Semiautomatic Segmentation with Compact Shape Prior

Piali Das
University of Western Ontario
London, Ontario

Olga Veksler
University of Western Ontario
London, Ontario

Vyacheslav Zavadsky
Semiconductor Insight Inc.,
Ottawa, Ontario

Yuri Boykov
University of Western Ontario
London, Ontario

Abstract:

We present a semiautomatic segmentation algorithm, that can segment an object of interest from its background based on a single user selected seed. We are able to obtain reliable and robust segmentation with such low user interaction by assuming that the object to be segmented is of *compact* shape (we define this assumption later). We base our work on the powerful Graph Cut segmentation algorithm of Boykov and Jolly [2]. As additional benefit of incorporating the compact shape prior we are able to bias the graph cuts segmentation framework towards larger objects. It helps to counteract the well known bias of [2] to shorter segmentation boundaries. Segmentation results are quite sensitive to the choice of parameters, and so another contribution of our paper is that we show how to select the parameters automatically. We demonstrate the effectiveness of our method on the challenging industrial application of transistor gate segmentation in an integrated chip, for which it produces highly accurate results in real-time.

Keywords: segmentation, shape prior, graph cuts

1 Introduction

In most applications, purely automatic segmentation is often ambiguous in presence of multiple objects and in absence of strong edges. These problems can be avoided by using semiautomatic or interactive segmentation methods which depend on user guidance. Hence their popularity is increasing in applications in different domains. Segmentation is goal-dependent and thus it is ill-posed in a general set up. However, for a specific domain, if we can make simplifying assumptions then very reliable segmentations can be obtained. The motivation behind our work is to reduce user interaction to the minimum, requiring that the user just chooses the object of interest by clicking anywhere inside. Our goal is to produce an accurate and robust segmentation.

Among the existing methods of segmentation found in the literature, techniques like region growing, split and merge, edge detection [6] are local and, in general, fail to obtain segmentation with global properties. The way to incorporate global properties into segmentation is to formulate an objective function or energy function and optimize it implicitly as in Live wire [4, 10] or explicitly as in active contour (snake) [7, 3], level set [11], normalized cut, [12] and graph cut [2] based methods. However, while formulating an energy function is relatively straightforward, finding a global minima of an energy function is computational prohibitive, in general. Out of the methods mentioned above, only the graph cut [2] energy optimization technique guarantees globally optimal solution for a family of energy functions, and our application falls in this family. An additional benefit of using the framework of [2] is that it easily allows to incorporate both regional and boundary properties of the segment and also provides the option to simplify the user interaction. These advantages make the graph cut [2] based method much more attractive than others.

As segmentation is a subjective problem, we make some assumptions based on the prior knowledge about the data. We also make assumptions about the regional and boundary properties of the segment and fit them into the framework of the algorithm in [2]. The most important assumption that we make is that an object to be segmented is *compact*[1] in

[1] we use the word *compact* here informally, we will explain what we mean by it later.

shape. While this assumption allows us to produce very robust segmentations, it is also our most restrictive assumption, making our algorithm not suitable for segmentation of objects of general shapes. However, there are important applications (industrial and medical) where the objects of interest are approximately compact in shape. Furthermore, we can also handle objects with somewhat more general shapes, specifically the objects that can be divided into several approximately collinear pieces, where each piece is compact in shape. There are several related methods [13, 9, 5] which incorporate shape priors into graph cut based segmentation, however these methods are either too computationally intensive and/or still require too much user interaction.

An important issue with the framework of [2] that we have to handle is that the values of parameters have direct impact on the result produced by the algorithm. Unfortunate choice of parameters can produce unacceptable segmentation results that have to be detected by the user and corrected by possibly considerable amount of user interaction. As we want to reduce the user interaction, we devise a simple but intuitive test to check the quality of the segment automatically. This "quality check" is application dependent. If current segment does not pass the quality check, the parameters are readjusted and the graph cut step is redone with the new parameters. We iterate this process using a search over parameter space until the resulting segment passes the quality check. Thus in our work, we estimate all the important parameters in the algorithm automatically.

A serious difficulty in practice for graph cut based segmentation is its bias towards producing segments with shorter boundaries, if the regional properties are weighted not as heavily as the boundary properties. Our goal is to have a very low input from the user, who just marks one object seed point. Thus we do not have enough samples from the user to construct a reliable model for the color distribution of the object, and we are forced to weight boundary information more heavily than regional information. In our framework, we can easily counteract the bias towards shorter boundaries of [2]. It turns out that due to incorporating *compact* shape prior in the framework of [2], we can introducing a new parameter *bias*, which biases the algorithm towards a larger object segment[2]. We search over a range of values to find the appropriate value of *bias* that produces a segment passing the quality check as discussed above.

[2]Without the compact shape prior, incorporating bias parameter results in an energy function which is not submodular [8], and thus cannot be minimized exactly with graph cuts, see section 3.4

Thus our main contributions to the graph cut segmentation framework of [2] are as follows. We introduce the idea of an application dependent "quality check" which can be effectively used for automatic parameter selection. We introduce compact shape prior in the framework of [2] which lets us deal with the objects of compact shape very robustly. Lastly, due to the shape prior, we are able to introduce a bias parameter which allows us to counteract the shrinking bias of graph cut based segmentation.

We evaluate our approach on a transistor segmentation application for Semiconductor Insights, which is an engineering consultancy company specializing in intellectual property protection and competitive intelligence in the integrated circuit domain. Our segmentation algorithm produces highly accurate results in real-time, and was used to upgrade their manual system to a semi-automatic one.

This paper is organized as follows. In section 2 we review the graph cut based segmentation framework of [2], in section 3 we describe our work, and finally we conclude with experimental results in section 4.

2 Graph Cut Segmentation

In this section we briefly review the graph cut segmentation algorithm in [2].

2.1 Graph Cut

Let $G =< V, E >$ be a graph consisting of a set of vertices V and a set of edges E connecting the vertices. Each edge $e \in E$ in G is assigned a non-negative cost w_e. There are two special vertices called the *terminals* identified as *source*, s and *sink*, t. A cut C is a subset of edges $C \subset E$, which when removed from G partitions V into two disjoint sets S and $T = V - S$ such that $s \in S$ and $t \in T$. The cost of the cut C is just the sum its edge weights:

$$|C| = \sum_{e \in C} w_e.$$

The minimum cut is the cut with minimum cost. Max-flow/Mincut algorithm can be used to obtain the minimum cut. We use a fast implementation of graph cuts described in [1].

2.2 Segmentation Algorithm

In the graph-cut segmentation of [2], the problem of segmenting an object from its background is interpreted as a binary labelling problem, which can be

solved in energy minimisation framework. The labelling corresponding to the minimum energy is chosen as the solution. Let P be the set of all pixels in the image, and let N be the standard 4-connected neighborhood system on P, i.e. N is a set of pixel pairs $\{p, q\}$ where p is immediately to the right, or left, or top, or bottom of q.

For the segmentation problem at hand each pixel of the image has to be assigned a label from the label set $L = \{0, 1\}$, where 0 and 1 represent background and object, respectively. Let $S = \{S_1, \ldots, S_p, \ldots, S_{|P|}\}$ be a binary set that defines a segmentation, where each $S_p \in L$ is the label assigned to pixel p. Thus the set P is partitioned into two subsets, where pixels in one subset are labelled 0 and the ones in the other subset are labelled 1.

The labelling problem for segmentation is formulated in the energy minimization framework with the energy defined in terms of labelling S as:

$$E(S) = \alpha R(S) + B(S). \tag{1}$$

In the energy function above, $R(S)$ is called the *regional* term because it incorporates the regional constraints into the segmentation. Specifically, $R(S)$ measures how well pixels fit into the object or background models under labelling S. It has the following form:

$$R(S) = \sum_{\forall p \in P} R_p(S_p),$$

where $R_p(S_p)$ is the penalty of assigning the label S_p to pixel p. If label S_p is likely for a pixel p, then $R_p(S_p)$ should be small. If label S_p is unlikely for a pixel p, then $R_p(S_p)$ should be large.

The term $B(S)$ in equation (1), is called the *boundary* term because it incorporates the boundary constraints in segmentation. A segmentation boundary occurs whenever two neighboring pixels are assigned different labels. Thus $B(S)$ is defined as a sum over neighboring pixel pairs:

$$B(S) = \sum_{\substack{\{p,q\} \in N \\ p<q}} B_{pq}(S_p, S_q),$$

where $B_{pq}(S_p, S_q)$ describes the penalty for assigning labels S_p and S_q to two adjacent pixels. We use B_{pq} to incorporate our prior knowledge that most nearby pixels tend to have the same label. Thus there is no penalty if neighboring pixels have the same label and a penalty otherwise. Typically, $B_{pq}(S_p, S_q) = w_{pq} \cdot I(S_p \neq S_q)$ where $I(\cdot)$ is an identity function of a boolean argument defined as:

$$I(S_p \neq S_q) = \begin{cases} 1 & \text{if } S_p \neq S_q, \\ 0 & \text{otherwise.} \end{cases}$$

To align the segmentation boundary with intensity edges, w_{pq} is typically chosen to be a non-increasing function of $|I_p - I_q|$, where I_p and I_q are the intensities of pixels p and q respectively.

Note that the term $\alpha \geq 0$ in (1), decides the relative importance of the *regional* and *boundary* terms. The larger the value of α is, the more importance the regional constraints $R(S)$ have compared with the boundary constraints $B(S)$. This parameter is one of the most important parameters in the graph-cut segmentation framework.

In [2] they show how to construct the graph such that the labelling corresponding to the minimum cut is the labelling optimizing the energy in (1).

3 Our Work

The main goal of our work is to reduce the user interaction for segmentation to the minimum. We seek to develop an interface where the user just has to choose the object of interest by clicking inside the object only once. At the same time the segmentation result has to be accurate and robust. We call such segmentation *semi-automatic*. Under the general set-up, the segmentation algorithm in [2] has several issues making it unsuitable for semi-automatic segmentation. We address these issues in our work.

The algorithm in [2] was proposed for a general framework, and when trying to use it for a specific application, there are several problems one typically runs into. In [2] the user has to initially select a few object seeds and a few background seeds. After running the algorithm the user has to inspect the quality of the segment. If required, he/she has to repeatedly add new seeds and rerun the algorithm until an acceptable segmentation is obtained. However the results of the algorithm depend heavily on the choice of parameters for the energy (1). If parameters are far from optimal, the user might have to perform a significant amount of interaction.

While general purpose semi-automatic segmentation is likely to remain an allusive goal in the foreseeable future, an application specific segmentation is a much more tractable problem. One of the main ideas in our paper is that for a specific application, it is not too hard to come up with a goal-dependent measure of segment quality. We produce a relatively simple "quality check" which lets us decide whether segmentation under current parameters in (1) is satisfactory. With this quality check at hand, we can then search over a range of parameters (typically using a variant of binary search) to quickly and automatically find parameters corresponding to a suitable segmentation. While our particular segment "quality check"

was tested on our particular application, the overall idea of segment quality test can be used for automatic parameter selection in any application where a segment quality test is possible to design.

In our particular application, the objects are of *compact* shape (or close to compact shape), we explain what we mean by *compact* in section 3.2. Thus we introduce a compact shape constrain as a hard constraint in our segmentation. Many objects can be approximated by a compact shape, so similar construction can be used in other applications. A major benefit of including the compact shape prior is that the objects of this shape are segmented more robustly and reliably. An additional and very important benefit is that we can include a new parameter in our energy function which incorporates bias to larger objects, as explained in section 3.4. This helps to solve another general issue in graph cut based segmentation of [2], namely its tendency to produce segments of smaller size.

This section is organized as follows. In section 3.1 we discuss the assumptions made by our algorithm, in section 3.2 we explain the *compact* shape prior, in section 3.3 we give the regional term that we use for the energy in (1), in section 3.4 we explain our boundary term and show that our energy function can be minimized exactly with a graph cut, in section 3.5 we discuss shapes more general than compact that our algorithm can handle, and in section 3.6 we give an overview of our algorithm.

3.1 Our Assumptions

In this paper, we make the following assumptions: (a) the intensity variation inside the object of interest is small compared to the strength of the intensity edge on its border; (b) minimum and maximum possible size of the objects to be segmented are known; (c) the objects to be segmented are *compact* in shape or can be divided into approximately collinear parts which are *compact*. The first two assumptions are relatively general, and are usually appropriate for a variety of segmentation applications. The last assumption is the most restrictive, but can be still satisfied by certain application, for example by the application we test our segmentation algorithm on.

3.2 Compact Shape

We assume that the object or the part of the object that is to be segmented is compact in shape. Incorporating the *compact* shape prior in the framework allows to obtain more robust segmentation. We use the word *compact* informally, borrowing the idea from [14]. Consider $Fig.1$. In this figure, the squares represent the image pixels, and the dark gray square represents the seed point that the user has selected. We divide the image into four slightly overlapping quadrants with respect to the seed, as shown in the figure. Let us name these quadrants P_1, P_2, P_3, and P_4. P_1 consists of all pixels above and to the right of the seed, P_2 consists of all the pixels above and to the left of the seed, P_3 consists of all the pixels below and to the left of the seed, and, finally, P_4 consists of all the pixels to the right and below the seed.

An object is defined to be compact if its boundary can be fully traced clockwise using only the edges in each quadrant shown in $Fig.1$. In [14], the word compact is chosen to reflect that for such segments, the perimeter to area ratio tends to be small.

Thus in order for the object segment to be compact, we must prohibit a certain set of label assignments to neighboring pixels. For example, for any neighboring pixels p and q in the first quadrant, we must prohibit assigning 0 to p and 1 to q if p is either to the left or below q. We will use notation $p <_l q$ to denote that pixel p is to the left of q. Similarly, notation $p <_a q$ means pixel p is above pixel q. If l, l' are labels, we will denote the assignment of l to pixel p and l' to pixel q by $(p \leftarrow l, q \leftarrow l')$. Now we can define the set of prohibited assignments:

$$A^p = \begin{array}{l} \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_1 \cup P_4, p <_l q\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_2 \cup P_3, q <_l p\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_1 \cup P_2, q <_a p\} \cup \\ \{p \leftarrow 0, q \leftarrow 1\} | p, q \in P_3 \cup P_4, p <_a q\} \end{array}$$

An object segment is of *compact* shape if no prohibited assignments are made in its segmentation.

3.3 Regional Term

In this section discusses our regional terms in equation (1). For the segmentation algorithm in [2], initially the user has to provide a few object seeds and a few background seeds. We reduce the user interaction to a single object seed selection. We find the background seeds automatically using the maximum object size information. Following [2], for the foreground seed pixel p, we set $R_p(0) = MaxInt$ and $R_p(1) = 0$, where $MaxInt$ is the maximum integer allowed by a programming environment. This insures that the foreground seed pixel will always be assigned to the foreground in the optimal labelling. Similarly, if p is the automatically detected background pixel, we set $R_p(1) = MaxInt$ and $R_p(0) = 0$.

In our application, as the background is unknown, we use a uniform distribution as the background intensity model. Since we have only one pixel marked

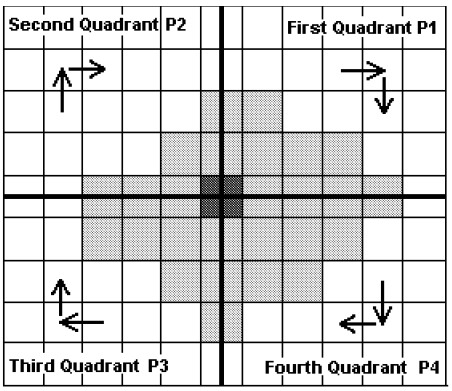

Figure 1: Shows how the segmented is restricted in different quadrants drawn wrt the object seed (marked in dark gray). The Quadrants intersect along the cells through which the bold lines pass.

as the object seed we use the knowledge of minimum object size to collect more data around the seed point to build the intensity histogram. But even then, the data is not sufficient to represent a good intensity distribution of the object. So we use a weighted mixture of uniform distribution and the smoothed normalized histogram. The actual costs $R_p(S_p)$ are taken as negative logarithms of these likelihood models. Therefore for pixel p,

$$R_p(1) = -ln\left(\gamma P_{hist}(I_p) + (1-\gamma)\frac{1}{256}\right),$$

and $R_p(0) = -ln(1/256)$, where we assume that there are 256 gray levels possible in an image, and $P_{hist}(I_p)$ is the likelihood of the object pixel to have intensity I_p according to the distribution modelled by the smoothed histogram.

3.4 Boundary Term

In this section we discuss the boundary term that we use in equation (1). Like in the framework of [2], the boundary terms serve to insure that most nearby pixels are assigned the same label (and thereby the object and the foreground regions form coherent blobs) and also that the boundary between the object and the background lies on image intensity edges. In our framework, in addition to the two purposes above, we use the boundary terms to make sure the object segment follows the compact shape described in section 3.2 and also to incorporate bias to a larger object segment.

Our boundary terms have the following form:

$$B_{pq}(S_p, S_q) = \begin{cases} 0 & \text{if } S_p = S_q \\ w_{pq} & \text{if } \{p \leftarrow S_p, q \leftarrow S_q\} \notin A^p \\ K & \text{if } \{p \leftarrow S_p, q \leftarrow S_q\} \in A^p \end{cases}, \quad (2)$$

where A^p was defined in section 3.2, the constant K is prohibitively large[3], and $w_{pq} = e^{-\frac{(I_p - I_q)^2}{2\sigma^2}} - bias$. The parameter σ in equation (2) affects the segmentation by controlling when intensity difference $|I_p - I_q|$ is large enough to be a good place for a segmentation boundary. When $|I_p - I_q| > \sigma$, the weight w_{pq} is typically small enough to allow a boundary. Thus we compute σ as the average difference of the intensities of two adjacent pixels in a region around the user marked object seed. The size of this region is same as the smallest possible object size which is known to us beforehand.

Parameter *bias* implements bias to a larger segmentation boundary, and it is chosen automatically. When the *bias* increases the boundary cost decreases, though the gradient of the function remains the same. We devise a simple intuitive test that automatically detects the quality of the segment. It requires the average intensity difference between neighboring pixels along the segmentation boundary to be greater than the average absolute intensity difference of neighboring pixels inside the object. We search over a range, to find an appropriate value of *bias* that results in segmentation passing this quality check. Note that by changing the value of *bias* we are changing the values of the boundary terms. This alters the relative importance between the regional and the boundary terms in the energy function (1) . Recall that the parameter α in equation (1) also weights the importance between the regional and the boundary terms. We found that it is enough to search over the *bias* parameter while keeping α fixed.

Now that the energy is fully specified, to minimize it globally and exactly with a graph cut, all that remains is to check that it is submodular, according to [8]. For our energy to be submodular, the binary terms of E(S) have to satisfy the following inequality:

$$B_{pq}(0,0) + B_{pq}(1,1) \leq B_{pq}(1,0) + B_{pq}(0,1). \quad (3)$$

The left hand-side of equation 3 is always 0, and the right hand-side is $w_{pq} + K$, which is always non-negative since K was chosen to be very large.

[3]It is enough to make K equal to the cost of $E(S')$ where S' is any segmentation not containing prohibited assignments.

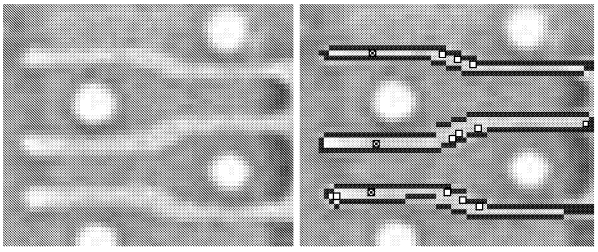

Figure 2: The left image shows the original image and the right image illustrates the extension to more general shapes. The white circles mark is the seed selected by the user, and the white squares show the automatically selected seeds for extension.

3.5 More General Shapes

Our algorithm can handle shapes somewhat more general than compact. Suppose the object can be divided into several approximately collinear pieces, where each piece is of compact shape. If we apply the algorithm above to such an object, we obtain an initial segment of compact shape around the user entered seed. We check if all the edges of the current segment satisfy the criteria for being a "strong edge". If an edge does not pass this test, a new seed point is chosen which lies inside the current segment, close to the weak edge and approximately collinear to the current seed. The last part emphasises our assumption that the object can be divided into approximately collinear compact pieces. Then the graph-cut is run again in the same way as already described, except we reuse the value for the *bias* parameter estimated previously, we found no need to re-estimate it. Thus by repeatedly finding new seeds and running the graph-cut, it is possible to segment the whole object accurately. *Fig.* 2 illustrates the above process. The white circles show the original seed selected by the user, and the white squares show the automatically selected extension seeds.

3.6 Our Algorithm

We build a graph of size greater than the maximum possible size of the object around the object seed. Once the initial segment is obtained, it is likely to contain only a portion of the object that agrees with the compact shape. This initial segment is extended in smaller pieces along the direction where the boundary of the segment does not meet the criteria for being strong boundary. Thus by iteratively running the graph-cut we can segment the whole object regardless of its length. Our algorithm can be summarized in the following steps:

1. User marks a single seed inside the object of interest.
2. Build a graph around the seed.
3. Run graph cut algorithm to obtain the initial segment
4. Test the quality of the segment.

   if it fails: search for *bias* using binary search and go to step 3.

   else: go to step 5
5. Find the part of the boundary which does not meet good boundary criteria

   if found: extend the current segment by using graph cut to obtain the next piece of the object and go to step 5.

   else: terminate.

4 Results

We applied our algorithm to a challenging industrial problem of transistor gate segmentation in the images of integrated chips. It is an important preliminary step for many reverse engineering tasks. To obtain the images, the integrated circuit is de-layered and SEM micro-photographed. *Fig.*3 shows some of the images of ICs provided by Semiconductor Insight. Notice the large variation in the noise level across the images. Hence accurate estimation of σ is a crucial part in our work in order to accommodate the variation. From *Fig.*3, it can also be seen that other challenges are the large variation in contrast and intensity range of the transistor gates. Another challenge is the wide variability in the transistor gates sizes,which range in length from 10 pixels to a few thousands of pixels.

For all the experiments in this section, the parameters were set to the following values: $\alpha = 0.007$, $\gamma = 0.4$. As discussed in section 3.4, the parameter σ is computed from the data collected around the seed using the knowledge of the minimum possible size of the object. Parameter *bias* is chosen automatically, as discussed in section 3.4.

We first compare our results with the results of the algorithm [2]. *Fig.*4(a) shows the result of algorithm [2] using only the boundary terms and the region terms which come from object/background seeds. On including the intensity model for describing the regional property of the segment, the algorithm [2] produces multiple segments with very complex boundaries, most of them being false alarms,

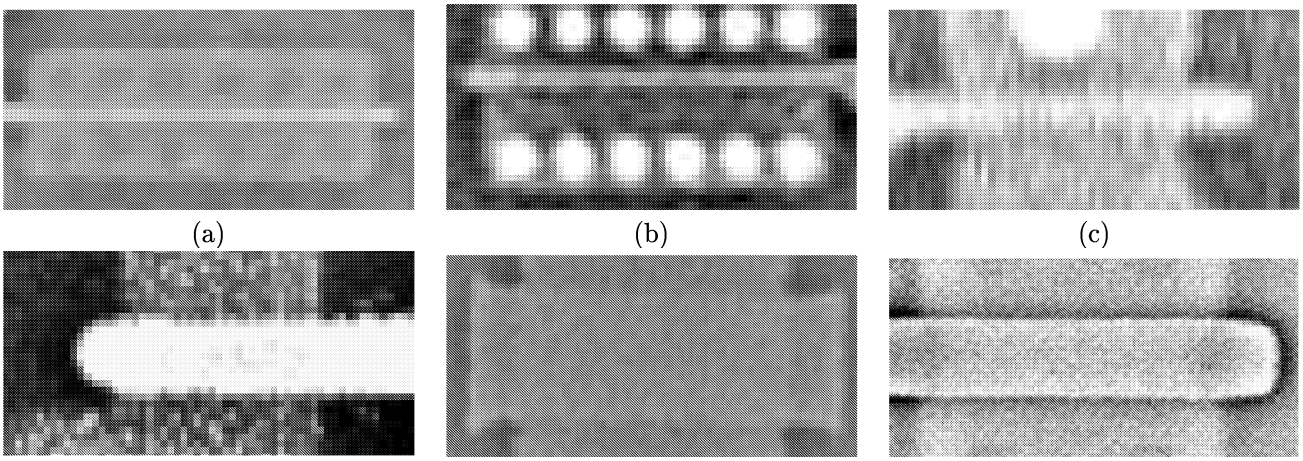

Figure 3: Sample of the images provided by by Semiconductor Insight Inc. showing the variation in image contrast, noise characteristics and the size of the object. Different scales are used for displaying.

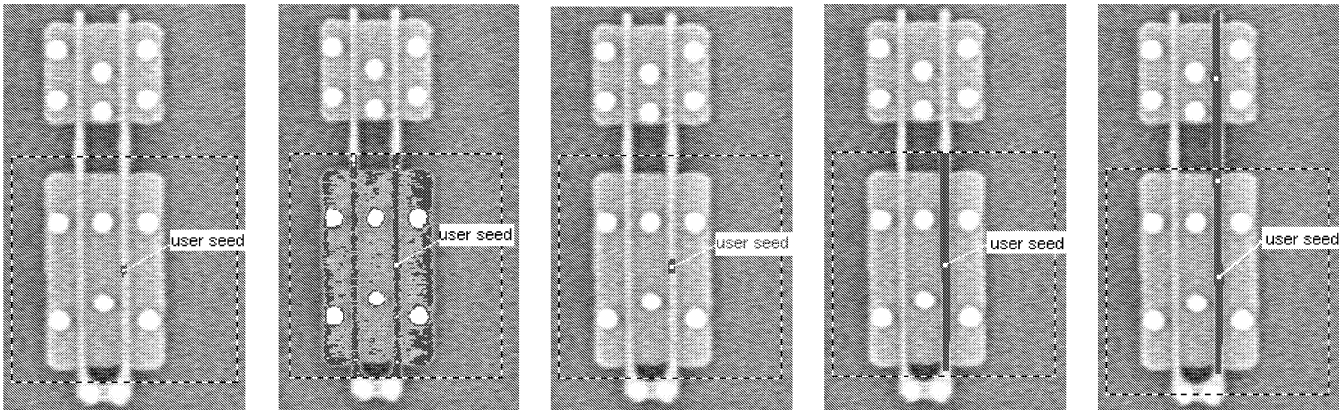

Figure 4: (a)-(b)Segmentation results obtained using algorithm of [2] (a) with the boundary term only (b) with intensity model as regional term along with the boundary term. (c)-(e)Segmentation results obtained with our algorithm (c)Initial segment obtained with $bias = 0$ (d)Initial segment obtained with $bias > 0$ (e)Final segmentation obtained by extending the initial segment obtained in (d). The seeds are marked with white squares, and the initial user entered seed is labelled. The large dotted square shows the maximum allowed segment size.

shown in $Fig.4(b)$. This happens because many pixels not inside the object of interest have intensities close to the intensity of the object seed pixel. On adding the compact shape prior to the framework, the segment obtained is *compact* in shape with smoother boundaries as shown in $Fig.4(c)$.

The parameter *bias* is initially set to 0. After we estimate an appropriate value for *bias*, that is a value which results in an initial segment passing our "quality check", we get the part of the transistor gate shown in $Fig.4(d)$, which corresponds to an acceptable initial segment. It is then extended iteratively to the whole object as shown in $Fig.4(e)$. Every time a segment is extended, a new seed point, marked with a white square in $Fig.4$, is located and σ is re-estimated using the current segment.

$Fig.5$ shows the segmentation results obtained using our algorithm on the images in $Fig.3$. The object seed pixel inside the transistor gate on which the user clicked is marked with white circles and the

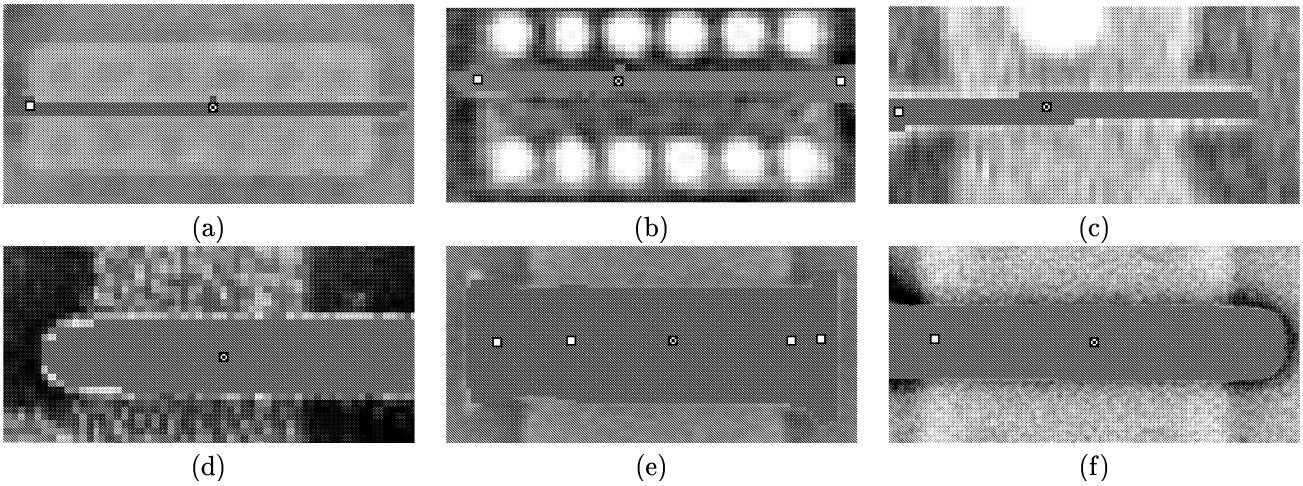

Figure 5: Shows the segmentation results obtained using our algorithm on the images of Fig.3.

automatically detected extension seeds are marked with white squares.

The application developed on the basis of our algorithm runs in real-time and is being used by Semiconductor Insight Inc.

Acknowledgements

We would like thank Stephen Begg and Dale Carlson of Semiconductor Insights for developing interactive application incorporating the method.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transaction on PAMI*, 26(9):1124–1137, September 2004.

[2] Yuri Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation. In *International Conference on Computer Vision(ICCV)*, volume I, pages 105–112, 2001.

[3] L.D. Cohen. On active contour models and ballons. *Computer Vision, Graphics and Image Processing*, 53(2):211–218, 1991.

[4] A. X. Falaco, J.K. Udupa, S. Samarasekara, and S. Sharma. User-steered image segmentation paradigms: Live wire and live lane. In *Graphical Models and Image Processing*, volume 60, pages 233–260, 1998.

[5] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. pages I: 755–762, 2005.

[6] Gonzalez and Woods. *Digital Image Processing.* Prentice Hall, Berlin Heidelberg New York, second edition, 1996.

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2:321–331, 1998.

[8] Vladimir Kolmogorov and Ramin Zabih. What energy function can be minimized via graph cuts? *IEEE Transaction on PAMI*, 26(2):147–159, February 2004.

[9] M.P. Kumar, P.H.S. Torr, and A. Zisserman. Obj cut. pages I: 18–25, 2005.

[10] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. In *International Conference on Computer Vision(ICCV)*, volume II, pages 904–910, 1999.

[11] S. Osher and J.A. Sethian. Fronts propagating with curvature dependent speed: Algorithm based on hamilton jacobi formulations. *JOurnal of Computational Physics*, 79:12–49, 1988.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision(ICCV)*, pages 731–737, 1997.

[13] G. Slabaugh and G. Unal. Graph cuts segmentation using an elliptical shape prior. pages II: 1222–1225, 2005.

[14] O. Veksler. Stereo correspondence with compact windows via minimum ratio cycle. *IEEE Transaction on PAMI*, 24(12):1654–1660, December 2001.