Image Processing on Raspberry Pi for Mobile Robotics

K. Horak
Brno University of Technology/Department of Control and Instrumentation, Brno, Czech Republic
Email: horak@feec.vutbr.cz

L. Zalud
Brno University of Technology/Department of Control and Instrumentation, Brno, Czech Republic
Email: zalud@feec.vutbr.cz

***Abstract*—We have designed a small robot called Cube for simple robotics tasks and students works. The Cube carries only small and lightweight computational platform Raspberry Pi equipped with a camera. This paper serves as an introduction to using image processing methods on a Raspberry Pi platform via Simulink in Matlab. We have designed a cheap framework for fast image processing in real-time including feature extraction and image transformation methods. In the paper a several selected methods frequently used not only in robotics are implemented and tested on the Raspberry Pi 2 equipped with a native camera board. Algorithms for edge, corner and line detection have been implemented by using Simulink with the Computer Vision System Toolbox and other in-build tools.**

***Index Terms*—image processing, Raspberry Pi platform, Matlab Simulink, feature extraction.**

I. Introduction

Machine vision is very often a substantial part of cybernetics systems for example in an industrial inspection, traffic monitoring, security surveillance systems, and naturally in mobile robotics [1,2].

A lot of various sensors (e.g. ultrasonic or lasers rangefinders, accelerometers, LED-based sensors, GPS, etc.) are used for data gathering and mobile robot itself navigation. Unfortunately, a machine vision instrumentation is regularly one of the most expensive part of mobile robots systems, which is inconvenient especially for students and research purposes. For that reason, we have decided to design and implement a concept of a cheap and universal mobile robotic platform. We have named it a “Cube” due to its simple shape and its straightforward purpose for beginners’ tasks.

A. Cube

Horizontal dimensions of the Cube are approx. 150 millimetres in both directions. A chassis and wheels of our new one-axle robot has been designed and printed by a low-cost 3D printer. The robot is equipped with two stepper motors driven and powered by a specially designed printed-circuit-board connected to the Raspberry Pi 2 platform [3]. A functional prototype of the Cube robot can be seen in the Fig. 1 on the left.

B. Raspberry Pi 2 Platform

The Raspberry Pi platform originally intended for educational purposes has become famous immediately after its introduction in 2012. The next-generation Raspberry Pi 2 (RPi2 hereinafter) was released in 2015 when almost six million Raspberry Pi of the first generation have been already sold. The second generation is based on the Broadcom BCM2836 system on a chip with a quad-core ARM Cortex-A7 processor and a VideoCore IV dual-core GPU with 1 GB of RAM. This small, universal and still good-performance platform is self-evidently suitable for such mobile applications in machine vision and robotics as mentioned above in the introduction chapter.

For machine vision purposes a native Raspberry Pi Camera board (RPiCam hereinafter) is available on the shelf. The RPiCam employs a 5-megapixels CMOS sensor Omnivision 5647 with a resolution of 2592 by 1944 pixels and transfer rates of 1080p/30 or 720p/60 fps.

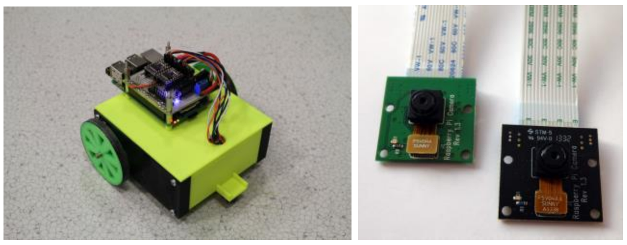
Figure 1. The small mobile robot Cube designed and printed on 3D printer at Brno University of Technology equipped with the Raspberry Pi 2 platform (on the left) and a couple of available native camera boards for visible and combined visible and IR light (on the right).

Manuscript received October 16, 2015; revised November 18, 2015; accepted November 18, 2015. Authors cordially thanks to the grant “The research of new control methods, measurement procedures and intelligent instruments in automation” No. FEKT-S-14-2429 from the Internal Grant Agency of the Brno University of Technology.

K. Horak is with the Department of Control and Instrumentation, Brno University of Technology, Brno, 61600, Czech Republic (phone: 00420-5-4114-6417; e-mail: horak@feec.vutbr.cz).

L. Zalud is with the Department of Control and Instrumentation, Brno University of Technology, Brno, 61600, Czech Republic (phone: 00420-5-4114-6416; e-mail: zalud@feec.vutbr.cz).

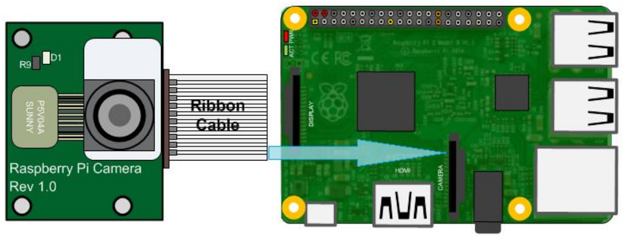

Figure 2. The Raspberry Pi Camera board connected to a camera connector on the Raspberry Pi 2 platform by ribbon flex-cable.

The image sensor is assembled in a fixed-focus module with either IR blocking filter or without it. We have used a NoIR version of the RPiCam with removed infra-red filter along our experiments. A camera is then sensitive to short-wavelength IR radiation around 880 nm besides general visible spectrum.

The RPiCam board itself is tiny at around 25x20x9 millimetres and is directly connected to a camera connector on the RPi2 platform by flat cable for higher operability. As is illustratively depicted in the Fig. 2, the RPi2 platform contains further interfaces for HDMI, USB, RJ45, GPIO, audio jack and Raspberry display devices.

We have only employed the power supply connector and Ethernet cable for communication between the RPi2 and PC during our development. All algorithms we have designed in the Simulink were deployed into the ARM processor on the RPi2 via the Ethernet cable.

II. Image Processing Methods

There is a plenty of various image enhancement methods, conversions, geometric and morphologic transformations, feature extraction and objects recognition techniques in a computer vision and image processing theory [4-9]. We have decided to implement and test several geometric-based feature detection methods frequently used in robotics for environment understanding and objects recognition [5,10].

The three representative algorithms for edge, corner and line detection have been selected and taken into account in this paper for implementing in the Simulink and performance measuring on the RPi2 platform.

A. *Edge Detection by Sobel operator*

Several different approaches how to detect edges in images exist [11]. The most common way to obtain an edge-image uses a convolution. An original input image is convolved with some gradient operator, which is selected from the set of well-known operators in advance. These gradient operators are of two categories (a first-order and second-order) depending on whether they

$$H_{Prewitt} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \qquad H_{Laplacian4} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{1}$$

$$H_{Sobel} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad H_{Laplacian8} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

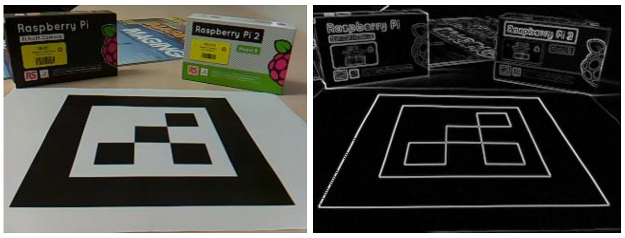
Figure 3. The input RGB image from RPiCam with resolution of 320x240 pixels and the relevant result of the Sobel operator.

approximate either a first derivative (Prewitt, Sobel and others) or a second derivative (zero crossing of the Laplacian) of the input image. The Prewitt and Sobel operator for vertical lines and the Laplacian operator for 4-neighbourhood (orthogonal) and 8-neighbourhood (orthogonal + diagonal) are given by the equation (1).

By comparing coefficients in matrices $H_{Prewitt}$ and H_{Sobel} can be quickly seen that the Sobel edge detector is very similar to the Prewitt with only difference in emphasizing centre pixels (i.e. nearest pixels in 4-neighbourhood of the central one). This results in insignificant differences between output edge-images of these two operators in practice. Fig. 3 shows an original image of a scene with an ARTag (marker with unique code used e.g. for indoor robots navigation) and the resulting edge-image using the Sobel operator.

Edge images, similar to the depicted one, are either important inputs to scene understanding blocks [12] or even necessary inputs to several image transforms as for example the Hough transform described below.

B. *Corner Detection by Harris operator*

A corner detection is the next qualitative level in an image understanding. All known corner detectors generally differ from edge detectors in stronger response in places where corner is present in comparison with places where only edge appears [11].

The most simple corner detector is the Moravec operator only summing eight differences between a central pixel and pixels in the 8-neighbourhood. More sophisticated corner detectors named after their authors are Harris&Stephens algorithm, Shi&Tomasi algorithm and finally Rosten&Drummond algorithm. We have used the Harris&Stephens corner detector in our Simulink tests. It is based on an autocorrelation function of a small patch of the input image. A first step in the Harris algorithm is to compute a Harris matrix, the matrix containing all combinations of gradients in horizontal and vertical direction. When the horizontal and vertical gradients are denoted as I_x and I_y, respectively, the Harris matrix A will be given by the equation (2).

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2}$$

Corners are places in an image where gradients are significant in both the horizontal and the vertical direction simultaneously. It follows eigenvalues λ_1 and λ_2 of the Harris matrix can distinguish flat patches, edges and corners from each other. In order to do this, easy to

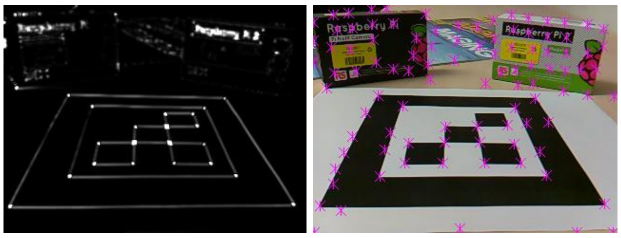
Figure 4. A metric map as a grayscale image computed by the Harris&Stephens algorithm (on the left) and strongest detected corners imprinted into the input image (on the right).

compute function H indicating corner by its high value were suggested by Harris&Stephnes. The H function is given by an equation (3), where the det(A) stands for a matrix determinant and the trace(A) stands for a sum of values along its main diagonal.

$$H = \lambda_1 \cdot \lambda_2 - \kappa \cdot (\lambda_1 + \lambda_2)^2 = \det(A) - \kappa \cdot trace^2(A) \quad (3)$$

Note that a second mentioned corner detector Shi&Thomasi directly computes a minimal value of λ_1 and λ_2.instead of the more complex function H in Harris&Stephens algorithm. Fig. 4 shows a metric map of the Harris&Stephens algorithm and a set of detected maximums in this metric map depicted into the original image by purple crosses.

Corners play, similarly as edges, very important role in image understanding step as input points for computation region descriptors e.g. SURF, MSER, BRISK, etc. [13]. Usage of such stable and reliable descriptors facilitates object tracking and recognition in an unknown scene.

C. *Line Detection by Hough transform*

The Hough transform is conversion of coordinates of all non-zero pixels in an input binary image to an abstract space of parameters. These parameters are exactly given by an analytical equation of a sought-after shape (e.g. line, circle, ellipse, etc.). In that resultant parameters space a desired number of peaks (i.e. global maximums) are detected and finally respective space coordinates are then transformed back into the image plane in form of detected entity (line, circle, etc.) [6,7].

The most common is the Hough transform for line and circles detection. The first one play a significant role in mobile robotics navigation and scene understanding. For example a thresholded (binary) version of the edge image in the Fig. 3 can serves as an input to Hough transform for lines detection.

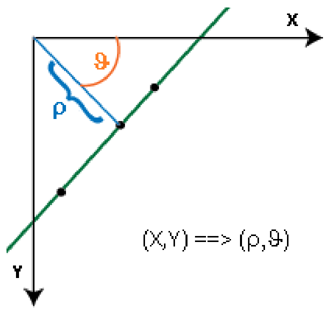

Figure 5. The coordinate space (X,Y) containing only one illustrative line with the three highlighted dots.

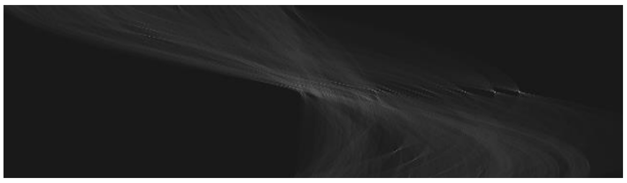
Figure 6. The accumulator of the Hough transform in form of a grayscale image containing several peaks representing respective number of lines in the input image.

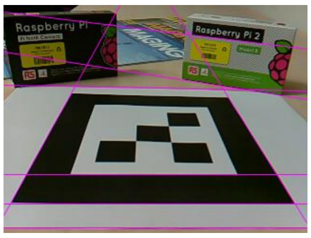
Figure 7. Eight lines related to the eight strongest peaks in the accumulator in the Fig. 6 imprinted into the input image.

An arbitrary line of perpendicular distance ρ (rho) from the origin and angle ϑ (theta) of inclination of a normal line from the x-axis can be represented in parametric form given by the equation (4).

$$\rho = x \cdot \cos(\vartheta) + y \cdot \sin(\vartheta) \quad (4)$$

In this case, a two-dimensional space of image coordinates (X,Y) is mapped into a new space of line-equation parameters (ρ,ϑ) often also called a Hough accumulator (see Fig. 6). As can be illustratively seen from the Fig. 5, the x and y coordinates of each nonzero pixel in the input binary image (three black highlighted dots) are transformed to the one couple of parameters (ρ,ϑ). A relevant point of the computed coordinates ρ and ϑ is increased by 1 in the accumulator. Such voting system causes that all pixels lying in one straight line create a peak in the accumulator with maximal value equal to a number of pixels in the line. Non-ideal (i.e. not exactly straight) shape of line results in a blurred peak, but still distinguished from generally lower values in a neighbourhood of this peak. Coordinates ρ and ϑ of each local maximum can be simply displayed in the input image as lines (see Fig. 7). User itself via an algorithm selects a number of strongest peaks to be displayed.

The lines detected in the input image are, again, very convenient features especially for robot navigation in both indoor and outdoor environments [1].

III. IMPLEMENTATION IN MATLAB SIMULINK

Matlab is well-known and, in certain sense, also powerful tool for scientific computations. It is equipped with the Image Processing Toolbox and Image Acquisition Toolbox for computer vision tasks. A plenty of code in form of various m-files have been written and is often shared at MathWorks exchange web pages. We decided to implement above mentioned image processing methods to the RPi2 by means of the Simulink®, a graphical tool for technical simulations and devices programming. Tools in Simulink enable design of such applications by using the Support Package for Raspberry

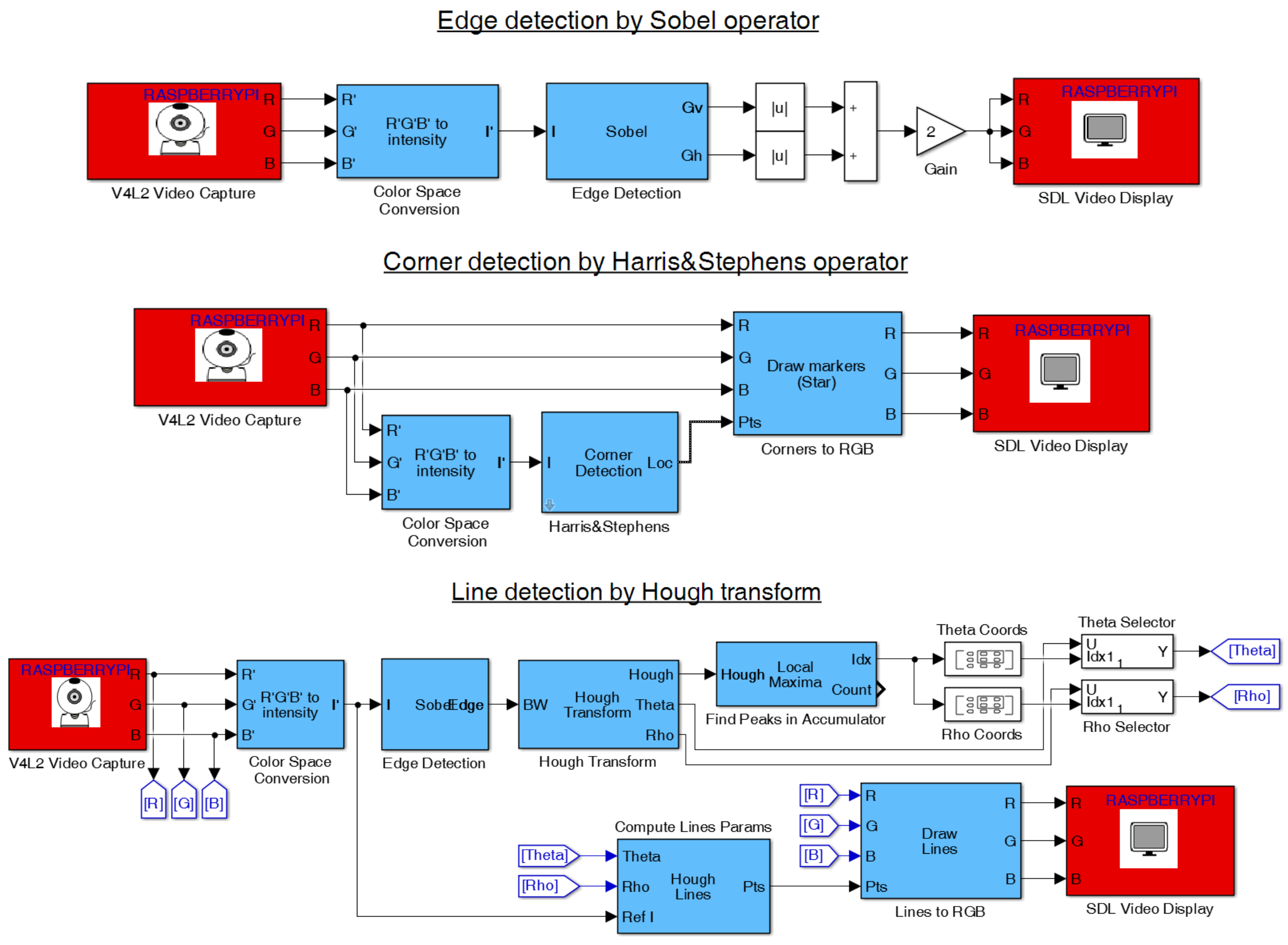

Figure 8. Simulink models for edge, corner and line detection.

Pi Hardware™ and the Computer Vision Toolbox. Each one of the three above mentioned methods has been designed in its own scheme. All the three Simulink models of edge, corner and line detectors are depicted in the Fig. 8. Red (dark in grayscale) blocks belong to the Support Package for Raspberry Pi Hardware, blue (pale in grayscale) blocks belong to the Computer Vision System Toolbox and white (transparent in grayscale) blocks are general Simulink math and other operations.

At a testing stage, each model have been deployed directly to the RPi2 platform by an Ethernet cable between the RPi2 and computer with Simulink models. A performance of the RPi2 platform on machine vision tasks were measured by the FPS indicator (frames per second) independently for all models. Moreover, each model has been tested at four generally used resolutions – 160x120, 320x240, 640x480 and 800x600 pixels.

At the most top position in the Fig. 8, the Simulink model of the edge detection is depicted. This model uses a block of the Sobel operator with the two output signals Gv and Gh related to the vertical and horizontal edges, respectively. The signals Gv and Gh are added up and straightforwardly distributed to an output display. Further, in the middle of the Fig. 8, the Simulink model of the corner detection is depicted. The corner detection block implements corners detection algorithm in the input grayscale image. The block has three options of algorithms used: the Harris&Stephens, the Shi&Tomasi (both described above) and the Rosten&Drummond. Note that the last one mentioned algorithm uses a local intensity comparison, an effective technique resulting in that algorithm is approximately twice as faster as the each one from the previous couple. Furthermore, a maximum number of detected corners were saturated by a number of 100 and the parameter κ (kappa) in the equation (3) were permanently equal to 0.04 during our experiments.

Finally, at the bottom position in the Fig. 8, the Simulink model for the line detection by means of the Hough transform is depicted. As can be seen from the Simulink model, the Hough transform block works with the thresholded (i.e. binary) image obtained by the Sobel operator applied directly on the input RGB image. This block outputs besides the rho and theta vectors also a Hough space, it means already mentioned accumulator. The next block „Local Maxima” finds no more than given number of strongest peaks in the accumulator and outputs rho and theta coordinates in it. Note that the range of theta is $-\pi/2 \leq \theta < +\pi/2$ with a step-size $\pi/180$ radians. It means lines in the input image are detected in precision of 1 degree and left and right side of the Hough accumulator in the Fig. 6 relates to line inclination of -90 degree and +90 degree (vertical lines). The last important block, however needed only for visualization purposes, is „Hough lines” intended for computation of ending points of the detected lines. These ending points are subsequently used for imprinting the lines into the input RGB image. Last, but not least, several submatrix and selector blocks have been employed for the rho and theta vectors operations.

IV. Experimental Results

As it was already mentioned in the text above, all the three Simulink models were designed, implemented and finally verified and tested on the RPi2 platform at four different resolutions. The performance of the feature extraction methods has been measured and compared in this paper by means of the frames per second indicator.

TABLE I. Comparison of FPS Values for the Designed Models and Different Image Resolutions

FPS	160x120	320x240	640x480	800x600
Edge detection (Sobel operator)	72.4	18.2	4.5	2.9
Corner detection (Harris&Stephens)	22.6	4.7	1.2	0.7
Line detection (Hough transform)	19.6	5.7	1.6	1.0

Both the internal (in Simulink) and external (deploying to RPi2) modes were tested, nevertheless only the second one is important for our purposes of the cheap robotics platform. Values of FPS are shown in the Table I.

Note that all models described in the previous chapter were deployed directly onto the RPi2 platform during the test stage, but resultant images were displayed back on a monitor of a workstation with the Simulink due to absence of a Raspberry Pi proprietary display. A transfer of image data between the RPi2 platform and the workstation may influences the FPS values in the table. It follows, the values in the Table I. relate to the worst cases and the FPS values should be higher if any visualization block had not been implemented or had been implemented directly on the RPi2 platform.

To sum up, we may conclude that the measured FPS values are relatively good and the RPi2 platform is certainly useable for majority of usual robotics tasks, especially when optimizations over the Simulink schemes and target code (C/C++) have been carried out.

Acknowledgment

This work was supported by the grant “Research of New Control Methods, Measurement Procedures and Intelligent Instruments in Automation” No. FEKT-S-14-2429 from the Internal Grant Agency at BUT.

References

[1] Zalud, L., Kocmanova, P., Burian, F., Jilek, T. Color and Thermal Image Fusion for Augmented Color and Thermal Image Fusion for Augmented Reality in Rescue Robotics. Lecture Notes in Electrical Engineering. Springer, 2014. pp. 47-56. ISBN: 978-981-4585-42- 2.

[2] Klecka, J., Horak, K. Fusion of 3D Model and Uncalibrated Stereo Reconstruction. Advances in Intelligent Systems and Computing. Springer, 2015. pp. 343-351. ISBN 978-3-319-19823- 1.

[3] Richardson, M., Wallace, S. Getting Started with Raspberry Pi. Maker Media, Inc., 2012. 176 pages. ISBN 978-1-4493-4421-4.

[4] Vernon, D. Machine Vision: Automated Visual Inspection and Robot Vision. Hemel Hempstead: Prentice Hall International Ltd., 1991. 260 p. ISBN 0-13-543398-3.

[5] Russ, J.C. The Image Processing Handbook. CRC Press Inc., Boca Raton Florida, 1994. ISBN 0-8493-2516-1.

[6] Kropatsch, W. G., Bischof H. Digital Image Analysis. Springer-Verlag New York, Inc. 2001, 505 pages. ISBN 0-387-95066-4.

[7] Gonzales, R. C., Woods R. E. Digital Image Processing - Third Edition. Pearson Education, Inc. 2008, 954 pages. ISBN 978-0-13-168728-8.

[8] Young, I.T., Gerbrands, J.J., Vliet, L.J. Fundamentals of Image Processing. TU Delft, 1998. 113 p. ISBN 90-75691-01-7.

[9] Nikhil, R. P., Sankar, K. P., A review on image segmentation techniques, Pattern Recognition, Volume 26, Issue 9, September 1993, Pages 1277-1294, ISSN 0031-3203.

[10] Horak, K. Detection of Symmetric Features in Images. *International Conference on Telecommunications and Signal Processing.* 2013. pp. 895-899. ISBN 978-1-4799-0402- 0.

[11] Sonka, M., Hlavac, V., Boyle, R. Image Processing, Analysis and Machine Vision. Toronto: Thomson, 2008. 829 p. ISBN 978-0-495-08252-1.

[12] Papageorgiou, C.P., Oren, M., Poggio, T. A General Framework for Object Detection, Proceedings of the 6th International Conference on Computer Vision, pp. 555-562, 1998, IEEE Computer Society, Washington. ISBN 81-7319-221-9.

[13] Lowe, D. G., Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, Springer Netherlands, 2004, pp. 91-110, ISSN 0920-5691.

Karel Horak was born in Pardubice, Czech Republic, on August 1, 1979. He received the M.Sc. and Ph.D. degrees in cybernetics, control and measurements in 2004 and 2008, respectively, from Brno University of Technology.

He is currently working as an assistant professor at the Department of Control and Instrumentation, BUT, where he heads the Computer Vision Group. He has authored and co-authored almost fifty papers and journal articles in areas of computer vision and image processing.

Ludek Zalud was born in Brno, Czech Republic, on August 18, 1975. He received the M.Sc. and Ph.D. degrees in cybernetics, control and measurements in 1998 and 2002, respectively, from BUT.

He is currently working as an associate professor at the Department of Control and Instrumentation, BUT, where he heads the Robotics Group. He has authored and co-authored more than one hundred papers and journal articles in area of robotics and bioengineering.