Maciej KŁYŚ, Magdalena SZYMCZYK, Piotr SZYMCZYK, Mirosław GAJER
AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Parallel implementation of neural networks with the use of GPGPU technology OpenCL

Abstract

The article discusses possibilities of implementing a neural network in a parallel way. The issues of implementation are illustrated with the example of the non-linear neural network. Parallel implementation of earlier mentioned neural network is written with the use of OpenCL library, which is a representative of software supporting general-purpose computing on graphics processor units (GPGPU). The obtained results demonstrate that some group of algorithms can be computed faster if they are implemented in a parallel way and run on a multi-core processor (CPU) or a graphics processing unit (GPU). In case of the GPU, the implemented algorithm should be divided into many threads in order to perform computations faster than on a multi-core CPU. In general, computations on a GPU should be performed when there is a need to process a large amount of data with the use of algorithm which is very well suited to parallel implementation.

Keywords: OpenCL, Artificial Neural Networks, GPGPU.

1. Introduction

The computers are now an integral part of life of many people. It is hard to mention all the fields in which computers have been applied. They have contributed to the enormous progress because they are capable of performing lots of millions of calculations per second. With the use of this mighty computing power, a great deal of applications have been created to make our everyday lives easier. It is just the software development which has contributed to such a large development of general purpose processors (CPU - Central Processing Unit). More and more complex applications consume more computer resources. Over the last few decades one can see an extremely large growth in the performance efficiency of general-purpose processors. In the sixties, one of the founders of Intel observed that the number of transistors in a chip doubles about every 24 months. Later, this observation was called Moore's Law. In addition to the number of transistors the frequency of processors has been considerably increased . In the recent years one has seen a slowdown in increasing the operating frequency of processors, because for higher and higher frequencies, it is more difficult to construct and cool the processor. In order to further increase the performance of the processors, one decided to introduce processors that contain more than one core.

Parallely with the development of processors, the graphics cards were also developed (GPU - The Graphics Processing Unit), in which a lot of processing units were used. It is one of the few characteristics common to the construction of the GPU and CPU. In the GPU, a number of cores performing calculations can be counted in thousands, while in the CPU there are up to the tens. Despite the increased amount of processing elements, high performance of graphics cards can only be achieved when in the calculations a lot of threads are used (hundreds, thousands).

Until recently, the creation of tools that support the development of multi-threaded software was focused around the CPU. Only a few years ago a common standard was introduced to assist creation of applications to be run on the GPU - OpenCL. Since that time, many implementations of the same problem for the CPU and GPU have been created in order to compare their performance. One of them are neural networks which is the problem very well suited for parallelization. In the process of learning and testing neural network there are a lot of the same operations performed, so one can share the calculations on multiple threads.

The another advantage of neural networks, in the addition to easiness of writing parallel implementations, there is a possibility of solving many of the problems through network learning. One of the oldest applications of neural networks is the pattern recognition. In this area, a particular property of neural network was used, which consisted in the fact that they are able to find relationships between the learning data, and then on this basis, the network is able to classify previously unknown data.

The article describes the implementation of a non-linear neural network using OpenCL library (Chapter 5). In paper [3] was described an implementation of two other neural networks: Kohonen and LVQ. While choosing neural networks one was guided by learning methods of various neural networks. Among the algorithms of adapting selected neural networks one can distinguish two types of learning: with the teacher and self-learning.

During writing an implementation of algorithms which are executed by multiple threads, it is necessary to take into account the limitations imposed on the programmer by the existing hardware. Dismissal of hardware limitations can lead to a situation that, despite of the fact that the algorithm is executed by multiple threads, the execution time is longer than for sequential algorithms. The limitations that should be considered while implementing an algorithm which is designed to perform on the graphics card, are presented in chapter six.

During creation of a neural network implementation, the emphasis was put on obtaining high performance in GPU computing. For non-linear neural networks, the received results and a comparison of results obtained from sequential and multi-threaded implementations, written in C ++ programming language, are presented in chapter seven. Test results for the other two neural networks are described in paper [3].

2. General purpose computations on GPU

For the first time term GPGPU (General-Purpose computation on Graphics Processing Units) was introduced by Mark Harris in 2002, who noticed the tendency towards using computation potential of graphics processors for computations which are not connected with showing graphics on the computer monitor.

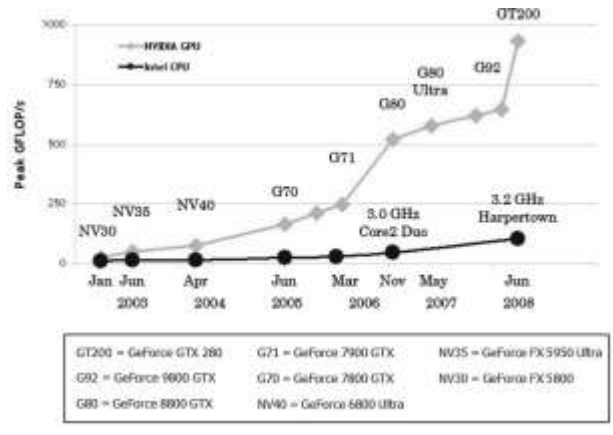

Fig. 1. Difference in floating-point capabilities of CPU and GPU. Source: [4]

Calculations began to move from general purpose processors to graphics processors when it turned out that they exceeded the performance of the CPU. During last decade the difference still grows (Fig. 2).

The reason for discrepancy in floating-point computing capabilities between the CPU and the GPU is that the GPU is specialized in processing the large amount of highly paralleled data. For this reason the GPU was designed in order that more transistors could be assigned to process data and not to cache data and flow control [4].

3. OpenCL

In 2008 was published the first specification of OpenCL framework [1, 2, 5]. It is a programming platform destined to perform computations on heterogeneous platform which consists of different types of computing units (coprocessors) (i.e. GPU, CPU). OpenCL provides a low-level modelling of hardware and framework that supports the creation of applications. Before the advent of OpenCL, it was necessary to use the proprietary software provided by manufacturer of particular computing unit. With the appearance of OpenCL, the software designed with the use of this technology can be launched on computing units from different manufacturers without changing the code.

OpenCL specifications assumes the division of a program into two parts:

- one program executed on host side,
- the other program executed on one or more computing units (OpenCL devices), which are divided into Compute Units, and further to the Processing Elements.

The computing code (called kernel) executed on the coprocessor-side is written in OpenCL C, which is a subset of ISO C99. The application which uses OpenCL sends the list of commands (kernels) from the host to the coprocessors to perform the calculations.

4. Artificial neural networks

An inspiration to create artificial neural networks is the human brain, which is considered as the most complicated human organ. The brain weights about 1,5 kilos and its volume and surface is 1400 cm^3, and 2000 cm^2 respectively. The number of connections between the cells is estimated at 10^{15}. The speed of human brain is estimated at 10^{18} operations per second [6] (for comparison, the theoretical performance of the fastest computer in the world (Titan - Cray XK7) is estimated at 27112.5 TFlop/s).

The first formal description of a nerve cell came into existence in 1943, and its author was Pitts McCulloh [7]. Over the next few decades, many neural network models were created. An artificial neural network is a very simplified model of human brain. It consists of tens to hundreds of processing elements. A single processing element is called a neuron. But it is not as complex as a real neuron, basically it is a very primitive model of real neuron. Neurons are tied together with links which have a parameter (weight), which is modified during the learning process. The connection topology and organization of neurons in larger groups form a neural network program, whose task is to solve the problems assigned to artificial neural network.

General properties of artificial neural networks

The primary criterion by which neural networks can be divided is the division into a network with one-way connections (called feedforward) and network with feedback loops (e.g. Hopfield network). The Neural networks with feedback loops are more often used in research than in practice [6].

One of the most important feature of the neural networks is their ability to adapt themselves and self-organize. This characteristic is used for most practical applications [7, 8]. It allows for the replacement of some algorithm by a neural network, which automatically finds the solution to the assigned problem. Finding a solution may take from a few to many thousands of iterations. The process of finding automatic solution search is called network learning. Another important advantage of neural networks is their reduced sensitivity to the failure of individual network elements. But the most important advantage of neural networks in the view of this article is the ability to work in a parallel way. Today's supercomputers consist of multiple cores which allow to accelerate the operations of neural network several times.

5. Parallel implementation of nonlinear neural network

In this chapter the parallel implementation of nonlinear neural network will be presented. It will be shown which structural elements can be executed in a parallel way on the GPU and, where it is necessary, to synchronize the results of calculations. Then, the selected structures of neural network are mapped onto the objects used in OpenCL library.

Funcionalities

The implementations allow to setup the following parameters:

- a number of layer: 1 - 3,
- a number of neurons in the layer,
- a backpropagation learning method with the possibility of simultaneous validation.

Nonlinear neuron

The basic structural unit of multilayer perceptron is a non-linear neuron. At the time of reception of input signals by the neuron, the stimulation of neuron is calculated. In practice, the distance between the input vector and a vector a single neuron's weights is calculated with the use of selected metrics. In this case, the scalar product is used.

The distance between individual elements of the input vector and vector of a neuron's weights can be calculated by n independent threads, where n is the number of neuron inputs. This is an example of *SIMD* unit, which executes the same instructions for different data. The stimulation of neuron (e) is calculated by summing up the results of individual threads. In the next step the response of neuron is calculated based on activation function (sigmoid) and stimulation of neuron.

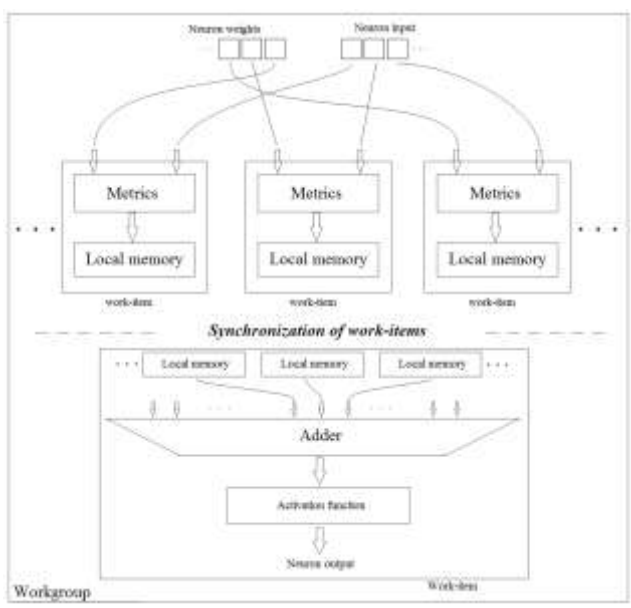

Fig. 2. Parallel implementation of neuron's response calculation

Each thread which calculates the distance between the elements of vectors in OpenCL nomenclature is called work-item. In the case when the calculation of stimulation of neuron is assigned to too few work-items, one work-item executes calculation for more than one input of neuron. The input data are collected by the work

item from the global memory. With this approach to the calculation of neuron output, it is necessary to share the results of calculations. For this purpose, the local memory, which is built into each Compute Unit of graphics card is used. All work-items involved in calculation of neuron stimulation are member of workgroup. Work-items with a local number 0 and 1 are responsible for calculating the sum of results from each thread W. Response of neuron is calculated and written into the global memory by work-item with a local number 0.

Layer of neurons

The largest structural element of neural network is a layer. Stimulations of neurons from the layer m are passed to a layer with number m+1. The layer consists of unlimited number of neurons, and the response of each neuron is calculated in a parallel way. Note, however, that too many layers, and neurons in the layer can contribute to some problem in finding a solution of the assigned problem.

The basic assumption in the implementation of a multilayer perceptron is the ability to use up to three layers, because more of them will not help to solve the problem and increase the amount of necessary calculations to be perform. There is no restriction on the number of parallely executed neurons in the layer. Each neuron is a workgroup in the OpenCL nomenclature. In one graphics card Compute Unit up to forty workgroups can be performed (ten for each unit vector). OpenCL standard itself does not specify whether creating, for example, sixteen workgroups, they will be executed in the same Compute unit, or they will be splitted into four groups containing four elements and will be executed in different Compute Units, thereby providing a parallel calculation of all neurons.

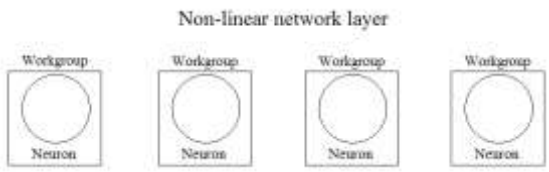

Fig. 3. Parallel calculation of each neuron's response

The calculation of neural network output requires the synchronization of the results in a couple of points. The points of synchronization of nonlinear neural network ale located after each layer. To synchronize the calculations is used in-order queue, which ensures that the next kernel will not be executed until the previous one has been completed.

Network learning - backpropagation algorithm

After calculating the network response to the currently presented input vector, change of weight vector for each neuron across all layers is calculated. Modification of n neuron weights can be performed by n threads (work-items) where n is the number of neuron inputs. The same n threads simultaneously calculate the error at the input of the neuron, and the result is stored in the global memory. If the number of work-items is fewer than the number of neuron inputs, then one work-item modifies more than one neuron weight.

6. Limitations

The approach to parallelization of the neural network presented in the preceding paragraphs has several limitations:

- The number of work-items per one neuron: 256. The limitation means that, in some cases (the maximum number of elements in the input vector is less than or equal to 256) only one thread performs the calculations for a single input.
- The total number of work-items possible to start in graphics card: 16777216.
- The available global memory: 3GB. In the case of very large training data sets, it is impossible to load all the training vectors into graphics card memory.
- Only a certain number of workgroups can be performed in the parallel way. Theoretically, the maximum number of workgroups executed in parallel can be estimated as follows: 32 Compute Units × 4 vector units = 128 workgroups (assuming that one workgroup is made up of 64 work-items). Increasing the number of work-items in a workgroup causes that fewer workgroups can be executed in a parallel way.

7. Performance tests of neural network

In this chapter, performance of implemented neural network algorithm will be presented. It will be checked how the execution time of the algorithm changes, which depends on the size of the input data and the number of neurons in each layer. The performance of the implemented algorithm intended to run on the graphics card, will be compared to the implementation of the same neural network in C++ programming language (the parallel and sequential algorithm using OpenMP library). It will also be shown how the performance of the implemented algorithm looks in comparison with the library built in Matlab - Neural Network Toolbox (in the tests only one processor core will be used).

The following table lists the calculation accelerators which are used for testing of implemented algorithms. Some identifiers (first column) which will be used in subsequent sections were assigned to devices.

No.	Name of calculation accelerator	Programming language/library
1	Intel Core i7 3770K C++	C++
2	Intel Core i7 3770K, C++ OpenMP	C++/OpenMP
3	Intel Core i7 3770K, Intel OpenCL	C++/Intel OpenCL
4	Intel HD 4000, Intel OpenCL	C++/Intel OpenCL
5	Radeon HD7970, AMD OpenCL	C++ AMD OpenCL
6	Matlab, Neural Network Toolbox	not applicable

Various size of input data

The following table summarizes the execution times for different sizes of the input data (image $n \times n$ pixels). In the table description there is an entry: *{x,y,z}*. It means the number of neurons in each layer, starting from layer number one. In any case, the neural network was tested by five thousand epochs.

Tab. 1. Nonlinear neural network built from three layers: {32, 16, 8}. Number of epochs: 5000. Learning coefficient: 0.2. Results given in seconds

n x n	1	2	3	4	5	6
16x16	7	8	9,2	14	33	11
32x32	10	9	10	19	40	14
64x64	24	12	13	27	48	33
128x128	70	23	34	54	69	121
256x256	256	96	156	122	45	699
512x512	1010	432	353	278	80	2589
1024x1024	3985	1712	892	647	140	10176

The results presented in the above-mentioned table, indicate that the implementation of a neural network which runs on general purpose processor is executed from two up to four times faster than on graphics cards. Increasing the size of the input image above 128×128 causes the program to execute faster on the graphics card than on a general purpose processor. There is also an evident difference in performance between the two graphics cards because the card Intel HD 4000 has many fewer resources to perform calculations than the AMD Radeon 7970.

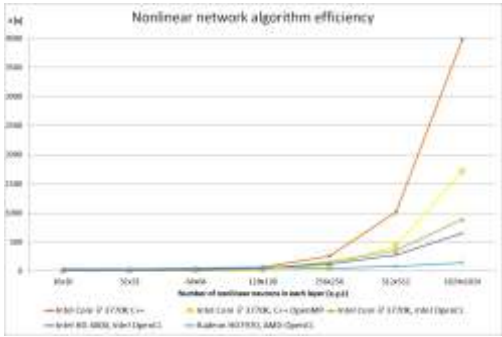

Fig. 4. Nonlinear neural network built out of three layers: {32, 16, 8}. Number of epochs: 5000. Learning coefficient: 0,2

The initially poor performance of AMD graphics card is caused by the fact that the number of work-items in one workgroup (computing output of one neuron) is much less than the minimum recommended size of the workgroup (recommended size of workgroup is a multiple of 64, but not more than 256). The best performance is achieved when one workgroup consists of exactly 256 working-items. It is worth noting that the computation time for an image of size 256×256 pixels is only five seconds worse than for the image of size 32×32 pixels. This shows how important is the size of the workgroup.

Various size of neural network layer

In the previous section is shown that the graphics card has some problems with the images of a smaller size. The purpose of this test is to check whether an implementation in graphics card can gain an advantage over a multi-threaded implementation intended to run in a general purpose processor, in case the size of neural network is increased. Tests are performed for two sizes of the input images:

- the smallest of the tested input images,
- the image for which the execution time of sequential implementation is close to the execution time in the GPU.

Tab. 2. Nonlinear neural network built from three layers. Input image size: 16×16. Number of epochs: 5000. Learning coefficient: 0.2. Results given in seconds

{*x*,*y*,*z*}	1	2	3	4	5	6
{32, 16, 8}	7	10,6	9,3	14	32	12
{64, 32, 8}	8	10,1	9,7	16	33	13
{128, 64, 8}	12	11	12	21	41	15
{256, 128, 8}	23	14	16	32	45	23
{512, 256, 8}	56	23	26	66	57	56
{1024, 512, 8}	240	75	78	175	46	174
{2048, 1024, 8}	1049	334	747	1150	120	780

Tab. 3. Nonlinear neural network built from three layers. Input image size: 128x128. Number of epochs: 5000. Learning coefficient: 0.2. Results given in seconds

{*x*,*y*,*z*}	1	2	3	4	5	6
{32, 16, 8}	71	24	31	20	70	111
{64, 32, 8}	133	42	61	27	30	261
{128, 64, 8}	263	45	141	41	43	646
{256, 128, 8}	530	130	279	70	69	1261
{512, 256, 8}	1081	499	546	84	93	2541

The results presented in Figs. 5 and 6 show that despite of unfavorable workgroup size, the implementation of neural network intended to run on a graphics card can gain advantage over the multi-threaded C++ implementation only when neural network is sufficiently large. However, during the test a negative effect of a large neural network was observed. Neural networks of size {256, 128, 8} (and smaller) were able to learn to recognize the training set. In contrast, the neural network of size {512, 256, 8} and bigger was not able to learn at all to recognize the training set.

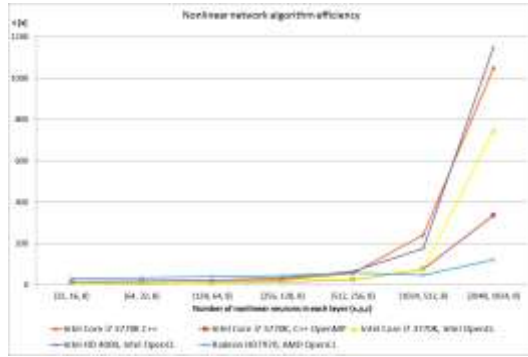

Fig. 5. Nonlinear neural network built from three layers. Input image size: 16×16. Number of epochs: 5000. Learning coefficient: 0.2

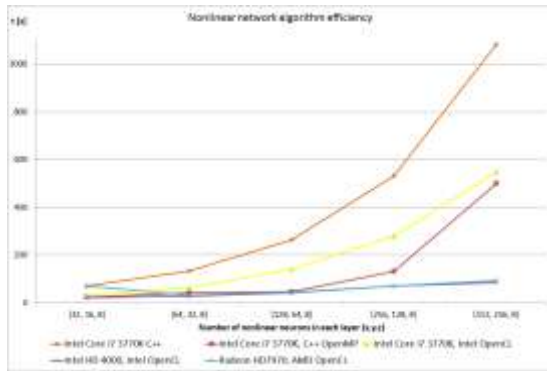

Fig. 6. Nonlinear neural network built from three layers. Input image size: 128×128. Number of epochs: 5000. Learning coefficient: 0.2

Access to global memory

An access to global memory on the AMD Radeon HD 7970 graphics card is accomplished through the twelve channels. Each channel consists of a number of memory banks. If two work-items reads data from memory area to which access is through two different channels, then the reading is in parallel. If from the selected memory area data is read through the same channel (by work-items from two different workgroups), then the reading is serialized, because there is a conflict of channels. In order to avoid conflicts of channels, work items should read data from the memory cells which are adjacent to each other and belonging to the same channel.

In the section Various size of input data the results of tests for optimized kernels on account of the access to the global memory are presented. The results of tests performed using the first version of the kernels were worse, especially for images larger than 128×128 pixels.

Tab. 4. Comparison of the results for two implementations of kernels. The results are given in seconds

$n \times n$	v. 1	v. 2
16x16	24	33
32x32	36	40
64x64	53	48
128x128	47	69
256x256	493	45
512x512	498	80
1024x1024	508	140

8. Conclusion

The main target of the article is to carry out performance tests which compare three implementations of one of the chosen neural

networks. The first two implementations are destined to run on a general purpose processor. They differ only in the number of used cores of processor (one core or all the available processor cores). The last implementation is destined to run on graphics processors.

The significant part of the paper is devoted to implementation and tests of non-linear neural network. During the implementation it is noted that certain structural elements of neural network are repeating (neuron, layer). Due to this fact, some parts of the code can be used multiple times in different types of neural network. This phenomenon has been used to implement two other types of neural networks, Kohonen and LVQ. The details of the implementation and performance tests are presented in the paper [3].

The implementation of neural network learning algorithms for GPU is not a difficult task. A programming language used to write kernels is very similar to C language, and additionally, the Intel and the AMD corporations delivered programs which check syntax errors. Some difficulties in the implementation appear only when one wants to achieve high performance of the implemented algorithms. During tests of neural networks, it was noted that for data of big size, computations on graphics cards are much slower than on quad-core processor. The time of computations of particular kernels can be checked due to a diagnostic tool (called the profiler) which is delivered by the AMD corporation. The implementation's bottleneck was the fact that global memory of a graphics card was hard accessible. In a code optimization a programming guide issued by AMD corporation was very helpful. In the programming guide one can find hints concerning the implementation, which differ depending on an architecture of an applied graphics card.

The conducted performance tests show that the graphics card gains an advantage over a multi-core processor in the two cases. The first one appears in a situation, when an image of large size is given on neuron input (more than 128×128 pixels). The second case occurs, when it is necessary to compute the outputs of many neurons in a parallel way (the size of input data is still relevant but not so much). The resulting acceleration varies from a few up to several times, depending on the type of neural network. Comparing acceleration of non-linear neural networks obtained in tests presented in the article to results for two other types of neural networks (Kohonen and LVQ network [3]), it can be noted that the greatest acceleration in comparison with the multi-thread implementation for CPU is obtained for Kohonen network and the lowest for LVQ network. The differences in performance arise due to the construction of neural networks, which requires the calculations to be carried out in steps. The more synchronization points of calculations, the less acceleration was achieved. In neural networks, in which victorious neuron is selected, applied sorting algorithm influences performance as well. For this reason, in the case of LVQ neural network a small acceleration is achieved mainly due to the repeated use of a slow sorting algorithm.

In the paper [3] it is described that one of the basic applications of neural networks - pattern recognition. The results obtained in the study show that satisfactory results can be achieved only when some features of the images are calculated earlier. Computing image features is an example of preprocessing, which may require a lot of calculations and hence computations can take a lot of time. Not all the features of an image are suitable for computations on a graphics card. Therefore in the next stage of creating an application, which uses OpenCL library, a general purpose processor can be joined to carry out preprocessing. On the basis of the constructed heterogeneous computing platform, one can build a pipeline to pre-process and classify the images. In this way one can get an acceleration which arises not only due to the use of accelerators capable of performing calculations in parallel way.

9. References

[1] AMD OpenCL programming guide, AMD Accelerated Parallel Processing, http://developer.amd.com/download/AMD_Accele rated_ Parallel_Processing_OpenCL_Programming_Guide.pdf

[2] Khronos, OpenCL documentation, http://www.khronos.org/registry/ cl/sdk/1.2/docs/man/xhtml/

[3] Kłyś M.: Master thesis „Rozpoznawanie kształtów z wykorzystaniem sieci neuronowych i technologii GPGPU OpenCL“, AGH, 2013.

[4] NVIDIA, OpenCL programming guide for the CUDA architecture, http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA _OpenCL\linebreak_ProgrammingGuide.pdf

[5] Scarpino M: How to accelerate graphics and computation, 2011.

[6] Tadeusiewicz R.: Sieci neuronowe, Akademicka Oficyna Wydawnicza, 1993.

[7] Tadeusiewicz R., Flasiński M.: Rozpoznawanie obrazów, PWN, 1991.

[8] Tadeusiewicz R., Korohoda P.: Komputerowa analiza i przetwarzanie obrazów. Wyd. Fundacji Postępu Telekomunikacji, 1997.

Received: 28.10.2014 Paper reviewed Accepted: 01.12.2014

MSc Maciej KŁYŚ

He graduated in 2013 in discipline of Automatic Control and Robotics at AGH University of Science and Technology in Cracow. Currently his scientific interest include artificial intelligence and general purpose computation on graphics processing units (GPGPU).

e-mail: kmaci3k@gmail.com

PhD Magdalena SZYMCZYK

She earned her Masters degree in Electronic Engineering in AGH University of Science and Technology (Krakow, Poland) in 1988, and Ph.D. degree in Computer Science also in AGH University of Science and Technology in 1999. Currently, she is a lecturer at AGH University of Science and Technology. Her research interests include real time computer systems, embedded systems, parallel programming and bioinformatics.

e-mail: Magdalena.szymczyk@agh.edu.pl

PhD Piotr SZYMCZYK

He earned his Masters degree in Electronic Engineering in AGH University of Science and Technology (Krakow, Poland) in 1988, and Ph.D. degree in Computer Science also in AGH University of Science and Technology in 1997. Currently, he is a lecturer at AGH University of Science and Technology. His research interests include real time computer systems, embedded systems, natural computing and bioinformatics.

e-mail: Piotr.szymczyk@agh.edu.pl

PhD Mirosław GAJER

He was born in Cracow on 25.04.1971. In 1996 he graduated in the discipline of electronics at AGH University of Science and Technology in Cracow. In 2000 he obtained the PhD degree in computer science. Currently his scientific interest concentrates in the field of artificial intelligence, evolutionary computations and systems, computational linguistics, natural language processing and machine translation.

e-mail: mirek.gajer@gmail.com