# A Fast One-Pixel Wide Contour Detection Method for Shapes Contour Traversal in Binary Images

Hrvoje Leventić, Tomislav Keser, Krešimir Vdovjak
Faculty of Electrical Engineering, Computer Science and Information Technology,
Josip Juraj Strossmayer University of Osijek, Osijek, Croatia

***Abstract*—Contour detection is an important step in contour based object analysis in 2D images. We present a novel method for contour detection optimized for contour traversal tasks in 2D images. The method is based on morphological operations and detects a one-pixel wide contour around objects in binary images. Performance of the method is evaluated on ground truth images from our dataset created by manually tracing the contour in images. Proposed method achieves high DICE coefficient overlap to ground truth images and superior detection compared to other state-of-the-art methods.**

***Keywords*—Contour tracing; contour detection; binary image**

## I. Introduction

Contour detection is an important step in a wide range of image processing methods, from various segmentation tasks to object shape recognition. Often the contour detection is the first step in the object shape recognition methods. Various contour detection methods have been proposed in the literature. We can divide contour detection methods in several categories [1]: methods based on local pattern analysis, methods based on contextual and global information and multiresolution methods. Some examples of methods based on local pattern analysis range from methods based on differential operators [2–4], methods based on statistical image properties [5], methods based on phase congruency and local energy [6] to methods based on morphological properties [7–9]. An examples of methods based on contextual and global information are methods based on machine learning [10], [11]. Another important category are images based on multiresolution [9].

The choice of the preferred method for the task is often dependent on the required accuracy of the detected contour, as well as the required performance. According to [1], statistical based methods are among the most accurate methods but suffer from long computation times. Methods based on morphological properties are among the most performant methods, but have problems handling the outliers. Additionally, the choice depends on the required representation of the detected contour. The contour can be represented as a discrete set (often as pixels in an image – as with the morphologically based methods [7], [8]), or as a parametric contour – as with active contours and similar methdos [2], [3], [12].

We present a method for contour detection based on morphological properties of binary image. Our method works on binary images created after thresholding the input image. This paper is organized in a following manner: section II presents the proposed method, secton III presents the achieved results, while in section IV we give concluding remarks.



## II. Proposed method

In this section we will present the method for contour detection in binary images based on morphological properties of the input image.

### A. Initial considerations and requirements

The goal for our method is to create a one-pixel wide contour of an object in 2D binary image. The method will be a part of a larger object shape analysis method (outside the scope of this paper). The shape analysis method is based on traversing and analyzing the detected contour. Shape analysis method has to run in real time, thus the detected contour has to be suitable for as simple contour traversal as possible. The considerations for the contour detection method are the following:

- Fast performance is more important than accuracy – the method has to be able to run in realtime on embedded hardware devices.
- Detected contour is one-pixel wide and represented by a binary image.
- Detected contour has to be optimized for simple contour traversal: starting from a pixel on the contour, for every pixel we simply follow its unvisited neighboring pixel until we reach the starting pixel. This traversal algorithm is not robust to loops and does not support backtracking while following the contour.

More formal description of the requirements is provided below. Let $\mathbf{p} \in \mathbb{Z}^2$ denote the pixel position in a 2D binary image $f$ where pixel value $f(\mathbf{p})$ is element of binary set $\{0, 1\}$. For the given pixel position $\mathbf{p} = (x_\mathbf{p}, y_\mathbf{p})$ we define the 8-neighborhood of pixel $\mathbf{p}$ as:

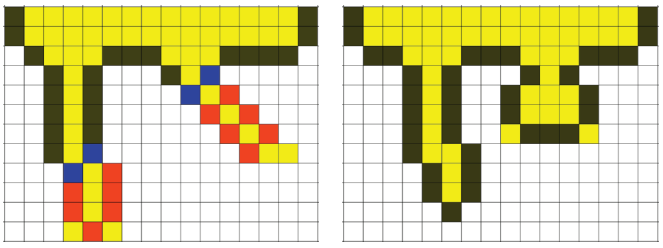
$$N_8(\mathbf{p}) = \left\{ \mathbf{q} \in Z^2 \middle| 1 \leq \sqrt{(x_\mathbf{q} - x_\mathbf{p})^2 + (y_\mathbf{q} - y_\mathbf{p})^2} < 2 \right\}. \tag{1}$$

Let $C$ denote the set of all pixels $\mathbf{p} \in \mathbb{Z}^2$ belonging to the resulting contour. For every pixel $\mathbf{p} \in C$ we define the set of pixels in contour connected to pixel $\mathbf{p}$ as:

$$N_C(\mathbf{p}) = \left\{ \mathbf{q} \in \mathbb{Z}^2 \middle| \mathbf{q} \in N_8(\mathbf{p}), \mathbf{q} \in C \right\}. \tag{2}$$

The suitability of the detected contour for the traversal algorithm is shown in definition 2.1.

978-1-5386-7189-4/18/$31.00 ©2018 IEEE



(a) (b)

Figure 1. Example of contours fitting the object. (a) Detected contour correctly fits the object. (b) Detected contour cannot completely fit the object because of violation of the suitability criteria defined in Definition 2.1

*Definition 2.1:* The contour $C$ is suitable for the traversal if for every $\mathbf{p} \in C$ the following is true:

$$|N_C(\mathbf{p})| \leq 2. \tag{3}$$

In other words, the contour $C$ is suitable if every pixel in the contour is connected to at most two other pixels in contour. Figure 1a illustrates the object (yellow) where the detected contour (black) respects the definition 2.1. Figure 1b illustrates the object where the detected contour cannot completely fit the object, otherwise the definition 2.1 will be broken. Blue pixels illustrate coordinates where, if the contour was fit to the edges of the object (red pixels), pixels will be connected to more than 2 neighboring pixels inside the contour. The contour represented with red pixels in the figure could result in loops during the contour traversal. When every pixel in the contour has at most two neighboring pixels, we can use very simple and fast traversal algorithm.

### B. Contour detection method

In this subsection we will present the proposed method for contour detection. The method consists of two steps: (i) morphological edge detection and (ii) the method for removing parts of the object which cause loops for traversal algorithm. We previously denoted $f$ as 2D binary image and $\mathbf{p}$ as a pixel position in $f$. For the given position $\mathbf{p} = (x_\mathbf{p}, y_\mathbf{p})$ we define the 4-neighborhood of pixel $\mathbf{p}$ as:

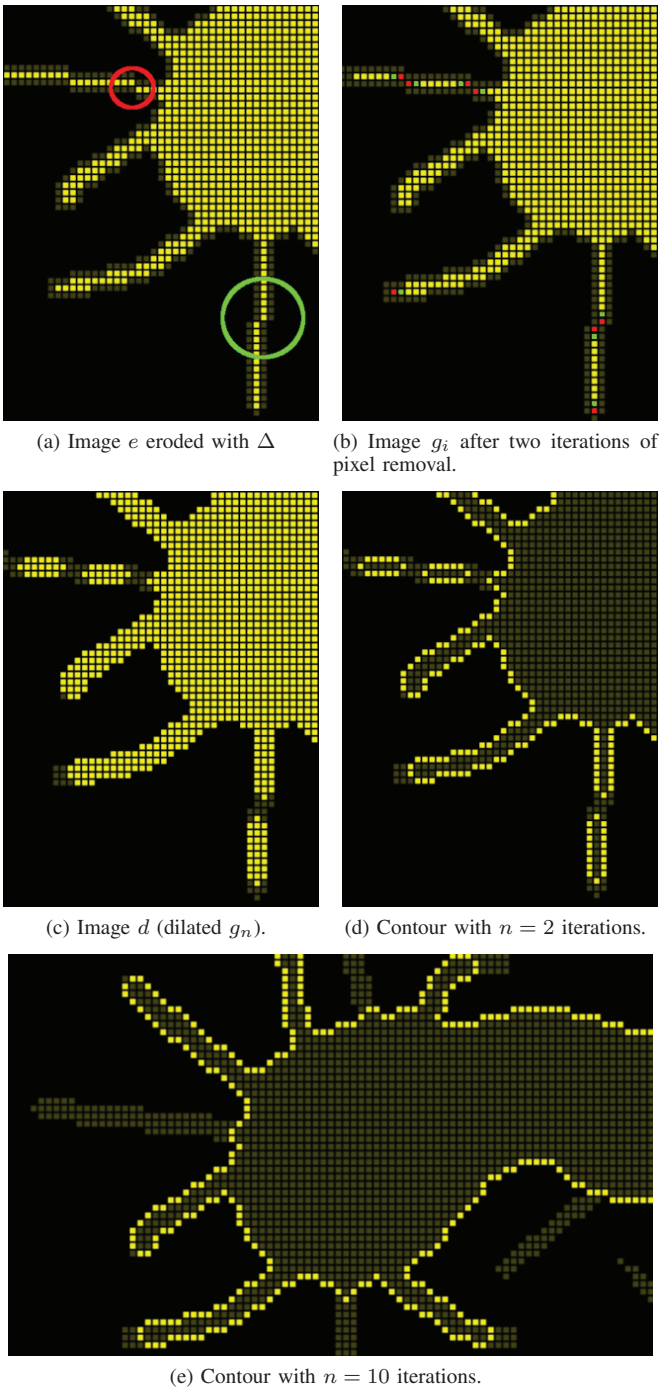
$$N_4(\mathbf{p}) = \left\{ \mathbf{q} \in \mathbb{Z}^2 \middle| \sqrt{(x_\mathbf{q} - x_\mathbf{p})^2 + (y_\mathbf{q} - y_\mathbf{p})^2} = 1 \right\}. \tag{4}$$

We perform the morphological erosion with the structuring element cross, denoted as:

$$\Delta(\mathbf{p}) = N_4(\mathbf{p}) \cup \{\mathbf{p}\}. \tag{5}$$

Let $e$ denote the image $f$ eroded with structuring element $\Delta$, as shown in Figure 2a. Red and green circles show a part of the object where the contour will not follow the object shape because of the suitability criteria (as shown in Figure 1b). Next, we iteratively detect and remove pixels from $e$ which will create problems for the contour detection (as shown in the Figure 2a with green and red circles). We define a set of pixels to be removed in an iteration $i$ as:

$$G_i = \left\{ \mathbf{v} \in \mathbb{Z}^2 \middle| \sum_{\mathbf{q} \in N_4(\mathbf{v})} g_{i-1}(\mathbf{q}) < 2 \right\}, \tag{6}$$



(a) Image $e$ eroded with $\Delta$ (b) Image $g_i$ after two iterations of pixel removal.

(c) Image $d$ (dilated $g_n$). (d) Contour with $n = 2$ iterations.

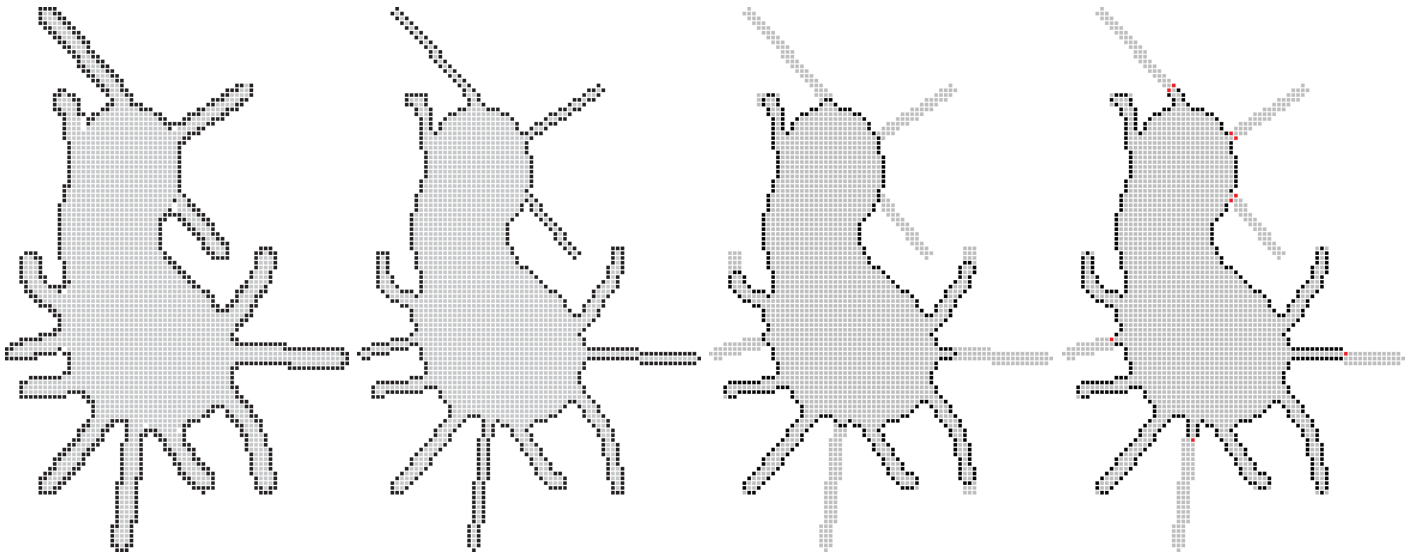
(e) Contour with $n = 10$ iterations.

Figure 2. Example of contour detection algorithm. Yellow represents the pixels with value 1, grey represents the pixels removed during the processing, black represents background pixels of the input image. Red pixels are removed in $i = 1$ and green pixels in $i = 2$.

where $g_i$ denotes the eroded image $e$ after $i \in N$ iterations of pixel removal. In the first iteration the starting image $g_0 = e$. The image $g_i$ after pixel removal in iteration $i$ is defined as:

$$g_i(\mathbf{v}) = \begin{cases} 0 \ ; \ \mathbf{v} \in G_i \\ g_{i-1}(\mathbf{v}) \ ; \ \text{otherwise} \end{cases} \tag{7}$$

Figure 2b shows removed pixels after $n = 2$ iterations. The



(a) Canny edge detector (b) Suzuki detector (c) Proposed method (d) Ground truth

Figure 3. Comparison of the results on one image from our dataset. Red pixels in image (d) show the locations where the ground truth contour stops unconditionally following the object shape. The reason is that the contour would violate the suitability criteria defined in Definition 2.1 if it did follow the object shape.

number of iterations parameter is user configured and depends on the size of objects in the image. After the iterative removal process the image $g_n$ is dilated. Let $d$ denote the $g_n$ image dilated with structuring element $\Delta$. The contour image $c$ is calculated as:

$$c(\mathbf{p}) = d(\mathbf{p}) - g_n(\mathbf{p}), \forall \mathbf{p} \in \mathbb{Z}^2 \tag{8}$$

while the set of pixels comprising the contour is defined as:

$$C = \left\{ \mathbf{q} \in \mathbb{Z}^2 \middle| c(\mathbf{q}) = 1 \right\}. \tag{9}$$

Figure 2c shows the dilated image $d$. Figures 2d and 2e show the final contour image $c$ after 2 and 10 iterations, respectively.

## III. Results and discussion

In this subsection we will present the evaluation results of the proposed method. Our method is evaluated on manually created ground truth contours. Figure 4 shows selected images from our dataset. The figure shows that the method will always detect the contour correctly when the width of the object is larger than 3 pixels. This is evident in images in Figures 4a and 4b. These two images show very little errors between ground truth contour and detected contour. Errors in detection appear when there is a long narrow part of the object – 3 or less pixels wide. In those parts of the image the detected contour has a potential to not respect the definition 2.1. This is especially evident in Figures 4c and 4f. Our implementation prioritizes the definition 2.1 over the contour accuracy.

Our method is objectively evaluated by calculation of the dice coefficient [13] between the ground truth contours and the detected contours. With the detected contour $C$ and ground truth contour denoted as $T$, dice coefficient is defined as:

$$dice(T, C) = \frac{2 * |T \cap C|}{|T| + |C|} \tag{10}$$

Our method achieves **0.9431** dice coefficient overlap on our dataset. The method is compared to Canny edge detector [14] implemented in MATLAB and contour detector implemented in openCV based on the work by Suzuki [15]. Figure 3 shows the results of Canny detector and Suzuki detector compared to ground truth image. Suzuki detector detects a contour that has a very good fit to the object in the images. However, both methods create contours which do not respect the definition 2.1 and create problems for contour traversal algorithms. Still, the Suzuki detector is vastly superior to Canny edge detector in terms of dice coefficient, as well as vastly superior to both methods in terms of runtime performance, as evident in Table I. Canny detector does not create a one pixel wide contour and creates a contour that is larger than the object in the image.

TABLE I. Methods tested on 1024x768 pixel images from our dataset

| **Method** | Canny | Suzuki | Proposed method |
|---|---|---|---|
| **Runtime** | 38ms | 0.8ms | 18ms |
| **Dice overlap** | 0.6355 | 0.9367 | 0.9431 |

## IV. Conclusion

We have presented a novel method for contour detection optimized for contour traversal tasks in 2D images. The detection is perfomed on binary images. Performance of the method is evaluated on ground truth images from our dataset as well as compared to other state-of-the-art methods. Proposed method achieves high DICE coefficient overlap to ground truth images and superior detection compared to other evaluated methods. The proposed method is suitable for shape analysis tasks based on contour traversal.

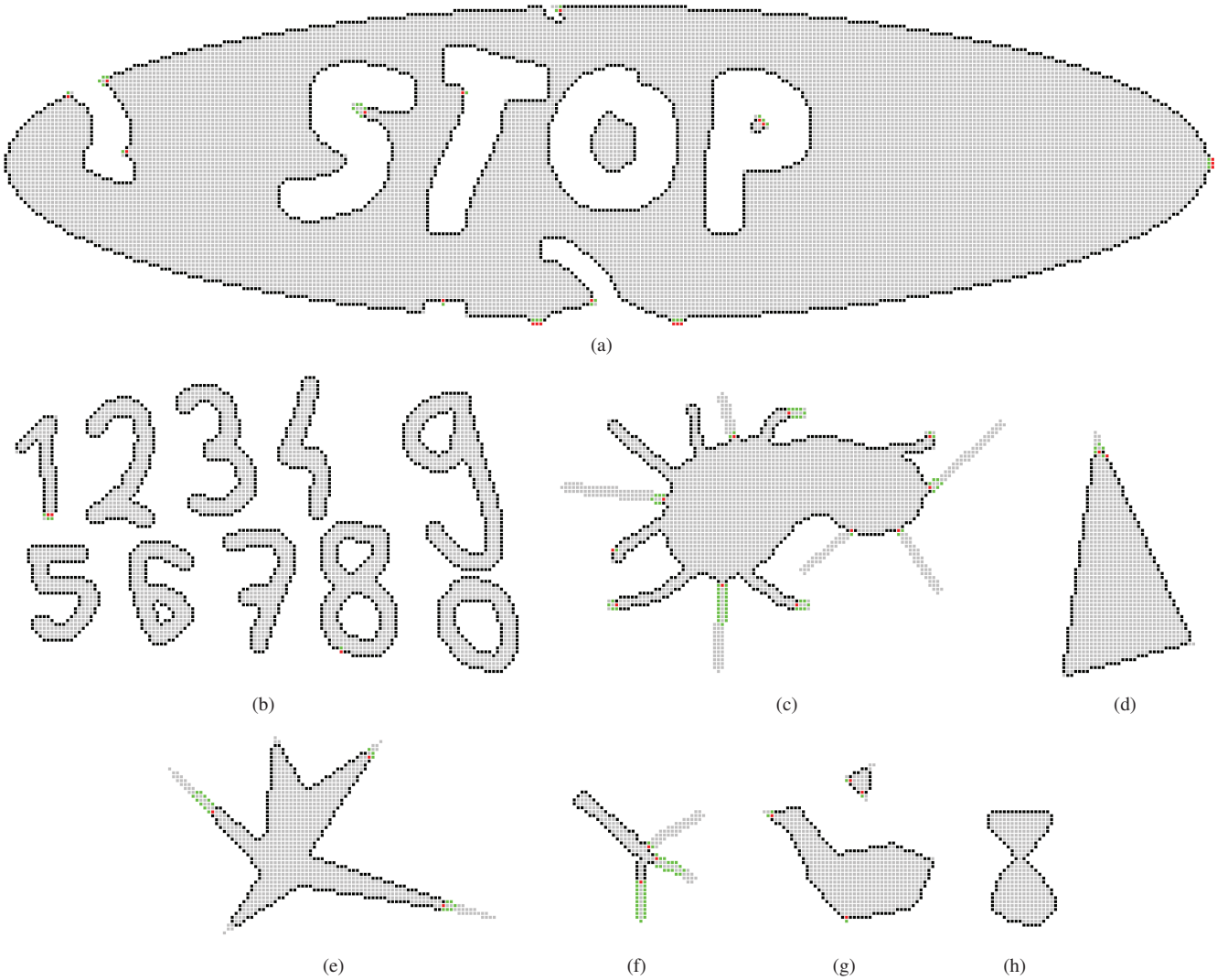



Figure 4. Results of contour detection on selected images from our dataset (10 iterations). Black pixels are the same in both the ground truth contour and the detected contour. Red represents pixels in the detected contour but not in the ground truth contour. Green represents pixels in the ground truth contour but not in the detected contour.

## REFERENCES


[1] G. Papari and N. Petkov, "Edge and line oriented contour detection: State of the art," *Image and Vision Computing*, vol. 29, no. 2-3, pp. 79–103, 2011.

[2] T. F. Chan, L. Vese, and others, "Active contours without edges," *Image processing, IEEE transactions on*, vol. 10, no. 2, pp. 266–277, 2001.

[3] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[4] H. Park, T. Schoepflin, and Y. Kim, "Active contour model with gradient directional information: Directional snake," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 2, pp. 252–256, 2001.

[5] Q. Chen, Z. M. Zhou, M. Tang, P. A. Heng, and D.-S. Xia, "Shape statistics variational approach for the outer contour segmentation of left ventricle MR images," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 3, pp. 588–597, 2006.

[6] P. Kovesi, "Image features from phase congruency," *Videre: Journal of computer vision research*, vol. 1, no. 3, pp. 1–26, 1999.

[7] J. Lee, R. Haralick, and L. Shapiro, "Morphologic edge detection," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 142–156, 1987.

[8] J.-F. Rivest, P. Soille, and S. Beucher, "Morphological gradients," *Journal of Electronic Imaging*, vol. 2, no. 4, pp. 326–337, 1993.

[9] B. Chanda, M. K. Kundu, and Y. V. Padmaja, "A multi-scale morphologic edge detector," *Pattern Recognition*, vol. 31, no. 10, pp. 1469–1478, 1998.

[10] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3982–3991.

[11] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference On*. IEEE, 2015, pp. 4380–4389.

[12] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International journal of computer vision*, vol. 22, no. 1, pp. 61–79, 1997.

[13] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[14] J. Canny, "A computational approach to edge detection," in *Readings in Computer Vision*. Elsevier, 1987, pp. 184–203.

[15] S. Suzuki, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.