Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(6):1549-1556

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

Hardware architecture for real-time license plate character recognition based on EWM2DPCA

Boyu Gu[1], Qiang Zhang[2], Zhenhuan Zhao[2], Yechun Li[3] and Shuli Huo[4]

[1]*School of Electronics and Information Engineering, Changchun University of Science and Technology. Changchun, China*
[2]*Continental Automotive Corporation (Lian Yun Gang) Co. Ltd. Changchun Branch, Changchun, China*
[3]*Jilin Agricultural University. Changchun, China*
[4]*Seagate Technology International (Wuxi) Co. Ltd., Wuxi, China*

ABSTRACT

An embedded hardware architecture for license plate character recognition is designed and implemented on an FPGA (field programmable gate array). The architecture is based on EWM2DPCA (eigen weighted modular two-dimensional principal component analysis) which is presented in this paper. Three kinds of processing elements are included in the hardware architecture, projection element is designed for matrix multiplication operations of feature extraction, weighted distances between input character and each class in training database are computed in distance element, and the nearest neighbor classification is carried out in classification element. Parallel and pipeline acceleration technique are combined in the hardware architecture. Experimental results show that the recognition rate of the proposed algorithm is improved, and the hardware architecture achieves high performance, which is practical and reasonable.

Keywords: License plate recognition; Character recognition; Parallel processing; FPGA.

INTRODUCTION

Automatic license plate recognition technology has numerous important applications in people's daily life[1,2]. The main difficulty of license plate character recognition is that implementations on embedded applications should operate fast enough to ensure the whole system fulfills the real-time requirement[3].

To efficaciously extract the feature in a high dimensional space is extremely crucial for speed and accuracy of character recognition. Since high dimensional image data could projected into low dimensional eigen space, As a statistical method, the PCA[4] approach is very suitable for embedded applications. To improve the accuracy of varying illumination and angle, modular PCA[5] was proposed, and 2DPCA[6] was proposed to reduce the computational complexity. Then, modular 2DPCA[7] was proposed, it performances more efficient and robust. However, the contribution to recognition of each sub-block is considered equally in M2DPCA, some important local features are ignored. Therefore, more robust method is required to emphasize the local features which are more significant for recognition (such as the bottom right of "0" and "Q") and distinguish similar characters ("8" and "B", "2" and "Z", etc.). An eigen weighted M2DPCA algorithm is presented, the contribution of each sub-blocks are described as a weight, and the final decision is taken based on a weighted sum instead of average.

In a license plate recognition system, very little time is available for character recognition. In recent years, many researchers have implemented character recognition for practical applications on FPGAs[8,9] since high speed can be achieved by means of embedded DSP units and parallel processing[10]. Since the processing elements for all sub-blocks could operate in parallel, FPGA implementations for character recognition based on M2DPCA could achieve a remarkable high speed.

This paper focus on achieving an embedded real-time license plate character recognition architecture on FPGA. An eigen weighted M2DPCA algorithm is presented and applied to handle the complexity of similar characters in license plates. The projection element and distance element are explored to operate the data of image sub-blocks, and the classification element is designed for nearest neighbor classification. The arithmetic and logic functions are described in Verilog HDL.

EXPERIMENTAL SECTION

2.1 Eigen Weighted M2DPCA

In the proposed algorithm, M2DPCA approach is applied to character images for feature extraction, and all sub-blocks are weighted according to its contribution for recognition in eigen space. Finally, the input character is classified by weighted distance. (See Refs. 5, 6 and 7 for details of PCA and M2DPCA).

2.1.1 Computation of eigen weights

Suppose that p classes are included in training database, each class contains q images, every image is divided into s sub-blocks (s=b^2), and the size of images is $m\times n$. Every image is a training sample, each sub-block is regarded as an $m/b\times n/b$ image matrix, and $p\times q\times s$ matrix are included in the sample space $\boldsymbol{I}=\{\boldsymbol{I}_{111}, \boldsymbol{I}_{112}, \dots \boldsymbol{I}_{11b}, \dots \boldsymbol{I}_{11s}, \boldsymbol{I}_{211}, \dots \boldsymbol{I}_{pqs}\}$. The kth sub-block of average character is described as:

$$\boldsymbol{a}_{\boldsymbol{k}} = \frac{1}{pq}\sum_{i=1}^{p}\sum_{j=1}^{q}\boldsymbol{I}_{\boldsymbol{ij}k} \tag{1}$$

Centralize every image matrixes: $\boldsymbol{Z}_{\boldsymbol{ijk}}=\boldsymbol{I}_{\boldsymbol{ijk}}-\boldsymbol{a}_{\boldsymbol{k}}$. Each centralized image matrix can be reformed to an r dimensional column vector (r=$m\times n/s$), all centralized image vectors compose a sample set $\boldsymbol{Z}$ which can be divided into s sub-sets. For any sub-set $\boldsymbol{Z}_{\boldsymbol{k}}$, each row can be considered as an l dimensional variable vector (l=$p\times q$). Therefore, $\boldsymbol{Z}_{\boldsymbol{k}}$ can be described as:

$$\begin{aligned}\boldsymbol{Z}_k &= \{\boldsymbol{Z}_{11k}, \boldsymbol{Z}_{12k} \dots \boldsymbol{Z}_{\boldsymbol{ijk}} \dots \boldsymbol{Z}_{pqk}\} \\ &= \{\boldsymbol{Y}_{k1}{}^{\mathrm{T}}, \boldsymbol{Y}_{k2}{}^{\mathrm{T}} \dots \boldsymbol{Y}_{kt}{}^{\mathrm{T}} \dots \boldsymbol{Y}_{kr}{}^{\mathrm{T}}\}^{\mathrm{T}}\end{aligned} \tag{2}$$

Where $\boldsymbol{Y}_{\boldsymbol{kt}}$ is the tth variable vector in $\boldsymbol{Z}_{\boldsymbol{k}}$. $\boldsymbol{Z}_{\boldsymbol{k}}$ is a space consisted of l image vectors, each vector is a r dimensional variable; also, the space $\boldsymbol{Z}_{\boldsymbol{k}}$ can be considered as a space consisted of r variable vectors. The covariance matrix of $\boldsymbol{Z}_{\boldsymbol{k}}$ can be obtained from statistics of the variable vectors:

$$\boldsymbol{V}_k = \frac{1}{r}\sum_{t=1}^{r}\boldsymbol{Y}_{kt}{}^{\mathrm{T}}\boldsymbol{Y}_{kt} \tag{3}$$

Compute all eigenvalues $\boldsymbol{\lambda}_{\boldsymbol{k}}=\{\lambda_{k1}, \lambda_{k2}, \dots \lambda_{kl}\}$ of $\boldsymbol{V}_{\boldsymbol{k}}$ and corresponding eigenvectors $\boldsymbol{e}_{\boldsymbol{k}}=\{e_{k1}, e_{k2}, \dots e_{kl}\}$. Eigenvalue describes the deformation in the direction of the corresponding eigenvector. The l eigenvalues of $\boldsymbol{V}_{\boldsymbol{k}}$ describe the energy distribution of l samples, the lager deformation energy, the higher contribution in the corresponding direction. The contribution of a sub-block in $\boldsymbol{Z}_{\boldsymbol{k}}$ is:

$$\Lambda_{fk} = \sum_{h=1}^{l}\lambda_{kh}\,(e_{kh,f}\big/v_{k,ff})^2 \quad f = 1,2\dots pq \tag{4}$$

Where f=$i\times j$, and Λ_{fk}=Λ_{ijk}. $e_{kh,f}$ is fth element of hth vector in $\boldsymbol{e}_{\boldsymbol{k}}$, $v_{k,ff}$ is the fth element on diagonal of $\boldsymbol{V}_{\boldsymbol{k}}$. Human can distinguish characters according to the representative part which can be considered as a part with higher contribution (such as the lower right of “0” and “Q”), and also, different sub-blocks of a character in sample space exist different contribution, as well as the projection of each sample in the eigen space.

Normalize the contribution of each sub-blocks of same sample to meet the condition that $w_{ij1}+w_{ij2}\dots+w_{ijs}=1$. The mean weight of kth sub-block in ith class is computed as:

$$w_{ik} = \frac{1}{q}\sum_{j=1}^{q}(\Lambda_{ijk}\Big/\sum_{t=1}^{s}\Lambda_{ijt}) \tag{5}$$

2.1.2 Weighted distance classification

Project the sample into eigen space. Suppose that $\boldsymbol{\mu}$ is the mean of all matrixes in $\boldsymbol{I}$, the projection of *k*th sub-block in *i*th class is:

$$\boldsymbol{\eta}_{\boldsymbol{ik}} = \frac{1}{q}\sum_{j=1}^{q}\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{I}_{\boldsymbol{ijk}} - \boldsymbol{\mu}) \tag{6}$$

Where $\boldsymbol{E}$ is eigenvectors corresponding to largest ε eigenvalues of the covariance matrix which is formed by statistics of sample space $\boldsymbol{I}$. The size of $\boldsymbol{E}$ is $m/b\times\varepsilon$, each $m/b\times n/b$ dimensional matrix in sample space is represented as a $\varepsilon\times n/b$ dimensional matrix in eigen space, $\varepsilon\leq m/b$ (generally, ε is much less than m/b). The Euclidean distance between *k*th sub-block of test sample and the same sub-block of *i*th class database is: $d_{ik}=\|\boldsymbol{\eta}_{\boldsymbol{test,k}}-\boldsymbol{\eta}_{\boldsymbol{ik}}\|$, and the weighted distance between test sample and *i*th class is:

$$\omega_i = \sum_{k=1}^{s} w_{ik} d_{ik} \tag{7}$$

Finally, the test sample is recognized as belonging to Γth class when Γ=argmin($\boldsymbol{\omega}$), where $\boldsymbol{\omega}=\{\omega_1, \omega_2, \ldots \omega_p\}$.

2.2 Hardware Architecture Design

2.2.1 Simplified equation

A simplified equation is introduced to simplify the operations for hardware implementation. The d_{ik} can also be computed as:

$$\begin{aligned} d_{ik} &= \left\| \boldsymbol{E}^{\mathrm{T}}(\boldsymbol{I}_{\boldsymbol{test,k}} - \boldsymbol{\mu}) - \frac{1}{q}\sum_{j=1}^{q}\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{I}_{\boldsymbol{ijk}} - \boldsymbol{\mu}) \right\| \\ &= \left\| \boldsymbol{E}^{\mathrm{T}}\boldsymbol{I}_{\boldsymbol{test,k}} - \frac{1}{q}\sum_{j=1}^{q}\boldsymbol{E}^{\mathrm{T}}\boldsymbol{I}_{\boldsymbol{ijk}} \right\| = \left\| \boldsymbol{E}^{\mathrm{T}}\boldsymbol{I}_{\boldsymbol{test,k}} - \boldsymbol{\gamma}_{\boldsymbol{ik}} \right\| \end{aligned} \tag{8}$$

In this equation, $\boldsymbol{\mu}$ is removed, and $\boldsymbol{E}^{\mathrm{T}}$, $\boldsymbol{\gamma}_{\boldsymbol{ik}}$ can be obtained off-line.

2.2.2 Hardware architecture on FPGA

The parallel and pipeline acceleration technique are combined in the hardware architecture. There are three novelties in the architecture. First, the architecture can be considered as a parallel accelerator, all the *s* sub-blocks can be processed simultaneously. Second, in order to enhance the efficiency of the architecture and minimize the idle time of each element, the distance element and classification element are run in pipeline. At last, the resource occupation can be optimized because of no matter how many operation units are instantiated, only one classification element is required.

The frame of proposed hardware architecture is shown in Fig. 1. An operation unit is consisted of a projection element and a distance element, the amount of which is equal to the number of sub-blocks, and only one classification element is needed in the architecture.

Since *s* operation units are contained in the hardware architecture, all the sub-blocks of input character can be transmitted to the operation units simultaneously and processed in parallel.

The projection element is designed to project a sub-block of input character into eigen space, which is essentially a matrix multiplier. The function of distance element is computing weighted Euclidean distances between input character and each class in training database. And nearest neighbor classification is carried out in the classification element. Data obtained in training phase which includes the eigenvectors, eigen weights and low dimensional eigen space projections of training database is stored in on-chip ROMs (read only memory).

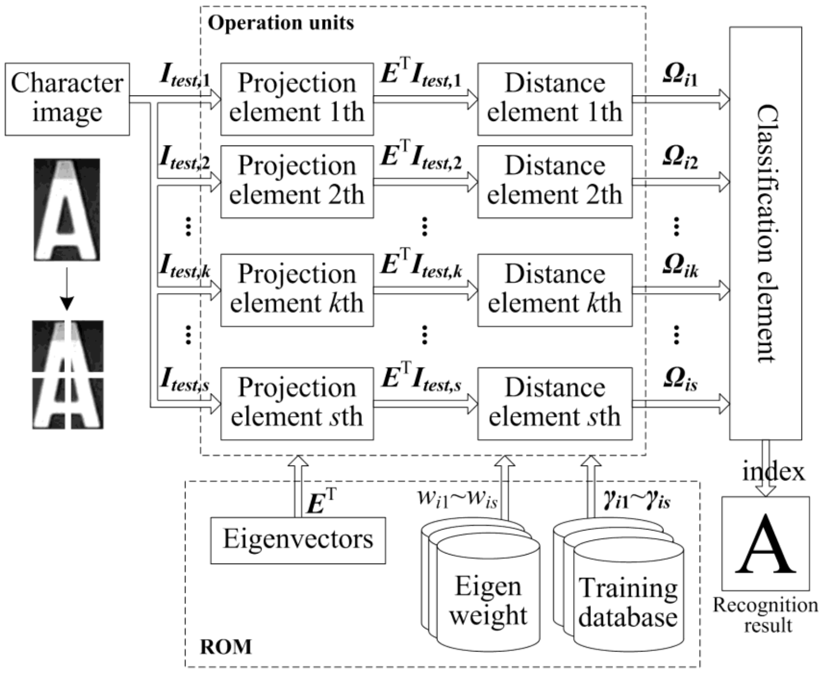


Fig. 1. Frame of hardware architecture

The core of proposed architecture is the processing elements which are shown in Fig. 2. The *k*th operation unit is described in detail, and others with same structure are omitted.

Matrix multiplication is actually a series of addition and multiplication, which can be performed by a multiplier, an adder, and a register in the projection element. The result is stored in an on-chip RAM (random access memory) instead of registers based on LC (logic cell), since amount of memory bits in FPGA is generally much greater than LCs, the matrix operation is more resource economical.

Euclidean distance is computed and multiplied by the eigen weight in the distance element. The square operated by a multiplier, the two inputs of which are connected to same data source, or a LUT (look-up table) is also practicable. The square root is computed by a LUT, in fact, the square root is not necessary.

In the classification element, a parallel adder is used to summate the weighted distances of all sub-blocks between the input character and each class in training database. The minimum distance and corresponding class index is obtained by the comparer.

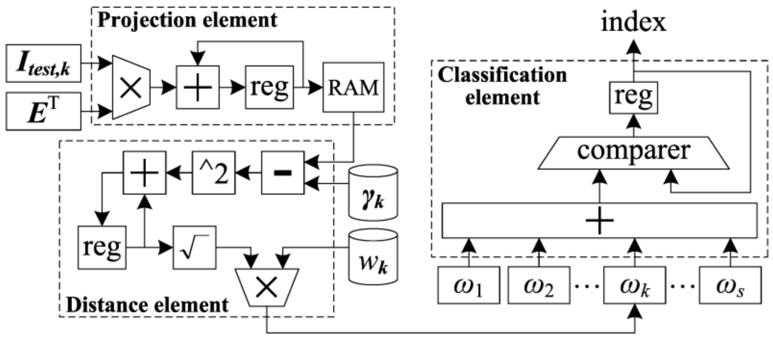


Fig. 2. Structure of processing elements

2.2.3 Timing analyze

The timing sequence of hardware architecture is shown in Fig. 3. There are *p*+2 periods included in recognition procedure when *p* classes are contained in training database. The projection element is active only in the first period, the classification element is disabled in first two periods, and the class index of the input character is outputted in last period. Except the first, second and last periods, all other *p*-1 periods are identical, and the (*i*+1)th period is demonstrated in detail.

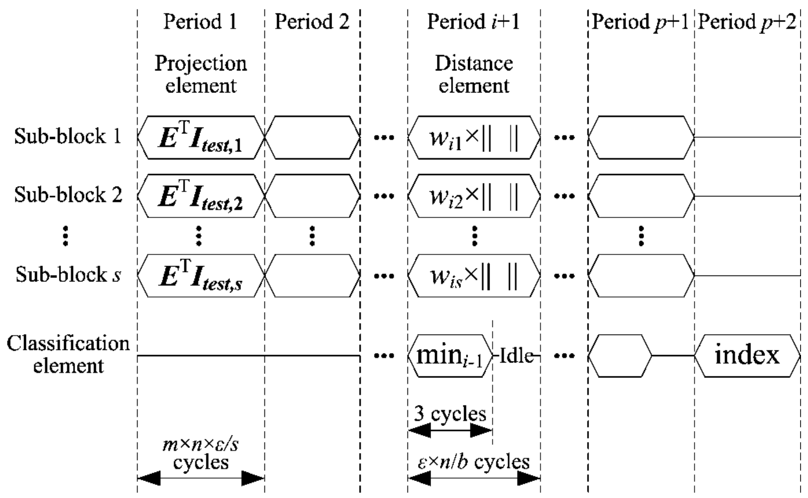


Fig. 3. Timing sequence of hardware architecture

Since there are *s* projection elements and distance elements are instantiated, all sub-blocks are processed in parallel. The distance between input character and (i-1)th class is obtained in each of 2th~(p+1)th period.

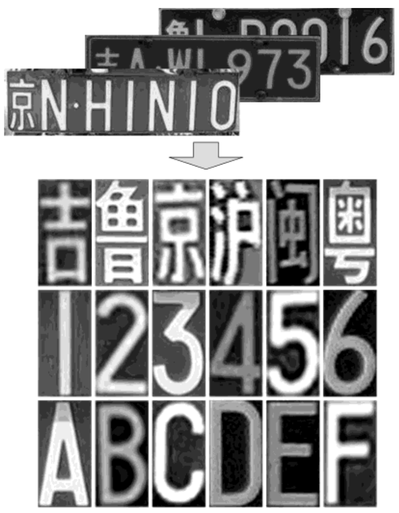
In Fig. 3, $\min_i$ means the minimum distance of first *i* classes, since the the distance element and classification element are run in pipeline, the $\min_i$ can be obtained in each of 3th~(p+2)th period, and therefore the minimum distance of all classes is determined in the last period.

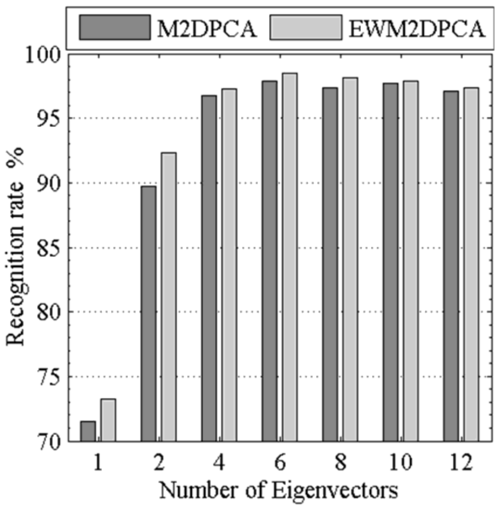
In the first period, $m\times n\times\varepsilon/s$ cycles are needed for the projection elements, the arithmetic and logic components lead to 3 cycles latency. For any other period, the weighted distance computing takes $\varepsilon\times n/b$ cycles and causes 6 cycles latency, and the $\min_i$ is computed in 3 cycles. The total time for character recognition is $m\times n\times\varepsilon/s+\varepsilon\times n/b\times p+12$ clock cycles.

RESULTS AND DISCUSSION

3.1.1 Improvement of eigen weighted M2DPCA

The performance of eigen weighted M2DPCA is tested and compared with conventional M2DPCA on a character database which is based on Chinese license plate. The database is consisted of 65 classes, include 31 Chinese characters, 24 upper case English letters (“O” and “I” are not contained in any Chinese license plate), and 10 Arabic numbers. Each class contains 320 samples, and there are 20 800 grayscale images in total.


(a)



(b)

Fig. 4. (a) Examples in database. (b) Recognition rate Comparison

In the database, 160 images of each class are used for composing the training set, and the others are for testing. Fig. 4(a) shows some examples in the character database, and The recognition rate of conventional M2DPCA and eigen weighted M2DPCA for varying number of eigenvectors (1~12) are demonstrated in Fig. 4(b).

The optimum number of eigenvectors is 6, the maximum recognition rate of conventional M2DPCA is 97.85%, and the proposed eigen weighted M2DPCA reaches up to 98.46%.

Keep the training set unchanged, and the similar characters are selected for composing the testing set. Table 1 shows the recognition rate of similar characters, experiment result shows that the accuracy rate is increased.

Table 1. Recognition rate of similar characters

Characte r	Recognition rate (%)	
	M2DPCA	EWM2DPCA
0	88.75	91.25
Q	91.88	93.75
D	91.25	92.50
2	93.13	95.00
Z	94.38	96.25
5	90.63	93.13
S	91.88	94.37
7	95.00	96.88
T	93.13	95.63
8	88.13	91.25
B	89.38	91.88
Average	91.59	93.81

The speed of the algorithm directly influences its applicability. The speed comparison of eigen weighted M2DPCA and conventional M2DPCA is shown in Table 2. The software implemented is based on C programming language on VC++ 6.0.

Table 2. Algorithm speed comparison

	Software speed		Hardware speed	
	M2DPCA	EWM2DPCA	M2DPCA	EWM2DPCA
Training	1174.9 s	1522.5s	--	--
Recognition	184.2 µs	187.1 µs	99.1 µs	100.4 µs

The speed of both software and hardware implementation based on Eigen weighted M2DPCA are compared with the conventional M2DPCA. The function of training step is obtaining the parameters off-line, which is not needed in the hardware implementation, and only needed to be run once on PC. As shown in Table 2, since the weight of each sub-block needed to be computed in training step, more time is consumed in Eigen weighted M2DPCA, however, in recognition step, the Eigen weighted M2DPCA takes only 2.9 µs more for software implementation, and 1.3 µs more for hardware implementation, the speed is almost the same.

It can be known that the Eigen weighted M2DPCA can achieve higher recognition rate while maintaining same speed.

3.1.2 Performance of hardware architecture
The hardware architecture is implemented with Verilog HDL utilizing Quartus II synthesis software, the target FPGA is an Altera Cyclone IV chip, which contains 114 480 LCs, 3 981 312 memory bits, and 532 embedded multipliers. In praxis, all images are scaled to 24×48, and the images are divided into 2×2 sub-blocks, ε is set to 6 for feature extraction, and the width of eigen space projection is 32 bit.

Resource consumption of the hardware architecture is shown in Table 3. 1074 LCs are needed to compose one operation unit, and the classification element occupies 502 LCs. With all other functions (such as image input/output, memory and synchronization control, etc), the architecture consumes 6% LCs, 13% on-chip memory bits, and 14% embedded multipliers in total.

Table 3. FPGA resource consumption

	Consumed	Total on-chip
Logic cells	6964	114 480
On-chip memory bits	524 960	3 981 312
Embedded multipliers	74	532

In the hardware architecture, 1747 cycles are needed for projecting the input character into eigen space, 4876 cycles are taken to compute the weighted distances, and the class index of minimum distance is obtained in 68 cycles. Speed of hardware architecture and equivalent software implementation are shown in Table 3.

The hardware architecture is capable of recognizing 9960 characters in one second. The recognition time is not fully match the timing analyze (Section 3.3) because of the latency such as memory addressing and image inputting are involved in practical application. Since a Chinese license plate contains 7 characters, the hardware architecture takes about 0.8 ms to recognize a license plate, meet the real-time requirement.

All character recognition functions in the hardware architecture are run at 66 MHz, compare it against the software implementation on an AMD dual-core 2.6 GHz PC with Matlab 7.6 and VC++ 6.0.

As shown in Table 4, the recognition time of software implementation is 187.1 μs, and the hardware architectures have performed significantly better. It can be known that the performance of the hardware is encouraging, which provides approximately 1.86 times speed of equivalent software implementation.

Table 4. Recognition speed

	Hardware	Software	Software
Target device	EP4CE115F29C7	PC	PC
Synthesis tool	Quartus II 12.0	Matlab 7.6	VC++ 6.0
Total clock cycles	6626	--	--
Clock frequence	66 MHz	--	--
Recognition time	100.4 μs	297.4 μs	187.1 μs

As the image size increases, the dimension of image matrix increases, and also the optimum number of eigenvectors. It is evident that the recognition speed is related to the size of character images. Speeds of the hardware architecture with respect to varying sizes of the images are shown in Table 5.

Table 5. Speed of characters with varying sizes

Size of character image	Recognition time
16×32	62.2 μs
24×48	100.4 μs
32×64	191.5 μs
48×96	414.4 μs
64×128	754.1 μs

The performance of the proposed hardware architecture can be evaluated by speedup (S_p) and efficiency (E_p). As seen from Table 4, the recognition time of software implementation T_1=187.1 μs; the recognition time of proposed hardware architecture T_p=100.4 μs; and the number of parallel modules P=4. The speedup and efficiency can be described as:

$$\begin{cases} S_p = T_1 / T_p = 1.86 \\ E_p = S_p / P = 0.465 \end{cases} \tag{9}$$

That means one more parallel module can achieve about 46.5% speedup. Since Ref. 5 indicates that the accuracy rate will be reduced significantly when the image is divided into more than 8×8 sub-blocks, the maximum number of parallel module is 64, and in that case, the speedup can reaches up to 29.76, in other words, the maximum achievable speedup is 29.76 times of the software implementation of C programming language.

CONCLUSION

High performance real-time hardware architecture for license plate character recognition was designed and implemented on an FPGA. An eigen weighted M2DPCA algorithm is presented to adapt the complexity of similar characters. The architecture achieved approximately triple times speed of equivalent software implementation. Experimental results indicate that PCA approaches are suitable for license plate character recognition on FPGA embedded applications.

The eigen weighted method is applied with M2DPCA in this paper, but it could be used with other algorithms, such as PCA, kernel PCA, and also ICA approaches. Although the architecture was tested on a database of Chinese license plates, it can be modified to adapt other license plates. The recognition speed of the hardware is related to the clock frequence, we did not have a device for testing, but there is no reason that a faster hardware would not be achieved on

FPGAs with faster speed grade. Although the hardware meets the real-time requirement, faster speed can be achieved by parallel every column or row of matrix operations.

This paper focuses on the character recognition step of automatic license plate recognition system. The license plate location and character segmentation steps will be combined together with the character recognition. Achieving higher speed and parallelism is the focus of future work, and then, entire license plate recognition system will be implemented on an FPGA.

REFERENCES

[1] Y.S. Huang, Y.S. Weng, and M.C. Zhou: *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 968-977, Dec. **2010**.
[2] O.A. Omitaomu, A.R. Ganguly, B.W. Patton, and V.A. *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 324-334, Jun. **2009**.
[3] C.N.E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas: *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377-391, Sep. **2008**.
[4] M. Turk, A. Pentland: Face recognition using eigenfaces, IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–591, **1991**.
[5] R. Gottumukkal, V.K. Asari: *Pattern Recognition Letters*, vol. 25, no. 4, pp. 429-436, Mar. **2004**.
[6] J. Yang, D. Zhang, A.F. Frangi: Two-dimensional PCA: a new approach to appearance-based face representation and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, pp. 131–137, Jan. **2004**.
[7] Fubing Chen, Xiuhong Chen, Shengliang Zhang, and Jingyu Yang: *Image and Graphics*, vol. 11, no. 4, pp. 580-585, **2006**.
[8] Rice K L, Bhuiyan M A, Taha T M, et al.: FPGA implementation of Izhikevich spiking neural networks for character recognition, Reconfigurable Computing and FPGAs, 2009. ReConFig'09. International Conference on. IEEE, **2009**: pp. 451-456.
[9] Tokunaga Y, Inoue T. A method for circular pattern recognition in a binary image and its implementation onto an FPGA. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 82, no. 2, pp. 246-254, Feb. **1999**.
[10] M.A. Bhuiyan, R. Jalasutram, and T.M. Taha: Character recognition with two spiking neural network models on multi-core architectures, IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing, Nashville, Mar.30-Apr.2, **2009**, pp. 29-34.