RESEARCH PAPERS
FACULTY OF MATERIALS SCIENCE AND TECHNOLOGY IN TRNAVA
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

2021 Volume 29, Number 49

AN APPROACH TO AUTOMATED PROGRAMMING OF INDUSTRIAL ROBOTS BASED ON GRAPHIC DATA

Igor HALENÁR[1], Lenka HALENÁROVÁ[1], Matej KOVAČIC[1]

[1]SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA,
FACULTY OF MATERIALS SCIENCE AND TECHNOLOGY IN TRNAVA,
INSTITUTE OF APPLIED INFORMATICS, AUTOMATION AND MECHATRONICS,
ULICA JÁNA BOTTU 2781/25, 917 24 TRNAVA, SLOVAK REPUBLIC
e-mail: igor.halenar@stuba.sk, lenka.halenarova@stuba.sk, matt.kov25@gmail.com
Received 25 August 2021, Accepted 22 October 2021, Published 24 November 2021

Abstract

The article contains the design and implementation of a system for automatic program generation for an industrial robot, based on the graphical input recorded by camera. The content is divided into two parts. The first section deals with processing the visual data acquired from an external source to appropriate form, using edge identification and transformation from raster to vector image. The second part deals with the automatic program generation for industrial robot. The article describes different possible approaches to solution. The selected way of solution was implemented in real environment in a chosen programming language. The final section of the article presents the results of experiments, together with the overall evaluation of the results.

Keywords

Robot, Programming, Automation, Picture processing, Algorithm

INTRODUCTION

Development of industrial technologies is characterized by increased growth of new technical knowledge, especially in the areas of information technologies, robots and automation. Modern research brings to practice implementation of new technologies, such as video processing, process visualization, simulation, additive production and automation of many production activities.

This situation allows many benefits, especially shortening the production time, increasing work efficiency and reducing the error rate. Automation together with programmable production units, such as robots, play a major role in this system. They are not only efficient and accurate, but they also have the ability of selflearning with the right software. This paper presents a proposal of a system able to create a real model from solid material according to the picture captured by a camera using industrial robot. Industrial robot was choosen thanks to variability of working toolsas well as flexibilty of object size and shape. The goal was to transform an achieved raster picture of a real object from camera (pattern) into a set of orders for an industrial robot movement. Data is processed in several steps, and the result is the system joining the field of video processing, programming and automation, whilethe final solution belongs to the field of industrial production or industrial 3D printing.

THEORETICAL FUNDAMENTALS AND THE RELATED PROCEDURE

The theoretical problem area of this paper consists of two comprehensive parts. The first one deals with a problem area of processing graphical picture data from external source (camera). Image processing via computer is one of the most frequently used activities in connection with computer technology.

Nowadays trend is the recognition of shapes and the evaluation of captured images using artificial intelligence methods [1, 2, 3], or process of reconstruction of low-quality images – image enhancement [4, 5, 6]. These two methods of image processing through artificial intelligence are used in a wide range of technical solutions and scientific disciplines. For example in medicine [7], astronomy [8], signal processing in automation [9, 10], physics [11], biology [12], chemical industry [13] and many others [14].

In this contribution, less complex activities are used, and the automated image conversion and processing within the solution (conversion of the acquired bitmap image from the CCD camera to another image data format, suitable for generating a program for the movement of the robotic arm) is done using mathematical operations. The core of process consists in identifying the important edges in a bitmap image, determining the importance of objects for preservation in the resulting image, and subsequent transformation into a suitable vector format. The main difference between the raster and vector graphics is the way that image data is stored. For raster images, the graphical information is represented by a set of individual points, stored in a regular grid together with information about the brightness and colour of each point. In the case of vector graphics, the image is created using basic geometric primitives, which can be expressed by mathematical equations. The conversion from vector image to raster image is simple. The opposite way is more complicated. From the available literary sources, it is clear, that problem area of edge detection in raster images for conversion is well processed and procedures are well known. The basic issues of conversion are described in the publication of authors of Olsen and Gooch [15]. They describe an algorithm and unsupervised system that transforms digital photographs in raster format as an input, and uses them to vector images, this resulting in a simplification of the source data in terms of bit encoding costs, as well as visual complexity. For picture processing, math functions are used. This is the basic approach and is generally used in many other solutions. Development is also advancing in this area, and the basic algorithm is being improved by a variety of methods. Very similar access is used in the publication of authors [16] L. Yuan and X. Xu. They use similar methods for edge detection in raster pictures – smoothing picture, gradient direction calculation, double threshold setting and non-maxima suppression for the gradient magnitude. Improvement is in the use of local weighted k- average method, solving the problems which exist in the traditional general edge detection algorithm. A new method of improving performance of Canny edge detection is core of the paper of authors Tasneem and Afroze [17]. Their improvement of classical access to edge detection lies in enhancing the image contrast of low contrast image to high contrast by stretching the grey values and application of high contrast histogram into process of edge detection.

From the listed literature sources, it is clear that general algorithm in a wide number of applications of edge detection in images is based on the use of the Canny edge detector developed in 1986 by John F. Canny, with some improvements in the final application.

Edge detection is just the first part of the problem area. Another problem is automatic generation of program and program data for industrial robot. The basic idea is to create vector image with graphical data based on the edge detection process in the first part of data processing. The transformation of edges into vector images is described in many publications [15, 18, 19]. From generally used approaches, it seems to be a simple and usable method described in the paper of Kirsanov et.al [20], based on the allocation of edges in the gradient transition using a threshold filter. Obtained data is then transformed to a vector output using straight line detection. This will be done by software implementation.

A problem of automatic generation of program for robot from vectorized image has a relatively simple solution and consists of generating program – list of commands - in the language suitable for robot. In our case, in the RAPID programming language. The way of data processing was similar, regardless of the type of data input (database, positional sensors, vision techniques). Myers et al. [21] describe a system that learns from demonstrations of a human operator, and then produces a sensor-driven program that can control a robot without operator intervention. The proposed system generates a procedural program, identical in function to a procedural program developed in a conventional robot programming language. The same approach (generating program for robot based on the data from positional sensors) is described in the publication of Friedrich et al. [22]. Other methods, but the same goal, is described in the paper of Chulhee et al. [23]. They used the basic computer vision algorithms (detection of brightness, motion and colour) for generating a program for a commercial edutainment robot. A team of authors in other publication [24] shows a CAD-based system to program a robot from a 3D CAD environment, allowing the users with basic CAD skills to generate the robot programs off-line in simulation. The proposed system generates program according to the objects defined in the CAD software.

From above-mentioned, it is clear, that the issue of program generation for robot is well-known and dealt with in various applications.

PROPOSAL OF THE SYSTEM IMPLEMENTATION

The aim of research was to construct a system with possibility to create a real solid model according to the picture taken by camera. The executive element was industrial robot. In our case, it was ABB robot IRB140 [25]. With certain modifications, our proposal is usable in many other robots. In addition, the proposed algorithm is applicable in various machines, e.g. CNC milling machine.

Thanks to the previous experience with solving the issues in the problem area, we can propose several solutions. There are different ways to solving this problem, while each has its pros and cons.

<u>Solution No. 1</u>: creating a standalone application for image processing and generating program code for the robot in the RAPID language. Achieved raster image was processed using a program in selected program language (C#). Algorithm in the solution identified separate points of given colour and intensity in source picture. After that, the data was exported to external file using a big number of the "MOVE" instructions for robot. The text file was imported to the Robot Studio software for program generation. After that, the program could be tested in virtual form in the Robot studio, or sent to the physical robot for execution. The disadvantage was the necessity of access to the Robot Studio program and large and slow program thanks to many "MOVE" commands.

<u>Solution No. 2:</u> using CAD software for path generation. In this proposal, the source picture was vectorized and loaded into the CAD software. There was a possibility of using software addons to convert vector image to a 3D object in such a way, that one surface represented original lines. After that, the geometry could be imported to the Robot studio, where the robot

can teach the path according to the defined surface. The only benefit of this solution is that robot generates program for moving itself. The disadvantage is the necessity of access to the Robot Studio program, suitable CAD program with addons and the necessity of a skilled user.

Solution No. 3: to create application in selected program language, directly connected to robot and cyclic processing of data. In this case, the created application directly communicates with the robot. The disadvantage was the necessity to create application with the option of graphic processing (edge find, contours find) and the use of libraries for direct communication with the robot. Advantage was the ease of use for operator and no need for additional software.

As the whole solution must in principle be easy to use, the approach No.2 seems to be the most appropriate. This method of implementation is original and user-friendly.

In addition, for successful implementation, the following specifications need to be determined:

- Define type of initial adjustment of input images (correction of size, colours, picture errors),
- Define how to find closed lines in images (contours), determine sensitivity limits in the identification of significant edges,
- Define a suitable method of vectorization of found contours,
- Design a method of transforming a vector image into a program for controlling the movement of a robot arm,
- Render a processed photo using an industrial robot.

These steps represent the basic process of the project implementation. As an industrial robot, the IRB-140 device was used, which is a compact industrial robot with six degrees of freedom. It has a long reach (810 mm) and handles load of 6 kg and a [25].

Fig. 1 *IRB 140 [25]*

It is also appropriate to set a requirements catalog for application:

- The application must allow the selection of a photo in selected bitmap formats or its capture directly from the recording device (camera),
- The application must allow image processing in order to find important graphic data enabling redrawing - canny edge detection, finding contours, changing colours to grayscale and more,

- The application must find the correct coordinates of the points and send them to the industrial robot,
- The solution will include an application program in rapid language in the robot's memory capable of communicating over the network with the created application,
- The program in the robot's memory (based on the data received over the communication network) ensures the movement of the robot's arm,
- The application will run under the Microsoft windows operating system,
- The application will be implemented in the Microsoft visual studio c # IDE.

Application can be created in C# language. For processing the source image, we used a specialized library for picture processing EmguCV, which is wrapper for OpenCV [26]. Using the functions of the library, it was possible to identify the individual contours in the raster image. After identification of the main lines in source image, the necessary data could be sent in the RAPID code format to the given robot. For communication with the robot, it is necessary to use the software development kit from ABB (SDK).

The input of data to the system was performed via graphical data from camera (Figure 5). Image can be scanned by the camera directly into the system. Eventually, an appropriate file stored in computer can be used as an input. The proposed application will cover the entire process. There are several steps and activities from the beginning (that is to get picture) up to the finish line (represented by movement of the robot arm). The block diagram of the architecture of the proposed system is shown in Figure 2.

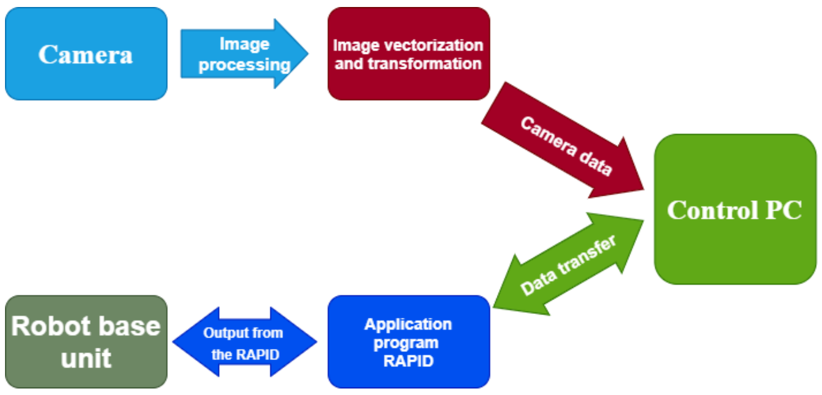

Fig. 2 *Block diagram of the proposed system*

Graphical data processing

The source of the data is a digital image in computer memory. All process is performed in two steps. In Step 1, there is the basic modification of the input data using the functions available within the used libraries of the OpenCV image processing module – size correction, colour correction. In Step 2, the EmguCV library software performs the identification of edges and contours in the source (raster) image. The threshold for detecting image edges must be visually checked by the operator. For edge detection, we used algorithm in the EmguCV library, which is implementation of basic Canny algorithm [27].

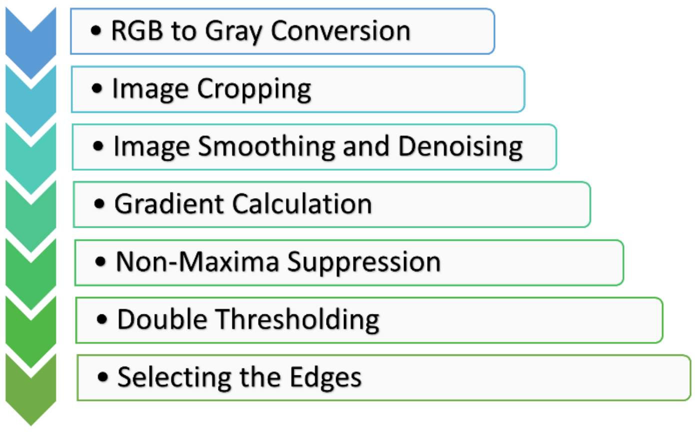

Fig. 3 *Steps of improved Canny algorithm used in application*

Phase 1 in data processing was noise reduction. Since edge detection is susceptible to noise in the image, the Step 1, we removed the noise in the image using a 5x5 Gaussian filter. Step 2 in the process of edge detection was to find the intensity gradient. Smoothened image was filtered using a Sobel kernel in both, horizontal and vertical directions to get first derivative in the horizontal direction (Gx) and vertical direction (Gy). From these two images, we could find the edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

The next step after image smoothing was elimination of unwanted pixels. This can be done by elimination of the pixels which are not part of any edge - non-maximum suppression. After getting the gradient magnitude and direction, a full scan of image was made to remove any unwanted pixels which may not constitute the edge. For this reason, every pixel is checked whether it concerns a local maximum in its neighbourhood in the direction of gradient [27].

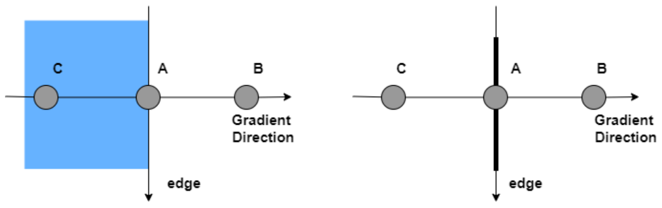

Fig. 4 *Elimination of pixel using gradient suppression*

The principle is shown in Figure4. Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Other two points (B and C) are in gradient directions. Algorithm will perform a check of point A if it forms a local maximum with points B and C. If so, it is considered for next stage, otherwise, it is suppressed and deleted. The final step in the process of edge detection was to set visibility of detected edges using double thresholding and hysteresis thresholding - minimum and maximum. The threshold values are given by operator visually. Any edges with intensity gradient more than maximum value are sure to be edges, while those below the given minimal value are sure to be non-edges, and so are discarded. The edges between the maximum and minimum values are classified as edges or non-edges, based on their connectivity. If they are connected to "sure-edge" pixels, they are considered as part of edges. Otherwise, they are also discarded [27]. For this process (set the values for thresholding), the operator's manual action is required. For easy setup of the limits for edge thresholding, the application provides available mouse-operated slider elements..

The output from picture processing is represented by matrix with the content of points of individual contours of the original image, and is stored in the memory. By cyclically displaying the individual points, it is possible to display found points again in the form of a bitmap for visual inspection.

The individual contours are drawn by "CvInvoke.cvDrawContours ()" library function with a specific redraw parameter, whether as a series of 4 connected points, 8 connected points, or anti-aliasing (smoothed lines). An example is shown in Fig.5.

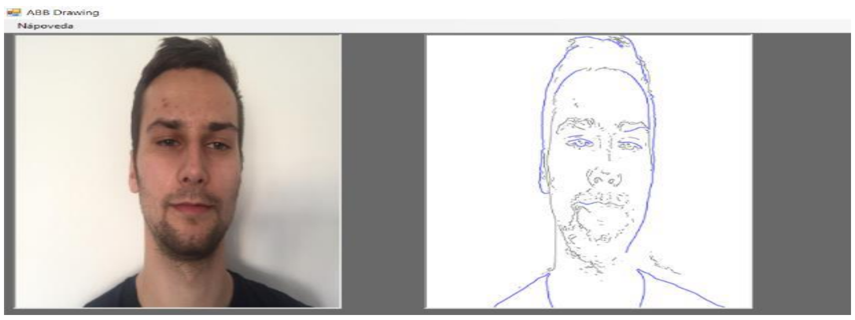

Fig. 5 *Example of edge detection*

Generating a program for a robot

The function available within the "DrawMe ()" libraries is used to export individual points of the found lines in the input image to the robot. The function is similar to the one mentioned above: generating the image. In this case, however, contours do not draw or return an image. The found points are be sent to the field prepared in the RAPID language code. This function always converts the currently processed contour to a point array creates an element of type "Pos", which, in the RAPID code data, represents the position of the point to which the robot has to move (defined by the X, Y and Z coordinates). Subsequently, in the "foreach-in" cycle for each point of contour, function creates one new position element and also increments the two auxiliary integer variables "i" and "j" by the value "1". The position "Pos" is then filled with data using the function of "FillFromString ()". This is used to write variables in RAPID code from an external source. The inserted values are the X and Y coordinates of the current point.

Since it is a test version of the program, we needed to simplify the whole process. The "Z" coordinate was set to fixed value - for movement of the robot arm in the 2D surface. The created position was then written in the data field in the RAPID code with the index "i". After finishing the "for-each" cycle, we created another object of class "Pos", which was filled with the data about the coordinates of the last point from the "for-each" cycle, but the coordinate "Z" was changed to the value of 50 mm. This ensured that the robot after the cycle moved up, and only then continued to the first point of the next line. In the end of the cycle, the variables "i" and "j" were cleared, the number of found points was written into the variable in RAPID. This was used in the next cycle.

```
Point[] pts = currentContour.ToArray();
If          (NumbOfPoint <= 2500) //check for max. number of points =<2500 (protection
for overflow in RAPID)
        {
                targetsNum = 0; // set the nuber of points
                j = 0;
                Pos rt;
                foreach (Point p in pts)
                {
                              i++;
                              j++;
                              rt = new Pos();
                              rt.FillFromString2(“[“ + (p.X) / 3 + “,“ + (p.Y) / 3 + “,“ +
-2 + “[“);
                              RD_targets.WriteItem(rt, i); //write to positions array
                    }
Pos rt1;
rt1 = new Pos();
rt1.FillFromString2(“[“ + (pts[j - 1].X) / 3 + “,“ + (pts [j - 1].Y) / 3 + “,“ + -50 +
“[“);
RD_targets.WriteItem(rt1, 1); // lift of arm
j = 0;
rd_tgNumValue.FillFromString2(i.ToString());
rd_targetsNum.Value = rd_tgNumValue;
```

Fig. 6 *Program for reading position data*

Program module for robot controller

Industrial robot is controlled by the application stored and running in a PLC robot. Native language for IRB 140 is RAPID language. The program running in the robot is relatively simple and contains only two functions in addition to declaring variables. The first one returns variable of “robtarget” type based on coordinates sent from the main application in control PC. The second one is a function for robot automatic movement and is called directly from “main()” program. It is important to set the arm with a command of linear movement to the basic position above the work surface of the robot:

MoveL Target_10,v200,z100,MyTool\WObj:=Surface;

where Targed10 is stored position, v200 means speed of robot movement (200 mm per sec.), z100 is zone for the robot. The target is given by the first element in the field of stored coordinates in the application returned by the function. The loop “while” is then started by MoveJ command, at the speed “v100” and zo setting “z-fine”. The loop starts to read all data from the stored data field. Then, the index is set to number 2, and is sequentially incremented by 1 in every run. This loop is repeated until the value of the iterator equals the number of read points. That means the whole field was drawn. At the end, the variables of the iterator and the number of elements of the array are cleared and the robot returns to the basic position.

PRACTICAL EXPERIMENTS

Simulation of proposed system can be simply performed in virtual space using IDE RobotStudio from ABB. Simplification of this solution lies in the limitation to 2D graphic. Enhancement of the third dimension is possible, but, for algorithm testing purposes, we used only one image from specimens. So, for the particular testing purposes, the application was used just for drawing lines on paper, using a robot arm according to the model.

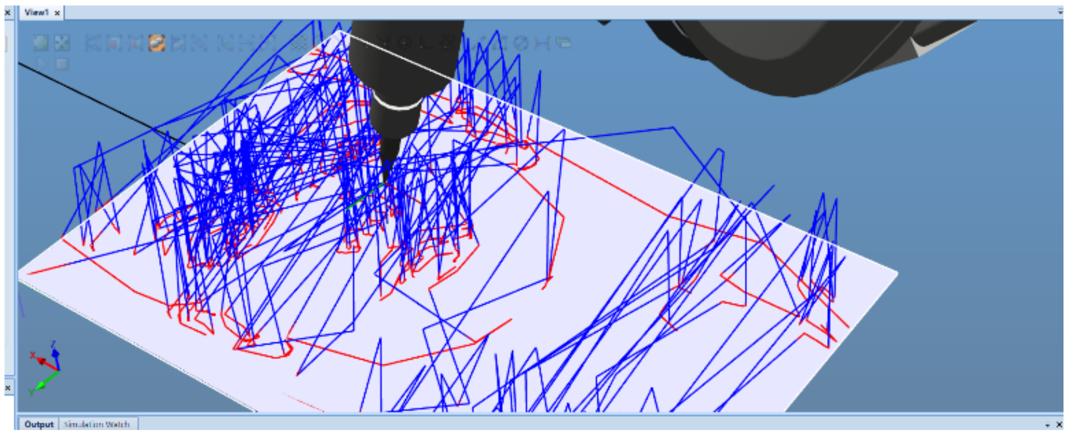

Fig. 7 *Experiment No. 1 – face render*

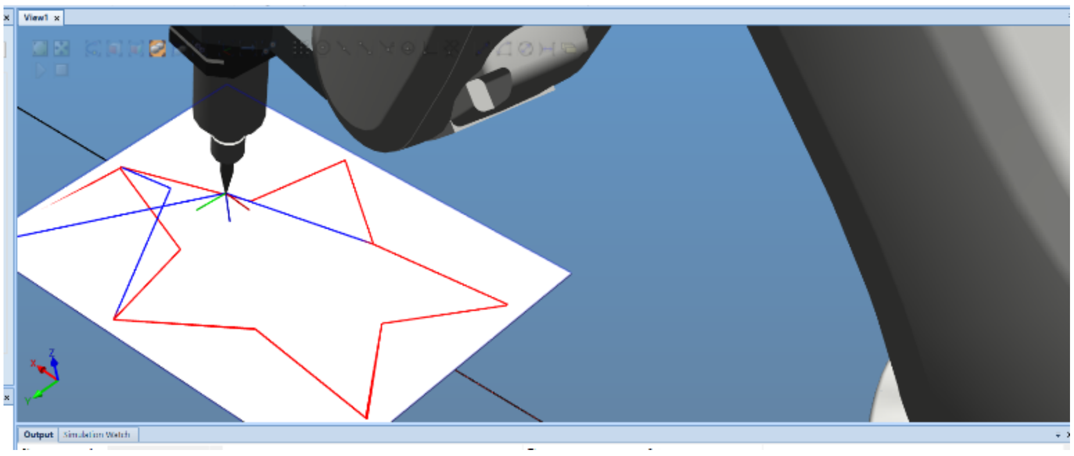

Fig. 8 *Experiment No. 2 –star render*

The tool movement tracking is expressed in lines. Movements on the paper, i.e. the main moves of the tool are marked in red colour. Blue lines indicate the moves of robot, in case the robotic arm is lifted above the surface. In the first image, we can see the rendering of the contours of the face. Due to complexity of image (human face) at the specified speed of v300 (30 cm per second), rendering took a long time, and making the first image took about 219 seconds. The second image is much simpler (star object), and rendering and drawing the second took about 15 seconds.

CONCLUSION

The aim of the research was to propose a system capable of generating data for a robot movement according to the input from external camera or another raster image. After analysing the problem area, several possible solutions are described. One of them was implemented to practice. The problem was divided into two steps, while the whole solution was covered by a simple application. First, the process of picture manipulation was done using the existing algorithms and program libraries for picture manipulation and edge identification. The second part represented the implementation dedicated to generating the data for the robot movement control.

The output from the designed system was tested in the experiments. The plan for future research is to extend the performed system to the work with three-dimensional shapes.

For the purposes of implementation, used were the existing libraries and functions. In conclusion we can say, that the whole system was simple and working, the application was easy to use for operator, and the whole system was easy to implement thanks to the utilisation of the existing libraries.

Acknowledgement

This contribution was published with the support of the Operational Program Integrated Infrastructure within the project of *Výskum v sieti SANET a možnosti jej ďalšieho využitia a rozvoja* (*Research into SANET and options of its further utilisation and development)* code ITMS 313011W988, co-financed by the ERDF.

References

[1] AMERINI, I., ANAGNOSTOPOULOS, A., MAIANO, L., CELSI, L.R., 2021. Deep learning for multimedia forensics *Foundations and Trends in Computer Graphics and Vision.* ISSN 15722740, DOI 10.1561/0600000096.

[2] BHATT, P.M., MALHAN, R.K., RAJENDRAN, P., SHAH, B.C., THAKAR, S., YOON, Y.J., GUPTA, S.K, 2021. Image-Based Surface Defect Detection Using Deep Learning: A Review (2021) *Journal of Computing and Information Science in Engineering*, **21**(4), art. No. 4049535. Cited 2 times. 1) https://www.scopus.com/inward/record.uri?eid=2-s2.0-85101153445&doi=10.11 15%2f1.4049535∂nerID=40&md5=a5213a78218aeb49275b303a6797f40b, DOI 10.1115/1.4049535.

[3] KHAN, M.A., MITTAL, M., GOYAL, L.M., ROY, S., 2021. A deep survey on supervised learning based human detection and activity classification methods. *Multimedia Tools and Applications*, **80**(18), pp. 27867-27923. Cited 3 times. 1) https://www. scopus.com/inward/record.uri?eid=2-s2.0-85108802610&doi=10.1007%2fs11042-021-10811-5∂nerID=40&md5=cee3db6c37193a7a2a94d91aa4295bd1, DOI: 10.1007/s11042-021-10811-5.

[4] GUAN, K.M., ANDERSON, T.I., CREUX, P., KOVSCEK, A.R., 2021. Reconstructing porous media using generative flow networks (2021) Computers and Geosciences, 156, art. No. 104905, . 1) https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112298638&doi=10.1016%2fj.cageo .2021.104905∂nerID=40&md5=5897ecfe28548319f8242a9c8a1d5daf, DOI: 10.1016/j.cageo.2021.104905.

[5] RAVEENDRAN, S., PATIL, M.D., BIRAJDAR, G.K., 2021. Underwater image enhancement: a comprehensive review, recent trends, challenges and applications. Artificial *Intelligence Review*, **54**(7), pp. 5413-5467. 1) https://www.scopus.com/inward/record.uri?eid=2-s2.0-85107324533&doi=10.1007%2fs10462-021-10025-z∂nerID=40&md5=979366750f1ddeedeac 28caeea8583cf, DOI: 10.1007/s10462-021-10025-z.

[6] HUANG, S.W., CHENG, H.M., LIN S.F., 2019. Improved Imaging Resolution of Electrical Impedance Tomography Using Artificial Neural Networks for Image Reconstruction. In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC*), pp. 1551-1554, DOI: 10.1109/EMBC.2019.8856781.

[7] JAKIMOVSKI, G., DAVCEY, D., 2018. Lung cancer medical image recognition using Deep Neural Networks. In: *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pp. 1-5, DOI: 10.1109/ICDIM.2018.8847136.

[8] WU, T. et al., 2019. T-SCNN: A Two-Stage Convolutional Neural Network for Space Target Recognition. In: *IGARSS 2019 – 2019. IEEE International Geoscience and Remote Sensing Symposium*, pp. 1334-1337, DOI: 10.1109/IGARSS.2019.8900185.

[9] DOERING, A., WITTE, H., 1997. Combination of adaptive signal processing and neural classification using an extended backpropagation algorithm. In: *Neural Networks for Signal Processing VII. Proceedings of the 1997. IEEE Signal Processing Society Workshop*, pp. 296-305, DOI: 10.1109/NNSP.1997.622410.

[10] JINGJING, X., JIAOYU, L., 2013. Neural network PID controller auto-tuning design and application. In: *25th Chinese Control and Decision Conference (CCDC)*, pp. 1370-1375, DOI: 10.1109/CCDC.2013.6561139.

[11] KIN, T., SANZEN, Y., KAMIDA, M., AOKI, K., ARAKI, N., WATANABE, Y., 2017. Artificial Neural Network for Unfolding Accelerator-based Neutron Spectrum by Means of Multiple-foil Activation Method. In: *IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pp. 1-2, DOI: 10.1109/NSSMIC.2017.8532892.

[12] LIN, J., ZENG, X., ZUO, Y., JU, Y., LIU, X., 2019. A Deep Neural Network for Antimicrobial Peptide Recognition. In: *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 82-85, DOI: 10.1109/BIBM47256.2019.8983034.
[13] DUMITRIU, T., DUMITRIU, R. P., MANTA, V., 2016. Application of Artificial Neural Networks for modelling drug release from a bicomponent hydrogel system. In: *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 675-680, DOI: 10.1109/ICSTCC.2016.7790744.
[14] BEKDAS, G., NIGDELI, S.M., YÜCEL, M., 2020. Artificial Intelligence and Machine Learning Applications in Civil, Mechanical, and Industrial Engineering. IGI Global. Retrieved from https://app.knovel.com/hotlink/toc/id:kpAIMLACM8/artificial-intelligence/artificial-intelligence
[15] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.229.1285&rep=rep1&type=pdf
OLSEN, S., GOOCH, B., 2016. Image Simplification and Vectorization [Online] [Accessed: 11-2017]. Available at https: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.229. 1285&rep=rep1&type=pdf
[16] YUAN, L., XU, X. 2015. Adaptive Image Edge Detection Algorithm Based on Canny Operator. *2015.* In: *4th International Conference on Advanced Information Technology and Sensor Application (AITS)*, pp. 28-31, DOI: 10.1109/AITS.2015.14.
[17] TASNEEM, T., AFROZE, Z., 2019. A New Method of Improving Performance of Canny Edge Detection. In: *2nd International Conference on Innovation in Engineering and Technology (ICIET)*, pp. 1-5, DOI: 10.1109/ICIET48527.2019.9290676.
[18] BERA, A., 2011. Fast vectorization and upscaling images with natural objects using canny edge detection. In: *3rd International Conference on Electronics Computer Technology*, pp. 164-167, DOI: 10.1109/ICECTECH.2011.5941823.
[19] NIDELEA, M., ALEXEI, A. M., 2012. Method of the Square — A new algorithm for image vectorization. In: *9th International Conference on Communications (COMM)*, pp. 115-118, DOI: 10.1109/ICComm.2012.6262618.
[20] KIRSANOV, A., VAVILIN, A., JO, K., 2010. Contour-based algorithm for vectorization of satellite images. In: *International Forum on Strategic Technology 2010*, pp. 241-245, DOI: 10.1109/IFOST.2010.5668109.
[21] MYERS, D.R., PRITCHARD, M.J., BROWN, M.D.J., 2001. Automated programming of an industrial robot through teach-by showing. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164*), pp. 4078-4083 Vol.4, DOI: 10.1109/ROBOT.2001.933255.
[22] FRIEDRICH, H., HOLLE, DILLMANN, J.R., 1998. Interactive generation of flexible robot programs. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, pp. 538-543 vol.1, DOI: 10.1109/ROBOT.1998.677029.
[23] CHULHEE, Y., JAEGON, A., YEON-HO, K., 2013. A fusion of computer vision technique and a visual programming language for edutainment robots. In: *IEEE ISR 2013*, pp. 1-5, DOI: 10.1109/ISR.2013.6695728.
[24] NETO, P., PIRES, J.N., MOREIRA, A.P., 2010. CAD-based off-line robot programming. In: *IEEE Conference on Robotics, Automation and Mechatronics*, pp. 516-521, DOI: 10.1109/RAMECH.2010.5513141.
[25] https://new.abb.com/products/robotics/industrial-robots/irb-140 [Online], [Accessed: 09-2021] Available at https://new.abb.com/products/robotics/industrial-robots/irb-140.
[26] https://www.emgu.com/wiki/index.php/Main_Page [Online], [Accessed: 09-2021] Emgu CV cross platform. Net wrapper. Available at https://www.emgu.com/wiki/index.php/Main_Page.
[27] https://docs.opencv.org/3.4.15/d7/de1/tutorial_js_canny.html [Online], [Accessed: 09-2021], Available at https://docs.opencv.org/3.4.15/d7/de1/tutorial_js_canny.html.

ORCID

Igor Halenár 0000-0002-5445-3625