Journal of AI and Data Mining
Vol 4, No 1, 2016, 9-17 *10.5829/idosi.JAIDM.2016.04.01.02*

Isolated Persian/Arabic handwriting characters: Derivative projection profile features, implemented on GPUs

M. Askari, M. Asadi, A. Asilian Bidgoli*and H. Ebrahimpour

Department of Computer Engineering, University of Kashan, Kashan, Iran.

Received 14 December 2014; Accepted 28 November 2015
**Corresponding author:asilian@gmail.com (A. AsilianBidgoli).*

Abstract
For many years, researchers have studied high accuracy methods for recognizing the handwriting and achieved many significant improvements. However, an issue that has rarely been studied is the speed of these methods. Considering the computer hardware limitations, it is necessary for these methods to run in high speed. One of the methods to increase the processing speed is to use the computer parallel processing power. This paper introduces one of the best feature extraction methods for the handwritten recognition, called DPP (Derivative Projection Profile), which is employed for isolated Persian handwritten recognition. In addition to achieving good results, this (computationally) light feature can easily be processed. Moreover, Hamming Neural Network is used to classify this system. To increase the speed, some part of the recognition method is executed on GPU (graphic processing unit) cores implemented by CUDA platform. HADAF database (Biggest isolated Persian character database) is utilized to evaluate the system. The results show 94.5% accuracy. We also achieved about 5.5 times speed-up using GPU.

Keywords: *OCR, Persian Character, Projection Profile, Parallel Algorithms, GPU, CUDA.*

1. Introduction

There are many definitions for OCR (optical character recognition); however, the best definition is the procedure of converting visual texts by an electrical or mechanical machine into intelligible codes (like ASCII) for a computer [1]. OCR systems have many applications in sorting letters, reading, information checking, and converting manuscripts to digital texts. OCR is mainly divided into two categories: printed and manuscript recognition. Manuscript recognition is always more difficult due to the variety of handwritings. Another categorization is the division between online and offline types. In an offline system, a fixed image, usually a bitmap, is given for recognition, while in an online system, writing text and the recognition process occur simultaneously. Instances of such recognitions can be witnessed in mobile phones and tablets. Due to temporal criteria like pen movements, recognition is easier with an online system.

Recently, researchers have shown high interest in the manuscript recognition field; in the English language, such systems have achieved a high accuracy level up to 98% [2] and THEY ARE CURRENTLY BEING USED in practice. On the other hand, while more than half a billion people in the world use languages with the same form of writing as the Persian form (like Arabic), little research has been carried out on such languages and most research studies have worked on Chinese, Japanese, and English forms of writing [3]. This is due to the lack of financial support, tools like databases, dictionaries, and the complex nature of these writing forms [4]. The production of Persian/Arabic OCR can have other advantages like the recognition of historical scripts. Persian/Arabic forms of writing have not had a great deal of change in the literature. Therefore, such systems can also be used for the recognition of historical scripts [5].

The remainder of this paper is divided into several sections. The following section presents an introduction to Persian characters and reviews past works in OCR. Section 2 explains a procedure of character Recognition. The proposed method and its parts are presented in section 3.

Section 4 provides an explanation about CUDA and its utilization in this paper. Finally, the implementation of the proposed method and the conclusions are respectively provided in sections 5 and 6.

1.1. Persian Scrip

Unlike English, Persian scripts (also known as Farsi) are written from right to left, letters have no capital form, and are linked together in order to form a word.

The Persian alphabet has 32 letters and the Arabic alphabet has 28 letters. Each letter in a word can have one of the three positions: initial, central, or final. The letters in the final position can sometimes be isolated from their preceding letters as figure 1. Dots in Persian play an important role; many of the letters like 'پ', 'ب' and 'ت' have the same main body and the only difference between them is the position and the number of dots.

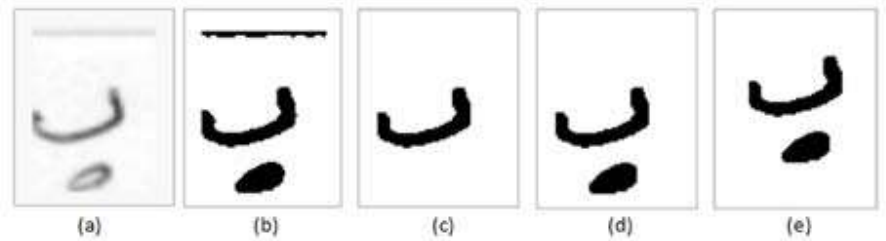

Figure 1. Preprocess stages. a)Main image. b) Binerized image. c) Biggest object of character. d)After applying morphology filter on b. e)Centralized action on d.

Lack of communication among Persian/Arabic OCR researchers is the cause of many repeated studies carried out in this field. Moreover, in the past, there was not any standard database to be used for comparison and each person used a different database for testing; therefore, it is not easy to compare different studies. For this reason, it is useful to set up competitions among researchers in this field. One of these competitions was ICDAR 2009 on the isolated Persian characters [6]. In this competition, the proposed systems were compared based on HODA, IFHCDB, and CENPARMI databases. The best result was obtained by a system based on a neural network.

1.2. Background

Persian OCR was first investigated in 1980s by Parhami, which could recognize some newspaper characters [7]. At the same time, another research was carried out by Badi on the Arabic alphabet. That system could recognize 12 printed letters with 90% accuracy.

In [8], Mozaffariet.al. proposed a new method for manuscript recognition. They used both structural and statistical features and the 1NN (nearest neighborhood) classifier. Their database consisted of 8 digits and they used 280 images for training and 200 images for testing. The accuracy was reported 94.4%.

In [9], a system is proposed for printed letters using chain code. In this system, the first image is thinned and then statistical features, such as the number of the dots, the position of the dots, the number of the holes, and the statistical features extracted from the chain code, are sent to the 1NN classifier. This system was proposed for the recognition of different fonts, i.e. training is carried out with one font and testing with another. The average accuracy of this system was 97%.

In [10], [11], Haraty and Ghader use skeleton presentation for preprocessing. The structural and statistical features applied in these methods include the number and density of black pixels and the number of joint points, final points, loops, and sides. The classifiers of these systems were two neural networks. The recognition rate of these systems was 73% using a database of 2132 characters.

Safabakhsh and Adibi used continuous HMM (Hidden Markov Model) for the recognition of Nastaaligh manuscripts (a type of Persian calligraphy in which letters are slanted and some letters fall over the others to produce an aesthetic effect). In this type of writing, the recognition of base lines is difficult. Before recognition, this system removes slopes to prevent incorrect ordering of letters. This system uses a lexicon of 50 words, including all types of characters. The test data consisted of 100 words written by two people. Their method was recursive, i.e. at each stage, falsely recognized words were ignored, and the test was repeated. This system could reach 69% accuracy rate after 5 iterations and 91% accuracy rate after 20 iterations [12]. In [13], a system is proposed, which is a combination of MLP (Multi-Layer Perceptron) classifiers. The preprocessing stage in this system is smooth and

binary. The feature vector includes structural features like the number of upper and lower dots and statistical features like densities of 57 regions in the image. They used 7 voting methods, such as MAX and SUM as classifiers, the best of which was Borda method with 96% precision rate. Their database consisted of 4800 images of words written by 100 people. About 50% of these images were used for training and the rest was used for verification and validation.

In 2003, a method was proposed for Arabic word recognition that contained 160 semi-continuous HMM [14]. This system thinned each word and used pixel rows of the thinned image as features. This system was used for 945 city names in Tunisia from IFN/ENIT database [15] and reached 89% accuracy.

Zahedi and Arjomandzadeh presented a system to correct space and half-space characters in Persian texts [16]. They used different statistical approach which uses a fertility-based IBM Model as word alignment by employing a parallel corpus which is created for the special purpose of Persian multi-part word edition.

In [17], a recognition system is introduced for recognizing printed Persian scripts. In this system, at first, lines are identified using a kind of projection and sub-words are separated with another projection. Subsequently, using an innovative algorithm, sub-words are divided into smaller parts and an MLP is used to detect these parts. If these parts are not recognized, they will return to the previous step to repeat the separation. These steps are repeated until no word is divided into smaller parts. If the system cannot recognize the words, the separation is performed again; however, this time, sub-word detection is performed from left to right.

The proposed method in this paper has 5% improvement compared with previous works [18] due to using DPP in four directions. In addition, parallel methods are employed to accelerate the algorithm 5.5 times. Since OCR is used in large contexts, such as Census, the running speed of the algorithm is an important factor in decreasing the cost.

2. Recognition procedure

The procedure of character recognition, like other forms of pattern recognition, consists of three steps: preprocessing, feature extraction, and classification.

In the preprocessing step, the image must be prepared for better recognition; in other words, preprocessing increases the accuracy using image features. In this step, some actions are performed, such as noise reduction, slant correction, and normalization. Normalization consists of assimilating the image to our patterns such as the assimilation of the image size to the trained images. One part of the preprocessing that can exist in the systems is the segmentation of words into characters. Persian/Arabic word recognition systems can be divided into two types: 1. recognition by segmentation 2. Holistic methods in which the whole word is recognized at once. This method was previously used for speech recognition [4].

In the feature extraction step, some information is extracted from the raw image, which is called a feature. Particular features must be extracted according to the pattern and we are trying to recognize. These features can include unitary transformations, gradient, or zoning [19]. In character recognition, features are divided into structural and statistical. Structural features are based on the geometric shape of the letters and are related to the nature of the letters. Three examples of these geometrical patterns are loop, slant, and dots. Statistical features are the result of calculations on regions of the image like FFT and projection profile.

In the next step, the classification step, the system tries to determine to which class the pattern belongs; in other words, to which class the pattern is similar using the feature vector. The most common classification methods are ANN (artificial neural network), SVM (support vector machine), and HMM.

3. Proposed System

In this system, two stages of recognition are used in order to increase the accuracy. In the first stage, the letters are classified into 9 groups according to the similarity of their main bodies. Figure2 shows these 9 groups. In the preprocessing step of this stage, after binarization and centralization, the main part of a character remains and the others are eliminated.

Figure 2. Persian character categories.

Other parts are removed so that only the main part of a character that is shared in the group is remained and additional parts are removed. After that, the DPP features are extracted as illustrated in section 4.2. These extracted features are given to the hamming network for classification. After this stage, characters are separated into 9 groups.

In the second stage, the character must be recognized in each group. In this stage, preprocessing consists of binarization, centralization, and morphology. In the first stage, the noise is removed using the main part; however, in the second stage, the noise must be eliminated by morphology. After that, similar to stage one, the DPP feature is extracted and a hamming classifier is used.

The next three sections describe the preprocessing, feature extraction, and classification of this system.

3.1. Preprocessing

We use the following operations for preprocessing the images in the database:

Binarization: This process is carried out for background elimination and noise reduction. Thus, by considering a threshold, the image is transformed into a zero and one image. This threshold is selected based on trial and error and the database using the classical method of Otsu. Subsequently, the image will be monochrome. Figure 1(a) presents the result of applying this method on the images.

Morphology: as shown in figure 1(b), a little noise remains after binarization. In this system, Erosion operator, which is one of the morphological operators, is applied to remove the noise [19]. Using this operation and a 4*4 mask, the noise with values smaller than this mask is removed (see Figure 1(d)).

Centralization: in this step, the character is moved to the center of the image using a linear function. This operation causes the features extracted from the images to be independent from the location of the character in the image. In the linear function, a rectangle surrounds the character and moves it to the center of the image. In figure 1(e), this operation is shown after applying it to figure 1(d).

Finding the main part: using this operation, the biggest part of the character (black pixels) remains and other parts are eliminated. This operation is good for noise removal; although, it may also remove the main information (see Figure 1(c)).

In this system, recognition has two stages. According to the stage at which preprocessing operations are carried out, only some of the operations mentioned above are applied. This issue will be discussed in the "the proposed system" section.

3.2. Feature extraction

One of the biggest problems in handwritten character recognition is the variance of the character size in the database. For example, figure 3 presents a few examples of one character with different sizes in the dataset. Therefore, it is helpful to have features, which are not sensitive to size. Projection profile is a statistical feature that is sensitive to size. In this paper, we obtain a new feature using the derivative of Projection Profile that is scale invariant. This feature is called DPP and it is described in the following paragraph.

Figure 3. Some variety samples of character 'ن'.

First, projection profile is calculated in four directions. Assuming the image is I (x, y) with the size w*h pixels, projection profile in four directions (vertical, horizontal, 35° diagonal, and 135° diagonal) is calculated using the following equations.

$$\forall j = 1,2,\ldots,h : D_1 = \sum_{i=1}^{w} I(i,j) \quad (1)$$

$$\forall j = 1,2,\ldots,w : D_2 = \sum_{i=1}^{h} I(i,j) \quad (2)$$

$$\forall i = (2-h),\ldots,w :$$

$$D_3(i+3-h) = \sum_{j=\max(1,i)}^{\min(w,i+h-1)} I(j-i+1,j) \quad (3)$$

$$D_4(i+3-h) = \sum_{j=\max(1,i)}^{\min(w,i+h-1)} I(j-i+1,w-j+1) \quad (4)$$

In fact, D1, D2, D3, and D4 show the pixel density of each letter in rows, columns, and diameters. Figure 4 shows the results of using the Projection Profiles formulas. After that, the derivative of these values must be calculated so that the final features are obtained. The first order discrete derivative is calculated using the following relation [20].

$$\frac{df}{dx} = f(x+1) - f(x) \quad (5)$$

As shown in [19], calculating the derivative using the two neighbors of the corresponding pixel

causes the derived value to become less sensitive to noise.

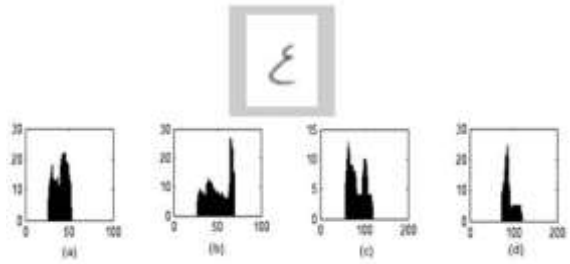

Figure 4. Projection Profile in 4 dimensions. a)Vertical b)Horizontal c)135° d) 35°.

Therefore, we use the following equation to calculate the derivative.

$$\frac{df}{dx} = f(x+1) - f(x-1) \qquad (6)$$

Since we want to track the changes of the derivative, the sign of the derivative is important. Therefore, after calculating the derivative, the sign function is applied in order to obtain different scales for projection profiles.

$$S_i = \text{sign}\left(\frac{dD_i}{k}\right), i = 1,..4, k = 1,\ldots,\text{size}(D_i) \qquad (7)$$

S_1, S_2, S_3, S_4 are signed vectors whose amounts are 0, -1 or 1 and we apply the following rules to normalize them and reduce class variance.

- For neighbors with the same value only one of them is considered.
- In order to remove noise, we consider only one threshold. If Si values change and return to previous values below the threshold, this change is ignored.

Figure 5 presents the derivative of projection profiles after applying the sign function. Now, the DPP feature is achieved and we can use it for recognition. By applying these rules to the database of character images (95 * 77 pixels), 20 values are obtained for each vector. Therefore, 80 elements are obtained as a feature vector for each image. As it is shown in figure 3, Persian characters in the dataset have different sizes, views, and places in the images. Thus, in the preprocessing step, all characters are aligned to the center of the image. Using DPP, the extracted features do not depend on the size of the character. Therefore, the shape and curvature of the character are important. In addition, the angle dependency can be reduced significantly by DPP in four directions.

3.3. Classification

In the proposed system, we used the Hamming network as a classifier. This network is a fully connected, competitive neural network, which is based on the hamming distance. After this network is trained with the training data, all the neurons compete until one of them wins and the test data is assigned to that class when a test data is given to the network [21]. This network calculates the similarity between the test and training data using the code in Algorithm 1.In this algorithm, 'x' is the test and 'y' is the trained data record with n dimensions. The term 'a' shows similar bits and 'd' shows different bits between the test and training samples. Finally, the negative value is the criterion for a winning neuron.

$$x^T.y = \sum_i x_i.y_i = a - d$$
$$d = n - a$$
$$x^T.y = 2a - n$$
$$a = 0.5(x^T.y + n)$$
$$-d = a - n = 0.5(x^T.y + n) - n$$
$$-d = 0.5(x^T.y) - 0.5n$$

Algorithm 1. Calculating hamming distance between two samples in Hamming Neural Network.

4. Using CUDA in parallel processing

Parallelism determines the future of computing science because on the one hand, when the number of transistors increases, increasing CPU speed will become very difficult. On the other hand, the need for simultaneous capabilities is ever increasing. Using multi-core processors is an endeavor to acquire parallelism, but these processors are expensive and the efficiency increases according to the maximum number of cores [22]. However, increasing the number of cores occurs very slowly. GPUs (Graphic Processing Units), which have recently gained importance, are appropriate tools to implement

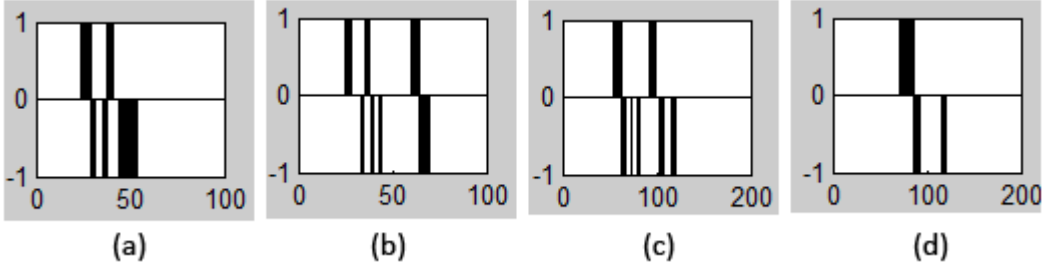

Figure 5. Derivative of projection profiles in after applying Sign function.

parallel algorithms.

The advantages of GPU over other parallelism methods are their high performance and availability. Figure 6 shows a comparison between different models of CPUs and GPUs [23].Each GPU contains many cores, which enable it to perform in parallel and carry out a number of operations at a speed many times more than the speed of a CPU. Figure 7 shows the inner structure of a GPU with 48 cores [25].

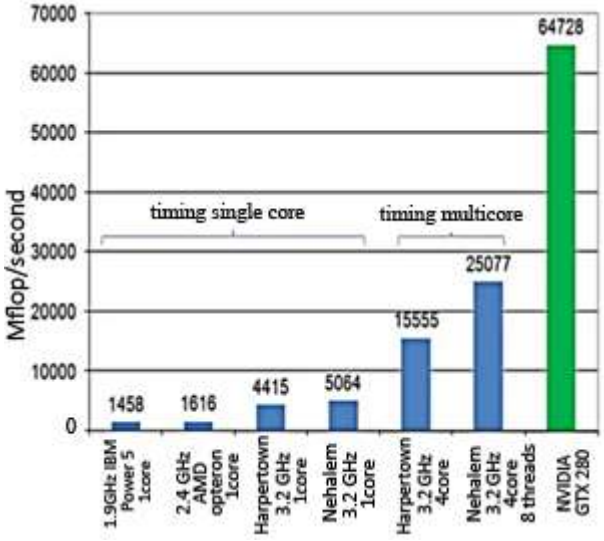

Figure 6. Performance of different CPUs and GPUs[24].

As we can see in this figure, the GPU has its own memory and there are not any intersections between GPU and CPU memories; thus, for executing a program, first, the data must be transferred to GPU memory and in the end, the outcomes must return to the main memory [23]-[26].In 2006, NVIDIA Company introduced CUDA platform for bulk parallel calculation with high efficiency on their GPUs. This platform was presented with a software environment that allows developers to write programs in C language and execute them on GPUs. Each CUDA program consists of two parts: a host and a device. Host is a program executed on the CPU and the device is executed in parallel on the GPU cores. In software application, each parallel program includes a number of threads, which are lightweight processes that are independent of the operations they perform. In CUDA, a set of threads form a block and a set of blocks form a Grid. There are different memory types in GPUs. Each thread has its own local memory and each Block has a shared memory that the inner threads have access to. There is also a global memory that all threads have access to. Moreover, there is another memory called the texture memory similar to the global memory, and all threads have access to, but its addressing mode is different and it is used for special data like images.

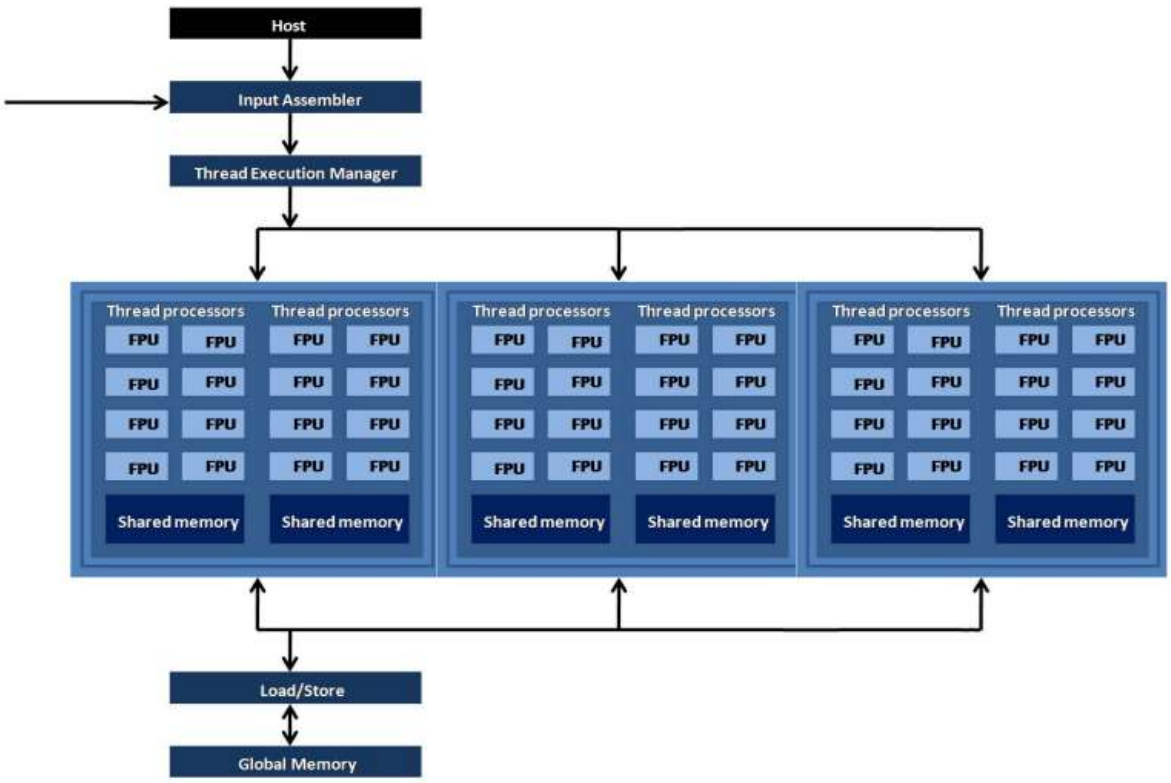

Figure 7. Nvidia Geforce 8800 architecture with 48 cores.

In the host part, the number of threads for the device part must be defined and the device part is executed using this number. Each thread can recognize its position using some CUDA functions and can work based on this position. Finally, the calculated results must return to the main memory.

GPUs are good tools to implement image-processing algorithms because many image operators are local and the operation must be performed on the pixels of each image. As a result, by considering one thread for each pixel (if the required number of threads can be defined), calculation time can be reduced to O(1). In [26], Yang implemented some of the best-known operators such as binarization and filtering by CUDA. Moreover, in [28], Gray uses CUDA for Motion Tracking.

The next section presents the implementation of the parallel algorithm using the hamming network in CUDA.

5. Hamming classifier implementation on GPUs using CUDA

The most time-consuming part of character recognition is searching for extracted features of the corresponding character in the hamming network. This time can be significantly reduced using CUDA and GPU configurations. First, the hamming distance should be calculated between the features of the corresponding character and each feature of the trained characters in the network. For this purpose, we use a two dimensional grid, whose configuration can be seen in figure 8. Up to 512 blocks can be defined in each block; however, for simplicity, we use 500 threads. If we have N training records, then the number of blocks, m, can be obtained by (8).

$$m = \sqrt{\frac{N}{500}} \tag{8}$$

By considering such a configuration, each training data has one thread that must calculate the distance between that and the test data. The pseudo code of each thread is shown in algorithm 2. In this pseudo code, distance is a vector, which has N elements that keep the distance between each training data and test data and each distance is valued by one thread. Using this method, the computational cost of calculating the distance vector is reduced from O(N) to O(1).

Now, in order to find the final answer, we must only calculate the least value of the distance vector. The cost of the search algorithm is a sequence of O(N) order. Although, using the SIMD architecture and tree Grid configuration, this cost can be reduced to O(log N) [29].

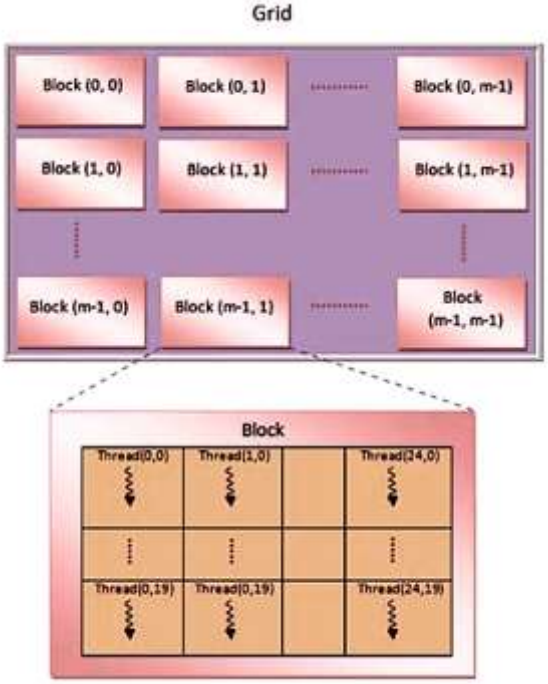

Figure 8. Grid configuration for calculating hamming distance with GPUs.

```
Function CUDA_Distance(Input_Data,Distance)
Begin
Bx: Block Id at X cordinate;
By: Block Id at Y cordinate;
Tx: Thread Id at X cordinate;
Ty: Thread Id at Y cordinate;
B_ addr = ((Bx * m) + By) * 500;
T_ addr = Tx * 20 + Ty;
Data_ addr = B_ addr + T addr;
TData = Train[Data_addr];
Distance[Data_addr] = Euclidean distance
between the 'TData' and 'Input_Data'
End
```

Algorithm 2.The pseudo code of calculating hamming distance between an input matrix and trained data using CUDA.

However, the grid must be reconfigured for each level of the tree in this way and this causes a decrease in speed and overloads. Thus, we only use two levels: There are $\sqrt{N}$ threads in the first level and $\sqrt{\sqrt{N}}$ threads in the second level. For the GPU configuration, a one dimension Grid is used and threads are in one dimension of blocks. The pseudo code related to the threads is shown in algorithm3. The output dimensions of this function, which is the result vector, are equal to the dimensions of the input vector. Therefore, the cost of finding the minimum is reduced from O(N) to O($\sqrt{\sqrt{N}}$). Finally, by giving the distance vector to this function and applying it two times, the least value of the function is obtained, which is considered the final answer. The architecture presented in this section is used to form hamming

networks in the target system. The results section shows the increase in the speed obtained by this performance compared to the normal state.

6. Experimental results

A part of Hadaf database is used to test the system [30]. Hadaf is the biggest Persian isolated characters database that has over 10,000,000 samples. Thus, a subset with 47,695 samples is selected as the dataset. The dispersion of each letter is shown in table 2. The training dataset has 11,796 samples separated by minimum similarity. More specifically, 24% of whole dataset was selected as the training data and the rest was used as the test data. The size of the images is 77*95 pixels.

Table 1.The dispersion of characters in selected database.

ا	9192	ر	3562	ف	456
ب پ	1836	ز	1955	ق	238
ت ث	249	ژ	61	ك	315
ج چ ح خ	1233	س	3453	گ	695
د ذ	30	ش	369	ل	2369
ر ز ژ	282	ص	218	م	4274
س ش ص ض	53	ض	249	ن	3572
ط ظ	1183	ط	173	و	1930
ع غ	493	ظ	49	ه	2305
ف ق	2344	ع	1249	ي	3183
ك گ ل م ن و ه ي	20	غ	105		

As it was mentioned, this system has two recognition stages, in the first stage, characters are clustered, and in the second stage, character class is recognized. The average recognition rate in stage one is 99.84%. The recognition rate of each cluster is shown in table 3.

Table 2. Recognition rate for each group presented in table 1.

Class	1	2	3	4	5	6	7
Detection Percent	99.95	99.90	99.94	99.95	99.70	100	99.61

Table 3. Recognition percent of each character within each group presented in table 3.

Char	**Rec%**	**Char**	**Rec%**	**Char**	**Rec%**
ا	100	ر	96.9	ف	93
ب پ	96.3	ز	94.3	ق	87
ت ث	97.4	ژ	100	ک ك	81.8
ج چ ح خ	94.7	س	89.5	گ	85.6
د ذ	100	ش	84.8	ل	93.4
ر ز ژ	94.5	ص	95.3	م	100
س ش ص ض	100	ض	87.3	ن	85.7
ط ظ	95.1	ط	88.3	و	89.1
ع غ	97.1	ظ	85.7	ه	91.5
ف ق	91.9	ع	84.2	ی	85.6
ك گ ل م ن و ه ي	100	غ	95		

The recognition rate of characters within each group is calculated in the second stage. Some clusters have only one character like clusters 1 and 9; thus, for these clusters, stage two is not needed and recognition rate is the same as stage one. Finally, the recognition rate of 94.5% is achieved. Table 4 shows the accuracy percentage of each character. The second aspect of this system is the usage of GPU parallel processing. For this purpose, the system is implemented by CUDA and it is executed using a 'GT 430' graphic card. This graphic card has 96 GPU cores. In table 5, the execution time in parallel mode is compared with the sequential mode on two different CPU models.

Table 4. Recognition execution time comparing between different processors.

Processor	**Model**	**Execution Time(s)**	**GPU Speedup toward processor**
GPU	GT 430	311.8	1
CPU	Core(TM) i7 CPU 3.2 GHz	1155.41	3.7
CPU	Core(TM)2 Duo 2.66 GHz	1705.69	5.4

7. Conclusion

In this research, two aspects of an OCR system are considered. From the first aspect, for the reliability and accuracy of the system, a new feature extraction method (DPP) is used. According to the experimental results, using this feature, the system can achieve a good recognition rate. Other proposed methods for Persian handwriting recognition did not use a similar database; thus, the exact comparison is not possible and the comparison of this method with others remains as a future work. However, this system has some advantages over other systems. In comparison to other methods, the recognition rate shows that this is one of the best methods.

Another advantage is the training dataset. In this system, the training dataset is a small part of the whole dataset (24%), whereas in some other studies, the majority of the dataset is used to train the data. Also in this system, the whole character set of Persian letters (32 classes) is considered, whereas only a part of the letters or digits is used in some previous studies. The second aspect of this system is its fast recognition. As mentioned before, some parts of this system are implemented on the CUDA platform and executed on GPU cores. As shown in the experimental results, this system was tested on some sequential and parallel configurations. In a common case, a normal CPU (Core(TM) 2 Duo 2.66 GHz) and a common GPU (GT430) were compared and speedup 5.4 was achieved. To our knowledge, in the previous works, there is no effort to use GPU parallel power for OCR systems.

References

[1] Rajavelu, A., Musavi, M. T., & Shirvaikar, M. V. (1989). A neural network approach to character

recognition. Neural Networks, vol. 2, no. 5, pp. 387-393.

[2] Holley, R. (2009). How good can it get? Analyzing and improving OCR accuracy in large scale historic newspaper digitization programs. D-Lib Magazine, vol. 15, no. 3/4.

[3] Mowlaei, A., & Faez, K. (2003). Recognition of isolated handwritten Persian/Arabic characters and numerals using support vector machines. 13th Workshop on Neural Networks for Signal Processing. NNSP'03. 2003.

[4] Amin, A. (1997). Off line Arabic character recognition: a survey. Proceedings of the Fourth International Conference on Document Analysis and Recognition, 1997.

[5] Khorsheed, M. S. (2003). Recognising handwritten Arabic manuscripts using a single hidden Markov model. Pattern Recognition Letters, vol. 24, no. 14, pp. 2235-2242.

[6] Mozaffari, S. & Soltanizadeh, H. (2009).Hndwritten Farsi/Arabic character recognition competition. 10th International Conference on Document Analysis and Recognition.

[7] Parhami, B., & Taraghi, M. (1981). Automatic recognition of printed Farsi texts. Pattern Recognition, vol. 14, no. 1, pp. 395-403.

[8] Mozaffari, S., Faez, K. & Ziaratban, M. (2005). Structural decomposition and statistical description of Farsi/Arabic handwritten numeric characters. Eighth International Conference on Document Analysis and Recognition.

[9] Zakian, H., Monadjemi, S., Ladani, B. T., & Zamanifar, K. (2008). Multi-font Farsi/Arabic isolated character recognition using chain codes. World Academy of Science, Engineering and Technology, vol. 43, pp. 67-70.

[10] Haraty, R. A., & Ghaddar, C. (2004). Arabic text recognition. Int. Arab J. Inf. Technol., vol. 1, no. 2, pp. 156-163.

[11] Haraty RA,C Ghaddar. (2003). Neuro-classification for handwritten arabic text. ACS/IEEE International Conference on Computer Systems and Applications, 2003.

[12] Safabakhsh, R., & Adibi, P. (2005). Nastaaligh handwritten word recognition using a continuous-density variable-duration HMM. Arabian Journal for Science and Engineering, vol. 30, no. 1, pp. 95-120.

[13] Nadir, F., Abdelatif, E., Tarek, K. & Mokhtar, S. (2005). Benefit of multiclassifier systems for Arabic handwritten words recognition. Eighth International Conference on Document Analysis and Recognition, 2005.

[14] Pechwitz, M. & Maergner, V. (2003). HMM based approach for handwritten Arabic word recognition using the IFN/ENIT-database. 12th International Conference on Document Analysis and Recognition.

[15] Pechwitz, M., Maddouri, S., Märgner, V., Ellouze, N. & Amiri, H. (2002). IFN/ENIT-database of handwritten Arabic words. Proc. of CIFED, 2002.

[16] Zahedi, M. & Arjomandzadeh, A. (2015). A new model for persian multi-part words edition based on statistical machine translation. Journal of AI and Data Mining.

[17] Zand, M., Nilchi, A. & Monadjemi, SA. (2008). Recognition-based segmentation in Persian character recognition. Proceedings of World Academy of Science, Engineering and Technology.

[18] Arabfard, M., et al. (2011). Recognition of Isolated Handwritten Persian Characterizing Hamming Network. Innovative Computing Technology. Springer Berlin Heidelberg, pp. 293-304.

[19] Nixon, M. (2008). Feature extraction & image processing. Academic Press.

[20] Gonzalez, RC. & Woods, RE. (2002). Digital image processing. Prentice hall Upper Saddle River.

[21] Mehrotra, K., Mohan, CK. & Ranka, S. (1997). Elements of artificial neural networks. MIT press.

[22] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. Proceedings of the IEEE, vol. 96, no. 5, pp. 879-899.

[23] Nvidia, C. (2011). Nvidiacuda c programming guide. NVIDIA Corporation, vol. 120, no., pp. 18.

[24] Michalakes, J., & Vachharajani, M. (2008). GPU acceleration of numerical weather prediction. Parallel Processing Letters, vol. 18, no. 04, pp. 531-548.

[25] Anderson, RF., Kirtzic, JS. & Daescu, O. (2011). Applying parallel design techniques to template matching with GPUs. High Performance Computing for Computational Science–VECPAR 2010. Springer, pp. 456-468.

[26] ATI Stream Computing user guide (2009). rev1.4.0a.

[27] Yang, Z. Y. & Zhu, Y. Pu. (2008). Parallel image processing based on CUDA. International Conference on Computer Science and Software Engineering.

[28] GrauerGray, S., Kambhamettu, C. & Palaniappan, K. (2008). GPU implementation of belief propagation using CUDA for cloud tracking and reconstruction. IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS 2008).

[29] Akl, S. G. (1989). The design and analysis of parallel algorithms. Prentice-Hall, Inc.

[30] Khosravi, S., et al.(2007). A comprehensive handwritten image corpus of isolated persian/arabic characters. 9th International Symposium on OCR development and evaluation, IEEE.

نشریه هوش مصنوعی و داده کاوی

دانشگاه شاهرود

حروف دست‌نویس جدا از هم فارسی/عربی: ویژگی‌های Derivative projection profile، پیاده‌سازی شده بر روی پردازنده‌های گرافیکی

میثم عسکری، میلاد اسدی، اعظم اصیلیان* و حسین ابراهیم پور

دانشکده مهندسی کامپیوتر، دانشگاه کاشان، کاشان، ایران.

ارسال ۲۰۱۴/۱۲/۱۴؛ پذیرش ۲۰۱۵/۱۱/۲۸

چکیده:

تاکنون پژوهشگران تلاش‌های زیادی را برای بالا بردن دقت روش‌های تشخیص حروف دست‌نویس انجام داده و پیشرفت‌های قابل توجهی در این زمینه داشته‌اند. اما نکته‌ی مهمی که کمتر در این تلاش‌ها موردتوجه قرار گرفته است مسئله‌ی سرعت این روش‌ها می‌باشد. با توجه به کاربردهای سیستم‌های تشخیص حروف دست‌نویس اجرای سریع آنها بسیار مهم می‌باشد اما محدودیت‌های سخت افزاری هزینه‌ی افزایش سرعت را بسیار بالا می‌برد. یک راه-حل مناسب استفاده از قدرت الگوریتم‌های موازی برای این نوع سیستم‌ها می‌باشد. در این مقاله یک روش استخراج ویژگی با نام DPP معرفی شده که به منظور تشخیص حروف دست‌نویس جدا از هم فارسی مورد استفاده قرار گرفته است. با وجود کارایی بالای DPP، محاسبات کمی برای استخراج آن لازم است. همچنین برای طبقه‌بندی ویژگی‌های به دست آمده شبکه‌ی همینگ به کار گرفته شده است. به علاوه برای افزایش سرعت، بعضی از قسمت های الگوریتم تشخیص روی واحدهای پردازنده گرافیکی(GPU) با استفاده از سکوی CUDA پیاده‌سازی شده است. برای ارزیابی روش‌های ارائه شده از پایگاه داده‌ی هدف(بزرگترین پایگاه داده‌ی حروف دست‌نویس جدا از هم) استفاده شده است. در پایان، نتایج دقت ۹۴٫۵٪ در تشخیص حروف و افزایش سرعت ۵٫۵ برابری در تشخیص حروف را نشان می‌دهند.

کلمات کلیدی: شناسایی حروف نوری، حروف فارسی، الگریتم های موازی، پردازنده‌های گرافیکی، Projection Profile ، CUDA.