

## Detection of the Road Pothole Contour in Raster Images

Vytautas Jakštys<sup>1</sup>, Virginijus Marcinkevičius<sup>1</sup>, Jevgenij Tichonov<sup>1</sup>,  
Povilas Treigys<sup>1,2</sup>

<sup>1</sup> Vilnius University, Institute of Mathematics and Informatics,  
Akademijos st. 4, LT-08663 Vilnius, Lithuania,  
e-mail: vytautas.jakstys@mii.vu.lt, virginijus.marcinkevicius@mii.vu.lt, jevgenij.tichonov@mii.vu.lt

<sup>2</sup> Vilnius Gediminas Technical University,  
Saulėtekio al. 11, LT-10223 Vilnius, Lithuania,  
e-mail: povilas.treigys@mii.vu.lt

**crossref** <http://dx.doi.org/10.5755/j01.itc.45.3.13446>

**Abstract.** The article analyses the issues related to detection and contour approximation of road surface defects, in particular, road potholes captured by a smart device camera. The assessment of measurements, obtained after image analysis and real objects, will be disclosed with a view to depict the measurement bias. The difference from all the other methods, mentioned in the literature, is that only 2D images captured by one camera were used in this investigation with a view to identify the contour of road potholes and get its parameterised representation. In practice, according to other authors, the most common approach to obtain a parameterised description of a pothole incorporates not only cameras, but also other various sensors such as accelerometer, global positioning systems, laser, hyperspectral imagery, infrared or ultrasonic sensors. In this article, we present a method that allows recognising a pothole object in terms of its colour, shape, and structure. The method discussed was applied to real world images to detect and outline the road pothole contour. Finally, the evaluation of approximation accuracy by empirical research techniques has been accomplished.

**Keywords:** pothole; pothole contour detection; pothole segmentation; shortest path analysis.

### 1. Introduction

The increasing level of traffic increases the need for road repair services accordingly. One of the causes of road damage is freezing liquids during winter periods. In such a case, it is very important to detect any impairments as soon as possible in order to avoid incidents. Often, it is not feasible to assess the condition of a road as there are no tools or devices to measure the damages. On the other hand, it is crucial for the road engineers to observe the dynamics of road conditions and evaluate the scope of the work needed to repair the road. The literature analysis has revealed that there exist approaches to solve such a problem. One of them is to use smartphones with accelerometers [1] and to detect the current location with a global positioning system [2]. Another approach is to use pressure sensors that can be built in shock absorbers to detect and quantify the intensity of a pothole [3]. Potholes are detected by using two ultrasonic sensors [4], laser line striper sensors [5], or using hyperspectral imagery [6]. One of the most convenient, well-known and inexpensive ways to

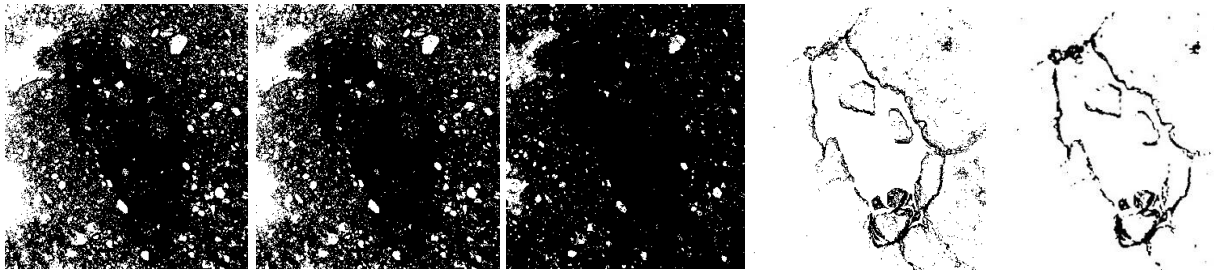
recognize road potholes is to use image analysis algorithms that analyse images obtained by a smart device [7,8,9].

At present, mobile devices became an inseparable attribute of our life. Their technical capabilities allow taking high quality photographs of objects, moreover, mobile networks have a possibility to provide a high data transfer rate, so it is rather practical to create methods capable of measuring parameters of road potholes and upload the collected data for public/-personal use of the road companies.

We analyse issues related to detection and evaluation of road surface defects that were captured by a smart device as well as assessment of measurements of particular road pothole objects (e.g. Fig. 1). In this article, we present a method that automatically detects and measures the size of extracted defect contours. The bias of the proposed method is evaluated by comparing differences between contours obtained applying the method and that drawn by an expert (Fig. 1).



**Figure 1.** Detection of a pothole contour: From left to right: original image, human-drawn contour, contour of the proposed method



**Figure 2.** Application of different threshold algorithms. From left to right: Huang, Otsu, Minimum, Triangle, Adaptive

The task of pothole contour detection is very difficult because of several factors. Impairments of the road surface have a very complex structure since there can be water, snow or other substances of various shades inside the pothole. The edge of a pothole is also not easily recognized as it does not have an even shape, asphalt is not smooth and monolithic, and in many cases it is fragmented into pieces. Even for a human it is challenging to recognize a boundary of a pothole, because it is a matter of subject, and cannot be drawn unambiguously. Moreover, a pothole does not have a predefined colour since it can have various shades, for example, when the surface is wet or dry. The interior of a pothole can be darker than the exterior and vice versa.

We present a method that does not depend on the smart device used, the size of the taken image, or the presence of non-uniform illumination and discuss the evaluation of its performance.

## 2. Image thresholding methods

In order to analyse pothole image the segmentation of foreground and background must be implemented [10, 11]. Frequently such a segmentation is done by thresholding the image, i.e. the image is converted to a binary image where white colour represents the foreground and the black colour corresponds to background structures, or vice versa. There are many different kinds of image threshold methods such as: Huang and Wang [12], Intermodes [13], IsoData [14], Li and Lee [15], MaxEntropy [16], Mean [17], MinError [18], Minimum [19], Moments [20], Otsu [21], Percentile [22], RenyiEntropy [16], Shanbhag [23], Triangle [24], Yen, Chang F. and Chang S. [25] and Adaptive thresholding [26]. However, the thresholding operation almost always guarantees the

presence of the salt-and-pepper noise, which we eliminate in this article by reducing the image size. Moreover, not all the algorithms are suitable for every image scene processing. Images in Fig. 2 illustrate the application of a few thresholding algorithms to the same pothole image. It can be seen that the best performance of separating the foreground and background is achieved by using the Adaptive and Triangle thresholding algorithms.

Thus, further two sections provide a short introduction to thresholding algorithms.

### 2.1. Triangle threshold

When an image histogram has the shape of unimodal distribution, most classical thresholding methods have difficulties when they try to correctly segment the image. To deal with this problem, a simple triangle segmentation method was proposed in 1977 [24]. The method was successfully applied in chromosome detection. Later, the method was used to analyse synthetic and real data and has proved that it is particularly effective when the object pixels produce weak peaks in a histogram [27].

The Triangle threshold method is very simple [24,27]. Suppose that we have an image intensity histogram of unimodal distribution shape, where the main peak is in the highest end of the histogram (Fig. 3). Then, a threshold that separates two classes could be computed as follows: a line is constructed between the last empty bin of the histogram before the first filled bin point  $A_{min}(\max_{f_i=0 \text{ and } f_{i+1} \neq 0}(i), 0)$  and the maximum frequency  $f_i$  histogram bin point  $A_{max}(j, \max(f_i))$ . Here  $i$  and  $j$  are intensity values. The threshold point  $T$  is selected as a histogram intensity value that the histogram bin maximises the perpendicular distance  $d$  between the

line and the histogram bin taken at each intensity from the interval  $[i, j]$ . Finally, pixels below the threshold  $T$  are set to the foreground value; those above the threshold will be set to the background value.

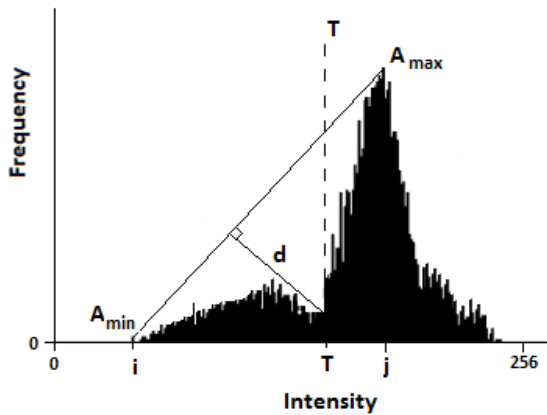


Figure 3. Scheme of the Triangle threshold method

### 2.2. Adaptive threshold

Usual thresholding methods use the global threshold for all pixels of the image. However, the Adaptive threshold method changes the threshold in such a manner that different threshold value is applied to each image pixel:

$$DI(x, y) = \begin{cases} 255 & \text{if } SI(x, y) > T(x, y), \\ 0 & \text{if } SI(x, y) \leq T(x, y). \end{cases}$$

Here  $SI$  is a source image;  $DI$  is a destination image;  $T(x, y)$  is a threshold calculated individually for each region at the pixel location  $(x, y)$ .

This method has a significant advantage for noisy images and images where the foreground or object is poorly defined [28]. In fact, this method can also be successfully applied in the edge detection process,

where boundaries between objects and the background are identified.

There are two variations of the method's algorithms: Chow and Kaneko approach [28] and local thresholding methods, where the threshold can be equal to: mean, weighted mean, median or mean of minimum and maximum of the region [29]. In the paper, a local adaptive threshold algorithm with the weighted mean was used. This algorithm is implemented as follows: at first, the region size ( $k_{size} \times k_{size}$ ) is defined; at the second step, the threshold  $T(x, y)$  is calculated for each region. Finally, the destination image (DI) is calculated by the formula above.

For experiments, OpenCV [30] implementation of the Adaptive threshold method with the weighted mean was used. In this implementation, all the weights can be uniform or distributed according to the Gaussian distribution [29]. In such a case, the weighted mean is equivalent to the Gaussian smoothing operator [29] that is known as a Gaussian blur. The Gaussian smoothing operator is used to process each and every pixel according to the central pixel  $(x, y)$  as a query pixel and  $T(x, y)$  is a weighted sum of the cross-correlation result after applying the Gaussian window of  $(x, y)$  to neighbouring pixels.

The threshold  $T$  depends on the following parameters of the Gaussian smoothing operator: the kernel size  $k_{size}$  or windows size and standard deviations  $\sigma_x$  and  $\sigma_y$ . Implementation of the OpenCV adaptive threshold enables us automatically calculate the standard deviations by the formula:  $\sigma_x = \sigma_y = 0.15 * k_{size} + 0.35$  [30]. Using this factor, only the kernel size has to be defined. To select an appropriate kernel size, an experiment was done. In the

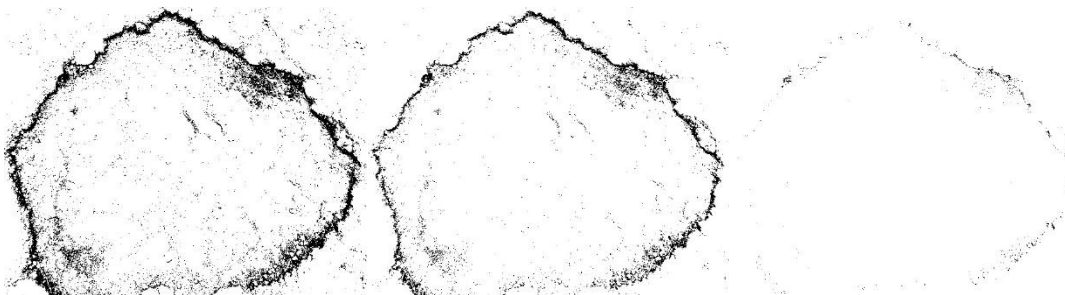


Figure 4. Dependence of the Adaptive threshold method on the kernel size. From left to right:  $k_{size}=951$ ,  $k_{size}=1075$ ,  $k_{size}=1201$

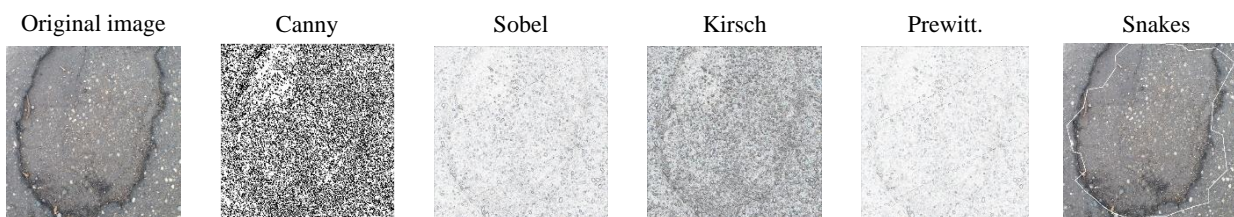


Figure 5. The result of application of Canny [32], Kirsch [33], Prewitt [13], Sobel [34], Snakes [31] algorithms to a pothole image

experiment, the kernel size was selected from the interval  $[1, 1500]$  and images were visually examined. The result has shown that the boundaries of potholes are clearly visible and images have less noise (Fig. 4), as  $k_{size} \in [1001, 1151]$ . Therefore, in the future investigation, a mean value of the interval was used, e.g.  $k_{size} = 1075$  pixels. It is important to note, that almost every time the kernel size exceeds the size of images used, therefore with to view to calculate the threshold  $T$  with the Gaussian smoothing operator, images were enlarged by duplicating their borders to fit the kernel size at each pixel.

### 3. Contour recognition

The analysis of the contour extraction methods has been accomplished. Such an analysis allowed us to select the best object contour detection method and use it in comparison with our proposed method for pothole contour detection. The Analysis results are presented in Fig. 5.

It is clearly seen that traditional approaches for contour extraction do not work well and the best result was obtained by applying segmentation of the Adaptive contour models (snakes) [31]. Thus, the next section describes in short the idea of models.

#### 3.1. Active contour models - snakes

The snakes is one of the classical frameworks for object tracking, shape recognition, segmentation, edge detection, which was proposed by M. Kass, A. Witkin and D. Terzopoulos in 1988 [31]. The basic concept of the framework is snakes or an active contour model – it is an energy-minimizing spline, guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. The model proposes effective energy functions that have few local minima and little dependence on the starting points. Nevertheless, it relies on other global mechanisms to place the starting points somewhere near the desired contours.

In short, the snakes model tries to draw a spline that minimizes the energy function:

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{img}(v(s)) + E_{con}(v(s)) ds,$$

where the position of active contour points  $v(s) = (x(s), y(s))$  in the image plane,  $E_{int}$  is internal energy of the contour,  $E_{img}$  represents the energy delivered from the image,  $E_{con}$  represents the energy of external constraint forces.  $E_{con}$  allows for a user to interactively control and guide the snake towards the contour.

The internal energy  $E_{int}$  serves to impose a piecewise smoothness constraint. It is defined as follows:

$$E_{int}(v(s)) = \alpha E_{cont}(v(s)) + \beta E_{curv}(v(s)),$$

where  $E_{cont}$  is the continuity energy,  $E_{curv}$  represents curvature energy, and  $\alpha, \beta$  are user-defined weights.

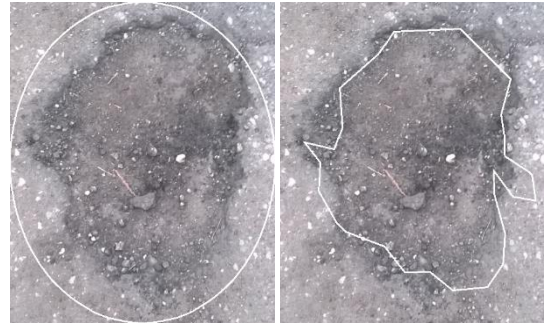
The image energy is decomposed into the weighted sum of the following energies:

$$E_{img} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$$

where  $E_{line}(v(s)) = I(v(s))$  is a line energy function represented by intensity of the image,  $E_{edge} = -|\nabla I(v(s))|^2$  is the edge energy based on the image gradient,  $E_{term}$  is the termination energy that helps to find termination of line segments and corners.

Optimization of the  $E_{snake}$  function is quite simple and can be performed iteratively by the gradient method. A full description of the snakes method can be found in [31].

For our experiments, we used the OpenCV implementation of the snakes method. In this implementation, all the weights of the  $E_{img}$  function are multiplied by the constant  $\gamma$ , that provides a separate control of image energy.  $E_{con}$  is represented only by the initial contour. Since the general form of a pothole is similar to the oval one, we had chosen the initial contour of ellipsoidal shape. The ellipse is drawn to fit the image (Fig. 6) and has 100 uniformly distributed points.



**Figure 6.** Snakes model results: From left to right: initial contour, final snake model contour ( $\alpha = 0.1$ ,  $\beta = 0.4$ , and  $\gamma = 10$ ).

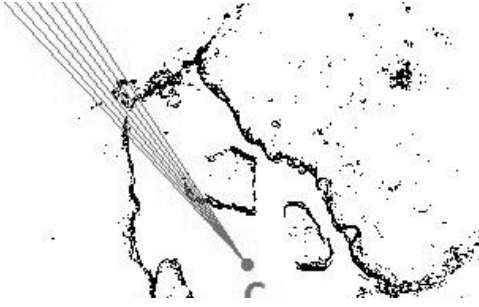
The constants  $\alpha$ ,  $\beta$ ,  $\gamma$  have been calculated experimentally and for the current data set the highest accuracy was obtained with the following values:  $\alpha = 0.1$ ,  $\beta = 0.4$ , and  $\gamma = 10$ . These values were used in the future experiments.

#### 3.2. Proposed edge detection method

To effectively find the contour of a pothole in a binary image, we have proposed a new heuristic edge detection method. Assume that we have a binary image of the pothole (Fig. 7). Then the contour can be obtained by processing the steps as follows:

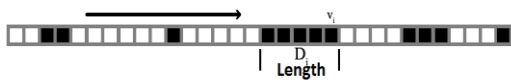
**Step 1.** Calculate the center of the image. We assume that the image center point falls within the area of the pothole.

**Step 2.** Draw the lines from the center point to each image border point. Fig. 7 depicts the lines drawing idea.



**Figure 7.** Lines crossing the pothole borders

**Step 3.** After drawing the lines, the algorithm collects the thresholded image points crossed by the lines. To determine whether the points lay on a line, we have used the Bresenham method [35]. For simplicity, assume that all the lines preserving their order are saved in set  $\mathbf{A}$ . Each line (a sample of the line is depicted in Fig. 8) can be understood as an ordered list of intensities of black and white pixels. The arrow on the top of the image shows the point ordering direction from the centre image point to the image boundary. The line depicts blobs of black pixels. The length  $D_i$  of each blob can be obtained by finding the length of the consequent black pixel count.



**Figure 8.** Example of a line in set  $\mathbf{A}$

**Step 4.** Since all the lines start from the centre of the image, there are a lot of the same blobs in different lines. Therefore, to cope with removal of duplicates, filtering of homogeneous points must be applied. With a view to eliminate them, each point that belongs to the line is considered. The coordinates of repeating points are removed from all the lines except the middle ones. Thus, a new set  $\mathbf{A}^*$  is obtained.

**Step 5.** Detection of the pothole contour path points is performed by Dijkstra's algorithm [36] with the structure  $\mathbf{S} = (\mathbf{G}, \mathbf{L})$  defined. The graph  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$  is constructed, where  $\mathbf{V} = \{v_i\}$  is composed of vertices  $v_i$  that are taken from the set  $\mathbf{A}^*$  and are presented by each line each blob the furthest black pixels (Fig. 8),  $\mathbf{E}$  consists of edges that connect all the vertices between two neighbour lines.  $\mathbf{L} = \{l_{ij}\}$  is a set of lengths between the  $i$ th and  $j$ th vertices of set  $\mathbf{V}$ . The lengths are calculated by the proposed formula:

$$l_{ij} = w_1(D_i + D_j) + w_2d_{ij} + w_3(d_{ic} + d_{jc}),$$

where  $D_i, D_j$  are lengths of line sections  $i$  and  $j$ ;  $d_{ij}$  is the Euclidian distance between the vertices  $i$  and  $j$ ;  $d_{ic}$  and  $d_{jc}$  are Euclidian distance between the vertex and image centre,  $w_1, w_2, w_3$  are weights such as  $\sum_{i=1}^3 w_i = 1$ .



**Figure 9.** Contour path after application of Dijkstra's algorithm

All the vertices compose an ordered set  $\mathbf{F} = \{v_i\}$ .

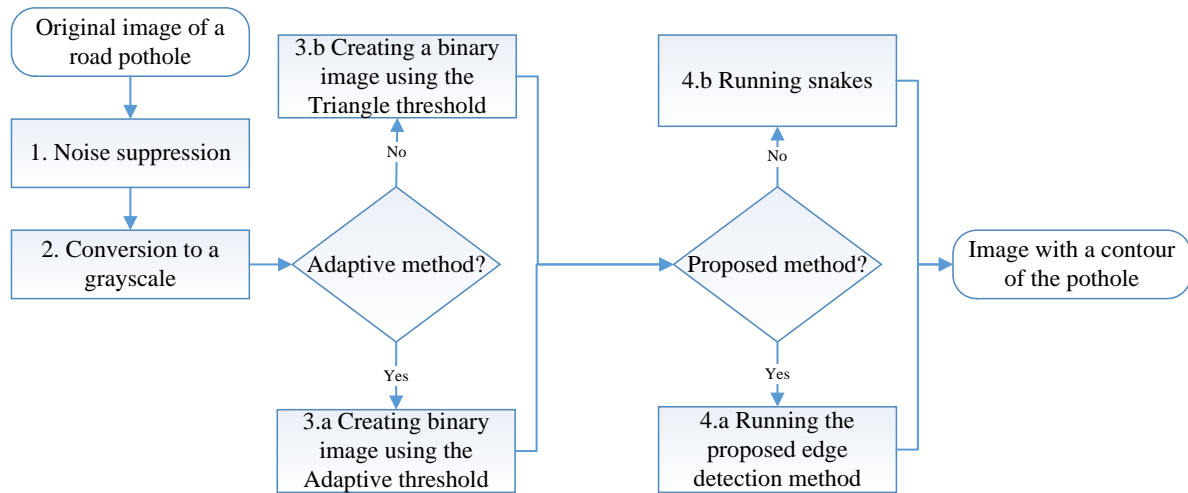
The weights  $w_i$  are selected so as to fit the line to contour. After a careful analysis, the weights were selected experimentally:  $w_1 = 0.6$ ;  $w_2 = 0.25$ ;  $w_3 = 0.15$ . The result is presented in Fig. 9.

**Step 6.** Elimination of the pothole contour path outliers is accomplished as described further. A new set  $\mathbf{F}^* = \{f_i\}$  is constructed from vertices of the set  $\mathbf{F}$ . At the beginning, we take the first vertex  $v_0$  from the set  $\mathbf{F}$  and insert it into set  $\mathbf{F}^*$ , i.e.  $f_0 = v_0$ . We make an assumption that the vertex  $v_0$  is not an outlier. Then if it is closest to  $v_0$  from 10 left-sided neighbours in set  $\mathbf{F}$ , we insert the vertex  $v_i$  into set  $\mathbf{F}^*$ . Further, we take the newly obtained vertex as a reference and start to look for its closest neighbour. The process continues until the whole set  $\mathbf{F}$  is checked. The filtering result is shown in Fig. 10.



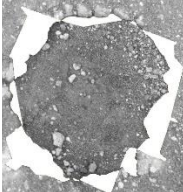
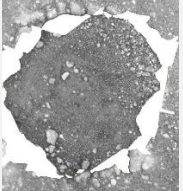
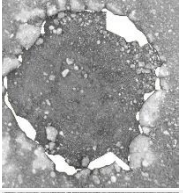
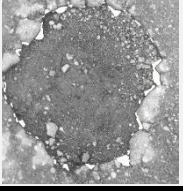
**Figure 10.** Smoothed contour of the pothole after filtering procedure

## Detection of the Road Pothole Contour in Raster Images



**Figure 11.** General contour recognition scheme for pothole detection

**Table 1.** The accuracy of methods

Method	Example	C=A/B*100%										Average
		100%-90%	90%-80%	80%-70%	70%-60%	60%-50%	50%-40%	40%-30%	30%-20%	20%-10%	10%-0%	
Snakes - Triangle		2	7	18	15	20	13	6	7	4	9	52%
Snakes - Adaptive		2	10	11	25	11	13	11	4	2	11	53%
Proposed - Triangle		13	25	25	16	11	4	2	0	1	4	70%
Proposed - Adaptive		15	38	25	11	9	2	0	0	0	0	78%

### 4. General contour recognition scheme

The general contour recognition scheme (Fig 11) was used to evaluate the proposed contour (edge) detection method, and to deliver the final version of the proposed pothole contour detection method.

The method for pothole contour detection, proposed in this article, is organized as follows:

1. Noise suppression. Reduction of the image size is accomplished by applying nearest neighbour interpolation technique [33]. The image size is contracted to 700 pixels in width and then the length is also reduced accordingly. Such a reduction enables us to eliminate salt-and-pepper noise and also to increase the algorithm speed significantly.

2. Conversion to a grayscale. The resized image is converted to a greyscale. During the conversion to a grey-scale image (from RGB) the YUV conversion scheme has been applied.
3. Thresholding of image for foreground and background segmentation. In this step, we used one of the presented thresholding methods - either Triangle threshold or Adaptive threshold.
4. Edge of the pothole detection. After obtaining a binary image in Step 3, the snakes method and our method for edge detection has been applied.

By mixing the parts, presented in the third and fourth steps, we can obtain four different combinations of the method: Snakes-Triangle (method that includes parts 3.b and 4.b), Snakes-Adaptive (3.a and 4.b), Proposed-Triangle (3.b and 4.a) and Proposed-Adaptive (3.a and 4.a). These modifications of the method were compared in the next section.

## 5. Comparison of the described methods

Experiments were done by using a set of 105 various resolution images, purposely gathered for the research and captured with a mobile device camera. The aim of the experiment was to evaluate the accuracy of the method for pothole contour detection we have proposed, and that of the snakes method by comparing them to that of the human-drawn pothole contour. The accuracy was expressed as  $C = A/B * 100\%$ , where  $A$  is the area (white colour) contour calculated applying either the proposed method or snakes method;  $B$  is the area of a human-drawn contour. The results are shown in Table 1.

These results show that when comparing the snakes method after adaptive and triangle thresholds, the adaptive threshold shows better results. The same is true for the method proposed in this paper. Also, comparing the snakes method and the proposed method, the latter shows better results.

## 6. Conclusion

In this article, the new method for pothole contour (edge) detection has been proposed and evaluated using 105 pothole images and the study conclusions are:

- When comparing the Snakes-Adaptive and Proposed-Adaptive methods, the better performance shows Proposed-Adaptive method. It increase difference accuracy by 25%.
- The highest concurrence (applying the Adaptive threshold and the proposed edge detection method) of the contour drawn when applying the proposed pothole contour detection method and the human-drawn contour was 97%. The lowest accuracy was 42% , mostly as a result of

insufficient lighting conditions or external impediments. The average accuracy of the proposed method is 78% . 15 images were recognised with the size accuracy of 90% – 100% , 38 images with the size accuracy of 80% – 90% , and only 2 images with that of 40% – 50% and below.

## References

- [1] **A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo.** Real Time Pothole Detection Using Android Smartphones with Accelerometers. *Researchgate*, 2011, DOI: 10.1109/DCOSS.2011.5982206.
- [2] **A. Kulkarni, N. Mhalgi, S. Gurnani, N. Giri.** Pothole Detection System using Machine Learning on Android. *International Journal of Emerging Technology and Advanced Engineering*, 2014, Vol. 4, No. 7, 360-364.
- [3] **S. Balakuntala, S. Venkatesh.** An Intelligent System to Detect, Avoid and Maintain Potholes: A Graph Theoretic Approach. *Conference on: Mobile Computing and Ubiquitous Networking, Seventh International Conference* 2014, 10.1109/ICMU.2014.6799066.
- [4] **A. Raibagi, S. Anand, Swetha.** Ultrasonic anti crashing system for automobiles. *International Journal of Advanced Research in Computer and Communication Engineering*, 2013, Vol. 2, No. 4, 1774-1778.
- [5] **A. Miraliakbari, M. Hahn, H. G. Maas.** Development of a Multi-sensor System for Road Condition Mapping. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2014, XL-1, 265-272.
- [6] **C. Jengo, D. Hughes, J. LaVeigne, I. Curtis.** Pothole Detection and Road Condition Assessment Using Hyperspectral Imagery. *Conference on: Geospatial Goes Global: From Your Neighborhood to the Whole Planet*, 2005.
- [7] **H. Punjabi, R. Nanwani, A. Vaswani, R. Jotwani, A. Kunte.** Intelligent Pothole Detection System. *International Journal of Emerging Technology and Advanced Engineering*, 2014, Vol. 4, No. 7, 590-595.
- [8] **M. Jahanshahi, S. Masri.** Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. *Automation in Construction*, 2012, Vol. 22, 567–576.
- [9] **S. Radopoulou, I. Brilakis.** Patch detection for pavement assessment. *Automation in Construction*, 2015, Vol. 53, 95-104.
- [10] **P. Treigys, V. Marcinkevičius, A. Kaklauskas.** Analysis of iris and pupil parameters for stress recognition. *Information Technology and Control*, 2012, Vol. 41, No. 1, 7-14.
- [11] **O. Niakšu, G. Balčiūnaitė, R. J. Kizlaitis, P. Treigys.** Semi-automation of Doppler spectrum image analysis for grading aortic valve stenosis severity. *Methods of Information in Medicine*, 2016, Vol. 55, No. 1, 23-30.
- [12] **L. K. Huang, M.J. Wang.** Image thresholding by minimizing the measure of fuzziness. *Pattern Recognition*, 1995, Vol. 28, No. 1, 41-51.

- [13] **J. Prewitt**. Object Enhancement and Extraction. Picture processing and Psychopictorics, B. Lipkin and A. Rosenfeld, Eds., New York: Academic Press, 75-149, 1970.
- [14] **T. Ridler, S. Calvard**. Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man and Cybernetics*, 1978, Vol. 8, No. 8, 630-632.
- [15] **C. Li, C. Lee**. Minimum Cross Entropy Thresholding. *Pattern Recognition*, 1993, Vol. 26, No. 4, 617-625.
- [16] **J. Kapur, P. Sahoo, A. Wong**. A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram. *Graphical Models and Image Processing*, 1985, Vol. 2, No. 3, 223-237.
- [17] **C. Glasbey**. An analysis of histogram-based thresholding algorithms. *CVGIP: Graphical Models and Image Processing*, 1993, Vol. 55, No. 6, 532-537.
- [18] **J. Kittler, J. Illingworth**. Minimum error thresholding. *Pattern Recognition*, 1986, Vol. 19, No. 1, 41-47.
- [19] **J. M S. Prewitt, M. L. Mendelsohn**. The analysis of cell images. *Annals of the New York Academy of Sciences*, 1966, Vol. 128, 1035-1053.
- [20] **W. H. Tsai**. Moment-Preserving Thresholding: A New Approach. *Computer Vision, Graphics, and Image Processing*, 1984, Vol. 29, 377-393.
- [21] **N. Otsu**. A Threshold Selection Method From Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1979, Vol. 9, 62-66.
- [22] **W. Doyle**. Operation useful for similarity-invariant pattern recognition. *Journal of the Association for Computing Machinery*, 1962, Vol. 9, No. 2, 259-267.
- [23] **A. Shanbhag**. Utilization of information measure as a means of image thresholding. *CVGIP: Graphical Models and Image Processing*, 1994, Vol. 56, No. 5, 414-419.
- [24] **J. C. Yen, F J. Chang, S. Chang**. A New Criterion for Automatic Multilevel Thresholding. *IEEE Transactions on Image Processing*, 1995, Vol. 4, No. 3, 370-378.
- [25] **E. R. Davies**. Machine Vision: Theory, Algorithms, Practicalities. *Elsevier Inc*, 2005.
- [26] **P. L. Rosin**. Unimodal thresholding. *Pattern Recognition*, 2001, Vol. 34, No. 11, 2083-2096.
- [27] **C. Chow, T. Kaneko**. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and Biomedical Research*, 1972, Vol. 5, No. 4, 388-410.
- [28] **R. Fisher, S. Perkins, A. Walker, E. Wolfart**. Hypermedia Image Processing Reference. HIPR, 2000.
- [29] **G. Bradski**. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.
- [30] **M. Kass, A. Witkin and D. Terzopoulos**. Snakes: Active contour models. *International Journal of Computer Vision*, 1988, Vol. 1, No. 4, 321-331.
- [31] **J. Canny**. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, Vol. 8, 679 - 698.
- [32] **R. A. Kirsch**. Computer Determination of the Constituent Structure of Biological Images. *Computers and Biomedical Research*, 1971, Vol. 4, No. 3, 315-328.
- [33] **I. Sobel and G. Feldman**. A 3x3 Isotropic Gradient Operator for Image Processing. In: *Conference on: Stanford Artificial Intelligence Project*, 1968.
- [34] **J. E. Bresenham**. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 1965, Vol. 4, No. 1, 25-30.
- [35] **E. W. Dijkstra**. A Note on Two Problems in Connexion with Graph. *Numerische Mathematk*, 1959, Vol. 1, No. 1, 269-271.

Received September 2015.