Field Programmable Gate Array-based Number Plate Binarisation and Adjustment for ANPR Systems

Xiaojun Zhai, Faycal Bensaali and Reza Sotudeh

School of Engineering & Technology

University of Hertfordshire

Hatfield, UK

{x.zhai2, f.bensaali, r.sotudeh}@herts.ac.uk

Abstract

Number Plate (NP) binarisation and adjustment are very important pre-processing stages in Automatic Number Plate Recognition (ANPR) systems and are used to link the Number Plate Localisation (NPL) and Character segmentation (CS) stages. Successfully linking these two stages will improve the performance of the entire ANPR system. This paper presents two optimised low complexity NP binarisation and adjustment algorithms. Efficient area/speed architectures based on the proposed algorithms are also presented and have been successfully implemented and tested using the Mentor Graphics RC240 FPGA development board, which together require only 9% of the available on-chip resources of a Virtex-4 FPGA, run with a maximum frequency of 95.8 MHz and are capable of processing one image in $0.07-0.17ms$.

1. Introduction

Automatic Number Plate Recognition (ANPR) systems have become a fundamental part of many Intelligent Transportation Systems (ITSs) to improve transportation safety and efficiency. Examples include electronic payment, access control, tracing of stolen cars and identification of dangerous drivers [1-3]. A typical ANPR system consists of three main stages: Number Plate Localisation (NPL), Character Segmentation (CS) and Optical Character Recognition (OCR) [4]. NPL is the stage where the Number Plates (NPs) are localised in the input image from the ANPR camera. CS is the stage where each character from the detected NP is segmented before recognition so that only useful information is retained for recognition. In the last stage, optical character information will be converted into encoded text by pre-defined transformation models [5].

Current ANPR systems use high performance workstations as processing units to meet real-time requirements under real-world environments. As ANPR systems are needed in different sections of a highway, the total cost and power consumption to cover a single highway are significant. Recent improvements in low-power high-performance Field Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs) for image processing have motivated researchers to consider them as a low-cost solution for accelerating such computationally intensive tasks [6]. By optimising the ANPR algorithms to take specific advantage of technical features and innovations available within new FPGAs, such as low power consumption and development time, it will be possible to replace the 3GHz roadside computers within small in-camera dedicated platforms.

Researchers normally focus on one aspect of ANPR system, such as NPL [7] [8] [9], CS [10] [11] or OCR [12]. However, for a robust ANPR system, an exhaustive and meticulous discussion of the pre-processing stages is desired. Two important pre-processing stages in ANPR systems are NP binarisation and rotation. NP image binarisation converts a 8-bit grey level NP image to a black and white image and the simplest way to perform this is to choose a fixed threshold value and classify all pixels in the image. However, brightness distribution in a NP image may cause some parts of character to be missed and noise impact to be increased after performing image binarisation due to the problem of uneven illumination. In such cases, there are two main approaches to deal with this problem which are global and local threshold based binary algorithms. One global threshold binary method is Otsu method [13], where the target and background in a given image are separated by maximizing the variances of the histogram. However, this method does not consider the correlation between the pixels in an image such as the one in NP images [14]. In this type of image, the correlation between pixels becomes more important than the grayscale values and using the global threshold with this type of images it is difficult to separate the NP characters from the background. The local binary method is often used to solve this problem as it considers the correlation between the pixels in a NP image. Adaptive local binary method is one of the local binary methods. In this method, an image is first divided into sub-blocks and then each sub-block is processed with a filter [15].

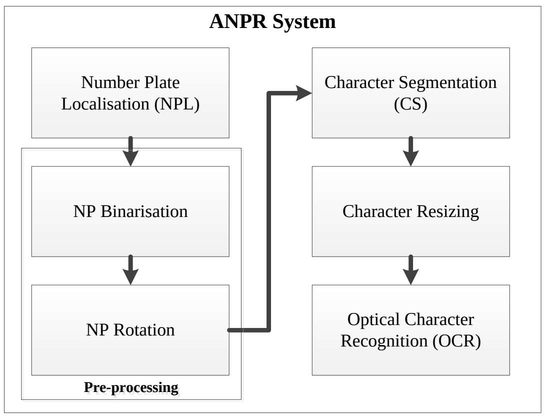
NP adjustment is also a very important pre-processing step in an ANPR system. Slanted NPs are very likely to cause failure of character segmentation in the CS stage. The main challenges in this step are how to correctly and efficiently calculate the rotation angles and adjust the NP accordingly. The most common used approaches to analyse the shape of the NP to calculate the rotation angles are pixel projection, Hough transformation or CCA [15] [16]. The main disadvantage of these methods is their computationally intensive nature to calculate the rotation angle, which could slow down the entire ANPR system.

This paper presents a NP binarisation algorithm which uses local binarisation method to solve the problem of uneven illumination and low complexity NP adjustment algorithms to automatically adjust NPs horizontally and vertically, which could improve the NPL result prior to CS stage. Two area/speed efficient architectures based on the proposed NP binarisation and adjustment algorithms are also presented and have been implemented and verified using a Mentor Graphics RC240 FPGA development board equipped with a 4M-Gate Xilinx Virtex-4 LX40.

The remainder of this paper is organised as follows: Section 2 describes the proposed NP binarisation and rotation algorithms. The proposed binarisation and rotation architectures are then described in Section 3. Section 4 is concerned with FPGA implementation and discussion of the experimental results. Section 5 concludes the paper.

2. NP Pre-processing

In ANPR systems there are several pre-processing stages such as NP binarisation, rotation and character resizing. In this paper, improved methods that take advantage of the NP image characteristics are presented for NP binarisation and rotation. Due to the fact of that NP images are taken from different lighting environments in real-world conditions, fixed or Global threshold binarisation methods are very likely to fail to separate the characters and background after NP binarisation. On the other hand, due to the angle of NP orientation, the NP image may have a slant and distortion which are very likely to cause failure of character segmentation in the CS stage. These two problems can significantly affect the recognition rate of the entire ANPR system. There is a need for efficient NP pre-processing algorithms and architectures to address these problems. Figure 1 illustrates the main building blocks of an ANPR system.


ANPR System
Number Plate Localisation (NPL)
NP Binarisation
NP Rotation
Pre-processing
Character Segmentation (CS)
Character Resizing
Optical Character Recognition (OCR)
Figure 1: The building blocks of an ANPR system.

The inputs of the first pre-processing stage (i.e. NP binarisation) are the localised grayscale NP images, where they will be binarised and then rotated in the next pre-processing stage (i.e. NP rotation). The processed images must meet the input requirements of the CS stage where the characters need to be clearly displayed in NP image and the shape and positions of the characters need to be adjusted properly.

2.1 NP binarisation

In ANPR systems the NP images are taken under different lighting conditions which give varying brightness distribution. If the well-known global threshold method is applied for NP image binarisation, the resulting images will not meet the input requirements mentioned above and are likely to fail the segmentation stage. Therefore, a local threshold method is proposed to solve this problem, which divides the entire NP image into many $m \times n$ blocks. Different thresholds are then calculated for each block and thus the entire NP image is binarised according to local illumination information.

In the proposed NP binarisation algorithm, a square $w \times w$ window is used to scan NP images column by column from left to right where each pixel from the NP image is the centre of the window. A mean filter is used to calculate the mean value for each window, and the mean value is used to calculate the local threshold.

Suppose that $f(x,y)$ denotes the grey value for pixel (*x*, *y*), which is always the centre point of a square window *B* with size $w \times w$. The window mean value $f_{mean}(x,y)$ is computed by equation 1:

$$f_{mean}(x,y) = \frac{\sum_{(x,y)\in B} f(x,y)}{w^2} \tag{1}$$

The local threshold $T(x,y)$ is then obtained by:

$$T(x,y) = f_{mean}(x,y) - \Delta t \tag{2}$$

Where Δt is an threshold offset which is used to adjust the threshold value.

The binary image is obtained by:

$$b(x,y) = \begin{cases} 0, & \text{if } f(x,y) \le T(x,y) \\ 1, & \text{else} \end{cases} \tag{3}$$

In this algorithm, *w* and Δt have significant impacts on the processing results, they are both identified by experimental tests. In the proposed system, these tests have shown that the constant value '6' for Δt has given the best binarisation results. *w* is determined by two other factors:

1) The size of characters in the NP image, for example, the stroke width of each character is normally around 8 pixels.
2) As the main aim of this research is to implement the entire ANPR system on one single FPGA [1] [17] [18], power of two numbers are used for *w* to avoid the need of multiplications as they consume a lot of on-chip FPGA resources and replace them with shifters.

Figure 2 shows a comparison between the use of global and local binarisation methods with different window sizes.

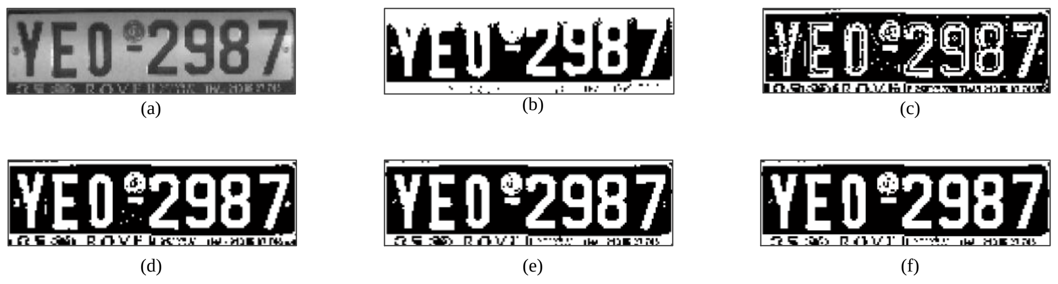


Figure 2: Results of using global and local binarisation methods with different window sizes. (a) Greyscale NP. (b) Global binarisation method. (c) Local binarisation method, $w = 4$. (d) Local binarisation method, $w = 8$. (e) Local binarisation method, $w = 16$. (f) Local binarisation method, $w = 32$.

As it can be seen from Figure 2, global binarisation method has failed to separate the NP image background and characters in the image, however, using local binarisation method better results can be achieved compared to the global method. The higher the value of *w* the better is the binarisation result, but high *w* value means more computations and hardware usage. For the proposed system and based on the obtained results from the experimental tests *w* has been chosen to be equal to 8.

2.2 NP Adjustment

In real-world scenarios, NP images can be slanted and distorted due to many factors such as the car and ANPR camera positions. Thus, horizontal and vertical adjustments are required after NP binarisation. In this section new algorithm for calculating the horizontal and vertical rotation angles is presented. Once the angles are found, a 2-D rotation method can be applied to adjust the NP image horizontally [19] followed by applying a cropping method to crop the Non-NP pixels from the rotated NP image. After cropping, the resulted image is vertically adjusted.

2.2.1 Horizontal Adjustment

The proposed algorithm calculates rotation angle by utilising the output image from the NPL stage. Existing algorithms to calculate the rotation angles require some characters analysis to obtain the angles. This affects significantly the computation time which is a very important factor in such real-time application. The proposed algorithm uses the output image from the NPL stage without the need to analyse the characters in the NP region of the image. Figure 3 shows and example of an input image to the NPL stage and the processed image.

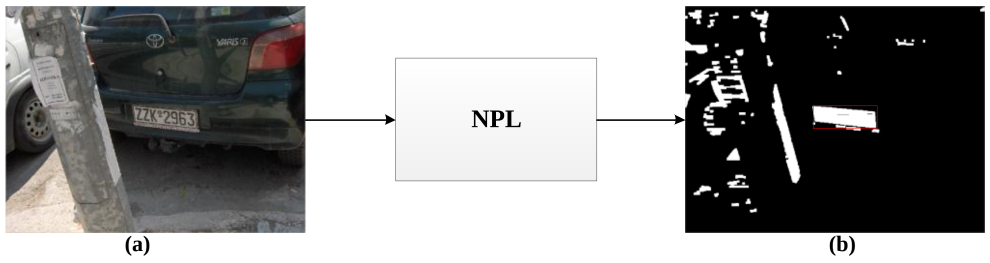


Figure 3: Input and output images of the NPL stage. (a) Input image. (b) Output image.

In Figure 3, the original colour image is processed in the NPL stage which produces a binary output image. Connected Component Analysis (CCA) is used to localise the NP region in the output image and once the NP is localised, the proposed rotation algorithm will be used to calculate the rotation angle. An example of a localised NP region that needs to be binarised and adjusted is shown in Figure 4.

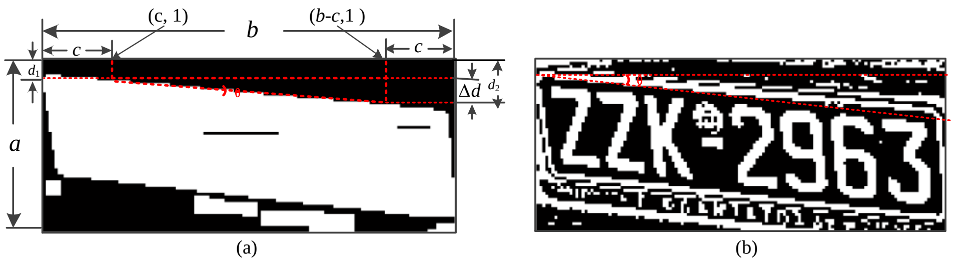


Figure 4 (a) Localised NP region. (b) Binarised NP image.

Let $a \times b$ be the size of the localised NP region and θ be the horizontal rotation angle. As illustrated in Figure 4, θ can be calculated by the proposed algorithm uses an approach that consists of the following two steps:

1) Search vertically the first NP pixel with value ‘1’ in the localised NP region from top to bottom at positions (1,1) and $(1, b-c)$. Then obtain two vertical distance values d_1 and d_2. c is an offset constant used to ensure that the first NP pixel is found in the correct NP top boundary. According to experimental tests, when c is close to $b/4$ the value of θ is more precise.
2) Calculate the difference $\Delta d = d_2 - d_1$.

According to trigonometric relations, θ is calculated using the following equation:

$$\theta = \tan^{-1} \frac{\Delta d}{b - 2 \times c} \tag{4}$$

After obtaining θ from the localised NP region, a 2-D rotation method will be applied to rotate the binarised NP image. In this paper, the nearest neighbour interpolation method has been chosen to perform the horizontal rotation which is based on the following equations:

$$x_2 = \cos\theta \times (x_1 - b/2) - \sin\theta \times (y_1 - a/2) + b/2 \tag{5}$$

$$y_2 = \sin\theta \times (x_1 - b/2) + \cos\theta \times (y_1 - a/2) + a/2 \tag{6}$$

Where a and b are height and width of the binarised NP image respectively, (x_1, y_1) and (x_2, y_2) are the old and new coordinates of a given pixel on the NP image respectively.

The rotation operation produces output locations (x_2, y_2) which may not be within the boundaries of the original NP image and they will be ignored. It is worth noting that the size $a \times b$ will be kept and as a result some pixels in the boundaries of the NP will be filled with value ‘0’. Due to the fact of the rotation algorithm may produce coordinates that are not integers, the proposed method uses the nearest integer coordinate values. The binarised NP region after the rotation is shown in Figure 5.

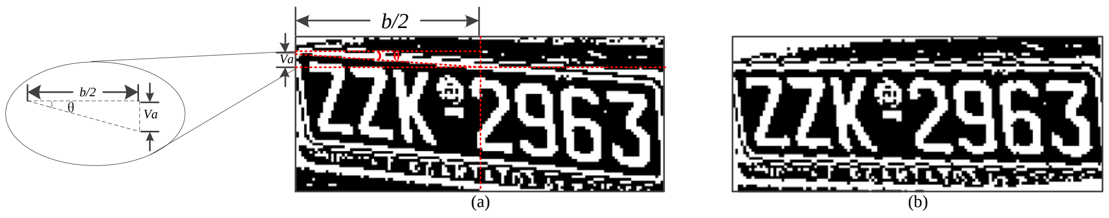


Figure 5 (a) Binarised NP image. (b) Rotated NP image.

As it can be seen from Figure 5(b), there are many non-NP pixels in the boundaries that are generated after rotation; therefore, a cropping process is needed to reduce the height of NP image. A simple method has been proposed to perform this operation. In this method, the rotated NP image will be cropped by V_a from the top and bottom of the NP which leaves a new NP height of $a - 2V_a$. The cropped NP image is shown in Figure 6.

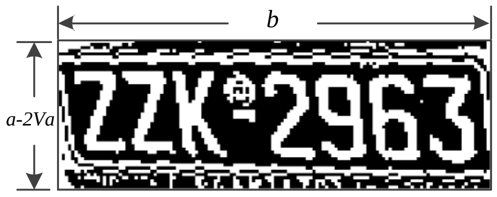


Figure 6: The cropped NP image.

From Figure 5(a), the following to trigonometric relation can be obtained:

$$tan\theta = \frac{V_a}{b/2} \tag{7}$$

Thus, the cropping parameter V_a can be calculated using the following equation:

$$V_a = tan\theta \times b/2 \tag{8}$$

2.2.2 Vertical Adjustment

After horizontal rotation, NP images may still need vertical rotation to adjust the slant as shown in Figure 7(a). Since there is a vertical slant, it is difficult to separate the character with CCA or other projection techniques. For such cases, a vertical adjustment method, which based on horizontal shifting of pixels, is proposed and presented in this section to solve the problem.

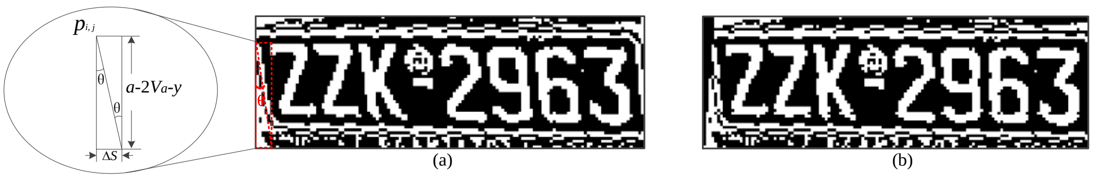


Figure 7 (a) NP image before vertical correction. (b) NP image after vertical correction.

In Figure 7(a), if an NP image was rotated horizontally with an angle θ the resulted NP image may have a vertical slant. To adjust this, the NP image must be shifted with a value Δs that depends on the horizontal rotation angle θ and a variable y which is the vertical coordinate of the pixel to be shifted. From Figure 7(a):

$$tan\theta = \frac{\Delta s}{a - 2V_a - y} \tag{9}$$

For a pixel $P_{i,j}$, y is equal j. Thus, the shifting value Δs for each pixel can be calculated using the following equation:

$$\Delta s = (a - 2V_a - j) \times \tan\theta \tag{10}$$

The shifted NP image after vertical adjustment is shown in Figure 7(b).

3. Proposed Pre-Processing Architectures

The proposed pre-processing architectures consist of the two main modules listed below:

1) ***Binarisation***: this module is used to convert the NP greyscale image to a binary image and is based on the local threshold binarisation method presented in Section 2.1.
2) ***Adjustment*****:** this module is used to calculate the horizontal rotation angle, perform 2-D horizontal image rotation, calculate the vertical shifting value and perform the vertical shifting operation. The module is based on the methods presented in section 2.2.

The block diagram of the proposed pre-processing modules is shown below in Figure 8.

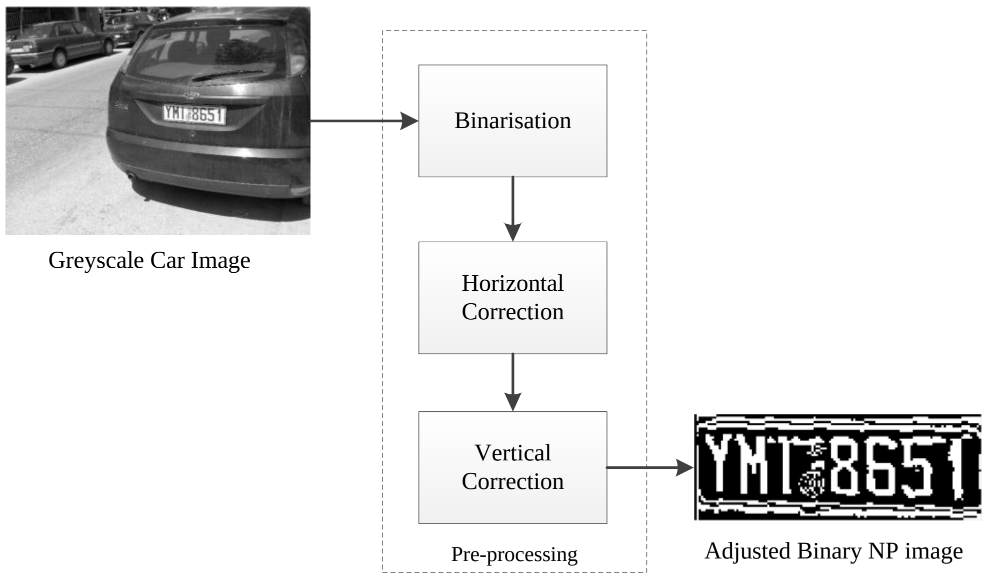


Figure 8: Block diagram of the pre-processing modules.

3.1 Binarisation Module

The Binarisation Module is the first module in the pre-processing stage which consists in three blocks which are the memory reader, mean and local threshold filters. The overall block diagram of the binarisation module is shown in Figure 9.

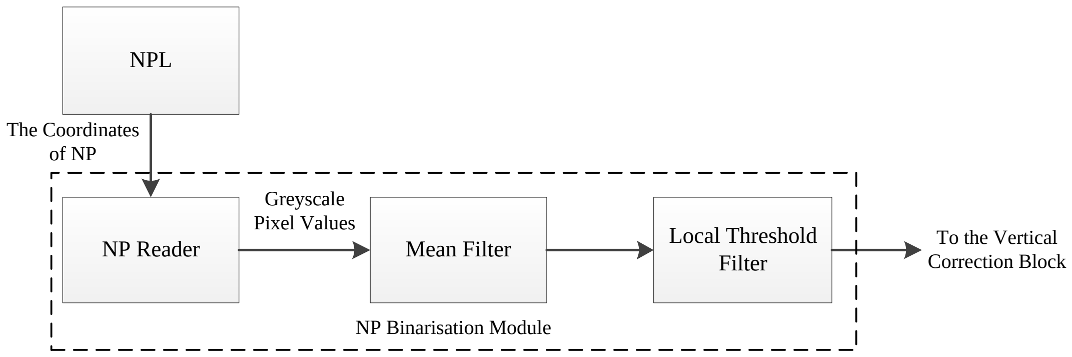


Figure 9: The overall block diagram of the binarisation module.

The input images to the NPL stage are 640×480 colour car images from two UK [18] and Greek databases [20]. During the NPL processing the input colour image is converted to greyscale and stored in one external memory. The greyscale image is used in further processing to generate the final outputs which consist of a binary car image and the NP region coordinates (i.e. top left and bottom right corners). The binary car image is stored in another external memory to be able to access it in parallel with the greyscale image [18].

In Figure 9, the NP reader first obtains the coordinates of NP region from NPL stage, and then uses them to calculate memory address of NP region. The greyscale pixel values of NP region are read from the first external memory where the greyscale image is stored and feed them to the mean filter block.

Let (x, y) denote the coordinates of a pixel in the NP rectangular region, (x_0, y_0) and (x_1, y_1) denote the left top and right bottom corner coordinates of the NP rectangular region respectively. The memory address of any pixel in the NP rectangular region can be calculated by:

$$address_{(x,y)} = 640 \times y + x, \quad x_0 \le x \le x_1 \ \& \ y_0 \le y \le y_1 \tag{11}$$

The pixels in the NP region are read column by column from left to right using the addresses calculated using the equation 11.

3.1.1 Mean Filter

The mean filter is the block that performs the main process of binarisation, which consists of a window shifter and an averaging filter.

Window Shifter

The window shifter is a 8×8 matrix used to buffer pixels from the NP image. This window shifter scans the NP image column by column from left to right. Figure 10 shows the architecture of the window shifter.

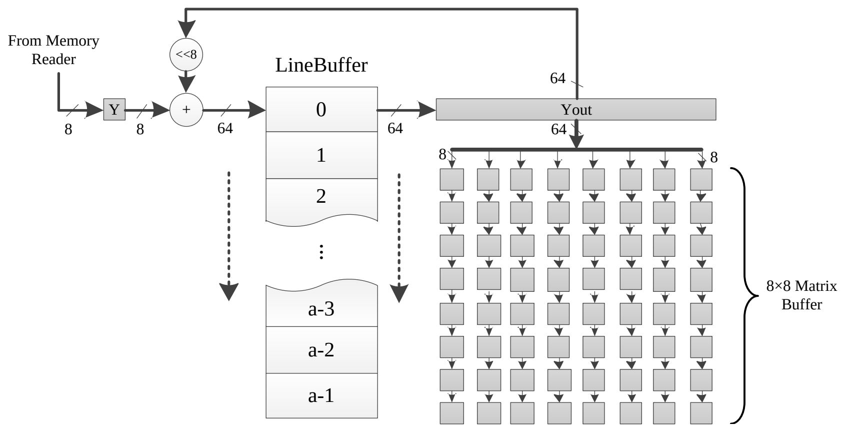


Figure 10: The window shifter.

In Figure 10, 'Y' is an 8-bit register used to temporarily store a pixel value from the NP Reader block. 'LineBuffer' is an $a \times 64$ dual-port RAM, where a is the height of the NP image, used to store NP pixels from eight columns and each memory location will contain eight pixels from the same row and to do this an eight-bit shifter and an adder are used. Thus, the content of each memory location in the 'LineBuffer' Y' is calculated using the following equation:

$$Y' = Y + Yout << 8 \tag{12}$$

Following this process, the first eight pixels from the first row of the NP image will be stored in the first memory location of 'LineBuffer' after $a \times 8$ clock cycles. After that the next eight-pixel rows will be stored in the corresponding location every clock cycle. ‘Yout’ is a 64-bit temporary register used to store the content of one location from ‘LineBuffer’ every clock cycle. Once all eight pixels from a row are saved in ‘Yout’ it will be transferred to the first row of the 8×8 matrix buffer which will be propagated to the next row every clock cycle. This process is repeated until the 8×8 matrix is full and transferred to the next module (i.e. averaging filter).

All steps described in this section need to be repeated for all pixels in the NP image.

Averaging Filter

The averaging filter consists of 21 adders and one 8-bit right shifter and is used to calculate the mean value of the 8×8 matrix buffer by simply adding all 8-bit values and divide the result by 64. The division is performed using the 6-bit right shifter. Figure 11 shows the architecture of the averaging filter.

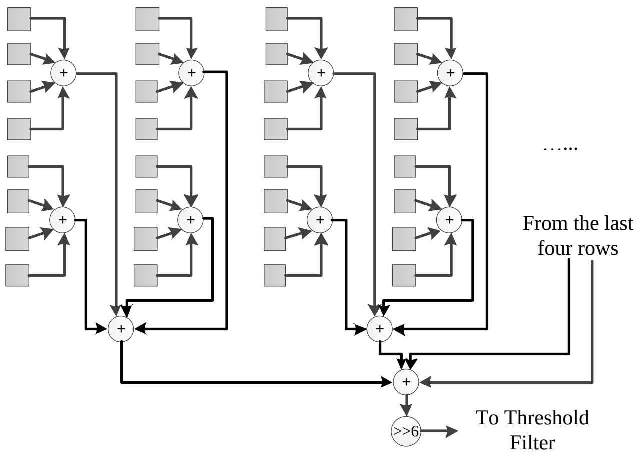


Figure 11: Architecture of the Average Filter.

To avoid long delay paths in the hardware implementation and as it can be seen from Figure 11, to obtain the final sum every four elements of are added together. The average value can then be calculated by right shifting the sum by eight bits.

Local Threshold Filter

The window shifter will be applied to the whole NP greyscale image pixel by pixel and each greyscale pixel will be the centre of the window. Local threshold filter module calculates a local threshold to be used to produce the binary value of the corresponding greyscale value of a pixel to produce a binary NP image. According to equation 2, the threshold can be calculated by applying the subtraction of average value and a constant Δt. This threshold is used as a condition to decide whether the current average value from the average filter should be set to '1' or '0'. The binary pixels will be stored in a 256×1 dual-port RAM, the adjustment module will read the pixels from this RAM simultaneously. Figure 12 shows the architecture of local threshold filter block.

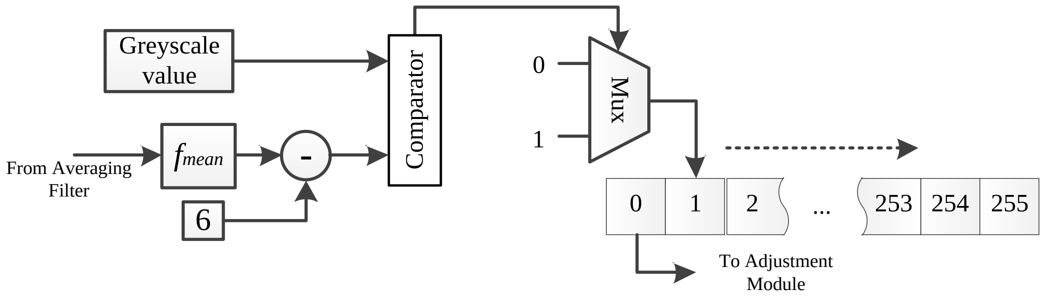


Figure 12: Architecture of the local threshold filter.

In Figure 12, the address of the 256×1 dual-port RAM is incremented by one every clock cycle, when it reaches the last location it will be initialised to '0'.

3.2 Adjustment Module

The adjustment module consists of three blocks which are rotation angle calculator, coordinates correction block and pixels reader block. The overall block diagram of the rotation module is shown in Figure 13.

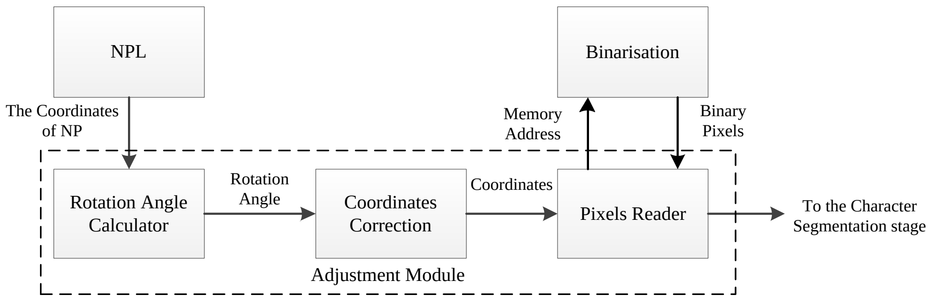


Figure 13: The overall block diagram of the rotation module.

3.2.1 Rotation Angle Calculator

Since the output image from NPL stage stored in the second external memory, the rotation angle calculator uses the coordinates of NP and calculates the addresses to read the binary pixels from the memory. According to Figure 4(a), the memory addresses for pixels in columns $x_0 + c$ and $x_0 + b - c$ can be calculated as follows:

$$MA_1(r_1) = 640 \times (y_0 + r_1 - 1) + x_0 + c \qquad (13)$$

$$MA_2(r_2) = 640 \times (y_0 + r_2 - 1) + x_0 + b - c \qquad (14)$$

Where x_0 *and* y_0 are the coordinates of the pixel in the left corner of NP, r_1 and r_2 are the NP row number, c is the offset constant and b is the width of the NP.

According to the proposed algorithm presented in section 2.2.1, the memory addresses MA_1 and MA_2 are calculated separately, where r_1 and r_2 are incremented by one until the first NP pixel with value ‘1’ is found, then their difference is calculated.

The rotation angle θ can be calculated using equations 4. In order to reduce the hardware usage and improve the performance $\alpha = 1/\tan\theta$ is calculated instead of the rotation angle θ as all needed calculations in the coordinates correction block are based on α.

$$\alpha = \frac{b - 2 \times c}{\Delta d} \qquad (15)$$

3.2.2 Coordinates Correction

The coordinates correction block performs horizontal and vertical adjustments. For horizontal adjustment, the main task is based on equations 5 and 6. However, in order to avoid the calculation of trigonometric functions and reduce hardware usage, simplified equations are used in this block.

The slant angles of NP images used from UK and Greek databases are always less than 10°. Therefore, the corresponding trigonometric functions *sin* and *cos* can be replaced *tan* and value '1' respectively as $\sin\theta \approx \tan\theta$ and $\cos\theta \approx 1$ when $|\theta| < 10^\circ$. Thus, equations 5 and 6 can be simplified as follows:

$$x_2 = x_1 - \frac{(y_1 - a/2)}{\alpha} \tag{16}$$

$$y_2 = y_1 + \frac{(x_1 - b/2)}{\alpha} \tag{17}$$

For vertical adjustment, the main task is based on equation 10. x_2 needs to be shifted horizontally by Δs in order to perform the vertical adjustment, thus, equation 16 can be written as:

$$x_2 = x_1 - \frac{(y_1 - a/2)}{\alpha} - \Delta s \tag{18}$$

Figure 14 illustrates the architecture of horizontal and vertical adjustments.

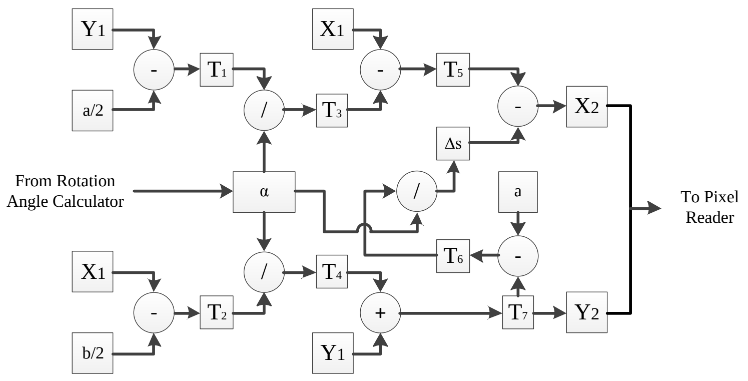


Figure 14: The proposed architecture for horizontal and vertical adjustments.

In Figure 14, ‘$T_1, T_2, ..., T_7$’ are buffers that are used to temporarily store intermediate results, which can efficiently reduce path delay and improve the throughput rate of the architecture. ‘X_1, Y_1’ and ‘X_2, Y_2’ are the registers used to store the original and new coordinates respectively. Each operation in equations 17 and 18 requires one clock cycle and data stored in the buffers are propagated from one buffer to the next every clock cycle. The final results are passed to the pixel reader block.

3.2.3 Pixel Reader

The pixel reader block reads binary pixels from the dual-port RAM in the local threshold filter block from the binarisation module. In this block two functions are performed:

1) Checking the new coordinates from coordinates correction block. If new coordinates exceed the boundary of the NP, they will be discarded.
2) Calculating the reading address and read the binary pixels from the dual-port RAM in the binarisation module, then feed them to two dual-port RAMs with sizes 256×1 and 2048×1 that will be used in CS stage.

The reading address of the dual-port RAM is calculated by:

$$MA_{read} = (x_2 \times a + y_2) / 256 \qquad (19)$$

Where x_2 and y_2 are the new coordinates, *a* is the height of the NP.

Figure 15 shows the proposed architecture of the pixel reader.

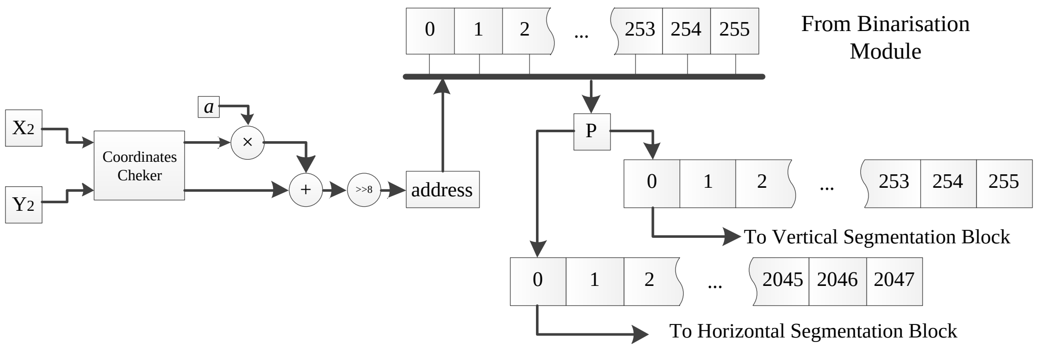


Figure 15: Architecture of the pixel reader.

In Figure 15, if the new coordinates stored in ‘X_2’ and ‘Y_2’ are valid coordinates, after passing the coordinate checker block, the corresponding binary pixel is read from the dual-port RAM in the binarisation module and stored the in the temporary buffer ‘P’. The stored value in ‘P’ will be simultaneously saved in the two dual-port RAMs to be used in the CS stage.

4. FPGA Implementation and Results

The proposed architectures for NP binarisation and adjustment have been simulated using the PAL Virtual Platform (PALSim) [21]. After simulation, the architectures have been successfully implemented and verified using the Mentor Graphics RC240 FPGA development board equipped with a 4M-Gate Xilinx Virtex-4 LX40 [22]. Handel-C has been used for hardware description of the proposed architecture, which is a high-level language that is at the heart of a hardware compilation system known as the Mentor Graphic DK Design Suite (DK). Handel-C has additional constructs to support parallelism and pipelining [23] [24].

The binarisation and adjustment modules run in parallel and pipelining has also been used in their implementation to achieve high throughput rate and an NP image can be processed by both modules in $(b+6)\times C$ clock cycles where:

- *b* is the width of the NP
- *6* is a constant delay that allows enough pixels to be stored in the dual-port RAM from the binarisation module
- *C* is the number of clock cycles required to complete binarisation for one column from the NP image

4.1 Proposed Environment for NP Binarisation and Rotation on FPGA

Figure 16 illustrates the proposed environment for NP binarisation and adjustment implementation. It contains a host application (GUI), NP database and the RC240 FPGA development board. The host application was developed using Visual Studio 2010 and gives the user the ability to select a car image from the database, display and send it to the FPGA for processing. Once processed, the localised, binarised and adjusted NPs are processed on the FPGA and send back to the host to be displayed in the same GUI.

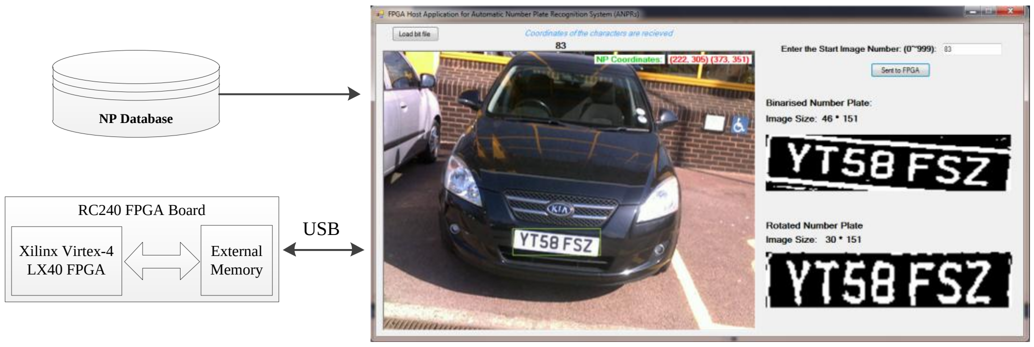

NP Database
RC240 FPGA Board
Xilinx Virtex-4 LX40 FPGA
External Memory
USB
FPGA Host Application for Automatic Number Plate Recognition System (ANPRs)
Load bit file
Coordinates of the characters are recieved
83
NP Coordinates: (222, 305) (373, 351)
Enter the Start Image Number: (0~999): 83
Sent to FPGA
Binarised Number Plate:
Image Size: 46 * 151
Rotated Number Plate
Image Size: 30 * 151

Figure 16: Host application for NP binarisation and adjustment.

4.2 Hardware Usage, Running Frequency and Power Consumption

Due to the low complexity of the proposed algorithms, the binarisation and adjustment architectures require only 9% of the on-chip FPGA resources. Table I summarises the required on-chip resources.

Table I: Usage of FPGA on-chip Resources

	Used	Available	Utilisation
Occupied Slices	1,763	18,432	9%
LUTs	2,649	36,864	7%
Block Rams	3	96	3%

According to our previous work [1, 17, 18] where new algorithms and architectures for NPL and CS stages were proposed and implemented on FPGA, the on-chip resources usage was 28% for NPL and 11% for CS. Therefore, the total hardware usage for NPL, pre-processing and character segmentation is 48%, leaving 52% of the FPGA area to be used for the remaining part of an ANPR system (i.e. character resizing and OCR).

The maximum running frequency is 95.8 MHz and the number of clock cycles needed for one image to be processed is 6297-16519, which depends on the resolution of the localised NP. The execution time for processing one frame can be calculated using the following equation:

$$T = \frac{c}{f} \tag{20}$$

Where *T* is the execution time; *c* is the number of clock cycles needed to process one image; and *f* is the maximum running frequency.

Based on equation 20, the proposed architecture can process one image (18×99 - 60×300) and produce a result in $0.07 - 0.17\ ms$. The difference in the execution time is due to the size of the images which affect the number of clock cycles. The smaller the size of the image, the lower the number of clock cycles is required. The execution times achieved mean that the proposed architecture satisfies the minimum requirement for real-time processing. The results achieved in terms of maximum running frequency and area used for implementing this part of the ANPR system show that there is enough room to implement the whole ANPR system on a single FPGA chip.

The power consumption of the designed circuit has also been analysed using Xilinx XPower [25], and the results obtained are shown in Table II.

Table II: Estimation of Power Consumption

Name of Power	Value of Power (mW)
Total Quiescent Power	446
Total Dynamic Power	236
Total Power	682

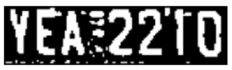
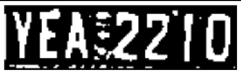
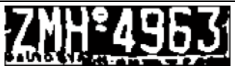
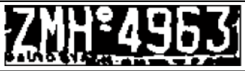
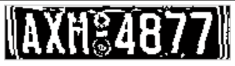
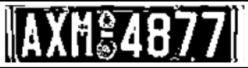
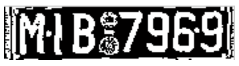
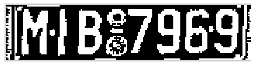
The total power consumption of FPGAs consists of quiescent and dynamic components. Table II shows that the total power consumption of the proposed architecture is 682 mW which is lower than the power consumption of a typical PC if it is used as the processing unit in an ANPR system.

4.3 Experiment Results

MATLAB implementations of the proposed algorithms were used as a proof of concept prior to the hardware implementation where floating-point arithmetic and functions from the image processing toolbox were used. However, the FPGA implementation uses a simplified integer based arithmetic. NP images from the Greek and UK databases have been used for testing the MATLAB and FPGA implementations.

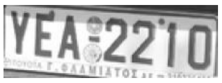
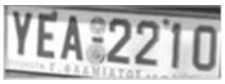
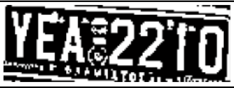
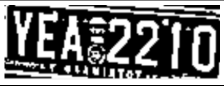
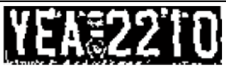
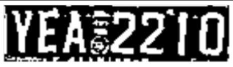
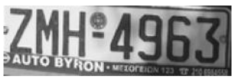
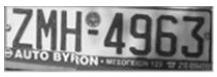
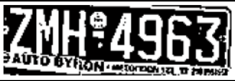
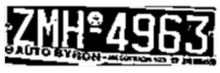
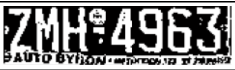
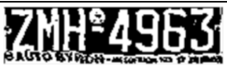
In order to compare the similarity of output images from MATLAB and FPGA implementations, the noise on the NP images is first removed using Gaussian filter and then 2-D correlation coefficient of the processed NP images are used to estimate the similarity of the two results [26]. As it can be seen from Table III the similarity of MATLAB and FPGA is around 67.2%.

Table III: Similarity result for MATLAB/FPGA

	MATLAB	FPGA	Similarity
NP Example 1			72.1%
NP Example 2			63.8%
NP Example 3			64.8%
NP Example 4			68.0%

In order to compare the software and FPGA-based implementations in term of the computation speed, the proposed algorithm has also been implemented in C using a PC equipped with an Intel Core i7 2.8GHz and 8G RAM. Table IV shows the results of the C and FPGA implementations in terms of computation time.

Table IV: C /FPGA result comparison

		C Implementation		FPGA Implementation	
		NP Image	Consumption Time	NP Image	Consumption Time
NP Example 1	Original greyscale NP		N/A		N/A
	Binarised NP		7 *ms*		0.11 *ms*
	Adjusted NP				
NP Example 2	Original greyscale NP		N/A		N/A
	Binarised NP		8 *ms*		0.12 *ms*
	Adjusted NP				

The images have been successfully binarised and adjusted using the proposed algorithms, where the characters are clearly isolated from each other and the vertical and horizontal positions of the NPs are properly adjusted. The FPGA processes a NP image 70 times faster than C implementation due to the low complexity of the proposed algorithms, the arithmetic techniques used, parallelism and pipelining exploited in the hardware implementation of the proposed architectures.

5. Conclusion

Current ANPR systems use high performance workstation as processing unit to meet the real-time requirement, however, the cost and power consumption issues that come with these systems have motivated researchers to look for other alternative platforms. Recent FPGAs have become a viable candidate for performing computationally intensive image processing task such as ANPR.

In this paper, two optimised low complexity NP binarisation and adjustment algorithms have been proposed to successfully link NPL and CS stages. Efficient area/speed architectures based on the proposed algorithms have also been presented and successfully implemented and tested using the Mentor Graphics RC240 FPGA development board, which together require only 9% of the available on-chip resources of a Virtex-4 FPGA, run with a maximum frequency of 95.8 MHz and are capable of processing one image in $0.07-0.17ms$.

The 91% remaining resources that allow the remainder of the ANPR system to be implemented on the same FPGA that can be placed within an ANPR camera housing to create a stand-alone unit which will remove the installation and cabling costs of bulky PCs situated in expensive, cooled roadside cabinets.

References

[1] X. Zhai and F. Bensaali, "Improved Number Plate Character Segmentation Algorithm and its Efficient FPGA Implementation," *Journal of Real-Time Image Processing,* 2012.

[2] S. Chang, Chen, L., Chung, Y. and Chen, S., "Automatic license plate recognition," *IEEE Transaction on Intelligent Transpotation Systerms,* **5**, 42-53 (2004).

[3] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A License Plate-Recognition Algorithm for Intelligent Transportation System Applications," *IEEE Transactions on Intelligent Transportation Systems,* , **7**, 377-392 (2006).

[4] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos and E. Kayafas "License plate recognition from still images and video sequences: A survey," *IEEE Transaction Intelligent Transportation System*, **9**, 377-391 (2008).

[5] X. Zhai, F. Bensaali and S. Ramalingam, "License plate localisation based on morphological operations," in *11th Int. Conf. Control Automation Robotics & Vision*, 2010, pp. 1128-1132.

[6] C. Arth, C. Leistner and H.Bischof, "TRIcam: an embedded platform for remote traffic surveillance," in *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, 2006, pp. 125-134.

[7] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location," *Pattern Recognition Letters,* **26**, 2431-2438 (2005).

[8] B. R. Lee, K. Park, H. Kang, H. Kim, and C. Kim, "Adaptive Local Binarization Method for Recognition of Vehicle License Plates," in *Combinatorial Image Analysis*. vol. 3322, J. Žunic, Ed., ed: Springer Berlin / Heidelberg, 2004, pp. 646-655.

[9] W. Jia, H. Zhang, and X. He, "Region-based license plate detection," *Journal of Network and Computer Applications,* **30**, 1324-1333 (2006).

[10] M. Pan, J. Yan, and Z. Xiao, "Vehicle License Plate Character Segmentation," *International Journal of Automation and Computing*, **05**, 425-432 (2008).

[11] X. Jia, X. Wang, W. Li, and H. Wang, "A Novel Algorithm for Character Segmentation of Degraded License Plate Based on Prior Knowledge," in *IEEE International Conference on Automation and Logistics*, 2007, pp. 249-253.

[12] X. Zhai, F. Bensaali, and R. Sotudeh, "OCR-Based Neural Network for ANPR," in *IEEE International Conference on Imaging Systems and Techniques*, Manchester, UK, 2012, pp. 393-397.

[13] N. Otsu, "A Tlreshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, **9**, 62-66 (1979).

[14] F. Yang, Z. Ma, and M. Xie, "A Novel Binarization Approach for License Plate," in *2006 1ST IEEE Industrial Electronics and Applications*, 2006, pp. 1-4.

[15] Y. Wen, Y. Lu, J. Yan, Z. Zhou, von Deneen K.M. and P. Shi, "An Algorithm for License Plate Recognition Applied to Intelligent Transportation System," *IEEE Transactions on Intelligent Transportation Systems*, **12**, 830-845 (2011).

[16] Y. Zhang and C. Zhang, "A new algorithm for character segmentation of license plate," in *IEEE Intelligent Vehicles Symposium*, 2003, pp. 106 - 109.

[17] X. Zhai, F. Bensaali, and S. Ramalingam, "Real-Time License Plate Localisation on FPGA," in *17th IEEE Workshop on Embedded Computer Vision and Pattern Recognition*, 2011, pp. 14-19.

[18] X. Zhai, F. Bensaali, and S. Ramalingam, "Improved Number Plate Localisation Algorithm and its Efficient FPGA Implementation," *IET Circuits, Devices & Systems*, 2012.

[19] H. Goldstein, *Classical Mechanics*, 2nd ed.: Addison-Wesley, 1980.

[20] Multimedia Technology Laboratory. *Medialab LPR Database*. Available: http://www.medialab.ntua.gr/research/LPRdatabase.html (Accessed on Jan, 2011)

[21] Mentor Graphics Corporation. *PAL User Manual*. Available: http://www.mentor.com/ (Accessed on June, 2011)

[22] Mentor Graphics Corporation. *RC240 Datasheet*. Available: http://www.mentor.com/ (Accessed on June, 2011)

[23] Mentor Graphics Corporation. *Handel-C User Manual*. Available: http://www.mentor.com/ (Accessed on June, 2011)

[24] F. Bensaali, A. Amira, and A. Bouridane, "Accelerating matrix product on reconfigurable hardware for image processing applications," *IET Circuits, Devices & Systems*, **152**, 236-246 (2005).

[25] Xilinx, Inc. *Xpower Tutorial: FPGA Design*. Available: http://www.xilinx.com/ (Accessed on June, 2011)

[26] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*: Psychology Press, 2002.