPGA. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 125–134. IEEE, Piscataway (2007)
25. Girshick, R., et al.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: International Conference on Computer Vision and Pattern Recognition. IEEE, Piscataway (2014)
26. Girshick, R.: Fast R-CNN. In: International Conference on Computer Vision. IEEE Computer Society, Los Alamitos (2015)
27. Ren, S., et al.: Faster R-CNN: Towards real-time Object Detection with Region Proposal Networks. Trans. on Pattern Anal. and Mach. Intell. (2017)
28. Dong, M., et al.: A CNN-based Approach for Automatic License Plate Recognition in the Wild. In: British Machine Vision Conference. Springer, London (2017)
29. Li, H., Wang, P., Shen, C.: Towards end-to-end car license plates detection and recognition with deep neural networks. IEEE Trans. Intell. Transp. Syst. 20(3), 1126–1136 (2017)
30. Liu, W., Anguelov, D., Erhan, D.: SSD: Single Shot Multibox Detector. In: European Conference on Computer Vision. Springer, Cham (2016)
31. Redmon, J., et al.: You Only Look Once: Unified, Real-time Object Detection. In: International Conference on Computer Vision and Pattern Recognition. IEEE, Piscataway (2017)
32. Redmon, J., Farhadi, A.: YOLO9000: Better, Faster, Stronger. In: International Conference on Computer Vision and Pattern Recognition. IEEE, Piscataway (2017)
33. Silva, S., Jung, C.: Real-time Brazilian License Plate Detection and Recognition using Deep Convolution Neural Networks. In: International Conference on Graphics, Patterns and Images. IEEE, Piscataway (2017)
34. Darji, M., Dave, J., Asif, N.: License Plate Identification and Recognition for Non-Helmeted Motorcyclists using Light-weight Convolution Neural Network. In: International Conference on Emerging Security Technologies, pp. 1–5.IEEE, Piscataway (2020)
35. Spanhel, J., Sochor, J., Juranek, R.: Holistic Recognition of Low Quality License Plates by CNN using Track Annotated Data. In: IEEE International Conference on Advanced Video and Signal Based Surveillance. IEEE, Piscataway (2017)
36. Hsu, G., et al.: Robust License Plate Detection in the Wild. In: IEEE International Conference on Advanced Video and Signal Based Surveillance. IEEE, Piscataway (2017)
37. Chen, X. R., Yuille, A. L.: Detecting and Reading Text in Natural Scenes. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2, pp. 366–373.IEEE, Piscataway (2004)
38. Ho, T. W., Lim, H. W., Tay, Y. H.: Two-Stage License Plate Detection Using Gentle Adaboost and SIFT-SVM. In: Asian Conference on Intelligent Information and Database Systems, pp. 109–114.Springer, Cham (2009)
39. Ho, T. W., Yap, W. H., Tay, Y. H.: Learning-Based License Plate Detection on Edge Features. In: International Symposium on Information and Communication Technologies. ACM, New York (2007)
40. Wang, S.-Z., Lee, H.-J.: A cascade framework for real-time statistical plate recognition system. IEEE Trans. Info. Forensics Secur. 2(2), 267–282 (2007)
41. Zheng, L., et al.: Number Plate Recognition Without Segmentation. In: International Conference on Image and Vision Computing pp. 164–168.IEEE, Piscataway (2007)
42. Prates, R. F., et al.: Brazilian license plate detection using histogram of oriented gradients and sliding windows. Int. J. Comp. Sci. Info. Technol. 5(6), 39–52 (2013)
43. Sarker, M. K., Yoon, S., Park, D. S.: A fast and robust license plate detection algorithm based on two-stage cascade adaboost. KSI Trans. Internet Info. Syst. 8(10), 3940–3957 (2014)
44. Zhang, X., et al.: License Plate Location Based on Adaboost. In: IEEE International Conference on Information and Automation, pp. 1705–1710.IEEE, Piscataway (2010)
45. Zheng, L., He, X., Wu, Q.: A Hierarchically Combined Classifier for License Plate Recognition. In: International Conference on Information Technology, pp. 372–377.IEEE, Piscataway (2008)
46. Panahi, R., Gholampour, I.: Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications. IEEE Trans. on Intell. Transp. Syst. 18(4), 767–779 (2017)
47. Xie, L., et al.: A new CNN-based method for multi-directional car license plate detection. IEEE Trans. on Intell. Transp. Syst. 19(2), 507–517 (2018)
48. Wang, Y.R., Lin, W.H., Horng, S.J.: A sliding window technique for efficient license plate localization based on discrete wavelet transform. Expert Syst. with Appl. 38, 3142–3146 (2011)
49. Chen, Z., et al.: Automatic license plate location and recognition based on feature salience. Int. J. of Comput. Cogn. 5(2), 1–9 (2007)
50. Samea, G. A., Khalefa, F.: Localization of license plate number using dynamic image processing techniques and genetic algorithms. Trans. on Evol. Comput. 18(2), 244–257 (2014)
51. Chiou, Y.C., et al.: Optimal locations of license plate recognition to enhance the origin-destination matrix estimation. Asian Trans. Studies 2(1), 80–92 (2012)
52. Khoshki, R. M., Ganesa, S.: Improved Automatic License Plate Recognition System Based on Single Pass Connected Component Labeling and Reign Property Function. International Conference on Electro/Information Technology, pp. 426–431.IEEE, Piscataway (2015)
53. Duan, T. D., et al.: Building an Automatic Vehicle License Plate Recognition System. International Conference in Computer Science, pp. 59–63.ACM. New York(2005)
54. Molder, C., et al.: Symbol Occurrence Probability Vectors for Syntax Correction in Automatic Number Plate Recognition Systems. In: International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, pp. 119–124.ACM, New York (2008)
55. Molder, C., et al.: Improved Automatic Number Plate Recognition System. In: International Conference on Visualization, Imaging and Simulation, pp. 49–54.ACM. New York (2008)
56. Sanyuan, Z., Mingli, Z., Xizhi, Y.: Car Plate Character Extraction under Complicated Environment. In: IEEE International Conference on Systems, Man, and Cybernetics pp. 4722–4726.IEEE, Piscataway (2004)
57. Wu, B.F., Lin, S.P, Chiu, C.C.: Extracting characters from real vehicle license plates out-of-doors. Inst. Eng. Technol. Comp. Vis. 1(1), 2–10 (2007)
58. Yoon, Y., et al.: Blob Detection and Filtering for Character Segmentation of License Plates. In: IEEE International Conference on Multimedia Signal Processing, pp. 349–353.IEEE, Piscataway (2012)
59. Fan, X., Fan, G.: Graphical models for joint segmentation and recognition of license plate characters. IEEE Signal Process. Letters 16(1), 10–13 (2009)
60. Kim, D., et al.: Effective Character Segmentation for License Plate Recognition Under Illumination Changing Environment. IEEE International Conference on Consumer Electronics, pp. 532–533.IEEE, Piscataway (2016)
61. Utkanova, A., et al.: Quality Assessment of Compressed Video for Automatic License Plate Recognition. In: IEEE International Conference on Computer Vision Theory and Applications, pp. 306–313.IEEE, Piscataway (2014)
62. Gao, Q., Wang, X., Xie, G.: License Plate Recognition Based on Prior Knowledge. In: International Conference on Automation and Logistics, pp. 2964–2968.IEEE, Piscataway (2007)
63. Zhai, X., Bensaali, F., Sotudeh, R.: OCR-Based Neural Network for ANPR. In: IEEE International Conference on Imaging Systems and Techniques, pp. 393–397.IEEE, Piscataway (2012)
64. Biyabani, A., Al-Salman, A. S., Alkhalaf, K. S.: Embedded Real-time Bilingual ANPR. In: International Conference on Communications, Signal Processing, and their Applications, pp. 1–5.ACM, New York (2015)
65. Bailey, G. D.: Design for Embedded Image Processing on FPGAs. 1st ed., pp. 1–482.John Wiley & Sons, Singapore (2011)
66. Caner, H., Gecim, H. S., Alkara, A. Z.: Efficient embedded neural-network-based license plate recognition system. IEEE Trans. Veh. Technol. 57(5), 2675–2683 (2008)
67. Zhai, X., Bensaali, F., Ramalingam, S.: Real-Time License Plate Localization on FPGA. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 14–19.IEEE, Piscataway (2011)
68. Hu, B., et al.: Real-Time Taxi Detection for Embedded System. In: IEEE International Conference on Intelligent Transportation Systems, pp. 3112–3115.IEEE, Piscataway (2014)
69. Yang, S.-Y, et al.: Hardware-Accelerated Vehicle License Plate Detection at High-Definition Image. In: International Conference on Robot, Vision and Signal Processing, pp. 106–109.IEEE, Piscataway (2011)
70. Clienti, C., Beucher, S., Bilodeau, M.: A system on chip dedicated to pipeline neighborhood processing for mathematical morphology. Eur. Signal Process. Conf. 2(1), 80–92 (2008)
71. Zhai, X., Bensaali, F., Ramalingam, S.: Improved number plate localisation algorithm and its efficient FPGA implementation. Inst. Eng. Technol. Circuits, Devices Syst. 21(2), 93–103 (2013)
72. Rosli, F., Elhossini, A., Juurlink, B.: Real-time vision system for license plate detection and recognition on FPGA. J. Parallel Algorithms Comp. Structure 32(1), 69–79 (2015)
73. Al-zawqari, A., Hommos, O., Al-Qahtani, A.: HD number plate localization and character segmentation on the zynq heterogeneous SoC. J. Real-Time Image Process. 16, 2351–2365 (2018)
74. Sborz, G. M., et al.: A Custom Processor for an FPGA-based Platform for Automatic License Plate Recognition. In: Symposium on Integrated Circuits and Systems Design, pp. 1–6.IEEE, Piscataway (2019)
75. Tan, Y. H., Xin, Y., Zhai, X. J.: A FPGA-Based Method for License Plate Localization. In: International Conference on Electrical Automation and Mechanical Engineering, pp. 534–537.Atlantis Press, Oxford (2015)
76. Bellas, N., et al.: FPGA Implementation of a License Plate Recognition SoC Using Automatically Generated Streaming Accelerators. In: IEEE Conference on Parallel and Distributed Processing, pp. 8-16.IEEE, Piscataway (2006)
77. Kanamori, T., Amano, H.: Implementation and Evaluation of a High Speed License Plate Recognition System on an FPGA. In: International Conference on Computer and Information Technology, pp. 567–572.IEEE Computer Society, Los Alamitos (2007)
78. Chhabra, S., Jain, H., Saini, S.: FPGA based Hardware Implementation of Automatic Vehicle License Plate Detection System. In: IEEE Conference on Advances in Computing, Communications and Informatics, pp. 1181–1187.IEEE, Piscataway (2016)
79. Wang, F., Man, L., Wang, R.: Fuzzy-based algorithm for color recognition of license plates. Pattern Recognit. Letters 29, 1007–1020 (2008)
80. Wang, Y.R., Lin, W.H., Yang, L.: A Novel Dsp Based Real Time License Plate Detection Algorithm. In: International Conference on Machine Learning and Cybernetics, pp. 1779-1781.IEEE, Piscataway (2012)
81. Zhai, X., Bensaali, F.: Improved number plate character segmentation algorithm and its efficient FPGA implementation. Springer J. Real-Time Image Process. 10(1), 91–103 (2015)
82. Zhai, X., Bensaali, F., Sotudeh, R.: Real-time optical character recognition on field programmable gate array for automatic number plate recognition system. Inst. of Eng. and Technol. Circuits, Devices and Syst. 7(6), 337-344 (2013)
83. Xie, M., Fu, H., Liu, Z.: One Design Method of License Plate Recognition System with High Recognition Rate. In: IEEE International Symposium on IT in Medicine and Education, pp. 24-49.IEEE, Piscataway (2008)
84. Jing, Y., et al.: An Efficient FPGA Implementation of Optical Character Recognition for License Plate Recognition. In: IEEE Conference on Electrical and Computer Engineering, pp. 1–4.IEEE, Piscataway (2017)
85. Farhat, A., Hommos, O., Al-Zawaqari, A.: Optical character recognition on heterogeneous SoC for HD automatic number plate recognition system. J. Image Video Process. 58, 1–17 (2018)