ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
UDC/UDK 004.75.052:004.932.72'1

OPTICAL CHARACTER RECOGNITION ON GRID AND MULTI-CORE SYSTEMS – PERFORMANCE ANALYSIS

Miran Karić, Zdravko Krpić, Goran Martinović

Original scientific paper

Requirements of digital libraries for computing power, as well as needs of smaller users for automated document processing and digitization are growing, making grid environments and multi-core systems a preferred platform for optical character recognition applications. This paper examines character recognition performance on the CRO-NGI grid and on two multi-core systems by using from a single core up to all available cores. In addition, different models of load distribution on processing units are used. Comparison of the processing speed and the recognition accuracy is conducted. A developed optical character recognition application for parallel environments is used to obtain the results.

Keywords: *grid computing, multi-core, optical character recognition, performance analysis*

Optičko prepoznavanje znakova na grid i višejezgrenim platformama – analiza performansi

Izvorni znanstveni članak

Zahtjevi digitalnih biblioteka za računalnom moći, kao i potrebe manjih korisnika za automatiziranom obradom dokumenata i digitalizacijom su u porastu, što čini grid i višejezgrena okruženja poželjnom platformom za aplikacije optičkog prepoznavanja znakova. U članku se ispituju performanse prepoznavanja znakova na CRO-NGI gridu i na dva višejezgrena računala, uz korištenje od jedne do svih dostupnih jezgara. Uz to, korišteni su različiti modeli raspodjele opterećenja na procesne jedinice. Provedena je usporedba brzine obrade i točnosti prepoznavanja. Za dobivanje rezultata korištena je razvijena aplikacija optičkog prepoznavanja znakova za paralelno okruženje.

Ključne riječi: *analiza performansi, grid okruženje, optičko prepoznavanje znakova, višejezgrena platforma*

1 Introduction

Growing demands of digital libraries, postal and banking services, as well as various automated paper form document processing systems, put greater demands on optical character recognition (OCR) systems. Therefore, the use of grid and multi-core computers is often needed to meet high demands in terms of computing power. In this paper the possibility of parallelization for character recognition, possible models of such systems and their performance were investigated.

A character recognition system usually consists of the following processes: image acquisition, pre-processing, segmentation, feature extraction and classification [1]. In this paper digit image datasets with preprocessed and segmented images of digits were used and authors dealt only with feature extraction and classification steps. These two steps are also generally most time consuming and greatly influence recognition accuracy [2, 3]. For feature extraction gradient features (GF) were used, described in [2] and [4], which are among the best features for recognition of handwritten digits [2]. These features can benefit from complementary features based on character concavities [2]. One type of concavity features (CF) is also used in experiments. Cost-effectiveness of the method in terms of increase in recognition accuracy and recognition time is explored. For classification step, simple and well known *k*-nearest neighbors classifier (K-NN) was used. Datasets used for the experiments are MNIST [5] and DIGITS [6].

However, main focus of this paper was the analysis of character recognition procedures on systems with multiple processing units (PUs) such as grid systems and multi-core computers.

Primarily, the time needed to complete the recognition of a specific set of input data on different platforms and using different numbers of PUs was observed. The CRO-NGI (Croatian National Grid Infrastructure) was tested, on which up to 20 PUs were used and also two multi-core computers with 6 and 4 cores.

Different models of communication between PUs were also tested. It is possible to configure whether the PU receiving input data (main PU) is responsible only for transmitting the work to other PUs (coordinator main PU) or in addition performs work in its idle time (coordinator/worker main PU). These two types of communication will be referred to as coordinator model (CM) and coordinator/worker model (CWM). Remaining PUs in the system are workers; they receive data from the main PU and return results. Packet size i.e. the number of images that the main PU sends to worker PUs in a single step, can also be adjusted.

Similar models for parallel OCR systems exist in literature. Reference [7] proposes a platform for distributed and cooperative OCR systems, called "OCRGrid". A model with many networked OCR servers which can communicate with each other is presented. Clients can request access to a server and send text images to it. The servers perform recognition and send results back to the client. Similar multicomputer OCR system for web library creation is presented in [8]. References [9] and [10] propose an algorithm for recognition of Arabic cursive characters and discuss its parallel implementation using volunteer grid computing.

The remainder of the paper is organised as follows: Section 2 describes used character recognition procedure. Section 3 shows the experimental setup. Obtained results are shown and discussed in Section 4. Conclusion is given in Section 5.

2 Character recognition procedure

For the input to the recognition system data sets with preprocessed and segmented images of digits were used, therefore only procedures for feature extraction and classification were conducted. Two data sets were used: MNIST [5] and DIGITS [6].

MNIST dataset is often used to validate the performance of feature extraction and classification methods. It consists of 60 000 training instances and 10 000 test instances of digit images. Each image size is 28×28 pixels with 256 levels of gray. Size of the digit image is actually 20×20 pixels, centred within a larger frame using the centre of mass. This dataset is freely available online together with the recognition accuracies of different methods on the dataset [11]. A few examples of images from this dataset are shown in Fig. 1a.

The second dataset is lesser known and contains a small number of instances, 1893 training and 1796 test instances. Digits are scaled to fit in 16×16 - pixel images with 256 levels of gray. Some experimental results on this dataset can be found in [6]. Examples of images from this dataset are shown in Fig. 1b.

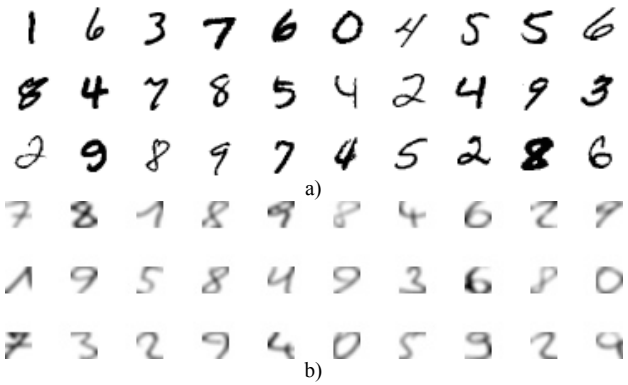


Figure 1 Samples from a) MNIST dataset, b) DIGITS dataset

Gradient features as in [2] and [4] are used in feature extraction step. According to [2] this method achieves the best recognition accuracy among several tested state-of-the-art feature extraction methods. The method and its implementation are explained in detail below. In addition, complementary features based on concavity were used. A transformation, $y = x^{0,5}$, known as Box-Cox transform [12], is applied to obtained feature vectors.

In the classification step, the K-NN, a simple and often used classifier in character recognition, was used. This classifier recognizes an instance of unknown class the following way. If we think of a feature vector of size n as a coordinate in an n-dimensional space, the classifier compares distances from a test instance of unknown class to all coordinates of instances in the training set. Class of the nearest instance in the training set is recognized as a class of the test instance.

2.1 Gradient features

Gradient features are based on image intensity gradient, obtained by filtering the image, usually using Sobel or Kirsh masks [2]. Local stroke directions are then extracted by dividing the image into zones by a grid, usually 4×4 or 5×5, and measuring total gradient intensity in each zone in several directions, usually four or eight.

In this paper Sobel masks were used, shown in Fig. 2, on character images to compute the gradient components. Values of gradient on image border are obtained by replicating missing pixels outside image bounds with nearest value on image border. Gradient vectors for every image pixel are computed from gradient components and then mapped eight standard directions, as shown in Fig. 3. Every vector is decomposed into a sum of two vectors on two nearest standard directions. Image is then divided into 5×5 zones. Total sum of vectors in each standard direction is then calculated, separately for every zone. These sums represent a feature vector. 5×5 zones and eight standard directions were used, giving a feature vector of $5 \times 5 \times 8 = 200$ feature variables.

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Figure 2 Sobel masks for gradient

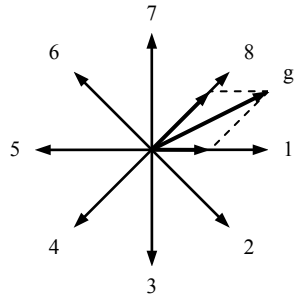


Figure 3 Gradient standard directions and vector decomposition

2.2 Concavity features

Features based on character concavity are used as complementary features since information they contain is not sufficient for successful character recognition, but can complement well certain features, like gradient features [2, 13]. They are structural features based on measurements of character concavities. The method used in this paper is based on measurements of character concavity regions in a binary image.

First the convex hull of a character is calculated. By subtracting character image from the hull, concavity regions are obtained. Position of the center of mass, width, height, and the area of each outer region, i.e. on the convex hull border, are calculated. For the inner regions, the center of mass position and the area are calculated plus one feature variable denoting that an inner concavity exists. There are 5 feature variables per outer region and 4 feature variables per inner region. 5 outer and 2 inner regions were defined, giving a total of 33 feature variables.

2.3 Classification

Classification is performed using the K-NN algorithm. It is a method for classification based on the nearest training objects in the feature space. When k=1, the class of the nearest training objects becomes the class of the test object. When k>1 the class of the test object is determined by the majority vote of its neighbours. A weighted version

of the algorithm is used in this paper. Each of the neighbours in the training set are assigned a weight equal to inverse of distance to the test object, as in Eq. 1.

$$w_k = \frac{1}{d_k}. \tag{1}$$

This ensures that closer neighbours contribute more to the decision than distant neighbours.

3 Multi-core implementation – complexity and performance analysis

There are several approaches when dealing with parallelization of character recognition process. One possibility is to send individual test instances to available PUs for processing. It is also possible to send packets of several test instances. On the other hand, it is possible to parallelize processing of a single instance on multiple PUs, by dividing each processing step to multiple parts that different PUs can execute.

The proposed algorithm is divided into most significant time consuming components: Training Set Feature Extraction (TSFE), Test Set Feature Extraction (TESFE), Classification and Utility Operations and Inter-process Communication (IPC). For the purpose of the code profiling the proposed algorithm is run on 10 cores of the CRO-NGI grid ETFOS installation. The execution ratio of these components with different parameters can be seen in Fig. 4, which shows that the most time and resource consuming component is the classification.

If the classification is parameterized, the number of operations inside the nested loops (`SUB`, `ADD` and `MUL`) is 4, making the total number of operations in the classification function:

$$O = 4\left(N_1 \cdot N_2\right), \tag{2}$$

where N_1 is the training set size and N_2 is the feature count. Considering that for single core execution the whole operation is done by the process coordinator, the whole classification is run N_3 times (test set size), giving the total number of operations as:

$$O = 4\left(N_1 \cdot N_2 \cdot N_3\right). \tag{3}$$

The complexity on a parallel machine of every core is approximately divided by the number of processing elements (cores):

$$O \approx \frac{4}{n}\left(N_1 \cdot N_2 \cdot N_3\right), \tag{4}$$

where n is the number of cores. Furthermore, because for every 4 operations (O) 4 64bit words are accessed from the main memory (W), the number of floating point operations per memory access is 1 $\left(V_a = 1\frac{FLOP}{word}\right)$:

$$V_a = \frac{O}{W} = \frac{\frac{4}{n}\left(N_1 \cdot N_2 \cdot N_3\right)}{\frac{4}{n}\left(N_1 \cdot N_2 \cdot N_3\right)} = 1\frac{FLOP}{word}. \tag{5}$$

The experimental results from Fig. 4a show that by using a small database DIGITS the classification time makes 38 % of the total execution time with Concavity Features (CF) turned on (outer circle), and roughly 44 % of the execution time with CF turned off (inner circle), although the total classification time remains almost the same. The same image shows that the TSFE makes 30 % of the total execution time when using CF, and fairly less (17 %) when not using them. The similar relation can be applied to the TESFE, which uses up 4 % of the execution time when using CF, and a half of that amount without using these features (2 % of the total time).

The situation per single PU changes drastically when using large database MNIST, as illustrated in Fig. 4b; the classification time share in the total execution time is bigger (89 %) in the situation without CF (inner circle), and 87 % with CF (outer circle). The TSFE time takes up 2 % of the total execution time more when using CF. TESFE time is negligible when using a large database.

When comparing the two proposed communication models (CM and CWM), it should be emphasized that the major part of the IPC time is the inter-process communication time, so further analysis will treat IPC time as inter-process communication time. Fig. 4c and 4d show that the non-blocking MPI communication in CWM (outer circles) ensures smaller execution times, but with greater overall communication share in the total time than by using CM (inner circles). This difference ranges from roughly 3 % when using small databases (such as DIGITS) without CF, as in Fig. 4c, up to 9 % when using CF on a large database (MNIST), shown in Fig. 4d. This brings up the conclusion that when using the non-blocking communication the execution times are a bit lower, but at the expense of the IPC time increase. This leads to limited scalability of the code.

4 Experimental setup

For research purposes, an application was developed that performs recognition on input data set i.e. digit images of unknown values. It is possible to adjust how many PUs are used during execution. In addition, it is possible to choose between two communication models among PUs and select the packet size, i.e. the number of images to be sent to a worker for processing in a single communication step.

The first communication model, CM, dedicates one PU only for receiving input data and transmitting jobs to other PUs and that PU is labelled a coordinator PU while other PUs are labelled worker PUs. In this case, coordinator PU may be idle while waiting for worker PUs to complete the job. Another model of communication, CWM, uses this idle time so the coordinator PU performs recognition jobs in its free time (with no communication), and then takes the role of coordinator/worker PU. The minimum packet size that coordinator PU sends to worker PUs is a single image that should be recognized. In

addition, the recognition speed of the entire data set when the packet size is 2, 4, 8, 16, 32, 64 and 128 images was tested.

Two multi-core computers were used − one with four cores (4-core) and one with 6 cores (6-core). In addition the computational grid CRO-NGI was used, on which up to 20 PUs were used. Number of PUs used is in the following order: 1, 2, 4, 6, 10 and 20, limited by the maximum number of PUs on a platform.

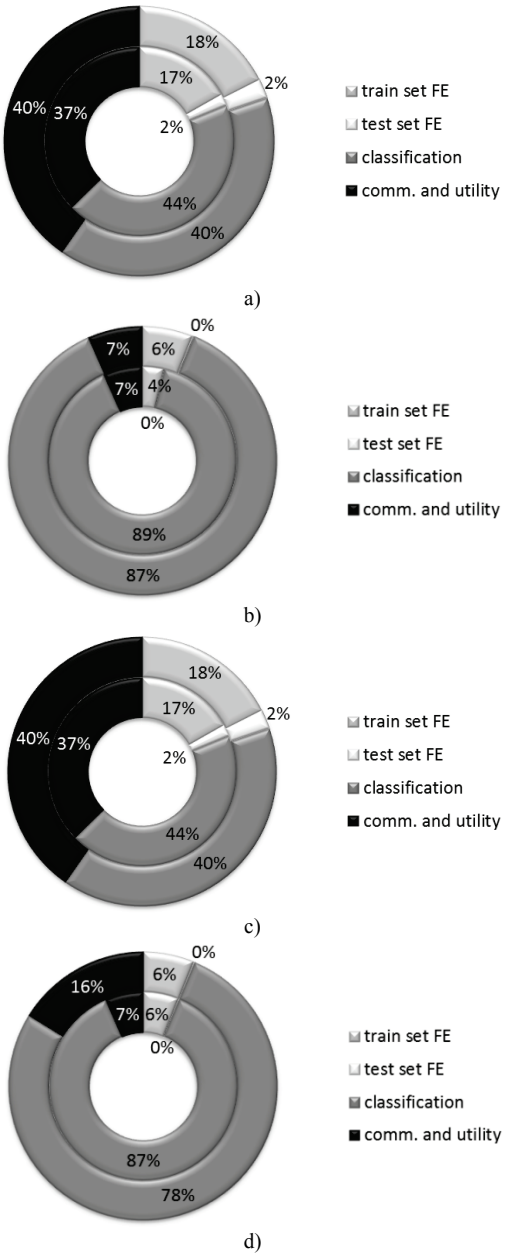
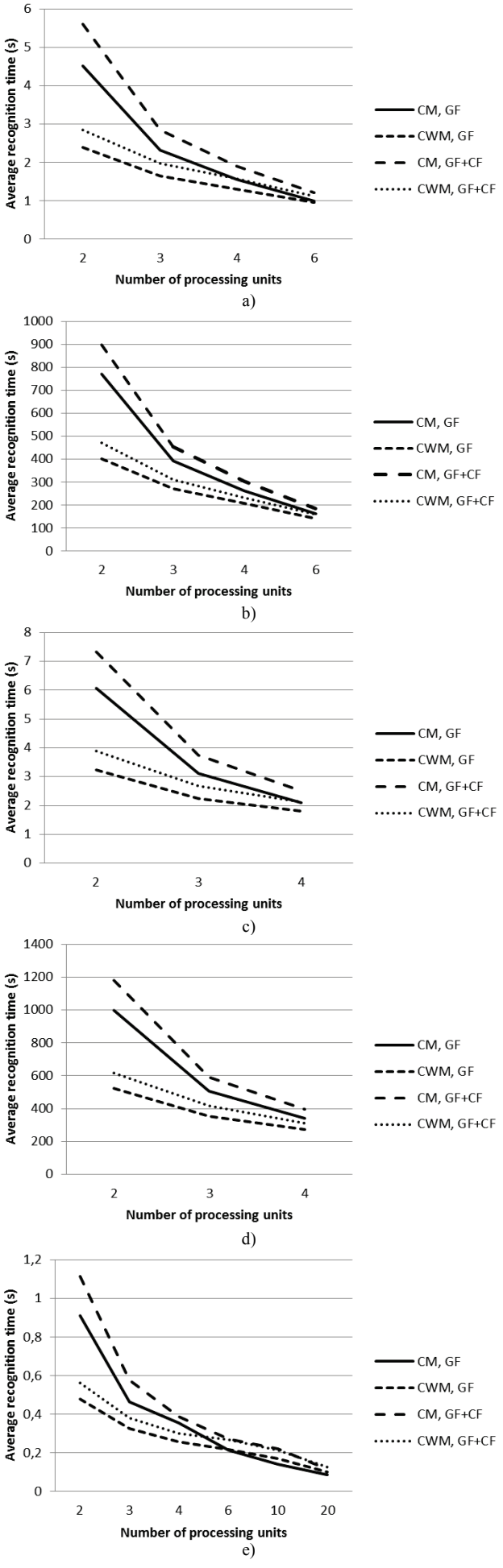
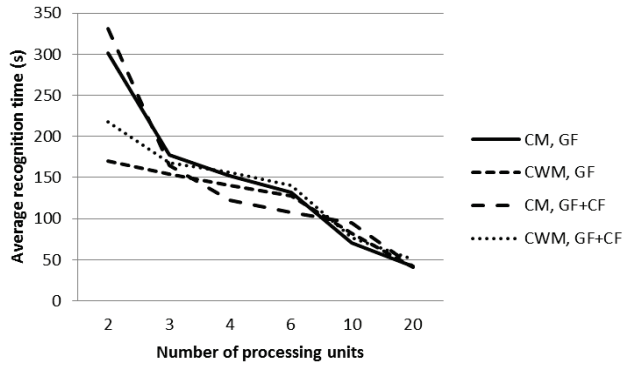


Figure 4 The ratios between execution times of the algorithms most time-consuming parts: a) (inner circle) CF=0, DIGITS database, CM, (outer circle) CF=1; b) (inner circle) CF=0, MNIST database, CM, (outer circle) CF=1; c) (inner circle) CF=0, DIGITS database, CM, (outer circle) CWM, d) (inner circle) CF=1, MNIST database, CM, (outer circle) CWM

In character recognition procedure experiments with two different feature vectors were conducted – GF (only gradient features) and GF+CF (gradient and concavity features). A single classifier is used – K-NN.





f)

Figure 5 Average recognition time on a) 6-core and DIGITS dataset, b) 6-core and MNIST dataset, c) 4-core and DIGITS dataset, d) 4-core and MNIST dataset, e) CRO-NGI and DIGITS dataset, f) CRO-NGI and MNIST dataset

Experiments were conducted on all of these platforms, using binarized images (obtained using Otsu's global threshold method as in [14]) from two different input datasets (MNIST and DIGITS), two communication modes (CM and CWM) and two feature vectors (GF and GF+CF). All experimental measurements are conducted five times and the average is taken, in order to reduce possible errors caused by the uneven environment load.

5 Results and discussion

Recognition performance is measured in time needed to classify whole input dataset. Recognition times are shown below. Since there are too many measurements to display them all, only average recognition times for different numbers of PUs are shown (Tab. 1 and Fig. 5), and average recognition times for different package sizes (Tab. 2 and Fig. 6). For Tab. 2 the results are shown as averages of ratios – recognition times in relation to longest time for any package size.

Results show that as the number of PUs is smaller, CWM model performs better. This is expected as the coordinator PU has a lot of idle time since there is not much communication needed. At 6 PUs the difference is very small while at 10 and 20 PUs coordinator model starts performing better. This can be explained with increased communication needs so coordinator PU has low idle time. Also as the number of PUs increases it becomes more probable that coordinator/worker PU will block one or more worker PUs while executing its job.

Packet size also needs to be balanced. Fig. 6 shows recognition times for different packet sizes and communication models. It can be seen that in communicator model (CM), lower to medium packet sizes provide a low recognition time. With the increase in packet size, the recognition time also increases. Communicator/worker model (CWM) shows different behaviour – medium packet sizes give the lowest recognition times. Lower and higher packet sizes give increasingly higher recognition times. Packet sizes of 8, 16 and 32 gave the lowest recognition times in general.

When recognition times of GF and GF+CF feature vectors are compared it can be observed that the time increases significantly when complementary CF features are used. On average, recognition times are higher by approximately 20 % on DIGITS dataset and 15 % on MNIST dataset. On the other hand, the recognition accuracy is higher when using CF. On the MNIST dataset the accuracy increases from 98,38 % for GF to 98,57 %. for GF+CF, while on the DIGITS dataset it increases from 92,37 % for GF to 92,82 % for GF+CF.

Table 1 Average recognition times for different numbers of processing units (in s)

6-core				
DIGITS				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	4,517	2,394	5,596	2,843
3	2,317	1,641	2,845	1,967
4	1,553	1,298	1,900	1,569
6	0,988	0,953	1,206	1,120
MNIST				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	770,2	401,3	898,0	471,6
3	391,1	271,0	453,0	309,7
4	263,3	206,8	302,7	233,1
6	161,1	140,2	184,1	158,2
4-core				
DIGITS				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	6,077	3,244	7,314	3,897
3	3,116	2,243	3,739	2,689
4	2,102	1,795	2,509	2,132
MNIST				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	997,1	521,7	1179,5	617,7
3	505,3	353,1	592,4	416,9
4	341,4	273,0	396,3	313,7
CRO-NGI				
DIGITS				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	0,911	0,478	1,113	0,560
3	0,463	0,325	0,575	0,378
4	0,355	0,257	0,387	0,300
6	0,215	0,215	0,270	0,268
10	0,142	0,168	0,219	0,213
20	0,087	0,100	0,117	0,126
MNIST				
PUs	CM, GF	CWM, GF	CM, GF+CF	CWM, GF+CF
2	301,3	169,8	330,7	217,9
3	177,8	154,3	164,9	168,2
4	151,9	140,0	122,9	156,0
6	131,5	127,2	107,2	140,3
10	70,7	81,9	95,3	77,4
20	42,2	40,7	42,4	50,6

Table 2 Average of ratios – recognition times for different package sizes

GF								
packet size	1	2	4	8	16	32	64	128
CM, GF	0,922	0,899	0,886	0,884	0,888	0,902	0,949	0,996
CM, GF+CF	0,971	0,970	0,970	0,969	0,970	0,978	0,987	0,998
CWM, GF	0,990	0,952	0,915	0,875	0,860	0,865	0,900	0,932
CWM, GF+CF	0,999	0,957	0,938	0,927	0,927	0,925	0,929	0,938
GF+CF								
packet size	1	2	4	8	16	32	64	128
CM, GF	0,937	0,907	0,889	0,872	0,878	0,895	0,931	0,960
CM, GF+CF	0,974	0,974	0,973	0,976	0,976	0,980	0,987	1,000
CWM, GF	0,988	0,940	0,904	0,868	0,860	0,862	0,888	0,931
CWM, GF+CF	0,997	0,960	0,928	0,912	0,907	0,904	0,909	0,917

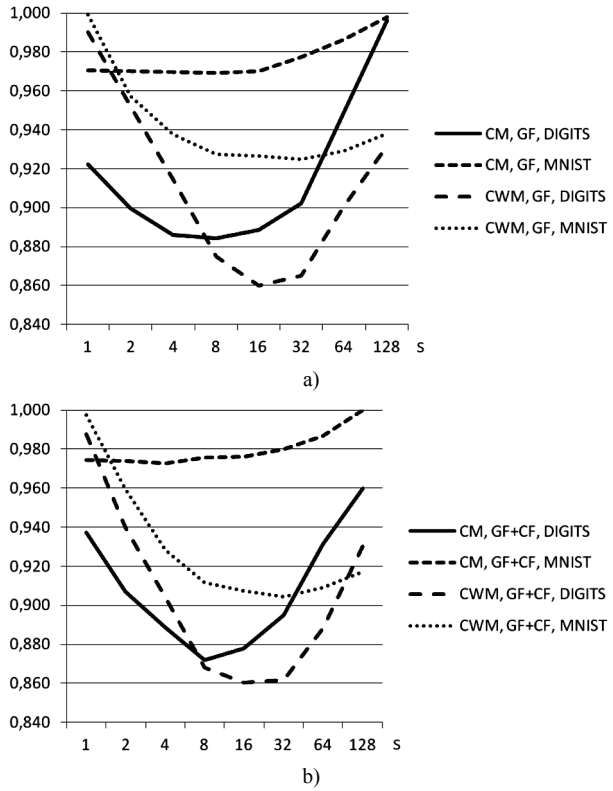


Figure 6 Average recognition time for different packet sizes and a) GF, b) GF+CF feature vector

6 Conclusion

In this paper parallel implementation of an OCR system is described and investigation has been conducted on recognition performance in terms of time and accuracy on grid and multi-core platforms, using different communication models between PUs, package sizes, feature vectors, datasets and a range of total PUs used. It can be concluded that with a low number of PUs, up to 6, lower recognition times are achieved using CWM model, despite larger IPC times. By increasing the number of PUs, CM and CWM start performing equally, although CM has the ratio between computation time and IPC time more in favour of the computation time. Average package sizes achieved best results (8 to 32 images per packet). Complementary CF features increased the recognition accuracy, but also significantly increased the processing time. Using these features can be beneficial when the timing constraints are met and it is important to achieve the greatest possible accuracy.

Future work will extend the research to the real-time system field, examining more complex parallel OCR systems capable of real-time adjustments to the input data in order to increase recognition accuracy and speed.

Acknowledgements

This work was supported by research project grant No. 165-0362980-2002 from the Ministry of Science, Education and Sports of the Republic of Croatia.

Nomenclature

- OCR – Optical character recognition
- CRO-NGI – Croatian National Grid Infrastructure
- CF – Concavity features
- GF – Gradient features
- K-NN – K-nearest neighbour
- PU – Processing unit
- TSFE – Training Set Feature Extraction
- TESFE – Test Set Feature Extraction
- IPC – Inter-process communication
- CM – Coordinator model
- CWM – Coordinator/worker model

7 References

[1] Liu, C.-L.; Nakashima, K.; Sako, H.; Fujisawa, H. Handwritten digit recognition: investigation of normalization and feature extraction techniques. // Pattern Recognition. 37, 2(2004), pp. 265-279.

[2] Liu, C.-L.; Nakashima, K.; Sako, H.; Fujisawa, H. Handwritten digit recognition: benchmarking of state-of-the-art techniques. // Pattern Recognition. 36, 10(2003), pp. 2271-2285.

[3] Nguyen, M. H.; de la Torre, F. Optimal feature selection for support vector machines. // Pattern Recognition. 43, 3(2010), pp. 584-591.

[4] Liu, H.; Ding, X. Handwritten Character Recognition Using Gradient Feature and Quadratic Classifier with Multiple Discrimination Schemes. Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR '05). Washington, DC (USA), 2005. pp. 19-25.

[5] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86, 11(1998), pp. 2278-2324.

[6] Seewald, A. K. Digits - A Dataset for Handwritten Digit Recognition. Austrian Research Institut for Artificial Intelligence Technical Report, Vienna (Austria), 2005.

[7] Goto, H. OCRGrid: A Platform for Distributed and Cooperative OCR Systems. 18th International Conference on Pattern Recognition (ICPR'06). 2, 2006. pp. 982-985.

[8] Martinovic, G; Cukic, B. Multicomputer system for optical character recognition in web library creation. // Library Collections, Acquisitions, and Technical Services. 32, 1(2008), pp. 19-30.

[9] Trifa, Z.; Labidi, M.; Khemakhem, M. Arabic Cursive Characters Distributed Recognition using the DTW Algorithm on BOINC: Performance Analysis. International Journal of Advanced Computer Sciences and Applications. 2, 3(2011), pp. 75-79.

[10] Khemakhem, M.; Belghith, A. Towards A Distributed Arabic OCR Based on the DTW Algorithm: Performance Analysis. The International Arab Journal of Information Technology. 6, 2(2009), pp. 153-161.

[11] LeCun, Y. The MNIST database of handwritten digits. URL: http://yann.lecun.com/exdb/mnist. (15.03.2012.)

[12] Heiden, R.V.D.; Gren, F.C.A. The Box-Cox metric for nearest neighbor classification improvement. // Pattern Recognition. 30, 2(1997), pp. 273–279.

[13] Favata, J.; Srikantan, G.; Srihari, S. Handprinted character/digit recognition using a multiple feature/resolution philosophy. Fourth International Workshop on Frontiers in Handwriting Recognition. Taipei (Taiwan), 1994, pp. 67 - 70.

[14] Gupta, M. R.; Jacobson, N. P.; Garcia, E. K. OCR binarization and image pre-processing for searching historical documents. // Pattern Recognition. 40, 2(2007), pp. 389 - 397.

Authors' addresses

Miran Karić, dipl. ing. el.
Faculty of Electrical Engineering
J. J. Strossmayer University of Osijek
Kneza Trpimira 2b, 31000 Osijek, Croatia
Tel: +385 31 495 424
E-mail: miran.karic@etfos.hr

Zdravko Krpić, dipl. ing. el.
Faculty of Electrical Engineering
J. J. Strossmayer University of Osijek
Kneza Trpimira 2b, 31000 Osijek, Croatia
Tel: +385 31 495 424
E-mail: zdravko.krpic@etfos.hr

Prof. dr. sc. Goran Martinović, dipl. ing. el.
Faculty of Electrical Engineering
J. J. Strossmayer University of Osijek
Kneza Trpimira 2b, 31000 Osijek, Croatia
Tel: +385 31 495 401
E-mail: goran.martinovic@etfos.hr