CANNY EDGE DETECTOR ALGORITHM OPTIMIZATION USING 2D SPATIAL SEPARABLE CONVOLUTION

Martin KRÁLIK, Libor LADÁNYI
SEC Technologies, s.r.o, 1. mája 4269, 031 01 Liptovský Mikuláš, Slovakia, tel. +421911307378,
E-mail: info@sec-technologies.com

ABSTRACT

In the case of real-time image processing, it is necessary to determine the computational complexity of the mathematical operations used. Reduction of computational complexity of 2D discrete convolution can be achieved by using a separable convolution. In this article, we focus on the application of a canny edge detector for different types of images. The main goal was to speed up the process of applying the kernel matrix to a given image using a separable convolution. By applying a separable convolution, we compared the duration of the Gaussian filter application, edges detection and the Hysteresis threshold level. Applying a separable convolution should speed up the duration of the 2D Gaussian filter as well as the edge detection. The main variable that interested us was time, but an important factor in the application of the filter and edge detection is the number of operating cycles. The use of a separable convolution should significantly reduce the number of computational cycles and reduces the duration of filter application and detection.

Keywords: *Canny edge detector, Gaussian filter, discrete convolution, separable convolution, kernel matrix*

1. INTRODUCTION

Edge detection is one of the most fundamental algorithms in digital image processing. The edge detection principle is to identify areas which large change in intensity occurs. It works by detecting discontinuities in brightness. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods. Among various edge detection techniques and methods, Canny edge detector (CED), is the one of most used.

Canny operator presents three good performance criteria as Signal to Noise Ratio (SNR), localization precision and single edge response [1,2,3]. Due to these characteristics, CED is the most implemented edge detection algorithm because of its ability to detect edges even in images that are intensely contaminated by noise. These properties predetermine its use in areas such as medical imaging [4-8].

Edge detection in medical imaging is a significant task for object recognition. Medical diagnostics is one area that has been significantly advanced by using digital images. Medical digital images include magnetic resonance imaging (MRI), positron emission tomography (PET) and computed tomography (CT). One of the problems with digital images takes by using one of these technics is that they typically contain significant speckle noise and other artifacts which complicate image interpretation and automatic processing [9,10,11]. This requires some image enhancement and image segmentation processing in order to make the image better and clearer for diagnostics purpose. In such a case, it is possible to use CED and this is also one of our goals, where we attempt for the best possible extraction of data from the image with high accuracy and short evaluation time.

When we talk about time, it is related to the volume of processed data, but it is also important to deal with the hardware equipment and the possibility of speeding up the operation using various sharing technics and parallel processing [12,13,14,15].

Currently, there is a significant implementation of automated processes in almost all scientific, industrial and medical areas like we mentioned in previous section. The main goal of our research is the detection of objects (buildings) using CED. Automatic object identification is a very complex issue, especially in cases where it is necessary to detect and recognize various geometrically complex and very similar objects [16,17].

As already mentioned in the previous sections, in the case of object detection, noise occurs in the acquired images. There are three main objectives for CED. The first is find the intensity gradients of the image. Second is to apply non-maximum suppression to get rid of spurious response to edge detection and the last step, apply double threshold to determine potential edges [18-22]. The Gaussian reduces the effect of noise present in the image. In this paper, we have used a separable convolution to speed up the process of Sobel edge detection. We were interested in the computation time of applying the Gaussian filter and edge detection, as well as the number of operations required by application of the filters and edge detection.

2. THEORETICAL BACKGROUND

In this part of the work are listed all the theoretical basis used in the Canny edge detector algorithm. In this section we will also describe the theory of computational complexity of Canny edge detection algorithm. Computational complexity of 2D discrete convolution can be achieved by using a separable convolution.

2.1. Gaussian filter

Gaussian filter also called Gaussian smoothing filter in digital image processing, is an effective way to reduce noise and enhance details in an image. Applying a Gauss filter to the image is the first step in edge detection using a Canny detector. The main advantage of the Gaussian filter is its frequency response, which also represents a Gaussian distribution centered around the zero frequency. Two-dimensional Gaussian function centered on x=y=0 is defined by equation [23-26]

$$G = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad (1)$$

The kernel of the two-dimensional (2D) Gaussian function is generally given by a matrix with minimum required dimension (three-sigma rule) n×n, where n=2×3×σ+1 and σ is standard deviation. Constructing a 2D Gaussian kernel can be realized by sampling of the continuous 2D Gaussian function defined by Eq. (1) followed by the normalization of all elements. In the case of σ=1, the minimum required dimension of the matrix **G** representing the 2D Gaussian kernel will be 7×7 with the following normalized elements

0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0018	0.0216	0.0966	0.1592	0.0966	0.0216	0.0018
0.0011	0.0131	0.0586	0.0966	0.0586	0.0131	0.0011
0.0002	0.0029	0.0131	0.0216	0.0131	0.0029	0.0002
0.0000	0.0002	0.0011	0.0018	0.0011	0.0002	0.0000

Since Eq. (2) describes the normalized elements of the Gaussian kernel, the term $1/(2\pi\sigma^2)$ in Eq. 1 can be neglected. The advantage of the 2D Gaussian kernel defined by matrix **G** is its separability, so in this case it is sufficient to generate a one-dimensional (1D) Gaussian kernel defined by a matrix $\mathbf{G}_x$ with dimension 1×7, matrix **G** can be then obtained as $\mathbf{G}_x^{\mathrm{T}}\times\mathbf{G}_x$. 1D Gaussian function centered on x=0 with the neglected term $1/(2\pi\sigma^2)$ is defined by equation

$$G = e^{-\left(\frac{x^2}{2\sigma^2}\right)} \quad (2)$$

The 1D Gaussian kernel (σ=1) with the normalized elements is defined by the following vector $\mathbf{G}_x$

0.0044	0.0540	0.2420	0.3911	0.2420	0.0540	0.0044

Another way to construct the Gaussian kernel is for example binomial approximation of Gaussian distribution. The smoothing of the input image using the Gaussian filter is realized using 2D discrete convolution of the input image and the Gaussian kernel. 2D discrete convolution of input image and Gaussian kernel is defined by equation

$$\mathbf{H}(x,y) = (\mathbf{I} * \mathbf{G})(x,y) = \sum_i \sum_j \mathbf{I}(x-i,y-j)\mathbf{G}(i,j) \quad (3)$$

where **I** is the matrix representing input image, **G** is the matrix representing Gaussian kernel and **H** is the resulting matrix representing filtered image (i and j takes values from 0 to n).

In the case of real-time image processing, it is necessary to determine the computational complexity of the mathematical operations used. The computational complexity of the 2D discrete convolution is defined as $O(A\times B\times n^2)$, where $A\times B$ is the size of the matrix representing the input image and n^2 is the size of the matrix representing the Gaussian kernel. Reduction of computational complexity of 2D discrete convolution can be achieved by using a separable convolution. There are two types of separable convolution, namely spatial (SSC) and depthwise (DSC) separable convolution. In the case of the use of the SSC, the computational complexity of the 2D discrete convolution is defined as $O(A\times B\times n)$. As already mentioned, the Gaussian kernel is separable, which allows to reduce the computational complexity using the 2D discrete spatial separable convolution. 2D discrete spatial separable convolution of input image and Gaussian kernel can be write as

$$\mathbf{H}(x,y) = \left[\mathbf{I}(x,y) * \mathbf{G}_y(y)\right] * \mathbf{G}_x(x) = \sum_i \left[\sum_j \mathbf{I}(x-i,y-j)\mathbf{G}_y(j)\right] \mathbf{G}_x(i) \quad (4)$$

where $\mathbf{G}_y=\mathbf{G}_x^{\mathrm{T}}$.

2.2. Calculation of the gradient components and their absolute magnitude and direction

The calculation of the gradient components $\mathbf{D}_a$ and $\mathbf{D}_b$ is performed as a convolution of the input image **I** with matrices $\mathbf{h}_a$ and $\mathbf{h}_b$ representing the Sobel operators. The advantage of the Sobel operators is that it is separable as in the case of the Gaussian kernel. The calculation of the gradient component $\mathbf{D}_a$ using the 2D discrete spatial separable convolution is defined by equation

$$\mathbf{D}_a(x,y) = [\mathbf{I} * \mathbf{h}_a](x,y) = \left[\mathbf{I}(x,y) * \boldsymbol{h}_{a_y}(y)\right] * \mathbf{h}_{a_x}(x) = \sum_i \left[\sum_j \mathbf{I}(x-i,y-j)\mathbf{h}_{a_y}(j)\right] \mathbf{h}_{a_x}(i) \quad (5)$$

The gradient component $\mathbf{D}_b$ is calculated in the same way as $\mathbf{D}_a$. The Sobel operators $\mathbf{h}_a$ and $\mathbf{h}_b$ are defined as follows [27]

$$\mathbf{h}_a = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad \boldsymbol{h}_b = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

wherein the matrices $\mathbf{h}_a$ and $\mathbf{h}_b$ can be written as follows

$$\mathbf{h}_a = \mathbf{h}_{a_y} \times \boldsymbol{h}_{a_x}, \quad \mathbf{h}_b = \mathbf{h}_{b_y} \times \boldsymbol{h}_{b_x} \quad (6)$$

$$\mathbf{h}_{a_x} = [1 \;\; 0 \;\; -1], \quad \boldsymbol{h}_{a_y} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{h}_{b_x} = [1 \;\; 2 \;\; 1], \quad \boldsymbol{h}_{b_y} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

The absolute magnitude **D** and the direction **Δ** of the gradient components $\mathbf{D}_a$ and $\mathbf{D}_b$ are defined by the following equations [25-28]

$$\mathbf{D} = \sqrt{\mathbf{D}_a^2 + \mathbf{D}_b^2} \quad (7)$$

$$\boldsymbol{\Delta} = \tan^{-1}\left(\frac{\mathbf{D}_b}{\mathbf{D}_a}\right) \quad (8)$$

3. RESULTS AND DISCUSSIONS

The results obtained in this paper are mainly focused on optimizing the computational complexity of the Canny edge detector. The high computational complexity of the Canny edge detector lies mainly in the use of several convolution operations used in the mathematical computational algorithm. The Canny edge detector algorithm requires a triple convolution. The first convolution is used in Gaussian smoothing. The second and third convolution is used in edge detection in the part performing the calculation of the gradient components $\mathbf{D}_a$ and $\mathbf{D}_b$. It is the convolutional calculation that represents the most time-consuming part in the detector. The table 1(a-b) shows the required number of arithmetic operations (image with a resolution of 576×720 pixels) using normal 2D convolution.

Table 1(a) Number of arithmetic operations using normal 2D convolution

σ	Gaussian smoothing		Gradient components	
	×	+	×	+
1	20 321 280	19 906 560	7 464 960	6 635 520
2	70 087 680	69 672 960		
3	149 713 920	149 299 200		
4	259 200 000	258 785 280		
5	398 545 920	398 131 200		
6	567 751 680	567 336 960		

Table 1(b) Total arithmetic operations (Gaussian smoothing + calculation of the gradient components)

σ	×, +
1	54 328 320
2	153 861 120
3	313 113 600
4	532 085 760
5	810 777 600
6	1 149 189 120

One possible way to reduce the computational complexity of the Canny edge detector algorithm is to use a 2D spatial separable convolution. The number of required arithmetic operations (image with a resolution of 576×720 pixels) in the case of 2D spatial separable convolution is given in table 2(a-b).

Table 2(a) Number of arithmetic operations using 2D spatial separable convolution

σ	Gaussian smoothing		Gradient components	
	×	+	×	+
1	5 806 080	4 976 640	4 976 640	3 317 760
2	10 782 720	9 953 280		
3	15 759 360	14 929 920		
4	20 736 000	19 906 560		
5	25 712 640	24 883 200		
6	30 689 280	29 859 840		

Table 2(b) Total arithmetic operations (Gaussian smoothing + calculation of the gradient components)

σ	Total
1	19 077 120
2	29 030 400
3	38 983 680
4	48 936 960
5	58 890 240
6	68 843 520

The measurement of the runtime of the computational algorithm of the Canny edge detector was performed in the MATLAB environment (using the "cputime" command). The parameters of the used computer were as follows: Intel(R) Core(TM) i5-7500 3.40GHz, 16 GB RAM 2400 MHz. The Canny edge detection durations are listed in the table 3.

Table 3 Duration of edge detection using a Canny edge detector

σ	Gaussian smoothing (ms)		Edge detection (ms)		Total (ms)	
	A	B	A	B	A	B
1	62.5	15.6	15.6		109.3	62.4
2	218.8	31.3			265.6	78.1
3	437.5	46.9			484.3	93.7
4	765.5	62.5			812.3	109.3
5	1156.3	78.1			1203.1	124.9
6	1703.1	93.8			1749.9	104.6

^{A} 2D normal convolution
^{B} 2D spatial separable convolution

Fig. 1-3 show images representing the steps in the Canny edge detector. As mentioned above, the first step of edge detection is smoothing the image using the Gaussian filter, which aims to remove noise and unnecessary backgrounds from the image. In this case, a compromise needs to be found between the computational complexity and the required output. Based on the results shown in Fig. 3, Fig. 3(b) can be considered as an acceptable output. Fig. 3(b) represents the smoothing of image using the Gaussian filter with the parameter σ=3, in this case it is necessary to use a matrix (2D Gaussian kernel) with a dimension of 19×19. In the case of a Gaussian filter with a 19×19 core, the number of required arithmetic operations is 38 983 680and the computation time is 46.9ms. The computation time 46.9 ms and the number of arithmetic operations 38 983 680 are valid under the assumption of using 2D spatial separable convolution. The second step in the presented edge detection algorithm is the calculation of gradient components $\mathbf{D}_a$ and $\mathbf{D}_b$ and absolute magnitude $\mathbf{D}$. In the second step, it is necessary to perform 8 294 400 arithmetic operations in the calculation of the gradient components, assuming the use of the 2D separable convolution. The absolute magnitude **D** of the images with different σ is shown in Fig. 2. The third step is the hysteresis threshold and is shown in Fig.3. The fourth step is a non-maximum suppression, which however was not performed in the case of Fig. 1 due to unfavorable results.

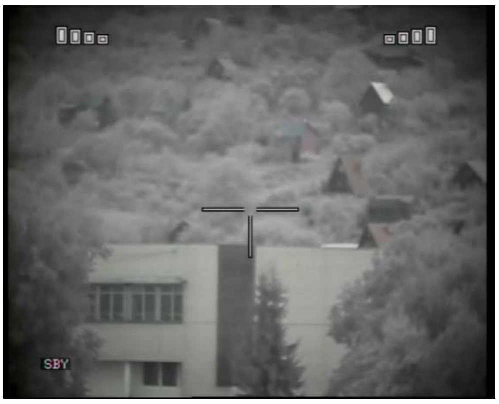

Fig. 1(a) Original image

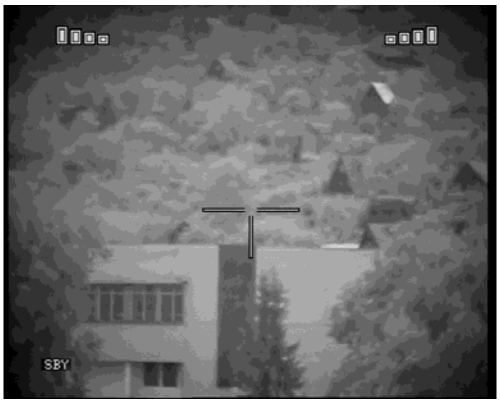

Fig. 1(b) Image converted to grayscale format

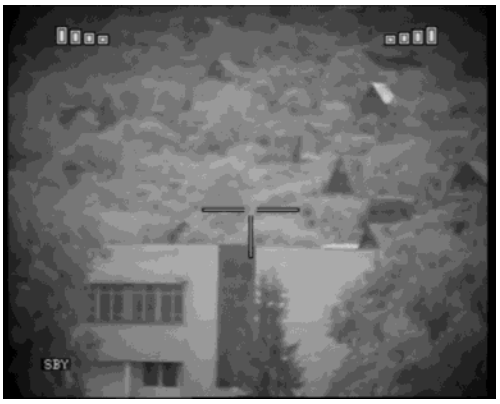

Fig. 1(c) Image smoothing using the Gaussian filter: σ=1

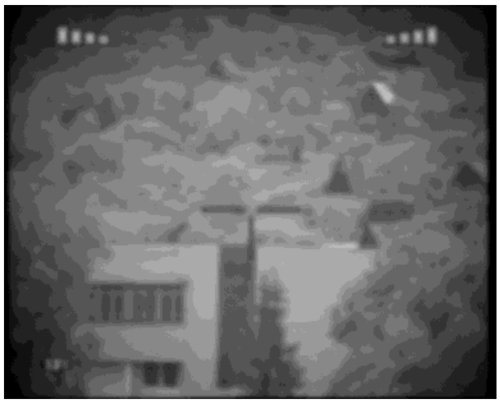

Fig. 1(d) Image smoothing using the Gaussian filter: σ=3

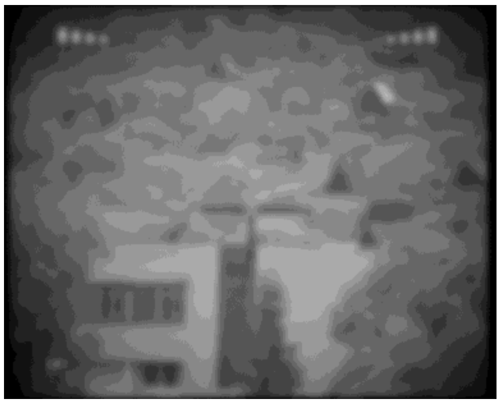

Fig. 1(e) Image smoothing using the Gaussian filter: σ=5

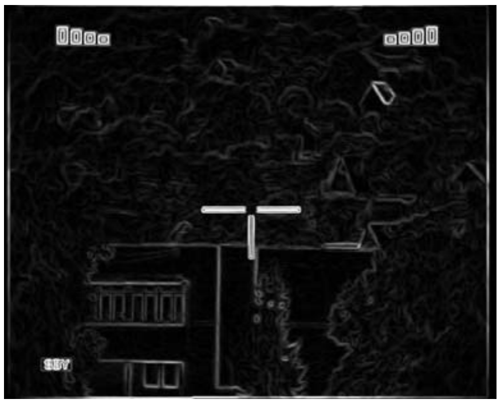

Fig. 2(a) Edge detection of smoothed image: σ=1

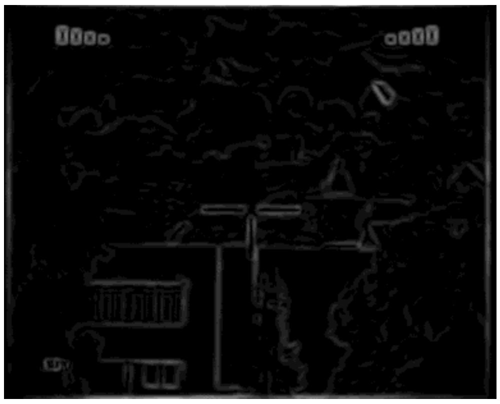

Fig. 2(b) Edge detection of smoothed image: σ=3

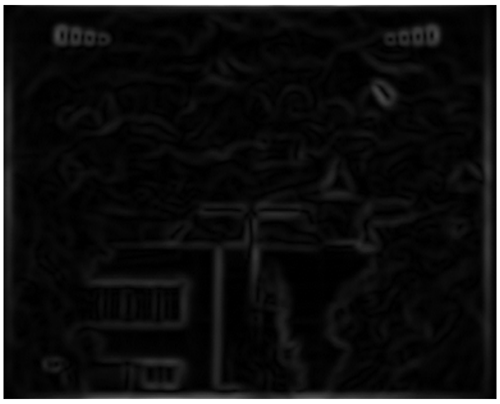

Fig. 2(c) Edge detection of smoothed image: σ=5

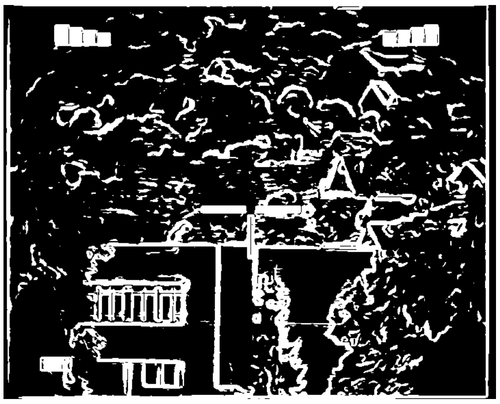

Fig. 3(a) Hysteresis thresholds after edge detection: σ=1

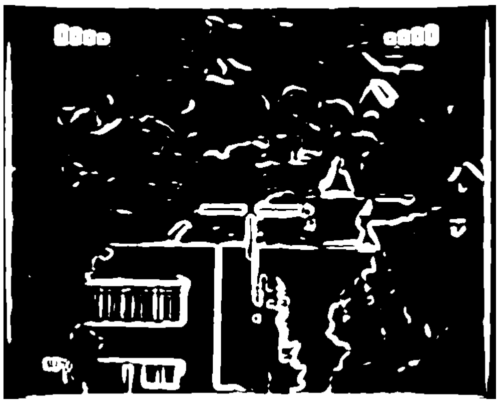

Fig. 3(b) Hysteresis thresholds after edge detection: σ=3

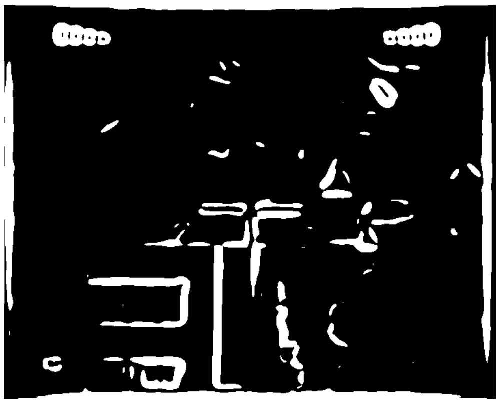

Fig. 3(c) Hysteresis thresholds after edge detection: σ=5

Fig. 4-7 show the detection of edges using the Canny edge detector as in the case of Fig. 1-3, but with the difference that the non-maximum suppression algorithm has been supplemented. The calculation of the non-maximum suppression was performed on the basis of the results given in [26,28]. Non-maximum suppression in the case of edge detection of Fig. 1(a) did not achieve satisfactory results. Fig. 5(a) was originally captured at a resolution of 3000×4000 pixels and subsequently converted to a resolution of 576×720 pixels. The Gaussian smoothing and gradient components calculation time was the same as in the case of Fig. 1(a). Sufficient background suppression with preservation of object contours was achieved in the case of Fig. 5(b) at the value σ=2. The time required to apply Gaussian smoothing (σ=2) was 31.3 ms, assuming the use of 2D spatial separable convolution.

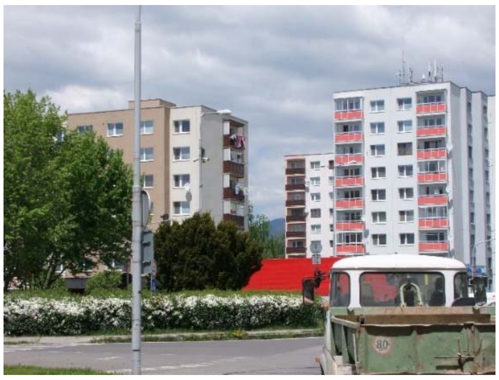

Fig. 4(a) Original image

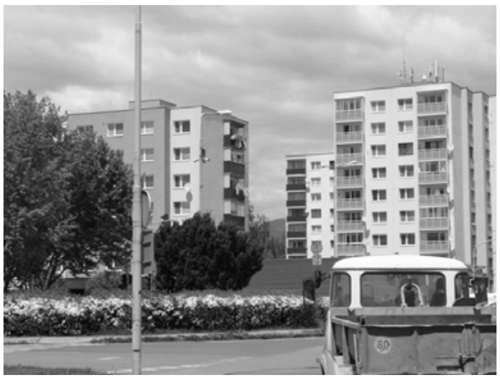

Fig. 4(b) Image converted to grayscale format

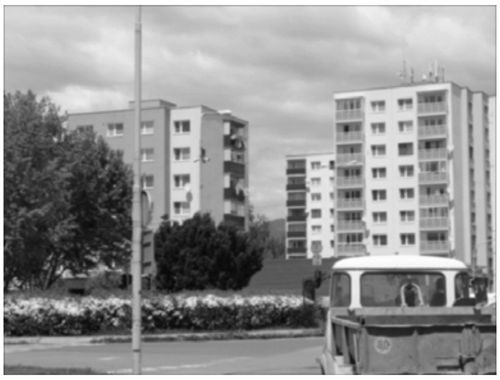

Fig. 4(c) Image smoothing using the Gaussian filter: σ=1

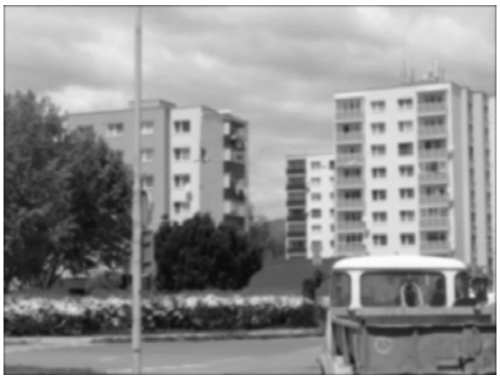

Fig. 4(d) Image smoothing using the Gaussian filter: σ=2

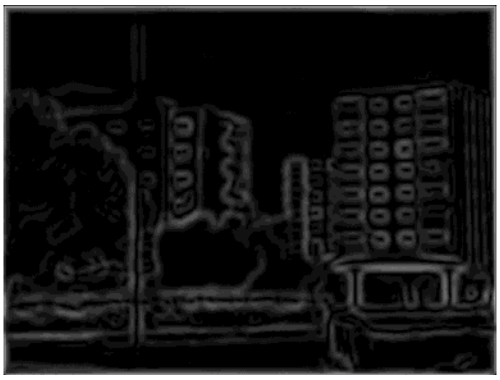

Fig. 5(c) Edge detection of smoothed image: σ=5

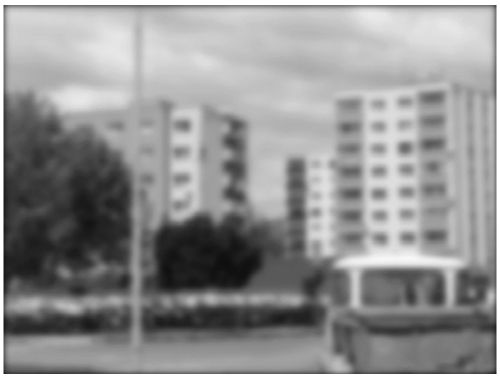

Fig. 4(e) Image smoothing using the Gaussian filter: σ=5

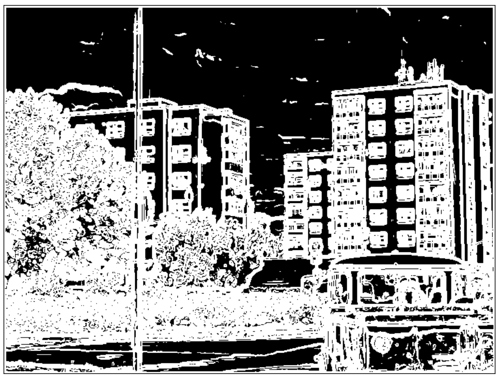

Fig. 6(a) Hysteresis thresholds after edge detection: σ=1

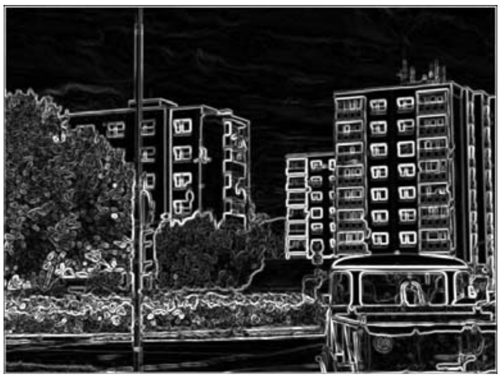

Fig. 5(a) Edge detection of smoothed image: σ=1

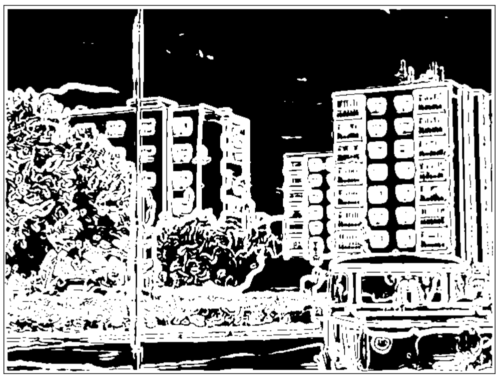

Fig. 6(b) Hysteresis thresholds after edge detection: σ=2

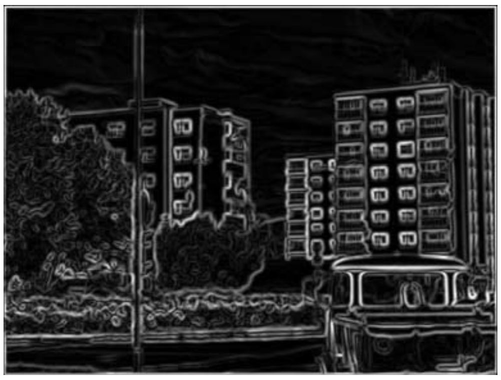

Fig. 5(b) Edge detection of smoothed image: σ=2

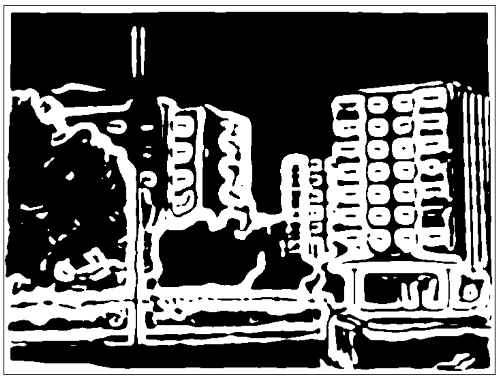

Fig. 6(c) Hysteresis thresholds after edge detection: σ=5

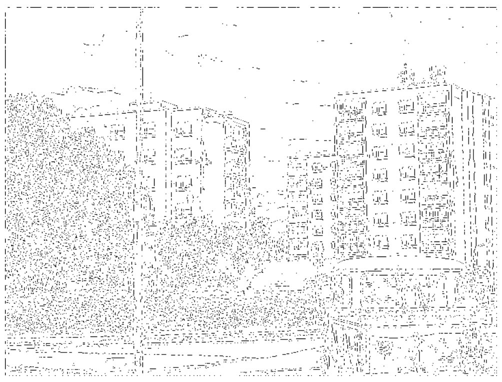

Fig. 7(a) Invert colors of image after the non-maximum suppression and hysteresis thresholds: σ=1

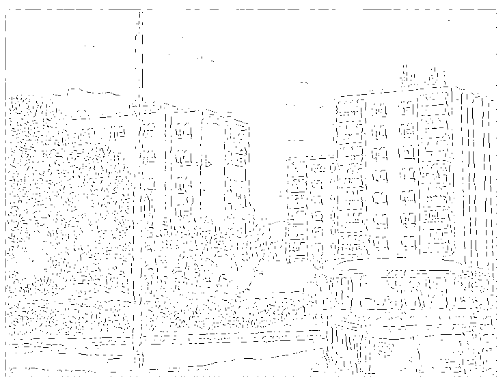

Fig. 7(b) Invert colors of image after the non-maximum suppression and hysteresis thresholds: σ=2

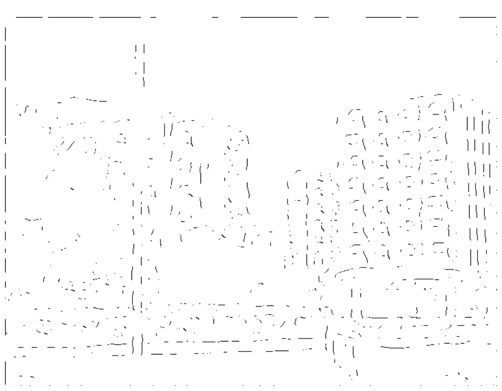

Fig. 7(c) Invert colors of image after the non-maximum suppression and hysteresis thresholds: σ=5

4. CONCLUSIONS

This work dealt with the analysis of the possibilities of optimization of computational complexity and related optimization of computation time of Canny edge detector. Since the Canny edge detection algorithm requires the use of several 2D convolutions, the optimization was focused on the computational complexity of the 2D convolution, as convolution is the most time-consuming operation. There are several ways to optimize the computational complexity of a 2D convolution, such as using a fast Fourier transform or using a 2D separable convolution. In this paper, the 2D spatial separable convolution method was analyzed. The most computationally complex part of the Canny edge detector was the first part performing Gaussian smoothing. Using the value σ = 3 in Gaussian smoothing, a saving of 268 323 840 arithmetic operations was achieved when using a 2D spatial separable convolution compared to using a normal 2D convolution. The time required to calculate the Gaussian smoothing algorithm was 390.6 ms less when using a separable convolution.

ACKNOWLEDGMENTS

This work was supported by the Slovak Research and Development Agency under the project APVV-19-0569.

REFERENCES

[1] BENHAMZA, K. – SERIDI, H.: "Canny edge detector improvement using an intelligent ants routing", Evolving Systems 12, pp. 397–406, 2021.

[2] SHRIVAKSHAN, G. T. – CHANDRASEKAR, C.: "A Comparison of various Edge Detection Techniques used in Image Processing", International Journal of Computer Science Issues, 2012.

[3] QIN, X.: "A modified Canny edge detector based on weighted least squares", Comput Stat 36, pp.641–659, 2021.

[4] AL-HAFIZ, F. – AL-MEGREN, S. – KURDI, H.: "Red blood cell segmentation by thresholding and Canny detector", Procedia Computer Science, vol. 141, pp. 327-334, 2018.

[5] KAZEMI, M. F. – MAZINAN, A. H.: "Neural network based CT-Canny edge detector considering watermarking framework", Evolving Systems, Springer, 2021.

[6] CHANDRASHEKAR, N. S. – NATRAJ, K. R.: "Detection of Lung Cancer by Canny Edge Detector for Performance in Area, Latency, ", 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEEECCOT), pp. 690-696, 2018.

[7] NIKOLIC, M. – TUBA, E. – TUBA, M.: "Edge detection in medical ultrasound images using adjusted Canny edge detection algorithm,", 24th Telecommunications Forum (TELFOR), pp. 1-4, 2016.

[8] ZHANG, Z. – CHEN, P. – SHI, X. – YANG, L.: "Text-Guided Neural Network Training for Image Recognition in Natural Scenes and Medicine", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 5, pp. 1733-1745, 2021.

[9] LAAROUSSI, S. – BAATAOUI, A. – HALLI, A. – KHALID, S.: "A dynamic mosaicking method for finding an optimal seamline with Canny edge detector, ", Procedia Computer Science, vol. 148, pp. 618-626, 2019.

[10] CHENGETA, K. – VIRIRI, S.: "Image Preprocessing Techniques for Facial Expression Recognition with Canny and Kirsch Edge Detectors, ", Computational Collective Intelligence. ICCCI 2019. Lecture Notes in Computer Science, vol 11684, 2019.

[11] GAOCHAO, W. – TSE, P. W. – YUAN, M.: "Automatic internal crack detection from a sequence of infrared images with triple-threshold Canny edge detector, ", Measurement Science and Technology, 2017.

[12] MANIKANDAN, L. C. – SELVAKUMAR, R. K. – NAIR, S. A. H., et al.: "Hardware implementation of fast biateral filter and canny edge detector using Raspberry Pi for telemedicine applications, ", Ambient Intell Human Comput 12, pp. 4689–4695, 2021.

[13] ALASSMI, N. S. – ZAGHLOUL, S. S.: "Speeding Up Canny Edge Detection Using Shared Memory Processing", International Journal of New Computer Architectures and their Applications. 7.pp. 68-76. 2017.

[14] PELLEGRINO, F. A. – VANZELLA, W. – TORRE, V.: "Edge Detection Revisited. IEEE transactions on systems, man, and cybernetics, ", Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics, 2004.

[15] MOGALE, H.: "High Performance Canny Edge Detector using Parallel Patterns for Scalability on Modern Multicore Processors, ", 2017.

[16] JIN LI, C. – QUA, Z. – YE WANG, S. – LIU, L.: "A method of cross-layer fusion multi-object detection and recognition based on improved faster R-CNN model in complex traffic environment", Pattern Recognition Letters, vol. 145, pp. 127-134, 2021.

[17] MATAS, J. – OBDRŽÁLEK, Š.: "Object recognition methods based on transformation covariant features", 2004 12th European Signal Processing Conference, pp. 1721-1728, 2004.

[18] XU, Q. – VARADARAJAN, S. – CHAKRABARTI, C. – KARAM, L. J.: "A Distributed Canny Edge Detector: Algorithm and FPGA Implementation, ", in IEEE Transactions on Image Processing, vol. 23, no. 7, pp. 2944-2960, 2014.

[19] GENTSOS, CH. – SOTIROPOULOU, C. L. – NIKOLAIDIS, S. – VASSILIADIS, N.: "Real-time canny edge detection parallel implementation for FPGAs, ". pp. 499-502., 2010.

[20] LEE, J. – TANG, H. – PARK, J.: "Energy Efficient Canny Edge Detector for Advanced Mobile Vision Applications,", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 4, pp. 1037-1046, 2018.

[21] LIN, J. – GUO, T. – YAN, Q. F. – WANG, W.: "Image segmentation by improved minimum spanning tree with fractional differential and Canny detector", Journal of Algorithms & Computational Technology, January 2019.

[22] YANG, Y. – ZHAO, X. – HUANG, M. – WANG, X. – ZHU, Q.: "Multispectral image based germination detection of potato by using supervised multiple threshold segmentation model and Canny edge detector, ", Computers and Electronics in Agriculture. 182.,2021.

[23] NIXON, M. S. – AGUADO, A. S.: "Basic image processing operations", Feature Extraction & Image Processing for Computer Vision (Third Edition), Editors M. S. Nixon and A. S. Aguado, ISBN 978-0-12-396549-3, pp. 83-136, 2012.

[24] LV, D. – PAN, S.: "Improved Canny edge detection algorithm based on deep learning", Scientific Journal of Intelligent Systems Research, vol. 3, no. 2, 2021.

[25] SHWETHA, V. – RENU MADHAVI, C. H.: "Design Techniques For Improvement Of Canny Edge Detection Algorithm", vol. 8, no. 8, 2020.

[26] MA, X. – LI, B. – ZHANG, Y. – YAN, M.: "The Canny Edge Detection and Its Improvement", Artificial Intelligence and Computational Intelligence, 4th International Conference, vol. 7530, pp. 50-58, 2012.

[27] JIANG, X. J. – SCOTT, P. J.: "Characterization of free-form structured surfaces", Advanced Metrology, Editors X. J. Jiang and P. J. Scott, ISBN 978-0-12-821815-0, pp. 281-317, 2020.

[28] LI, B. – SÖDERSTRÖM, U. – RÉHMAN, S. U. – LI, H.: "Restricted Hysteresis Reduce Redundancy in Edge Detection", Journal of Signal and Information Processing, vol. 4, no. 3B, pp. 158-163, 2013.

Received November 24, 2021, accepted December 16, 2021

BIOGRAPHIES

Martin Králik is absolvent of University of Žilina, Faculty of Electrical Engineering and Information Technology. Since September 2020 university teacher. The field of research is electrotechnology and materials, specifically the analysis of optical and structural properties of porous silicon. Additional area of research is numerical implementation and optimization of computational algorithms. 2012-2015 (bachelor's degree) Digital Technology. 2015-2017 (engineer's degree) Telecommunications. 2017-2020 (PhD study) Electrotechnology and materials.

Libor Ladányi received his M.Sc. in 2011and his Ph.D. in telecommunications in 2014 both from the Department of Telecommunications and Multimedia, the Faculty of Electrical Engineering, the University of Žilina. His research interests include optical communication networks and systems. He is also involved in research with external companies, currently cooperating on a project focused at detecting and recognizing objects from images and video. 2006-2009 (bachelor's degree) Digital Technology. 2009-2011 (engineer's degree) Radiocommunication and Telecommunication ingeneering. 2011-2014 (PhD study) Telecommunications.