ISSN 1392 – 1215 ELECTRONICS AND ELECTRICAL ENGINEERING 2010. No. 10(106)
ELEKTRONIKA IR ELEKTROTECHNIKA

T 171 MICROELECTRONICS
MIKROELEKTRONIKA

Time-Efficient Adaptive Segmentation Algorithm forIC Layers Images

G. Masalskis, R. Navickas
Vilnius Gediminas Technical University, Department of Computer Engineering,
Naugarduko g. 41, LT-03227, Vilnius, phone: +370 684 23229, e-mail: giedrius.masalskis@el.vgtu.lt

Introduction

Each automated image analysis task requires segmentation step to separate foreground objects from image background. There are many different approaches to solving segmentation problem, such as thresholding, region growing, graph analysis [1], region splitting and merging, rule-based segmentation, histogram-based segmentation and mixed algorithms [2]. Most suitable approach is selected depending on properties of input image. Threshold function is one of the most often selected operations used to segment images. It translates the image from colour or greyscale data (multiple bits per pixel) to the binary representation of only foreground or background pixels (1 bit per pixel). This enables further application of binary image specific analysis and processing algorithms. Fundamental bi-level threshold function, used in every threshold algorithm, is defined as

$$B = \begin{cases} 0, \text{when } P < T, \\ 1, \text{when } P \geq T, \end{cases} \tag{1}$$

where P represents pixel intensity value in source image and T represents threshold value [1]. Many threshold algorithms concentrate on selecting suitable threshold value *T* for complete image. While global *T* method is useful in many situations, there are cases where more sophisticated approach is required. For example, if image contains regions of variable intensity, global *T* becomes useless when background intensity exceeds *T* value in some regions. And it is very important to separate only specific foreground areas while disregarding brightness inequality during image segmentation.

In our research we perform analysis of integrated circuits images which are obtained using optical microscopes. Sometimes these images are on the edge of physical capabilities of optical microscopy. General traits of such images are: varying brightness throughout image, blurred or out of focus features of interest, lens dust, sensor dust and sensor noise. It is possible to filter out dust and noise but the biggest problem for further analysis remains uneven brightness in different parts of image, since our analysis algorithms depend on binary representation. Using global threshold function produces inconsistent, generally unsatisfying results. This is why it is important to have a more sophisticated, locally adaptive threshold function which would be immune to changing brightness conditions and which would produce good binary image with analysis objects set as foreground pixels.

Many different algorithms have been already proposed for calculation of image global threshold *t* in scientific publications. Approaches such as iterative (bi-level) threshold [1], variance based threshold [3] or entropy based algorithms [4–6] provide good results on uniform intensity images [7] but are not suitable for adaptive *T* applications. Locally adaptive threshold *T* calculation proposals [8-12] are computationally complex and thus unsuitable for our type of input images either because of unsatisfactory segmentation results or calculation time required for each image.

We chose to create new specific adaptive threshold function, suitable for our needs in IC layer image segmentation.

Objectives

The goal was to create segmentation algorithm with the following characteristics:

- It should be able to separate foreground objects in varying brightness conditions;
- Separated foreground objects should have smooth, accurate edges;
- Noise immunity would be an advantage;
- Possible computationally fast implementation.

Methodology

Because it is able to operate in varying brightness conditions, our custom threshold function will be referred as *adaptive threshold* in this paper. Its algorithm consists of the following steps:

1. Input greyscale image M_{in}.
2. Calculate global threshold value T_g.
3. Take pixel P_i.
4. Calculate brightness value I_i in pixel window W_{pi}.
5. Calculate local threshold value T_{li} in threshold window W_{ti}.
6. Calculate absolute difference D between I_i and T_{li}.

7. Assign pixel binary value B_i: use global threshold T_g if D is less than sensitivity s, else use local threshold T_{li}. Set B_i as foreground if I_i is greater or equal to threshold value.
8. Repeat steps 3 to 7 until every pixel P_i has been assigned a binary value B_i.
9. Output binary image M_{out}.

This algorithm is visualized as flowchart in Fig. 1 and Fig. 2 displays image–threshold window–pixel window interaction.

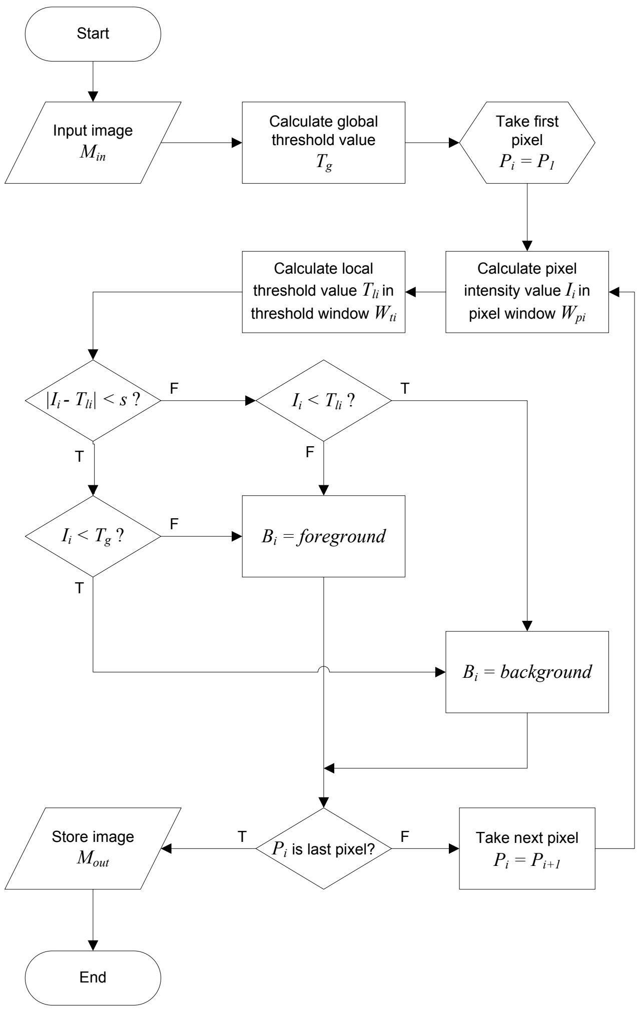

Start
Input image M_{in}
Calculate global threshold value T_g
Take first pixel $P_i = P_1$
Calculate local threshold value T_{li} in threshold window W_{ti}
Calculate pixel intensity value I_i in pixel window W_{pi}
$|I_i - T_{li}| < s$?
$I_i < T_{li}$?
$I_i < T_g$?
$B_i = foreground$
$B_i = background$
Store image M_{out}
P_i is last pixel?
Take next pixel $P_i = P_{i+1}$
End
T
F

Fig. 1. Adaptive threshold algorithm flowchart

Our method is based on multiple average value calculations. Algorithm proposed in this paper has three tuneable values: w_p - length of pixel window side, w_t - length of threshold window side, s - sensitivity value.

There are two windows of adjustable sizes. Average value in smaller window (*pixel window* W_p) defines its centre pixel brightness value I. It is expressed as arithmetic mean of all pixel intensities in the window:

$$I = \frac{1}{n}\sum_{i=0}^{n} P_i \,, \tag{2}$$

where $n{=}w_p^2$ – pixel count in small window, P_i – i-th pixel's intensity value in window W_p.

Average value in larger window (*threshold window* W_t) defines local threshold value T_l which is also calculated as arithmetic mean of pixel intensities in larger window

$$T_l = \frac{1}{m}\sum_{j=0}^{m} P_j \,, \tag{3}$$

where $m{=}w_t^2$ denotes pixel count in threshold window W_t.

Sensitivity s represents lowest reliable value of absolute difference between pixel intensity value I and local threshold value T_l. If value $|I - T_l|$ is higher or equal than s then we have a reliable calculation of foreground or background pixel (2). If absolute difference is lower than s then threshold value of such pixel is calculated using expression in third line of equation (2):

$$B = \begin{cases} 0, \text{when } I < T_l \text{ and} \left|I - T_l\right| \geq s, \\ 1, \text{when } I \geq T_l \text{ and} \left|I - T_l\right| \geq s\,, \\ B_{gt}, \text{when} \left|I - T_l\right| < s\,, \end{cases} \tag{4}$$

where B is final pixel binary value after threshold operation, B_{gt} indicates binary value assignment by comparison to arithmetic mean of all pixel intensity values of complete image T_g:

$$B_{gt} = \begin{cases} 0, \text{when } I < T_g\,, \\ 1, \text{when } I \geq T_g\,. \end{cases} \tag{5}$$

Our proposed threshold calculation approach has several advantages:

- it is more noise immune than simple bi-level threshold function,
- segmented binary features have smooth edges,
- some irrelevant features get eliminated depending on W_p and W_t values.

Some of weak points of the algorithm:

- redundant features might emerge, if image contains areas of continuous intensity of size larger than threshold window size W_t,
- calculation complexity of $O(n^2)$, because intensity averages for pixel and threshold windows must be calculation for every pixel in image.

Optimization

We were able to solve calculation complexity problem by applying an optimization based on sum table, which is described in paper [13]. We have adopted sum table approach to pre-calculate image intensity and use it during average intensity calculation in windows W_t and W_p. It works by initially parsing every pixel in the image and storing integral sum (running sum) which depends on upper and left neighbour pixels

$$\begin{aligned} s(x,y) = P(x,y) + s(x-1,y) + \\ + s(x,y-1) - s(x-1,y-1). \end{aligned} \tag{6}$$

In equation (4) $P(x,y)$ is the intensity level of pixel at coordinates x and y and $s(x,y)$ is the integral sum for that pixel. Having integral sum table, total intensity I_{total} for any window of size w in the image can be calculated using the following equation:

$$\begin{aligned} I_{total}(x,y) = {} & s(x+w-1, y+w-1) - \\ & - s(x-1, y+w-1) - \\ & - s(x+w-1, y-1) + \\ & + s(x-1, y-1). \end{aligned} \tag{7}$$

This reduces computational complexity from $O(n^2)$ to $O(n)$ which means calculation time now increases linearly with image size increase instead of quadratic increase.

Innovations

Our algorithm introduces several innovations to classic adaptive threshold approach:

- It uses local average intensity value not only for threshold calculation but also for pixel value calculation, which provides smoother edge lines to segmented objects.
- Sensitivity parameter s is the solution for continuous intensity regions larger that local threshold window, which otherwise would introduce unnecessary objects into segmented image.
- It introduces speed optimization based on integral tables [13] which enables fast adaptive filtering applications.

Experimental Testing

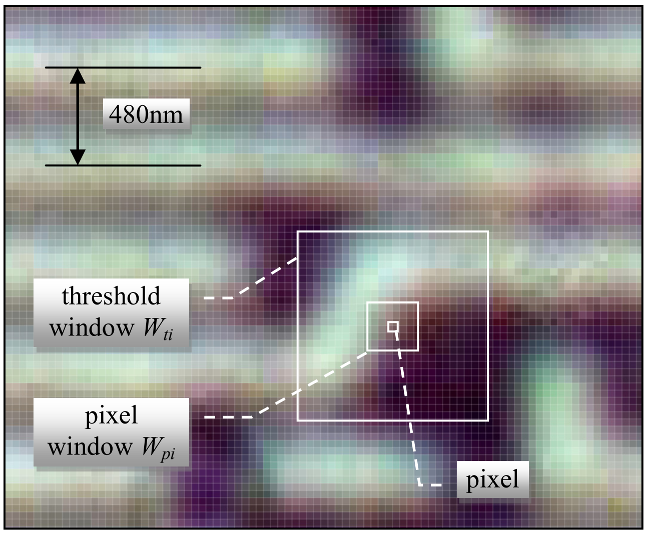


Fig. 2. Pixel-level image fragment

Our adaptive threshold algorithm was experimentally tested on Metal1 and Metal3 IC layer images obtained using optical microscope at 4000 times magnification. Image resolution is 29.99 *pixels per micrometer* which means physical size of each image is approximately 42 μm × 34 μm. The IC was manufactured using CMOS 0.18 μm node, size of smallest features in the images is 240 nm (metal wire lines) or 7.2 pixels. Such feature sizes approach physical capabilities of optical microscopy for diffraction effect reason. This is the cause of low image quality which manifests as blurry objects and variable brightness areas. Our threshold algorithm takes these image properties into consideration and has means to extract most usable binary object data.

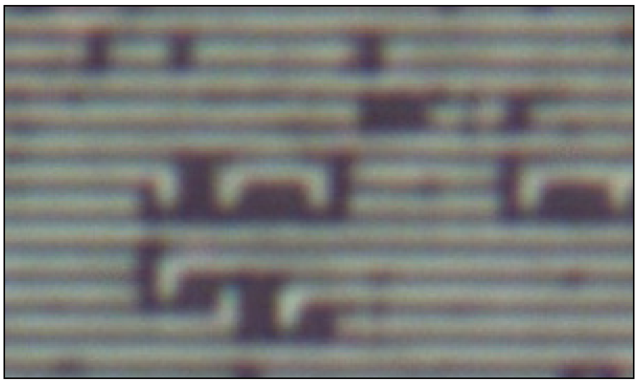
Unprocessed (direct from camera) images were used as input to threshold functions. Before statistically analysing the results, all images were filtered to discard objects smaller than minimal size of IC manufacturing technology node, which is equals 240 nm and approximately to 7.2 pixels, respectively.

The following images show filtering results of our algorithm compared to global t threshold algorithms (Fig. 3).

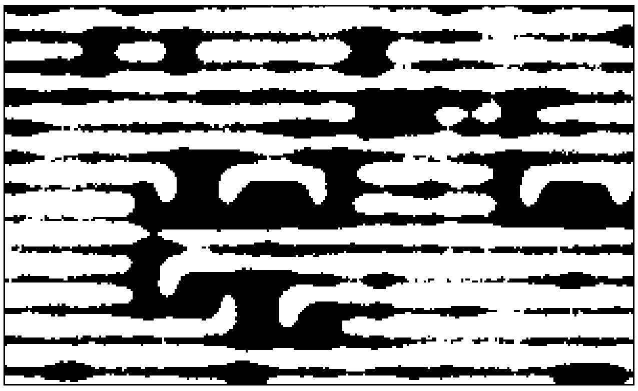
To measure how our adaptive algorithm performs compared to other algorithms, we analysed resulting binary images to calculate how close quality of each of them comes to a perfect segmentation results, obtained by human operator. The following criteria were evaluated and compared:

- Statistical evaluation of object edge pixel distances from human-segmented object edges pixels.
- Quantity of object pixels in segmented images, which exceed maximum allowed distance from perfectly segmented object edges. The distance, designated by d_{max} is set to ½ size of minimum feature size in images. d_{max} = 120 nm ≈ 3.6 pixels.

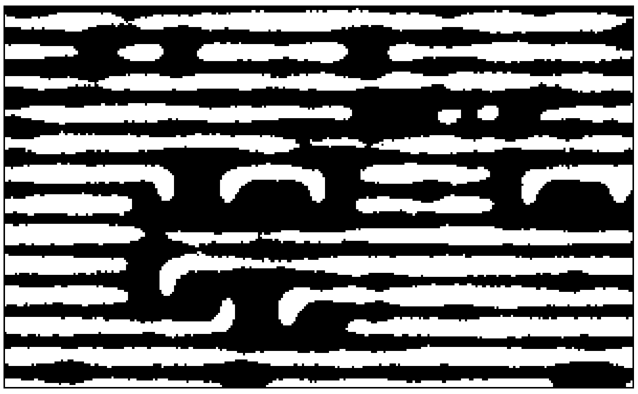
Since iterative threshold algorithm gave unmistakably worst results, it was removed from comparison and only adaptive threshold and manual bilevel threshold were quantifiably compared to the perfect segmentation results.



(a) Original image



(b) Iterative threshold image


(c) Manual threshold image

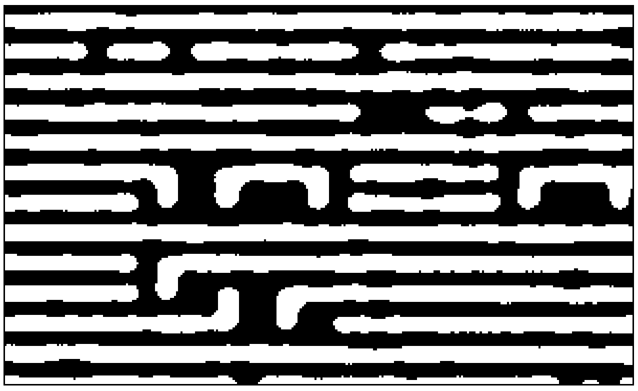

(d) Adaptive threshold image

Fig. 3. Experimental results of second fragment thresholding

In Fig. 4 we can see part of image, where object contour obtained from perfect segmentation is overlapped with contour from our adaptive threshold segmentation. In both sample image segmentation results, there were only limited number of locations where fractures occurred in objects and no cases of missing objects or maximum allowed pixel distance error exceeded.

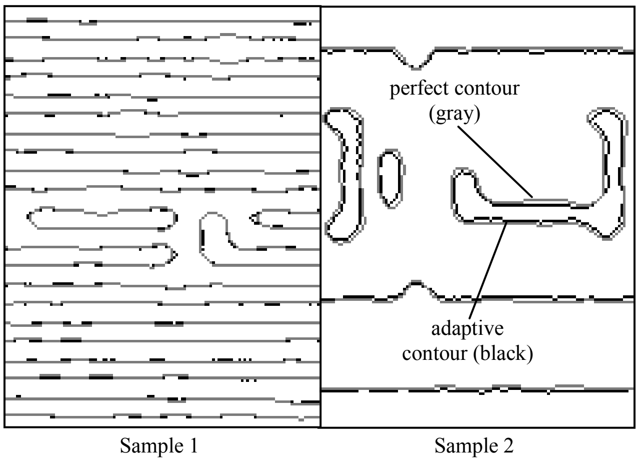


Fig. 4. Object contour differences between perfect segmentation and proposed threshold algorithm

In Fig. 5 we can see overlapped contour comparison between best manual bilevel threshold result and perfect segmentation result. In two sample images many erroneous object fractures and junctions are visible, along with redundant objects and maximum allowed pixel distance errors.

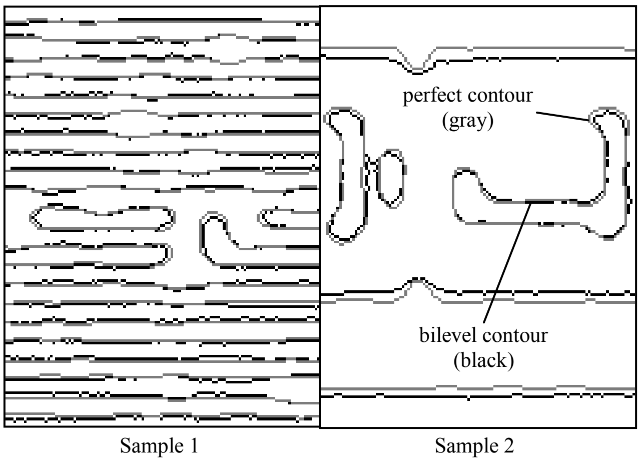


Fig. 5. Object contour differences between perfect segmentation and bilevel threshold algorithm

Summary

Table 1 gives summary results of statistical comparison of contour pixel distribution between perfectly segmented image edges and edges obtained from foreground objects of other two segmentation algorithm results.

Table 1. Statistical figures for distance between ideally segmented object contour pixels and object, obtained using adaptive and bilevel segmentation algorithms, pixels

Sample / algorithm	Mean value, pixels	Maximum value, pixels	Standard deviation
S1 adaptive	0,18	2,23	0,38
S1 bilevel	1,20	29,27	1,81
S2 adaptive	1,18	12,64	0,60
S2 bilevel	2,75	31,21	3,09

All the statistical parameters show that objects obtained using our adaptive segmentation algorithm are reliably better in comparison to bilevel thresholded objects statistical data. Sample 1 is generally easier to segment than sample 2. Its object distribution and brightness is more even compared to sample 2. Both samples represent a common case of IC layer images. Sample 1 ideal contour image contains 168983 edge pixels and sample 2 image contains 60884 edge pixels.

To compare segmentation accuracy, difference between each ideal contour pixel and corresponding edge pixel in segmented image was calculated. In our case corresponding edge pixels are the closest ones to the ideal contour pixels.

Mean values in Table 1 represent how far on average segmented edge pixels were from ideal contour pixels. Maximum values represent worst cases in each sample and segmentation algorithm. Standard deviation actually shows how consistent are edge pixel distances from ideal contour when using different segmentation methods. Our proposed adaptive segmentation algorithm gives standard deviation 5 times smaller than ordinary bilevel segmentation algorithm. This clearly shows that our algorithm gives object edges much more similar and close to ideal segmented object edges.

Calculation results visible in the Table 1 support the visual observation results of Fig. 6 and Fig. 7.

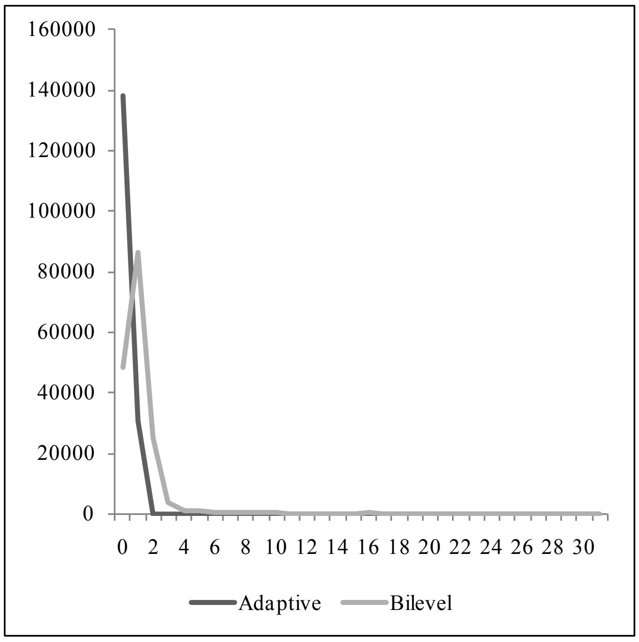


Fig. 6. Adaptive threshold algorithm performance compared to manual bilevel threshold on sample 1 image

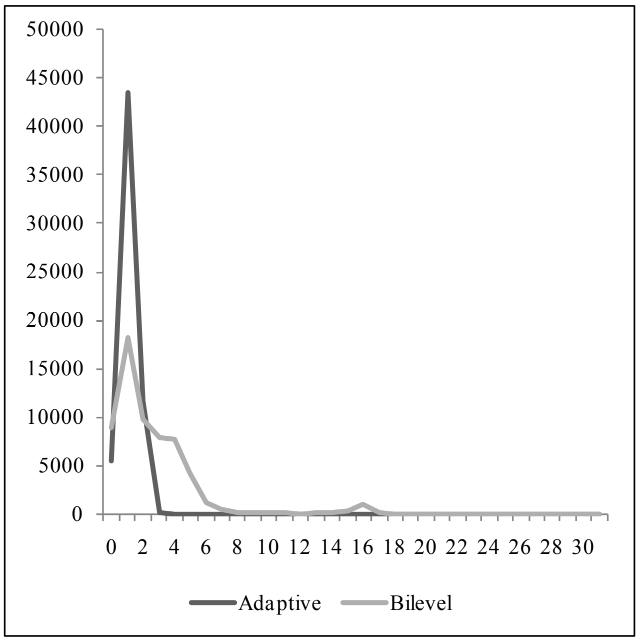


Fig. 7. Manual bilevel threshold algorithm performance compared to manual bilevel threshold on sample 2 image

In fact, more than 99% of all contour pixels are located closer than 3.6 pixels from perfect contour edge when using adaptive threshold. This means that our algorithm is suitable for practical application because its precision is high enough and error rate in segmented images is very low. High precision in combination with low calculation complexity due to sum table application allowed us to perform segmentation of thousands of images with very successful results.

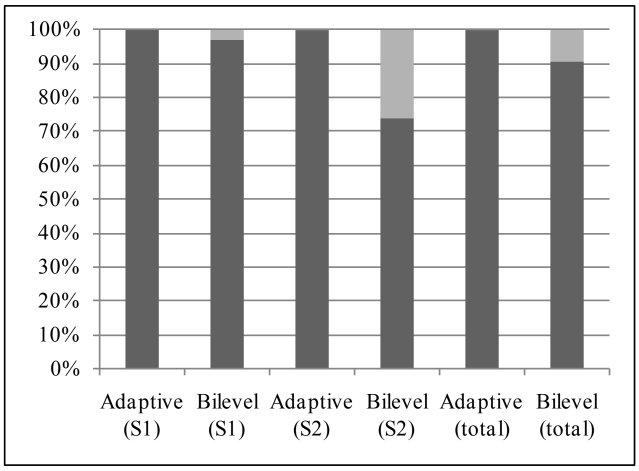


Fig. 8. Contour pixel distribution within maximum allowed distance

Conclusions

Adaptive segmentation algorithm was proposed, implemented and experimentally tested. The algorithm is suitable for image segmentation of integrated circuits layers under varying contrast and lighting conditions and varying IC sample quality. Our proposed adaptive segmentation algorithm gives standard deviation 5 times smaller than ordinary bilevel segmentation algorithm. The algorithm was optimized for computation speed which in combination with high segmentation precision makes it suitable for practical applications.

References

1. **Acharya T., Ray A.K.** Image Processing Principles and Applications. – Wiley–Interscience, 2005.
2. **Paulinas M., Miniotas D., Meilūnas M., Ušinskas A**. An Algorithm for Segmentation of Blood Vessels in Images // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 3(83). – P. 25–28.
3. **Otsu N.** A Threshold Selection Method From Gray Level Histograms // IEEE Transaction on System, Man, and Cyber, 1979. – Vol. 9. – P. 62–66.
4. **Kapur J. N., Sahoo P. K., Wong A. K. C.** A New Method of Gray Level Picture Thresholding Using the Entropy of the Histogram // Computer Vision, Graphics & Image Processing, 1985. – Vol. 29. – P. 273–285.
5. **Brink A. D., Pendcock N. E.** Minimum Cross–Entropy Threshold Selection // Pattern Recognition. – 1996. – Vol. 29. – P. 179–188.
6. **Abutaleb A. S.** Automatic Thresholding of Gray Level Pictures Using Two–Dimensional Entropy // Computer Vision, Graphics & Image Processing, 1989. – Vol. 47. – P. 22–32.
7. **Masalskis G., Navickas R.** Reverse Engineering of CMOS Integrated Circuits // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 8(88). – P. 25–28.
8. **Yanowitz S. D., Bruckstein A. M.** A new method for image segmentation // in Proc. 9th Int. Conf. on Pattern Recognition, 1988. – Vol. 1. – P. 270–275.
9. **Feigin M., Sochen N.** Segmentation and Denoising via an Adaptive Threshold Mumford–Shah–like Functional //

Proceedings of the 17th International Conference on Pattern Recognition, 2004. – Vol. 2. – P. 98–101.
10. **Sun–Gu Sun, Dong–Min Kwak, Won Bum Jang, Do–Jong Kim.** Small Target Detection Using Center–Surround Difference with Locally Adaptive Threshold // Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005. – P. 402–407.
11. **Blayvas I., Bruckstein A., Kimmel R.** Efficient computation of adaptive threshold surfaces for image binarization // Computer Vision and Pattern Recognition, 2001. – Vol. 1. – P. I–737–I–742.
12. **Tabbone S., Wendling L.** Binarization of color images from an adaptation of possibilistic c–means algorithm // Proceedings of the 17th International Conference on Pattern Recognition, 2004. – Vol. 1. – P. 704–707.
13. **Lewis J. P.** Fast Normalized Cross–Correlation. – 1995. Online: www.idiom.com/~zilla/Work/nvisionInterface/nip.pdf.

Received 2010 06 24

G. Masalskis, R. Navickas. Time-Efficient Adaptive Segmentation Algorithm forIC Layers Images // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 10(106). – P. 133–138.

In this paper we introduce adaptive threshold algorithm for grayscale images which substantially enhances object extraction precision while maintaining robust calculation performance even for very large images. The algorithm was developed for feature analysis in low quality integrated circuit images obtained using optical microscopy equipment. Our proposed adaptive segmentation algorithm gives standard deviation 5 times smaller than ordinary bi-level segmentation algorithm when comparing segmented object contour differences to manually segmented image. This technique has been applied in industry. Ill. 8, bibl. 13, tabl. 1 (in English; abstracts in English and Lithuanian).

G. Masalskis, R. Navickas. Spartus adaptyvaus segmentavimo algoritmas integrinių grandynų vaizdams // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2010. – Nr. 10(106). – P. 133–138.

Šioje publikacijoje aprašytas adaptyvaus slenkstinio filtro algoritmas, skirtas prastos kokybės skaitmeniniams integrinių schemų vaizdams, gautiems optiniu mikroskopu, apdoroti ir analizuoti. Pažymėtina jo savybė yra tikslus objektų kontūrų išskyrimas (standartinis kvadratinis nuokrypis buvo apie 5 kartus mažesnis nei taikant dviejų lygių segmentacijos algoritmą) ir spartus skaičiavimo tempas netgi esant labai didelės apimties skaitmeniniams vaizdams. Šis būdas taikomas pramonėje. Il. 8, bibl. 13, lent. 1 (anglų kalba; santraukos anglų ir lietuvių k.).