Comparison of Computing Performance of Image Processing on Different HW Platforms

Jakub Kolarik, Radek Martinek, Jakub Stefansky, Petr Bilik, and Jan Nedoma

***Abstract*—The work describes the evaluation of selected platforms in computing performance on a defined task. The work includes a description of the individual platforms and their hardware equipment. The chosen representatives are from categories of personal computers, embedded devices and industrial controller with on board FPGA. The evaluation of selected platforms is executed by the rising difficulty of given problem by changing the size of input data. In this case, it is the resolution of the image used by the Canny edge detecting algorithm. The result of this work is the relative comparison of the platforms, even with the increase in the volume of data processed by the algorithm.**

This experiment can be used to simplify architecture and hardware selection in practical applications due to presented performance in account of time complexity of given task.

***Index Terms*—Image processing, LabVIEW, FPGA, cRIO.**

I. Introduction

This work deals with the evaluation of hardware equipment that can be used for fast data processing. Two industrial platforms and three variants of personal computer use were selected for this experiment. The industrial devices used to exploit the FPGA (Field Programmable Gate Array) technology but have different performance parameters. However, the FPGA interface and the processor [1]-[3], whose data throughput may limit the performance of the entire device, can become a weak point for these devices. The work includes a brief description of the LabVIEW (2017) development tools and hardware that was used to program all of the algorithms. The work also includes a description of library functions for image processing.

The document also describes the algorithm itself, by means of all its partial steps. The algorithm consists of image conversion to the RGB (Red-Green-Blue) model, Gaussian filtering, Sobel operator edge detection, and the Canny edge detector itself [4]. The document also contains a description of the design and the implementation of the experiments.

Another interesting work on a similar theme of different approach to data processing is work of Naidila Sadashiv *et al* [5]. This work compares data processing differences using cluster, grid and cloud computing.

Similar work in the field of smartphones is the work of Raquel Trillo *et al*. [6], which describes the performance of mobile platforms when launching typical mobile applications.

Manuscript received May 29, 2019; revised January 13, 2020.

Jakub Kolarik, Radek Martinek, Jakub Stefansky, and Petr Bilik are with the Department of Cybernetics and Biomedical Engineering, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, Ostrava, Czech Republic (e-mail: jakub.kolarik@vsb.cz, {jakub.kolarik, radek.martinek, jakub.stefansky, petr.bilik}@vsb.cz).

Jan Nedoma is with the Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB - Technical University of Ostrava, Ostrava, Czech Republic (e-mail: jan.nedoma@vsb.cz).

II. Description of Industrial Controllers

The hardware parameters used in the experiments are listed in the table below (please see Table I). The actual description of the industrial platforms is included in the subchapters. Thanks to this table, it is possible to compare the ability of the individual platforms to effectively use their hardware.

However, we cannot convert these parameters to a ratio that expresses the effectiveness of the use of the available devices per memory unit by means of indirect proportion (time/memory size). The contribution of the FPGA chip to the time complexity of the operation and clocking cannot be taken into account in this coefficient

TABLE I: Description of Platforms.

Platform	NI myRIO 1900	IC - 3173	Dell Vostro 5568
Processor	Dual-core AR Corte A9	Dual-core Intel i7 2,2GHz	Intel Core i5-7200U
Non-volatile memory	512MB	64 GB	256 GB SSD
Operation memory	256MB 533MHz	8 GB	8 GB
FPGA chip	Xilinx Artix-7	Xilinx Kintex-7 XC7K160T	-
USB interface	2x USB 2.0 Hi-Speed	2x USB 3.0	3x USB 3.0
OS system	RTOS	Windows Embedded Standard 7 NI Linux Real-Time	Windows 10 Pro

A. LabVIEW Environment

The LabVIEW environment consists of two basic parts, a front panel and a block diagram. Both panels are interconnected. The front panel represents the user interface of the application being created. Specifies the appearance and behavior of the application relative to the user. Displays controls and indicators to control runtime and view application results and statuses. The second part is a block diagram in which the necessary algorithm is implemented using graphical elements that can be interconnected by data links. The shape and color of the data link indicates the data type and type of the object, such as an array or cluster, and others. The selected control and indication elements from the front panel are automatically displayed on the block diagram side. These elements can then be interconnected with other

doi: 10.18178/ijmlc.2020.10.2.944

elements from the library functions palette.

B. FPGA Programmable Chip

FPGA is a programmable chip. FPGA has the same flexibility as running software on a processor system but is not limited by the number of processor cores. Unlike processors, processing is naturally parallel. Therefore, each independent processing task is dedicated to a certain part of the chip and can work independently without the influence of other logical blocks. As a result, the performance of one part of the application is not affected when additional processing is added. The main advantage is that the program is implemented by hardware and therefore does not run in the operating system. This enables very fast output response based on the input signal.

Each chip consists of a finite number of predefined resources with programmable jumpers to implement a reconfigurable circuit and I / O blocks that allow access to the outside world. FPGA resources typically include several configurable logic blocks, fixed-function logic blocks such as multiplication blocks, and embedded RAM blocks.

Configurable Logic Blocks (CLB) are the FPGA base logic unit. Sometimes also referred to as logical cells, most often include flip-flops and lookup tables (LUTs). The flip-flop is used as a shift register, mostly to store True or False information for the next time cycle. And LUTs provide logical operations. LUT contains a truth table with a defined list of outputs for each combination of inputs (AND, OR, NAND and others).

C. NI myRIO

This is a built-in device manufactured by NI. This device is primarily designed for academic purposes and is an industrial solution to the compactRIO device.

The device has a configurable IO and can be connected to a host device via USB or 802.11b wireless protocol. The essence of the NI myRIO consists of a programmable system on the Zynq-7010 SoC chip (Fig. 1), which includes the ARM Cortex-A9 dual-core processor and the Xilinx Artix-7 programmable gate array. The real-time operating system runs on the processor side.

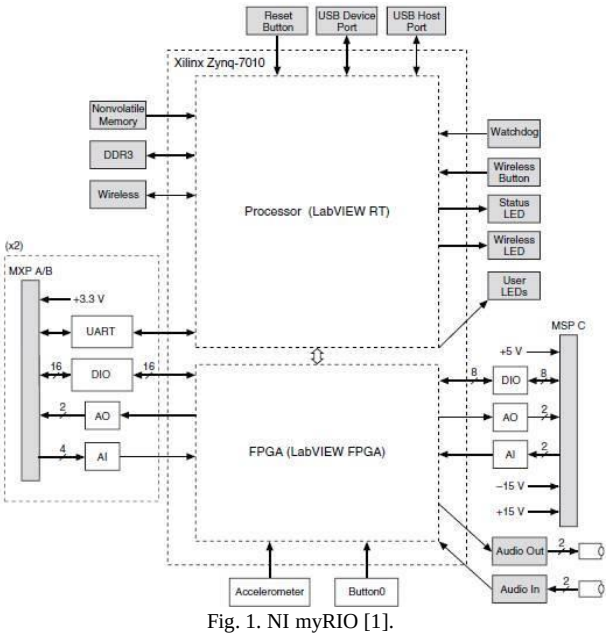


Fig. 1. NI myRIO [1].

D. NI FlexRIO

NI FlexRIO is hardware made by National Instrument. Its main advantage is the direct connection of a suitable input data source with the FPGA. Due to this feature, the associated time lag occurring during the data transfer to the FPGA chip through the processor is avoided. The data thus processed can then be transferred to the processor for further processing.

The FPGA chip can be programmed using LabVIEW without the use of external programs. The module is connected to the computer via a PCI or PCI express bus. For full use of the NI FlexRIO potential, there is a Camera Link NI 1483 module adapter with Camera Link 1.2 standard support. The maximum data throughput of this combination of modules is 850 MB/s.

E. NI Industrial Controller

Industrial controllers (Fig. 2) are also interesting NI products. They offer high levels of processing power and connectivity for automated image processing, data acquisition, and control applications in extreme environments. Industrial Controllers (IC) [7] are high-performance, fanless controllers that provide connectivity for communication and synchronization to automation equipment. You can use LabVIEW system design software to create, debug, and deploy logic to both the onboard FPGA and the processor.

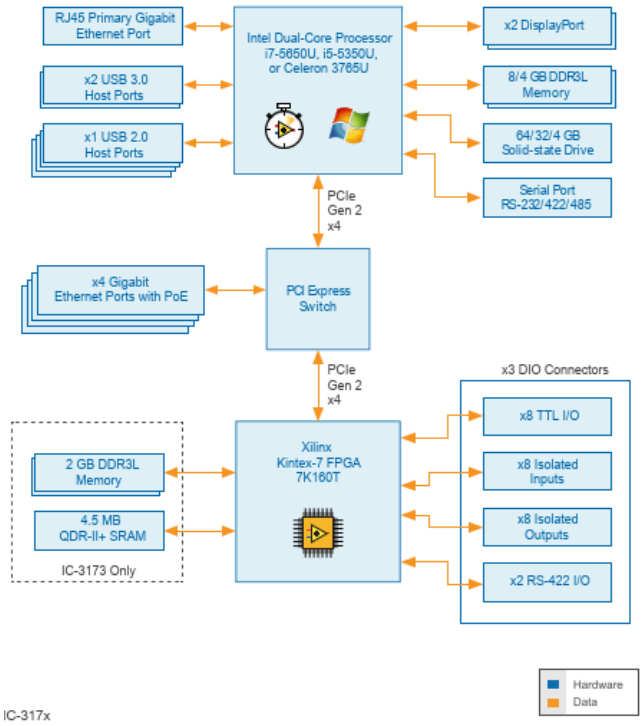


Fig. 2. NI IC-317x [7].

III. Description of the Algorithms Application

This chapter includes a description of the algorithm used to compare HW platforms. The individual parts are described according to the complexity of the mathematical and logical operations they use. IC are high-performance, fanless controllers that provide connectivity for communication and synchronization to EtherCAT and Ethernet CompactRIO chassis, EtherCAT motion drives, GigE Vision and USB3 Vision cameras, and other automation equipment. Controllers

also have onboard isolated, transistor-transistor logic (TTL), and differential digital I/O. You can use LabVIEW system design software to create, debug, and deploy logic to both the onboard FPGA and the processor. LabVIEW contains over 950 signal processing, analysis, control, and mathematics functions to accelerate development.

One of the first and most frequently used colour models is RGB. Each pixel in the image of this model is composed of three values that correspond to the representation of red, green and blue. CMY (Cyan-Magenta-Yellow) is an alternative model to RGB. Another way of writing a colour is, for example, the HSV (hue-saturation-value) model, which indicates colour using colour tone, saturation, and brightness. However, it is necessary to convert the image to a grayscale model for edge detection. According to formula (1), the brightness level for the grayscale model is calculated. The weight coefficients are set according to the sensitivity of the human eye.

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \tag{1}$$

Filtration is conducted for the purpose of noise reduction, smoothing, highlighting and edge detection. Convolution describes the image passage through a linear filter, which is the basis for image function filtering. The image is processed as a product of the mask coefficients with the values of the input image with the surroundings *O*; this procedure is performed sequentially for each pixel of the image.

$$g_b(u,v) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} h(s,t) \cdot f_b(u+s, v+t) \tag{2}$$

Formula (2) is a mathematical description of the discrete convolution used for filter application. The output image is $g_b(u,v)$. The convolutional mask is h with a size equalling to mxn where $m = 2 \cdot a + 1$ and $n = 2 \cdot b + 1, a, b \in N$. The mask dimensions are odd so that the pixel calculated could be in the middle of the mask. The most commonly used mask is a square mask where *a=b.*

A. *Gaussian Filter*

This type of filter is called Gaussian smoothing that uses a convolution mask. Where the pixels closer to the centre have a higher weight than the marginal ones. The weight distribution follows from the Gaussian curve. The filtration rate is based on parameter σ, which indicates the slope of the Gaussian curve. Fig. 3 shows a Gaussian curve for a 1D signal.

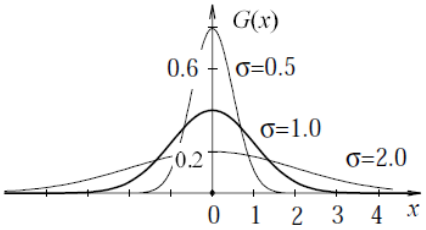


Fig. 3. Gaussian function.

For detection of edges is used algorithm which evaluate changes in brightness of nearby pixels. Gradient is vector with given value and direction. Edge detection is performed by the Sobel operator using a 3×3 convolution mask (formula 3). For edge detection, one vertical edge mask G_y and one horizontal edge mask G_x are used.

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{3}$$

The two-dimensional Gaussian distribution is defined from minus to plus infinity, but when implementing discrete convolution, it is necessary to confine to the most commonly used 5×5 pixel area where the central pixel is (0.0). When calculating mask coefficients, individual values must be normalized. The normalized coefficients must be 1 after the sum of the sum, so it is always necessary to divide the result by adding the sum of the coefficients in the matrix (formula 4).

$$h(x,y) = \frac{1}{2 \cdot \pi \cdot \sigma^2} \cdot exp\left(-\frac{x^2+y^2}{2 \cdot \sigma^2}\right) \tag{4}$$

B. *Sobel Operator*

An edge is defined as a location with a sudden change in the brightness value of the image function $f_b(u,v)$. To find these changes, partial derivatives are used, and the change of the function is indicated by its gradient as vector ∇f_b. The gradient determines the direction of the greatest growth of the function and the steepness of this growth (formula 5).

$$\|h(x,y)\| = \sqrt{\left(\frac{\partial f_b(u,v)}{\partial u}\right)^2 + \left(\frac{\partial f_b(u,v)}{\partial v}\right)^2} \tag{5}$$

The direction is given by the angle φ between the coordinate axis *u* and the radius to the point (u,v), in radians (formula 6).

$$\varphi = arctan\left(\frac{\frac{\partial f_b(u,v)}{\partial u}}{\frac{\partial f_b(u,v)}{\partial v}}\right) \tag{6}$$

The formulas are for continuous image function but are also adjusted for discrete image functions. Try practical calculations to decide if a pixel lies on the boundary of an object. In a single beep, a pixel beyond the boundary can be expected if the gradient size is above the desired threshold. This procedure is subject to fines. The first is that the boundaries of the object come out to be thicker than one pixel. And this procedure does not solve the problem of pre-filling and removing unnecessary border areas. These available solutions imitate the Canny Edge detector.

Local gradient operators are used to search for edges, which approximate the first partial derivative. It is assumed that the pixels with a high gradient value are edgewise. Then, these pixels merge into boundaries and direction of the vector ω is perpendicular to the gradient direction. An example is shown at 0

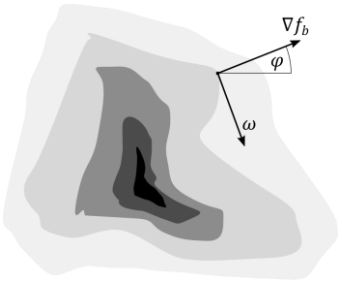


Fig. 4. Example of edge gradient.

C. Canny Edge Detector

Detection according to this algorithm consists of several steps [9]. At the beginning of the algorithm, significant edges are found to avoid omission or duplicate detection. Depending on the localization criterion, the actual and found edge position is then assessed, and such a deviation must be minimal. The last step is the requirement for one response, which is focused on shaded and particularly non-smooth edges not covered by the first requirement.

Edge search is predominantly performed in the basic two matrix dimensions. This edge detection is mainly performed by the Sobel operator. Then the gradient and the gradient angle of all pixels are calculated, so we can find the local maxima.

The last step of the algorithm is thresholding. It is advisable to use hysteresis thresholding that prevents the edges from being disconnected. The thresholding process evaluates the pixels according to the upper and lower thresholds and distinguishes the real edge from the background. This effectively eliminates lonely pixels.

D. Image Processing on FPGA

NI offers ready-made libraries and tools for working with the FPGA. These functions are divided into several categories according to their specific focus in the chain of processes that comprise image processing [10].

- Basic functions ensuring communication. They attend to the transfer of information between the FPGA and the processor using FIFO memory.
- Image processing functions to change the contrast, brightness, pixel inversion and image segmentation by thresholding.
- Filtering functions that are used for image smoothing, noise removal, edge detection, or entering a convolution mask.
- Functions of morphological operations such as dilation or erosion of grayscale images.
- Functions for color image processing, histogram creation or thresholding.
- Arithmetic and logic character functions that are used for bit operations such as adding, subtracting, or multiplying the image by a constant.
- Analytic functions that only work with an image with pixels in grayscale or described by a binary value.

IV. Test Implementation

The test contains the use of the Canny Edge Detection algorithm to determine the time complexity of this algorithm for each of the aforementioned platforms. This test was chosen because of its high demands on computing performance, which may result in platform deficiencies. The input data will be in three different resolutions in order to compare the results when increasing task complexity and to analyze the trends [11], [12]. The chosen resolutions were 320×240, 640×480 and 1280×720. To ensure comparability between tests, the code used in single-core and multi-core test was written without use of the NI Library functions.

On the myRIO and IC platforms, there are two SW (software) parts, one program runs on the RT processor and the other on the FPGA. Completely identical programs, except for minor differences in frequency and reference settings, will run on both platforms.

Two SCTL loops are used to perform the Canny detector. In the case of IC, the sufficient size of the FPGA chip allowed both clock loops to 90MHz. Formula 6.1 shows the period of this loop 11.11 μs. In contrast, myRIO loops can only be set to 55MHz. The frequency therefore depends on the complexity of the code and the size of the chip used.

A. Real Time Application

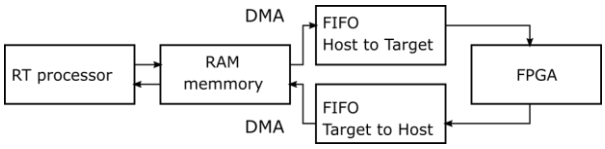


Fig. 5. Time complexity of Canny edge detection algorithm including array operations.

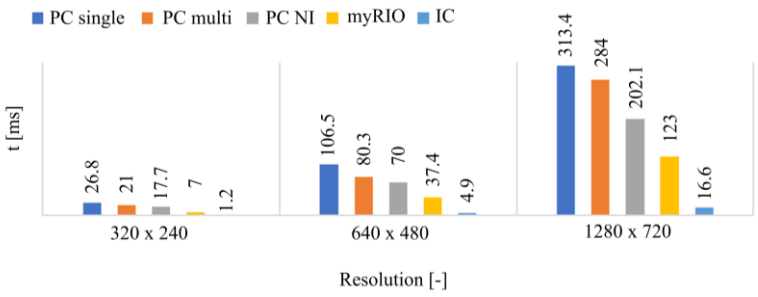


Fig. 6. Time complexity of Canny edge detection algorithm including array operations.

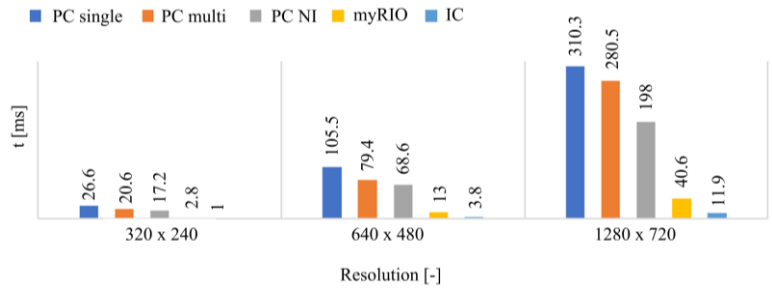


Fig. 7. Time complexity of Canny edge detection algorithm.

At the beginning, the program will initialize the camera for continuous scanning, image resolution and configuration of other parameters. The next step is uploading the FPGA application to the chip and running it. Based on the reference made to the FPGA application, two memories are initialized. The first memory is the DMA (Direct Memory Access) FIFO (First In-First Out) that is set to make the Host to Target transmission, representing the transfer from the RT processor to the FPGA chip. The second memory with the Target to Host transmission type transfers the data from the FPGA chip to the RT processor. The size of both memories corresponds to the number of pixels of the image scanned (see Fig. 4).

After these initialization steps, a sequence that will ensure sequential processing of the subtasks is run.

The FPGA application used the Single-Cycle Time (SCTL) main loop to optimize the codes for the FPGA platform. The content of this loop is then performed according to the specified clocking frequency. Due to the use of this loop, it was not possible to work with a decimal floating-point.

B. Evaluation of Performance Tests

The results of the tests enabled a comparison of the platforms according to the effectiveness of their hardware in the standardized image processing task. Fig. 5 shows the time complexity of the entire test. In the case of non-PC platforms, the next time stamp (please see Fig. 6) was measured. This represents the time complexity of the detection algorithm without a delay caused by data transformation before and after the algorithm execution. This delay consists in converting a 2D field to a 1D field, and a reverse transformation after performing the Canny edge detector.

TABLE II: THE VALUES OF THE CCT AND THE CHROMATICITY COORDINATES FOR THE SELECTED EXCITATION WAVELENGTHS.

Platform		Relative time complexity [%]					
		Detection algorithm			*Array manipulation included*		
		320x240	640x480	1280x720	320x240	640x480	1280x720
PC	Single	-	-	-	-	-	-
	Multi	77.4	75.3	90.4	78.4	75.4	90.6
	NI	64.7	65.0	63.8	66.0	65.7	64.5
myRIO		10.5	12.3	13.1	66.0	65.7	64.5
IC		3.8	3.6	3.8	4.5	4.6	5.3

*Single - Single core processor; Multi – Dual core processor; NI – LabView Library function.

Fig. 8. Source image for detection algorithms.

To compare and highlight the effectiveness of HW platforms [13], the relative determination of time complexity according to the worst outcome (PC - single processor) was used. The Table II shows that, in the case of IC, the relative time complexity is maintained. Fig. 8 shown source image, Fig. 9 the result of Sobel edge detection algorithm and Fig. 10 the result of Canny detection algorithm.

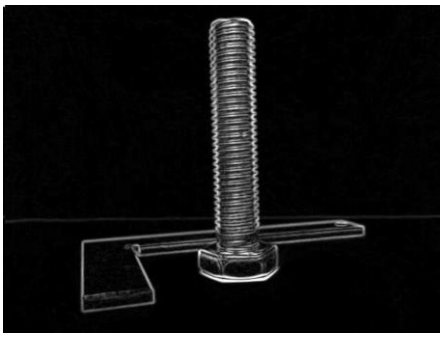


Fig. 9. The result of Sobel edge detection algorithm.

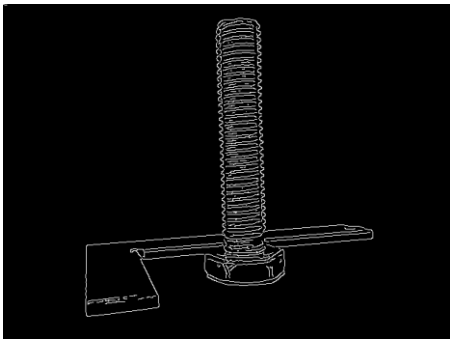


Fig. 10. The result of Canny detection algorithm.

V. CONCLUSION

Based on the results, it is obvious that the results of the industrial system using the FPGA chip come out significantly better than in the case of using a personal computer. However, these results may not be so disadvantageous for a personal computer due to a possible inefficient interpretation of the detection algorithm. When using the NI libraries, up to a fifty percent time saving was achieved compared to using the actual algorithm implementation on a single processor core.

When comparing the times measured, the IC overhead on the processor side is less than 2 ms, while the time overhead from the process side of the much less powerful myRIO is less than 24 ms. These data show that, in the case of myRIO, the processing time is most affected by the performance of the RT processor used other hardware components running under the processor.

Another limitation may be the transfer method, where the IC uses the PCIe between the processor and the FPGA, and myRIO uses transmission via the High performance AXI protocol.

Future research could include other platform, operating systems, programming languages and various number of basic algorithms. These tests would evaluate usability of hardware and software architecture in specific cases of applications.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

JS conducted the research and experiment as part of his

Bachelors's work under supervision of RM, PB supported all this research by necessary hardware and guidance in specific LabVIEW problematics, JN drafted the manuscript and contributed to the interpretation of the results, JK wrote the paper, made revisions and designed the figures, all authors had approved the final version.

ACKNOWLEDGMENT

This article was supported by the Ministry of Education of the Czech Republic (Projects No. SP2019/85). This work was supported by the European Regional Development Fund in the Research Centre of Advanced Mechatronic Systems project, project number CZ.02.1.01/0.0/0.0/16_019/0000867 within the Operational Programme Research, Development and Education.

This article was supported by the project SP2019/107, Development of algorithms and systems for control, measurement and safety applications V" of Student Grant System, VSB-TU Ostrava.

REFERENCES

[1] N. H. Shan and A. Hazanchuk, "Adaptive edge detection for real-time video processing using FPGAs," *Global Signal Processing*, vol. 7, no. 3, pp. 2-3, 2004.

[2] M. Eric and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Transactions on industrial Electronics*, vol. 54, no. 4, pp. 1824-1842, 2007.

[3] A. Shuichi, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *Proc. International Conference on Field Programmable Logic and Applications*, 2009.

[4] M. Raman and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing*, vol. 3, no. 1, pp. 1-11, 2009.

[5] S. Naidila and K. S. M. Dilip, "Cluster, grid and cloud computing: A detailed comparison," in *Proc. 2011 6th International Conference on Computer Science & Education*, 2011, pp. 477-482.

[6] T. Raquel, I. Sergio, and M. Eduardo, "Comparison and performance evaluation of mobile agent platforms," in *Proc. Third International Conference on Autonomic and Autonomous Systems*, IEEE, 2007, pp. 41-41.

[7] IC-317x User Manual. (2017). National Instrument. [Online]. Available: http://www.ni.com/pdf/manuals/375285c.pdf

[8] N. S. Ranjan *et al.*, "Design and implementation of real time video image edge detection system using MyRIO," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 5, pp. 1793-1800, 2016.

[9] K. L. Amruta and V. G. Sangam, "Canny edge detection algorithm," *International Journal of Advanced Research in Electronics and Communication Engineering*, vol. 5, no. 5, pp. 1292-1295, 2016.

[10] X. Qian *et al*., "A distributed canny edge detector: Algorithm and FPGA implementation," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2944-2960, 2014.

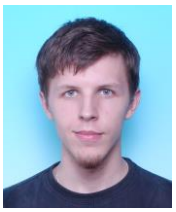
[11] V. Lyashenko *et al.*, "Study of composite materials for the engineering using wavelet analysis and image processing technology," *International Journal of Mechanical and Production Engineering Research and Development*, vol. 7, no. 6, 2017, pp. 445-452.

[12] A. F. Dwi, T. W. Purboyo, and R. E. Saputra, "Cotton texture segmentation based on image texture analysis using gray level co-occurrence matrix (GLCM) and euclidean distance," *International Journal of Applied Engineering Research*, vol. 13, no. 1, 2018, pp. 449-455.

[13] J. D. Owens *et al*., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*., vol. 26, no. 1, UK: Blackwell Publishing Ltd, 2007.

Jakub Kolarik was born in 1992 in Ostrava, Czech Republic. He received his bachelor degree at the VSB–Technical University of Ostrava, the Department of Cybernetics and Biomedical Engineering in 2014. Two years later at the same department, he received his master degree in the field of control and information systems. He is currently pursuing his Ph.D in technical cybernetics. His current research interest includes development of alternative systems for MRI triggering, methods of fetal ECG monitoring, analysis of vibrations caused by biological function of human body and also vibration analysis of road traffic.

Jakub Stefansky was born in 1992 in Ostrava, Czech Republic. He received his bachelors degree at the VSB–Technical University of Ostrava, the Department of Cybernetics and Biomedical Engineering in 2018. He is now also in the VSB–Technical University of Ostrava.



Radek Martinek was born in 1984 in the Czech Republic. In 2009 he received a master degree in information and communication technology from the VSB-TU of Ostrava. Since 2012 he has worked there as a research fellow. In 2014 he successfully defended his doctoral thesis titled: The Use of Complex Adaptive Methods of Signal Processing for Refining the Diagnostic Quality of the Abdominal Fetal Electrocardiogram. He has worked as an associate professor at the VŠB-TUO Technical University of Ostrava since 2017.

Petr Bilik was born in 1968. He received his master degree in power electronics from VSB Technical University in1991, he finished his PhD study in the field of Technical Cybernetics in 2004. From 2000 until 2012 he worked as the head of Power Quality Measurement System department in commercial company. Currently he is vice-head of Department of Cybernetics and Biomedical Engineering. His main interests are automated test and measurement systems design, data acquisition, graphical programming.

Jan Nedoma was born in 1988 in Prostejov. In 2012 he received a bachelor degree from VSB-Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science, Department of Telecommunications. Two years later, he received his master degree in the field of Telecommunications in the same workplace. He is currently an assistant professor of Department of Telecommunications at VSB-Technical University of Ostrava. He works in the field of fiber-optic sensor systems.