Journal of Machine Engineering, 2020, Vol. 20, No. 4, 74–85
ISSN 1895-7595 (Print) ISSN 2391-8071 (Online)

Received: *19 October 2020* / Accepted: *25 November 2020* / Published online: *29 November 2020*

edge detection, Cortex microcontroller, Canny detector,
Marr-Hildreth edge detector, fuzzy logic detector,
cost function detector

Grzegorz BUDZYN[1*]
Janusz RZEPKA[1]

REVIEW OF EDGE DETECTION ALGORITHMS FOR APPLICATION IN MINIATURE DIMENSION MEASUREMENT MODULES

Continuous improvement of semiconductor chip technology allows more and more complex functions to be performed by an increasing number of modules with limited space and power. The deep miniaturisation and cost reduction of such modules has a positive impact on their application areas. They can be used, for example, inside machine tools for dimensional inspection of workpieces or deformation of the tool tip. Modules of this type must contain as many standard components as possible and, in particular, a processing unit responsible, among other things, for processing the optical sensor signal. For this reason, this study reviews and compares algorithms for object edge detection useful in embedded systems based on standard single chips, cores based on Cortex-M4F. The complexity of processing and the quality of algorithm output data was analysed. The results of experimental studies are presented. It was found that the required processing times significantly reduce the use of single chip embedded systems only for edge detection on small images.

1. INTRODUCTION

The functionality of the camera modules increases with every improvement in semiconductor technology. Such modules are currently used e.g. in many fields of robotics and mechanics [1], medical image analysis [2], geographical sciences [3], measurements or even sport [4]. The main applications in mechanics include tool deformation monitoring [5], tool wear monitoring [6] and measurements of small objects. For these applications the most difficult and demanding is the analysis of the signal from the camera sensor. Usually an important part of this analysis is an edge detection algorithm. This technique allows to find boundaries of different objects in the image [7]. Thanks to this, an object can be recognized, classified or measured. In general, the available algorithms are computing intensive. FPGA, ASSP or ASICs are usually used for embedded applications [8]. Nowadays, edge detection is becoming so popular mainly due to easy access to cheap and efficient 32-bit microcontrollers capable of efficient signal processing.

[1] Faculty of Electronics, Wroclaw University of Science and Technology, Wroclaw, Poland
[*] E-mail: grzegorz.budzyn@pwr.edu.pl
https://doi.org/10.36897/jme/130876

Due to the complexity of the topic, many techniques have been implemented to develop an optimal edge detection algorithm [9–13]. These techniques use filters to recognise edges as a sudden change in grey scale pixel intensity. Most of the edge detection operators are gradient operator, Laplacian operator or Laplacian-of-Gaussian operator. Generally, in these algorithms, edge pixels are defined as those where the first-order derivative of the pixel intensity value exceeds a certain threshold or the second-order derivative of the pixel intensity value passes through zero. These methods, although simple and not requiring intensive calculations, have disadvantages such as susceptibility to noise causing edge fragmentation or a limited number of types of detected edges. A number of algorithms solving these problems can be found in the literature, based on optimal filtration [14], residual analysis [15] fuzzy logic [16] and anisotropic diffusion [17]. Unfortunately, most of them are based on the assumption that the edges in the image are edges of gradual intensity, which is not true for many profiles in real applications. For this reason a class of cost function algorithms was developed and described [14, 18].

The article reviews and compares selected edge detection algorithms suitable for use in 32-bit single chip microcontrollers, i.e. Canny, Marr-Hildreth and Fuzzy Logic. For comparison, a mathematical algorithm using a genetic cost minimization algorithm for edge detection has also been implemented.

The tests described in this article were carried out on a fairly simple and cheap Cortex-M4F core based on a single chip microcontroller clocked at 100 MHz. The core consists of a single-precision floating point FPU used in the tests. The calculations were made on the example image shown in Fig. 1 with the target application of the object measurement. Depending on the algorithm tested, two versions of the image were analysed – with and without backlighting. The analyzed algorithms were written in ANSI C and compiled using the same compiler and linker settings.

2. CANNY EDGE DETECTORS

Canny detector first described by J. Canny [9] is now commonly used edge detection algorithm. For the proper operation it was decided to implement procedures given in [10]. First the image was blurred to filter out noises. For this operation a Gaussian blur kernel was generated with the use of formula

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{m^2+n^2}{2\sigma^2}) \tag{1}$$

Where: m, n are ordinal numbers and σ is a parameter.

In the next step the blurred image gradient was calculated. For this purpose the image was convolved with the Prewitt masks [19] with the use of formula

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j]h[m-i,n-j] \tag{2}$$

The Prewitt kernels for horizontal G_x and vertical directions G_y were defined as

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \; G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3}$$

Calculated gradients were combined into one matrix with the formula

$$|G| = \sqrt{G_x^2 + G_y^2} \, . \tag{4}$$

For every pixel of the image an angle of gradient was checked. The angle was calculated as the arctangent of G_y to G_x ratio. Pixel value was compared with its two neighbours along the proper direction. If the pixel didn't have the highest value, then its value was set to 0.

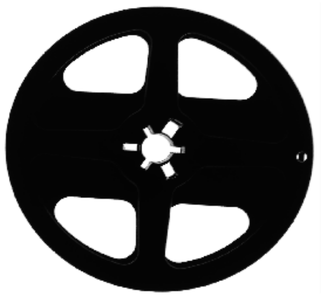


Fig. 1. The original image in greyscale. For the calculations the image resolution was scaled down to 30×30, 60×60, 90×90, 120×120 or 150×150 pixels

In the last step of the procedure every pixel of the image was compared with low and high thresholds (set as parameters). If the pixel value was lower than the low threshold, then the pixel was set to "Non Edge". For the pixel value between the low and the high threshold, the pixel was set to "Weak Edge". In the other cases the pixels were classified as "Strong Edge". In the last step, "Weak Edges" with connection with "Strong Edges" were classified as "Strong Edges". The others were set as "Non Edge".

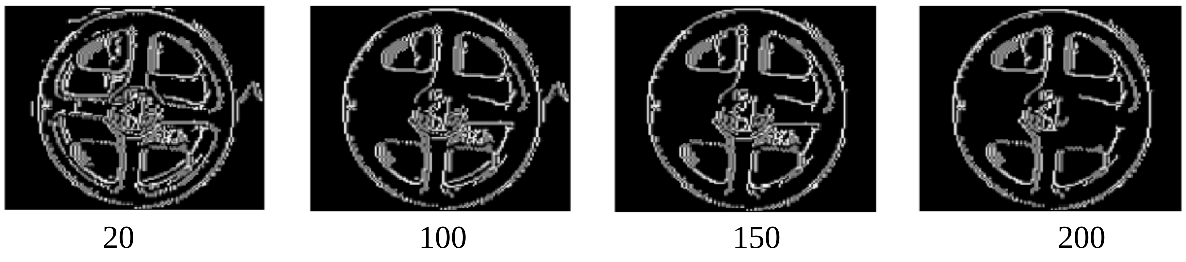


Fig. 2. Canny detector results for following parameters: no backlight, low threshold = 10, high threshold = 20; 100; 150; 200

The original picture is shown in Fig. 1. In Fig. 2 there are shown results of the edge detection with different high threshold values – from 20 to 200. The higher value of the threshold was set, the cleaner picture of the edges was obtained yet still the quality was not good. A significant improvement was observed when lower threshold was modified. It is shown in Fig. 3. Also the presence of white backlight helps improving the amount of visible details – see Fig. 4.

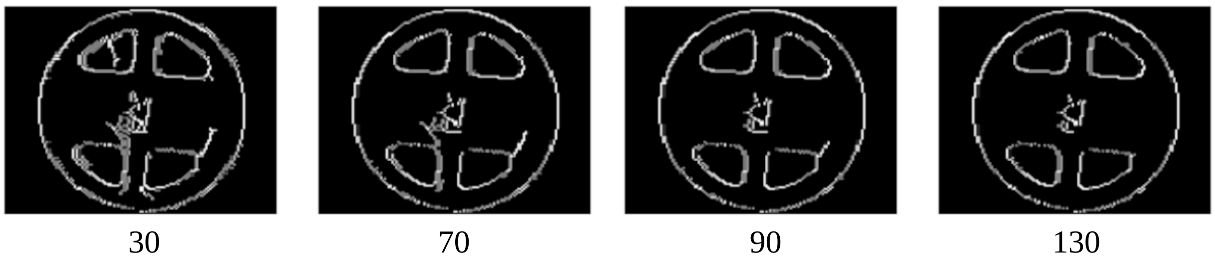


Fig. 3. Canny detector results for following parameters: no backlight, low threshold = 30; 70; 90; 130, high threshold = 200

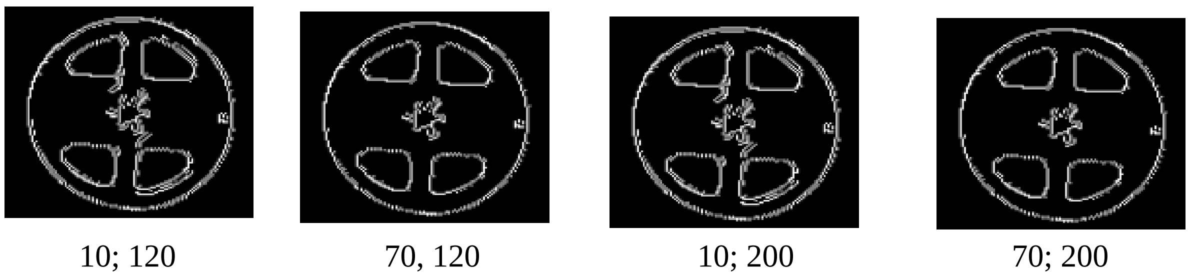


Fig. 4. Canny detector results for following parameters: white backlight, no blurring, low threshold = 10; 70; high threshold = 120,200

The algorithm was also tested for speed of calculations on the Cortex-M4F based single chip microcontroller. The chart of the dependence of processing time on the image size with the FPU usage as parameter is shown in Fig. 5. For very small input images the processing time keeps below the useful value of 2 seconds. For larger, but still quite small images the processing time rises beyond 3 seconds.

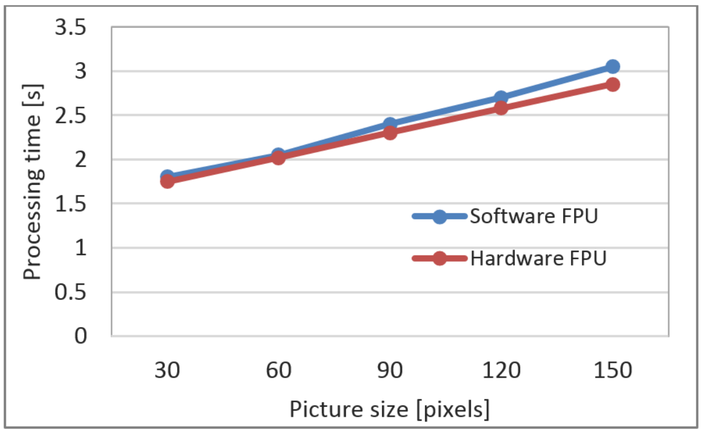


Fig. 5. Canny Edge Detector: time of calculations vs size of picture with and without FPU

As shown in Fig. 5, the Canny Edge Detector gives accurate, narrow edges and is quite fast for small input images. There is no big difference in calculation time when comparing work with and without FPU – it is only 20 ms difference for a 150×150 pixel image. The algorithm can be tuned with threshold parameters to avoid detection of false edges.

3. MARR-HILDRETH EDGE DETECTOR

In the Marr-Hildreth MH algorithm [20] implementation the image was first convolved with the Laplacian of Gaussian function known for its shape as Mexican hat wavelet. The LoG kernel was calculated with the formula

$$\nabla g(x,y) = \frac{-1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right) exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \tag{5}$$

Afterwards the processed image was filtered and checked for zero crossing between neighbouring pixels. If there was a zero crossing and the difference was higher than defined threshold then the pixel was classified as the edge.

As it is shown in Fig. 6 and 7 the MH detector was not functioning properly without the backlight. Only after applying the light source behind the scene, the edges of the object were properly detected. Also the proper setting of the settings was important.

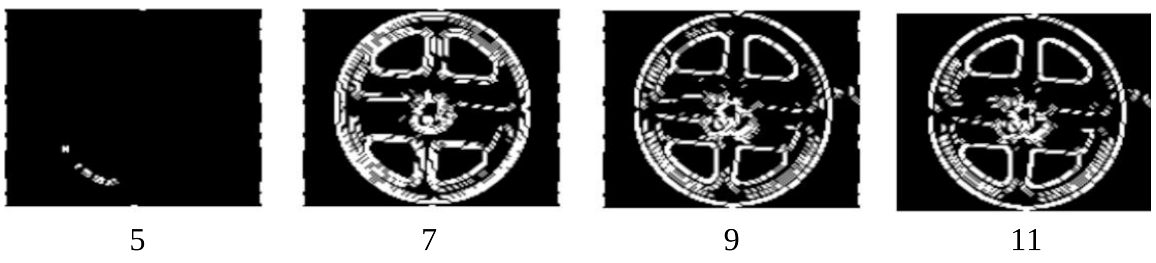


Fig. 6. MH detector results for following parameters: no backlight, LoG size = 5;7;9;11, σ = 1.4, threshold = 0

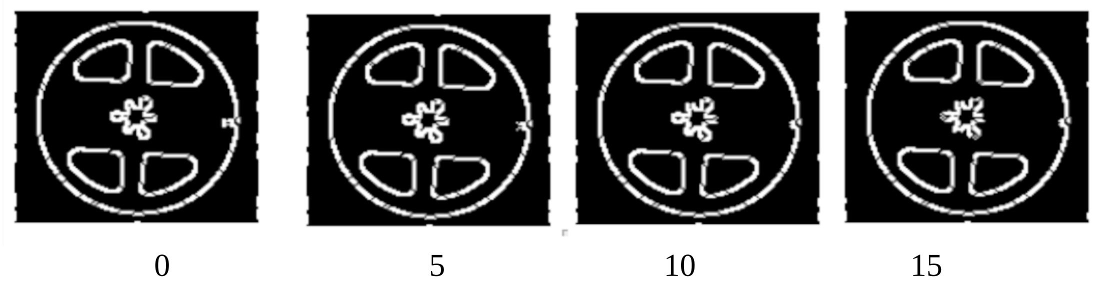


Fig. 7. MH detector results for following parameters: white backlight, LoG size = 9, σ = 1.5, threshold = 0; 5; 10; 15

Figure 8 shows the measured computational performance of the MH algorithm with and without a hardware FPU. Compared to Canny's detector, MH was faster with the hardware FPU but slower with programmed floating point calculations. Also the Marr-Hildreth algorithm did not give as accurate and narrow edges as the Canny detector.

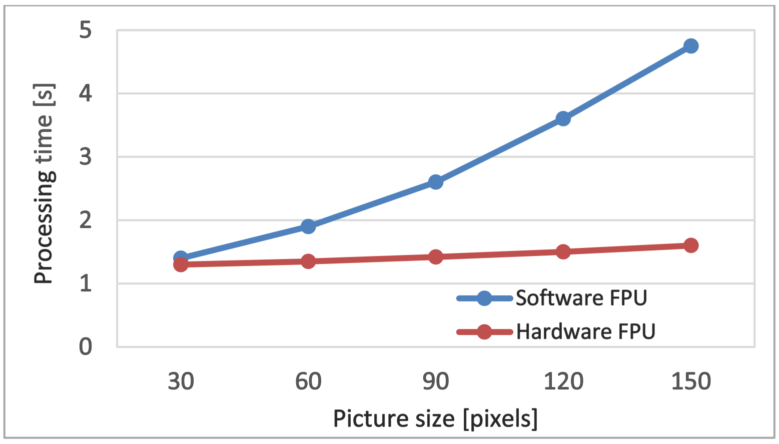


Fig. 8. MH: time of calculations vs size of picture with and without FPU

4. FUZZY LOGIC EDGE DETECTOR

The third type of detector implemented was based on the so-called fuzzy sets theory introduced by Lofti A in 1965. Zadeh [16]. This method performs mathematical and logical reasoning based on approximations and not on sharp values. This technique significantly reduces the complexity of problems where constant values cannot be reached or predicted. Based on [21] the input image is first blurred and then the sets are checked using coded rules and a sharp output is generated on this basis.

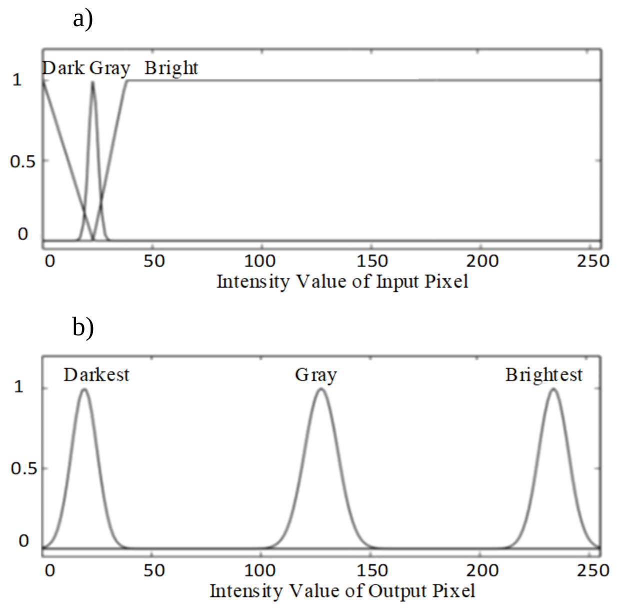


Fig. 9. Contrast adjustment: input (a) and output (b) membership functions [21]

In the first stage of this algorithm, the image contrast has been enhanced based on a fuzzy logic approach. Figure 9 and 10 shows the affiliation function for assigning output pixel intensity values to the input pixel value. The affiliation function is used for edge detection. In this algorithm, the only adjustable parameter was the threshold above which the pixel was classified as a slope.

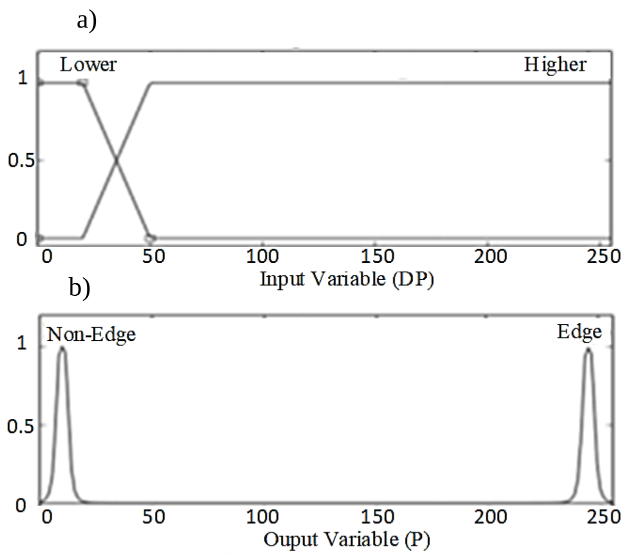
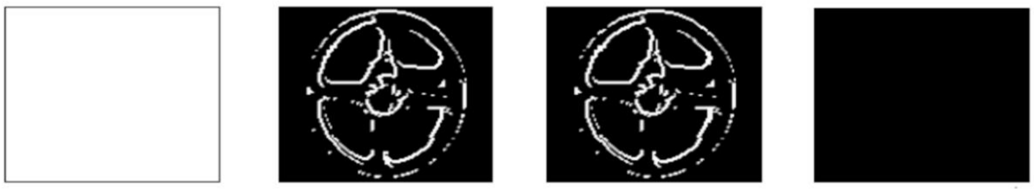
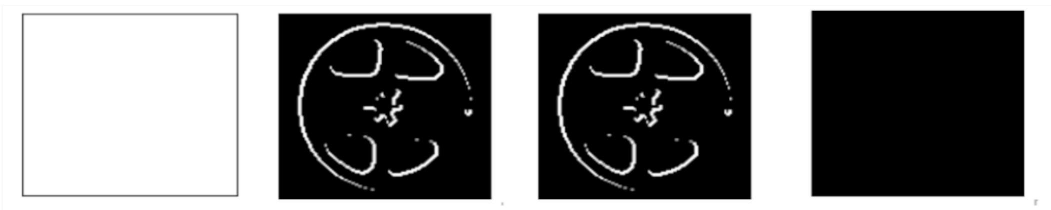


Fig. 10. Fuzzy Logic Edge detection: input (a) and output (b) membership functions [21]



5 90 170 250

Fig. 11. Fuzzy logic approach results for threshold = 5; 90; 170; 250 without backlight



5 90 170 250

Fig. 12. Fuzzy logic approach results for threshold = 5; 90; 170; 250 with backlight

The results obtained using the fuzzy logic method are shown in Fig. 11 and 12. The quality of detection was strongly dependent on the threshold value, but it was not possible to obtain such good results as in the case of other methods. The image processing time was also much longer than in the previously described methods. The advantage of this detector was the correct detection of edges in images of different contrast without reconfiguration of parameters.

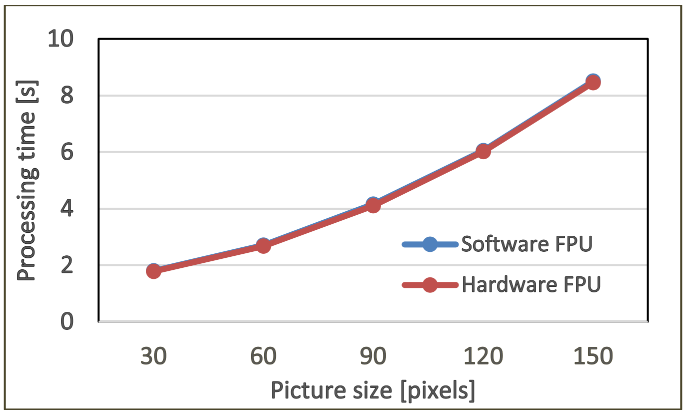


Fig. 13. Fuzzy Logic approach: time of calculations vs size of picture with and without FPU

5. COST FUNCTION EDGE DETECTOR

The algorithm of edge detection as a function of cost is not deterministic, in contrast to the above described, i.e. the same input may have a different output. It has been implemented based on [22]. In this method, not only the punctual presence of pixels in the edge is taken into account, but also the structure of adjacent pixels. The algorithm is able to correctly detect even fragmented or detached edges. Another important advantage of this method is that it does not assume a predetermined edge concept in terms of differences between regions on both sides of the edge, so it is possible to detect different types of edges. The cost minimisation function uses information from image data, but also from the local edge structure. The main problem with this approach is the very large number of possible solutions equal to 2^P, where P is the number of pixels in the image.

The Cost Function algorithm is divided into several steps. First, the differences between the image regions are detected and amplified with a special function to measure these differences. Then five cost function elements (factors) such as the cost of curvature, the cost of region dissimilarity, the cost of number of edge points, the cost of fragmentation and the cost of edge thickness are defined. In the next step, the cost function, i.e. a linear combination of cost factors, is minimised using a simulated annealing algorithm [23]. Simulated annealing is a stochastic optimisation technique derived from Monte Carlo methods in statistical mechanics.

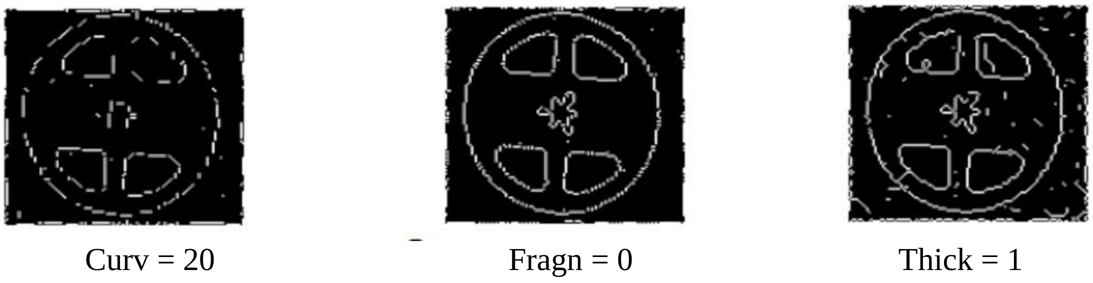


Fig. 14. Cost function approach: different weights of cost factors. Edge curvature weight set to 10 (left), fragmentation weight set to 0 (central), edge fragmentation weight set to 0 (right). Final temperature set to 0.2 degrees

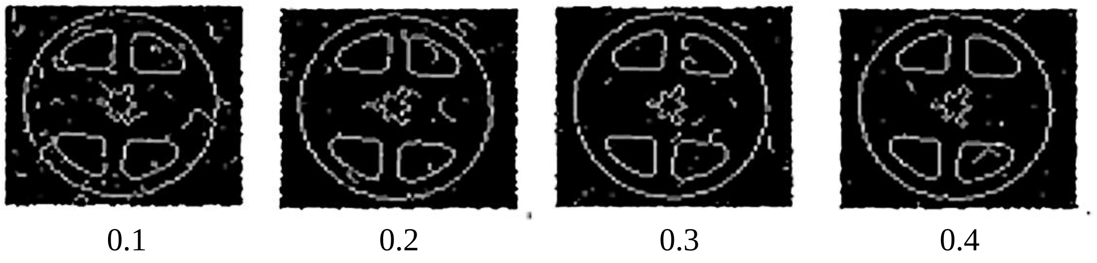


Fig. 15. Cost function approach results – influence of initial temperature on the simulated annealing algorithm. Temperature values from the left respectively: 0.1, 0.2, 0.3, 0.4.

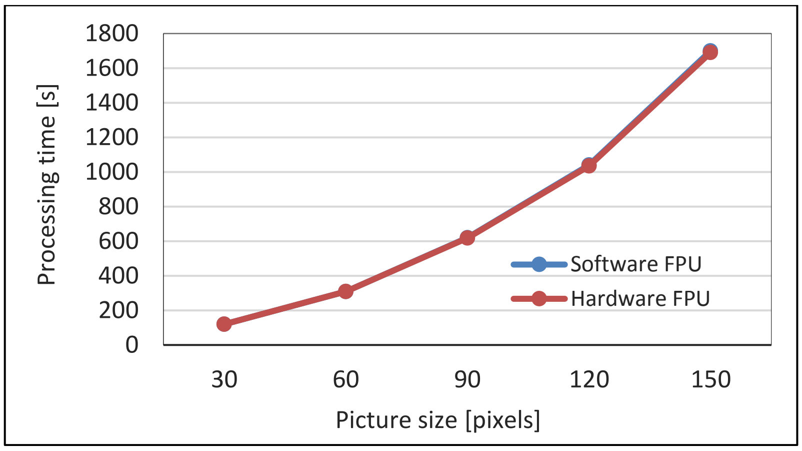


Fig. 16. Cost function approach results – time of calculations vs size of picture with and without FPU

The results obtained using a cost-function approach are shown in Figs 14 and 15. The temperature value is a parameter. It possible to obtain high quality of the edge, but the detector is incomparably slower than others, due to the huge number of operations performed by the algorithm (Fig. 16). The advantage of this detector is that the user can easily configure edge properties. The calculation time with and without FPU is very similar, but since the algorithm is not deterministic, the calculation time for the same input image may vary.

6. COMPARISON OF ALGORITHMS

The performance of the algorithms was additionally compared in the application of object size measurements. On the original object shown in Fig. 1, three dimensions were selected, as shown in Fig. 17. After applying the edge detection mechanisms, all three dimensions were read as the distance between the detected edges, internal and external. As the line thicknesses rarely had a single pixel width, the final dimension was therefore the average of these two values.

The results for all detectors are aggregated in Table 1. The error presented is the sum of the differences between the real size of the object measured by the calliper (Real column in Table 1) and the average of the various digital measurements made on the detected edge images. The source of the images were 5 different photos of the same object. These photos were taken with the same camera in similar conditions.

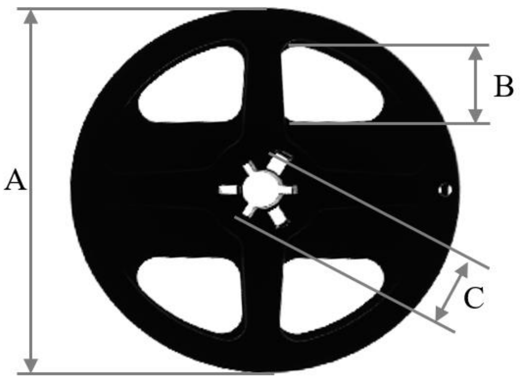


Fig. 17. Parts of the original image selected for measurements. Three openings A, B and C were chosen for comparison

Tab. 1. Differences $\boldsymbol{\varepsilon}$ with standard deviations $\boldsymbol{\sigma}$ between mean values of measured dimensions A, B and C and the real value measured with a calliper for each algorithm. Index denotes the algorithm: ca – Canny, mh – Marr-Hildreth, fu – Fuzzy Logic and co – Cost Function

	Real [mm]	ε_{ca} [mm]	σ_{ca} [mm]	ε_{mh} [mm]	σ_{mh} [mm]	ε_{fu} [mm]	σ_{fu} [mm]	ε_{co} [mm]	σ_{co} [mm]
A	128.0	–3.5	1.6	–3.6	1.5	–4.2	2.0	3.2	1.4
B	25.1	1.8	1.5	3.3	1.4	2.5	1.2	1.1	0.2
C	27.0	–0.5	0.4	0.6	0.3	–0.5	0.5	1.0	0.3

For straight comparison of the edge detection comparison we used a Summary Error measure defined as

$$SE_{xx} = |A_R - A_{xx}| + |B_R - B_{xx}| + |C_R - C_{xx}| \tag{6}$$

where index *R* denotes real values and index *xx* the indexes of the algorithm like in Table 1. The calculated summary errors are shown in Fig. 18.

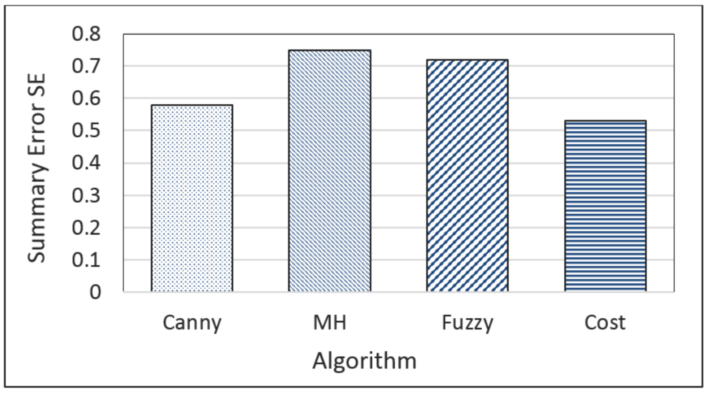


Fig. 18. Comparison of the performance of the reviewed algorithm in the object size measurement

The cost function detector seems to be the best solution for the assessment method used. Considering the speed of execution, the most optimal choice would be Canny's method. The biggest error is generated by the MH algorithm, probably because of the thick edges. It should be stressed that the Fuzzy Logic Approach is the only algorithm that has failed to detect all edges of an object.

7. CONCLUSIONS

Edge detection tests for microcontrollers based on Cortex-M4F were carried out on 4 different algorithms. The first two (Canny and Marr-Hildreth) work best using a hardware floating point unit, while the other two can be used for microcontrollers without FPU. The quality of edge detection in the fuzzy logic approach was weakest. Other methods, with optimal parameter setting, gave similar output results. The difference was in processing time- Cost Function approach was much slower than Canny and MH methods. For the 32-bit Cortex-M4F, Canny and MH edge detectors seem to be the optimal choice. The first one is slower but gives an image with thinner edges, which allows for more accurate measurements of the detected objects.

The tests carried out prove that the selected 32-bit microcontroller core can be used in various camera-based modules, e.g. to detect object dimensions or shapes. The quality of edge detection can be high, but on rather low resolution images. In the case of high-resolution cameras, a much larger processor is required.

REFERENCES

[1] HASKAH J., PIES M., MACHACEK Z., 2014, *Image Signal Processing, Analysis and Detection for Robotic System,* 14th International Conference on Control, Automation and Systems (ICC AS 2014) Oct. 22–25, 2014 in KINTEX, Oyeonggi-do, Korea.

[2] RULANINGTYAS R., AIN K., 2009, *Edge Detection for Brain Tumor Pattern Recognition. Instrumentation, Communications*, Information Technology and Biomedical Engineering(ICICI-BME), International Conferenceon; Nov 23–25; Bandung, Indonesia. IEEE. 1–3.

[3] LIN H., DU P.J., ZHAO C.S., SHU N., 2004, *Edge Detection Method of Remote Sensing Images Based on Mathematical Morphology of Multi-Structure Elements*, Chinese Geographical Science, 14/3, 263–268.

[4] YANG W., 2019, *Analysis of Sports Image Detection Technology Based on Machine Learning*, EURASIP Journal on Image and Video Processing, 17.

[5] WANG K., 2020, *Tool Thermal Deformation Measurement Research Based on Image Processing Technology*, Proceedings 3rd International Conference on Electron Device and Mechanical Engineering, ICEDME, May, 714–717.

[6] ZHICHAO Y., 2020, *On-Line Milling Cutter Wear Monitoring in a Wide Field-of-View Camera*, Wear, 460–461, 203479.

[7] DUAN L., YAO M., ZHANG H., 2019, *The Research and Design of a High Efficient Ball Image Recognition System Based on Microcontroller,* IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC.

[8] SATO T., KRUPA G., ZOCCAL L., *Performance Measurements of Sobel Edge Detection Implemented in an FPGA Architecture and Software*, Proceedings of the 4th Brazilian Technology Symposium (BTSym'18), Smart Innovation, Systems and Technologies, 140.

[9] CANNY J., 1986, *A Computational Approach to Edge Detection. Pattern Analysis and Machine Intelligence,* IEEE, 8/6 679–698.

[10] ZHANG J., CHEN Y., HUANG X., 2009, *Edge Detection of Images Based on Improved Sobel Operator and Genetic Algorithms*, Image Analysis and Signal Processing, IASP International, April 11–12; Taizhou, China. IEEE, 31–35.

[11] ROSENFELD A., 1981, *The Max Roberts Operator is a Hueckel-Type Edge Detector. Pattern Analysis and Machine Intelligence*, IEEE, 3/1 101–103.

[12] LEI Y., DEWEI Z., XIAOYU W., HUI L., JUN Z., 2011, *An Improved Prewitt Algorithm for Edge Detection Based on Noised Image*, Image and Signal Processing (CISP), 4th International Congress on, Oct. 15–17; Shanghai, China. IEEE, 1197–1200.

[13] Ulupinar F., Medioni G., 1990, *Refining edges detected by a LoG operator*, Computer Vision, Graphics and Image Processing, 51/3, 275–298.

[14] TORTE V., POGGIO T.A., 1986, *On Edge Detection*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8, 147–163.

[15] CHEN M.H., LEE D., PAVLIDIS T., 1991, *Residual Analysis for Feature Detection*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-13, 30–40.

[16] ZADEH., L.A., 1984, *Fuzzy Probabilities, Information Processing & Management*, 20/3, 363–372.

[17] PERONA P., MALIK J., 1990, *Scale Space and Edge Detection Using Anisotropic Diffusion*, IEEE Trans. Pattern Anal. Mach. Intell. PAMi-12, 6296–39.

[18] SHEN J., CASTAN S., 1992, *An Optimal Linear Operator for Step Edge Detection*, CVGIP: Graph. Models Image Process, 54, 112–133.

[19] PREWITT, J.M.S., 1970, *Object Enhancement and Extraction,* Picture processing and Psychopictorics. Academic Press.

[20] MARR D., HILDRETH E., 1980, *Theory of Edge Detection*, Proceedings of the Royal Society of London. Series B, Biological Sciences, 207 (1167), 187–217

[21] HAQ I., ANWAR S., SHAH K., KHAN M.T., SHAH S.A., 2015, *Fuzzy Logic Based Edge Detection in Smooth and Noisy Clinical Images*, PLoS ONE 10/9, e0138712.

[22] BHANDARKAR S.M., ZHANG Y., Potter W.D., 1994, *An Edge Detection Technique Using Genetic Algorithm - Based Optimization*, Pattern Recognition, 27/9, 1159–1180.

[23] KIRKPATRICK S., GELATT C.D.Jr, VECCHI M.P.Jr, 1983, *Optimization by Simulated Anneal*ing, Science, 220, 671–680.