Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Image and Vision Computing xxx (2008) xxx–xxx
image and vision COMPUTING
www.elsevier.com/locate/imavis

Semiautomatic segmentation with compact shape prior ☆

Piali Das [c], Olga Veksler [a,*], Vyacheslav Zavadsky [b], Yuri Boykov [a]

[a] *University of Western Ontario, Computer Science, Middlesex College, 361, London, Ont., Canada N6A 5B7*
[b] *Semiconductor Insight Inc., Ottawa, Ont., Canada*
[c] *Atamai Inc., London, Ont., Canada*

Received 9 February 2007; received in revised form 7 February 2008; accepted 7 February 2008

Abstract

In recent years, interactive methods for segmentation are increasing in popularity due to their success in different domains such as medical image processing, photo editing, etc. We present an interactive segmentation algorithm that can segment an object of interest from its background with minimum guidance from the user, who just has to select a single seed pixel inside the object of interest. Due to minimal requirements from the user, we call our algorithm semiautomatic. To obtain a reliable and robust segmentation with such low user guidance, we have to make several assumptions. Our main assumption is that the object to be segmented is of *compact* shape, or can be approximated by several connected roughly collinear compact pieces. We base our work on the powerful graph cut segmentation algorithm of Boykov and Jolly, which allows straightforward incorporation of the *compact* shape constraint. In order to make the graph cut approach suitable for our semiautomatic framework, we address several well-known issues of graph cut segmentation technique. In particular, we counteract the bias towards shorter segmentation boundaries and develop a method for automatic selection of parameters. We demonstrate the effectiveness of our approach on the challenging industrial application of transistor gate segmentation in images of integrated chips. Our approach produces highly accurate results in real-time.

Keywords: Segmentation; Shape prior; Graph cut; Parameter estimation

1. Introduction

Segmentation is generally defined as the problem of partitioning an image into two or more constituent components, where each component has a short summary representation. This definition is rather vague, because general purpose segmentation is not well defined. Segmentation becomes a much better defined problem when it is developed for a particular application, since then one frequently has a clearer idea of the properties a segmentation should have.

There are mainly three approaches to segmentation: automatic, manual and interactive. Manual segmentation is labor extensive and extremely time consuming. Purely automatic segmentation is very challenging, due to ambiguities in the presence of multiple objects, image noise, weak edges, etc. Ambiguity problems can be eased with user guidance, which is the idea of interactive segmentation methods. Hence, their popularity is increasing in applications in different domains [18,24,5,3,23,1,4].

The motivation behind our work is to reduce interaction to the minimum, asking the user to just choose the object of interest by clicking inside it. We call our approach *semiautomatic* segmentation, to distinguish it from general interactive segmentation, where the user is allowed to provide a potentially unlimited amount of guidance. The name semiautomatic is used to emphasize that our algorithm is only a step away from the automatic segmentation, since only one seed point is required from the user. General

☆ This research was partially supported by NSERC and Semiconductor Insights.
* Corresponding author. Tel.: +1 519 858 4403.
E-mail addresses: pdas@imaging.robarts.ca (P. Das), olga@csd.uwo.ca (O. Veksler), vyacheslavz@semiconductor.com (V. Zavadsky), yuri@csd.uwo.ca (Y. Boykov).

0262-8856/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2008.02.006

interactive segmentation can be quite far from automatic segmentation if lots of input is required from the user in order to achieve satisfactory results.

To produce an accurate and robust segmentation, we have to develop our algorithm with some application in mind, since, as we have already mentioned, general purpose segmentation is an ill-defined problem. We chose to design our algorithm in the context of an interesting industrial application, which requires transistor gates to be segmented from the images of integrated chips.

Over the years, researchers have developed different techniques for segmentation. Some of the primitive methods that have been popular because of their simplicity are region growing, split-and-merge, edge detection and thresholding, see, for example, Gonzalez and Woods [15]. Although these methods and their variants are still widely used, they are not robust as they are based on local decisions. For example, the major problem with region growing is the "leaking" through weak points in the boundary, which is inevitable in most images. Likewise, thresholding fails when the object of interest is not homogeneous. In particular, objects with smoothly varying intensities are split into several segments.

To overcome problems due to local decision strategies, global properties have to be included in the segmentation. Graph theoretic approach to segmentation allows us to do so. Various graph based algorithms have been proposed over the years [33,27,30,5,17,12,32,4,16]. They differ in the way the segmentation is interpreted and in the techniques employed to solve the problem. However, all these methods typically involve two main steps – formulating an objective function and optimizing it.

In some approaches, such as live wire [11,24], a global objective function is implicit. Live wire is a paradigm for segmentation that requires the user to mark a seed on the object boundary. As the user moves the cursor (the free point) close to the object boundary, a curve (livewire) clings to the object boundary and segments the object. The curve position is optimized by finding the shortest path on a certain graph. In this approach considerable amount of interaction may be required in order to find the appropriate segmentation.

Level sets sets [25], normalized cut [27], active contour (snake) evolution [18,7,2], and graph cut [5] formulate the energy function explicitly based on various global properties that the segmentation is expected to have. Unfortunately, for many energy functions that one may wish to formulate, finding their global minimum is computationally prohibitive. Normalized cut computes only an approximation to the global minimum, and in most cases, active contours and level sets compute only a local minimum (a few notable special case exceptions are Cohen and Kimmel [8,21]).

The advantage of the graph cut compared to the above listed methods is that it guarantees a globally optimal solution for a family of energy functions. An additional benefit is that one can easily incorporate both regional and boundary properties of segmentation. Also, unlike most active contour/level set methods, graph cut is not sensitive to the initialization [4]. Furthermore, level sets/snakes would be unsuitable for our semiautomatic approach since they require the user to initialize a contour, not just one point. These advantages make the graph cut method much more attractive than others in achieving our goal.

As segmentation is a subjective problem, we start with the already mentioned application of transistor gate segmentation in the images of integrated chips. We make several assumptions based on the prior knowledge of our data and fit them into the framework of the algorithm in Boykov and Jolly [5]. The most important assumption that we make is that an object to be segmented is *compact*[1] in 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, apart from the transistor gates there are important applications (industrial and medical) where the objects of interest are approximately compact. Furthermore, we can also handle objects with somewhat more general shapes, specifically the objects that can be divided either vertically or horizontally into several approximately collinear pieces, where each piece is compact in shape.

There are several related methods that incorporate shape priors into graph cut segmentation. In Slabaugh and Unal [28] the authors incorporate an elliptical prior in an iterative refinement process. The disadvantages of this approach is that it is iterative and the elliptic shape assumption is overly restrictive for many applications. In Freedman and Zhang [14], the shape prior can be arbitrary, but their method requires a very accurate registration of the assumed shape with the actual location of the object of interest in the image, which is a difficult task in itself. In Kumar et al. [20], they also require fitting of a model of a certain shape to an image, and their method, which uses sampling for estimation of model's parameters, is very computationally intensive.

The use of shape priors for segmentation has been investigated before. Recently there has been a lot of work on using shape priors in level set segmentation, some examples are Leventon et al. [22], Tsai et al. [29], Rousson and Paragios [26], Cremers et al. [10], Cremers et al. [9]. However, level set segmentation is not numerically stable and the solution is prone to getting stuck in a local minimum.

Another issue that we address is the parameter selection. In the framework of Boykov and Jolly [5], the values of parameters have a 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 a considerable amount of interaction. This is not acceptable for our

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

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

semiautomatic approach, since our goal is to reduce user interaction to a single click. If the segmentation algorithm is used for a collection of images that do not exhibit large variability, then it is possible to select the parameters that work well for that type of images beforehand. However, we found that for our application, the images do exhibit considerable variability and selecting fixed parameters that work well for most instances is not possible. For each image, there is an optimal setting of parameters that works well, but estimating that range is difficult. Our solution is to run the segmentation algorithm for a range of parameters and choose the highest quality segmentation. This, of course, requires some way of judging the quality of segmentation. We devise a simple but intuitive test to check the quality of the segment automatically. This "quality check" is application dependent. If the 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 of the algorithm automatically.

If we could directly incorporate our"quality check" into the energy function, then we would not have to search over a range of parameters but could compute the best quality segment in one step. Unfortunately we cannot incorporate our quality check into the energy function in such a way that it still can be minimized with a graph cut.

When the user provides many seed points, or when an accurate color model of the object of interest is known, the regional properties of the object can be relied on, and are included in the graph cut segmentation with a large weight. 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. In this case we have to allow the object to deviate from the unreliable color model, and therefore the regional terms are given a smaller weight (the smaller the weight of the regional terms, the more is the object allowed to deviate from the color model). When regional terms have smaller weight, boundary terms become relatively more important. It makes sense intuitively, since if there is no reliable color model, we must rely more on the fact that we expect the object boundary to aligns with intensity edges in the image. A serious difficulty in graph cut segmentation in the case when regional terms have a small weight is that there is a bias towards producing segments with shorter boundaries. In our framework, we can easily counteract this bias. It turns out that due to incorporating *compact* shape prior in the graph cut framework, we can introduce a new parameter *bias*, which biases the algorithm towards a larger object segment.[2] The *bias* is exactly the parameter for which we search over a range of values to find the segmentation that passes the quality check mentioned above.

Thus our main contributions to the graph cut segmentation framework of Boykov and Jolly [5] are as follows. We introduce the idea of an application dependent "quality check" which can be effectively used for automatic parameter selection. We introduce the compact shape prior, 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 the graph cut 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,[3] and was used to upgrade their manual system to a semiautomatic one.

This paper is organized as follows. In Section 2, we review the graph cut segmentation framework of Boykov and Jolly [5], in Section 3 we describe our work, in Section 4, we present our experimental results and we finally conclude with a discussion in Section 5.

2. Graph cut segmentation

In this section we briefly review the graph cut segmentation algorithm in Boykov and Jolly [5].

2.1. Graph cut

Let $G = \langle V, E \rangle$ 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 *terminals* identified as the *source*, s and the *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 the smallest cost. The max-flow/mincut algorithm of Ford and Fulkerson [13] can be used to obtain the minimum cut. We use the max-flow algorithm developed by Boykov and Kolmogorov [6], which was designed specifically for computer vision applications and has the best performance in practice.

2.2. Segmentation algorithm

In Boykov and Jolly [5], the problem of segmenting an object from its background is interpreted as a binary label-

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

[3] The system is real time in the sense that the user does not have to wait more than a couple of seconds after he/she places a seed in the image of the transistor gate to be segmented.

ing problem, which can be solved in energy minimization framework. The labeling 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, that is N is a set of pixel pairs $\{p, q\}$ where p is either immediately to the right, or left, or top, or bottom of q.

Each pixel in the image has to be assigned a label from the label set $L = \{0, 1\}$, where 0 and 1 represent the background and the 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 labeled 0 and the ones in the other subset are labeled 1.

The energy function has the following form:

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

In Eq. (1), $R(S)$ 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 labeling S. It has the following form:

$$R(S) = \sum_{\forall p \in P} R_p(S_p), \tag{2}$$

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

$$B(S) = \sum_{\substack{\{p,q\} \in N \\ p<q}} B_{pq}(S_p, S_q), \tag{3}$$

where N is the set of all neighboring pixels, and $B_{pq}(S_p, S_q)$ describes the penalty for assigning labels S_p and S_q to two neighboring pixels. The term B_{pq} is used to incorporate the 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, $B_{pq}(S_p, S_q) = w_{pq} \cdot T(S_p \neq S_q)$ where $T(\cdot)$ is an identity function of a boolean argument defined as:

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

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 $\alpha \geqslant 0$ 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 $B(S)$. Larger values of α result in a segmentation which obeys the regional model more. Smaller values of α result in a segmentation with smaller boundary cost, which usually means shorter boundary length. Therefore, this parameter is one of the most important parameters in the graph cut framework, and the hardest parameter to pick beforehand. Typically different images have different optimal values for parameter α.

In Boykov and Jolly [5], it is shown how to construct the graph such that the labeling corresponding to the minimum cut on that graph is the labeling optimizing the energy in (1).

3. Our work

The goal of our semiautomatic segmentation is accurate and robust segmentation with user interaction restricted to a single click inside the object of interest. The graph cut algorithm [5] has several issues which make its direct use unsuitable for semiautomatic segmentation. We address these issues in our work.

In Boykov and Jolly [5], the user has to initially select a few object and background seeds. After running the algorithm the user has to inspect the quality of the segmentation. If required, he/she has to repeatedly add new seeds and rerun the algorithm until an acceptable segmentation is obtained.[4] Moreover, the results of the algorithm depend heavily on the choice of parameter α for the energy function in Eq. (1). If the choice of α is far from optimal, the user might have to perform a significant amount of interaction.

Application specific semiautomatic segmentation is a more tractable problem than general purpose semiautomatic segmentation. One of our main ideas is that for a specific application, it may not be too hard to come up with a goal-dependent measure of segment quality. We develop a relatively simple "quality check" which lets us decide whether segmentation under current parameters in Eq. (1) is satisfactory. With this quality check at hand, we can then search over a range of parameters to quickly and automatically find the parameter value corresponding to a suitable segmentation. Our particular segment "quality check" was designed for a specific application, but it may be possible to design suitable quality checks for other applications. For example, when it is known that an object has a specific shape, a quality check can be based on the shape of the object segment.

In our particular application, the objects are of compact shape (or close to compact shape), we explain what we mean by compact in Section 3.1. Thus, we introduce a compact shape as a hard constraint in our segmentation. Many

[4] Rerunning the algorithm after the addition of new seeds usually takes much less time than the first run of the algorithm because the flow from the previous iteration can be reused and the max flow program does not start from scratch. However, the time required from the user to enter the new seeds can still be considerable.

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. Weak boundaries, background clutter, image noise are easier to overcome with the use of a shape prior. An additional and very important benefit of using the compact shape prior is that we can include a new parameter in our energy function which incorporates a bias to larger objects, as explained in Section 3.4. This helps to solve another general issue in graph cut segmentation, namely its bias to produce segments of smaller size.

This section is organized as follows. In Section 3.1, we explain the *compact* shape prior, in Section 3.2, we discuss the assumptions made by our algorithm, 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 3.6, we discuss shapes more general than compact that our algorithm can handle, and in Section 3.7, we give an overview of our algorithm.

3.1. Compact shape

In this section, we define the compact shape precisely. As we have already mentioned, incorporating a shape prior helps to achieve a more robust segmentation, because all shapes inconsistent with the assumed shape are ignored. This results in an increased robustness to weak boundaries, noise, and clutter. However, incorporating a shape prior within graph cut framework is a difficult task, we are aware of only three previous approaches: Slabaugh and Unal [28], Freedman and Zhang [14], Kumar et al. [20], their disadvantages have been discussed in Section 1.

We develop a shape prior which can be incorporated in the graph cut framework directly, without the need for iterative optimization or registration. We call our shape prior *compact*, borrowing the idea from Veksler [31]. The word compact is used informally. In Veksler [31], they chose the word compact to reflect the fact that for compact shapes, the perimeter to area ratio tends to be small. Intuitively, this shape prior encourages objects with boundaries that are relatively simple. Our shape prior is especially appropriate for industrial parts, and includes rectangles and ellipses as a special case.

We now formally define our shape prior. 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. Quadrant P_1 consists of all pixels above and to the right of the seed, including the seed. Quadrant P_2 consists of all the pixels above and to the left of the seed, including the seed. Notice that quadrants P_1 and P_2 have in common all pixels exactly above the seed, including the seed. Similarly, 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. We say that an object is compact if its boundary can be fully traced using only the edges in each quadrant shown in Fig. 1. Intuitively, in each quadrant, the boundary of the object is allowed to follow along only two out of four possible direction. This implies that the boundary in each quadrant is relatively simple and short.

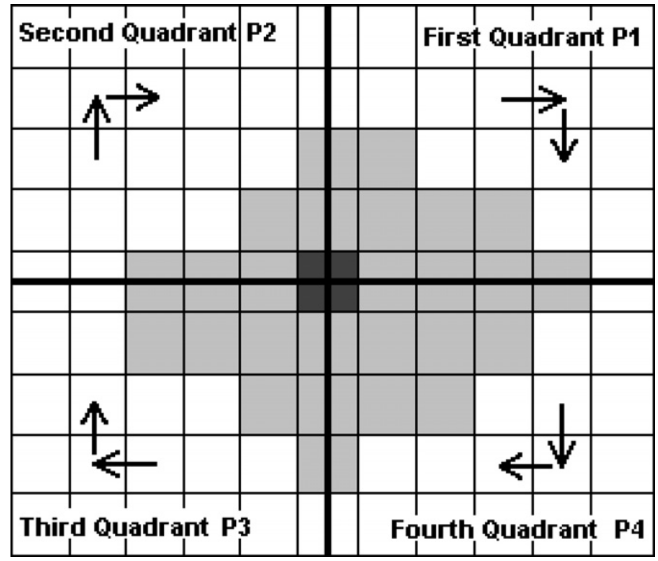

Fig. 1. This figure shows how the object segment is restricted in different quadrants drawn with respect to the object seed (marked in dark gray). The quadrants intersect along the pixels through which the bold lines pass. The segment in light gray color shows an example of a compact shape.

In graph cut framework, 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.[5] We will use notation $p<_l q$ to denote that pixel p is to the left of q. Similarly, notation $p<_a q$ means that 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:

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

We say that an object segment is of *compact* shape if no prohibited assignments are made in its segmentation.

Our definition of a compact shape might sound similar to that of a convex shape, but these two types of shapes are actually quite different. The classes of compact and convex shapes overlap but neither class contains the other. There are convex shapes which are compact, for example

[5] Recall that we use a 4-connected neighborhood system.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

the rectangular object in Fig. 2(a). The shape of the object in Fig. 2(b) on the other hand is not convex but it is compact. The object in Fig. 2(c) is an example of an object which is neither compact nor convex. The object in Fig. 2(d) is convex but not compact.

A weakness of the compact shape prior is that it is not rotationally invariant, since the definition relies on the vertical and horizontal axes. Suppose an object is compact with respect to the vertical and horizontal axes rotated through an angle θ. If we can compute θ (for example, if the object is rectangular but rotated by angle θ), then we can use our algorithm by defining the compactness of the object with respect to the calculated axis.

Another weakness of the compact shape prior is that it is defined with respect to the seed location. Depending on where the user clicks, the object may or may not be compact. We have noticed that users tend to click in the center of the object (or can be specifically instructed to click in the center of the object), therefore we make an implicit assumption here that the shape of interest is compact with respect to its center. Notice that some common shapes, such as rectangles and ellipses, are compact with respect to any seed location.

3.2. Our assumptions

In this paper, we make the following assumptions: (a) the average magnitude of the gradient along the boundary of the object of interest is larger than the average magnitude of gradient among pixels inside the object. In other words, on average, the intensity difference between pairs of pixels both of which lie inside the object is smaller than the intensity difference between pairs of pixels of which one is inside the object and the other is outside the object; (b) the minimum and maximum object sizes are known; (c) the objects to be segmented are *compact* in shape or can be divided either vertically or horizontally into approximately collinear compact parts. The first assumption is often satisfied in practice, since an object of interest frequently has a boundary corresponding to a strong intensity edge. The minimum/maximum size of the object can frequently be determined for a specific application. The last assumption is the most restrictive, but can still be satisfied by certain applications, for example by the application we test our segmentation algorithm on.

3.3. Regional term

In this section, we discuss the regional term that we use in Eq. (1). In the segmentation algorithm of Boykov and Jolly [5], initially the user has to provide a few object and background seeds. We only have one object seed provided by the user, therefore we find the background seeds automatically using the maximum object size information. 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 the programming environment. This insures that the foreground seed pixel will always be assigned to the foreground in the optimal labeling. Similarly, if p is the automatically detected background seed pixel, we set $R_p(1) = MaxInt$ and $R_p(0) = 0$.

Since the background is unknown in our application, we use a uniform distribution as the background intensity model, that is the probability of each intensity is 1/256, given that there are 256 intensity levels in the images. For the object, we do have one but only one pixel marked as the object seed. We use the knowledge of the minimum object size to collect more data around the seed point to

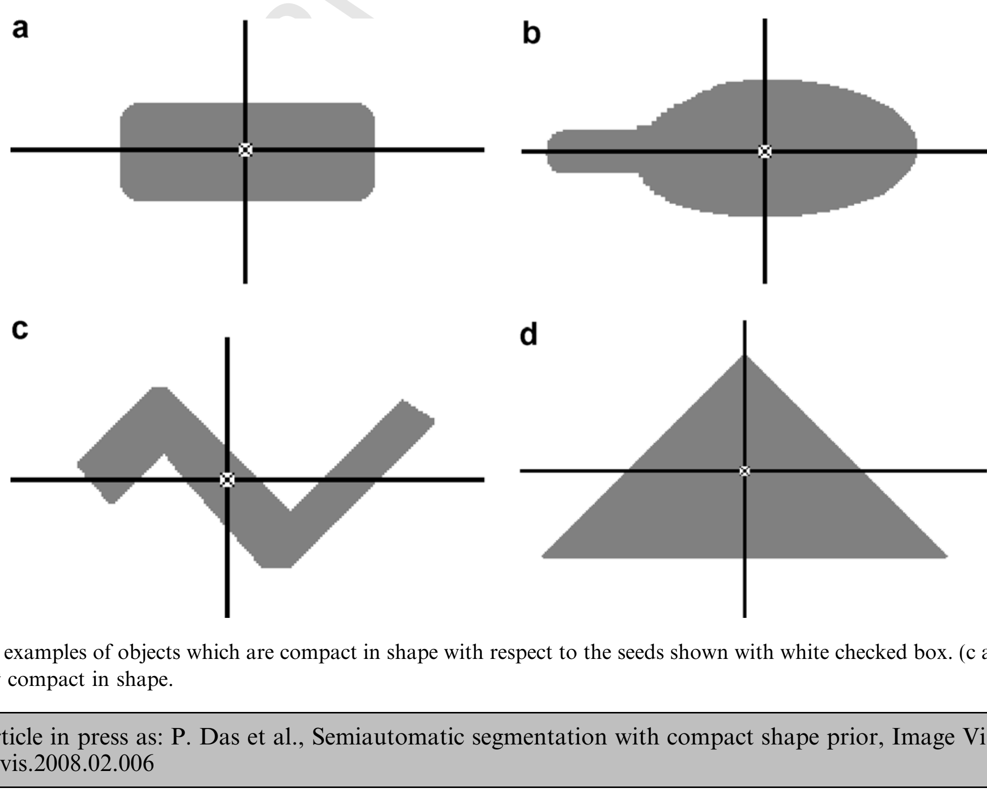

Fig. 2. (a and b) Are examples of objects which are compact in shape with respect to the seeds shown with white checked box. (c and d) Are examples of objects which are *not* compact in shape.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

build the intensity histogram. Our assumption here is that the user clicks roughly in the center of the object. The user can be instructed to click close to the center, but we have noticed that in many cases users will intuitively prefer to click close to the center. In case the click was not close to the center, the object histogram maybe inaccurate if the object size is actually the minimum size. However, this case is not very frequent, and as we will see below, we do not overly rely on the object histogram, so the object can still be segmented accurately in most cases.

Even after we collect more data around the object seed, we do not have a sufficient amount of data to faithfully model intensity distribution of the object. Therefore we use a weighted mixture of a 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,

$$R_p(1) = -ln\left(\gamma P_{hist}(I_p) + (1-\gamma)\frac{1}{256}\right), \quad (4)$$

and

$$R_p(0) = -ln(1/256), \quad (5)$$

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 modeled by the smoothed histogram. We smooth the histogram with a Gaussian with $\sigma = 2$ to avoid the problems due to sparse sampling. Notice that adding the uniform model to the histogram-based model in Eq. (4) makes the regional terms more robust. We know that our histogram based model is not very accurate. By adding to it a uniform model, we make sure that the penalty for an intensity that is not present in the histogram is not so large as to prohibit a pixel with this intensity to be a part of the foreground.

3.4. Boundary term

In this section, we discuss the boundary term that we use in Eq. (1). Like the framework of Boykov and Jolly [5], the boundary term serves 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 the intensity edges. In addition to the two purposes above, we use the boundary term to make sure the object segment follows the compact shape described in Section 3.1 and also to incorporate a bias to a larger object segment.

Our boundary terms have the following form:

$$B_{pq}(S_p, S_q) = \begin{cases} 0 & \text{if} S_p = S_q \\ w_{pq} & \text{if}(p \leftarrow S_p, q \leftarrow S_q) \notin A^*, \\ K & \text{if}(p \leftarrow S_p, q \leftarrow S_q) \in A^* \end{cases} \quad (6)$$

where A^* was defined in Section 3.1, the constant K is large enough so that any assignment in A^* is prohibitively expensive,[6] and

$$w_{pq} = e^{-\frac{(I_p - I_q)^2}{2\sigma^2}} - bias. \quad (7)$$

The parameter σ in Eq. (6) 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* in Eq. (7) implements a 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. The first part of our quality check requires the average intensity difference between pairs of neighboring pixels such that one pixel is in the object segment and the other pixel is in the background segment to be greater than the average absolute intensity difference between pairs of neighboring pixels such that both pixels in the pair are inside the object. This test comes directly from our first assumption in Section 3.2, that is we simply check to see if the segmentation satisfies our first assumption. The second part of our quality check simply makes sure that the object size is within bounds specified by the minimum and maximum object sizes, and this part follows from our second assumption in Section 3.2. For a too small value of *bias* the object segment is very small due to the bias of the graph cut to a small segmentation boundary. In this case, most likely our first assumption will not be satisfied, since the segment consists of a small area around the seed which usually does not have strong intensity edges on its boundary. For a too large value of *bias*, the object segment is too large, larger than specified maximum object size. We search over a range to find an appropriate value of *bias* that results in a segmentation passing this quality check.

Our search algorithm is a very simple iterative search. We search for *bias* in the range [0, 0.8] using a step size of 0.1. We are looking for the smallest value of *bias* in that range such that the object passes the quality check described above. That is we start with $bias = 0$ and increment it in steps of 0.1 until the object segment passes the quality check. Occasionally, there is no *bias* value in the allowed range so that the object passes the quality check. In this rare case, we perform segmentation with the smallest value of *bias* which results in an object segment at least

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

passing part of our quality check, namely the object size should be within the minimum–maximum allowed bounds.

Observe that when changing the value of *bias* we are changing the values of the boundary terms. This leads to altering the relative importance between the regional and the boundary terms in the energy function (1). Recall that the parameter α in Eq. (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. We chose a fixed value of α which works well for all the images.

3.5. Regularity of the energy function

In this section, we prove that our energy function can be minimized globally and exactly with a graph cut. We rewrite the energy function in Eq. (1) with the boundary and the regional term defined in the Eq. (3) and Eq. (2), respectively:

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

where $E(S)$ is a function of $|P|$ binary variables, which is a sum of functions of up to 2 binary variables. This energy function can be minimized using a graph cut if it is submodular, that is if the following property is satisfied [19]:

$$B_{pq}(0,0) + B_{pq}(1,1) \leqslant B_{pq}(1,0) + B_{pq}(0,1). \quad (8)$$

According to the definition of $B_{pq}(S_p, S_q)$, the left hand-side of Eq. (8) is always 0, and the right hand-side is always non-negative, even when w_{pq} is negative, since K is chosen to be very large. Thus $E(S)$ is submodular.

3.6. More general shapes

We started with the assumption that the object to be segmented has to be of compact shape. However, we can somewhat relax this assumption, making it possible to segment objects of shapes more general than compact. Suppose the object can be divided either vertically or horizontally into several approximately collinear adjacent pieces, where each piece is of compact shape. If we apply our algorithm above to such an object, we obtain an initial segment of compact shape around the user entered seed, but either vertical or horizontal boundaries of this initial segment do not align with the object boundaries and therefore do not lie on strong intensity edges. We check if all the edges of the current segment satisfy the criteria for being a "strong edge". In our application, we require 85% of the pixels lying on that edge to have intensity difference greater than the standard deviation inside the object. For this test, we use the pixels of the boundary which lie inside the object (as opposed to those lying on the outside of the object boundary). Other criteria can be also used, of course. If an edge does not pass the "strong edge" test, a new seed point is chosen which lies inside the current segment, at the center of the weak edge but slightly inside the current segment (to be precise, two pixels inside). The last part emphasizes 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 in this section, except we reuse the value for the *bias* parameter estimated at the previous step, we found that there is no need to re-estimate it. Experiments show that the value of *bias* parameter, if re-estimated for each extension piece, is so close to the value of *bias* inside the first piece, that almost no difference in segmentation results is observed. Thus by repeatedly finding the new seed and running the graph-cut algorithm, it is possible to segment the whole object accurately. Fig. 3 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.

This approach is especially helpful for our semiautomatic segmentation, since information about the exact size of the object is not provided. We can segment the objects in smaller pieces, saving the computational time. In addition, we can segment thin and long objects which would be

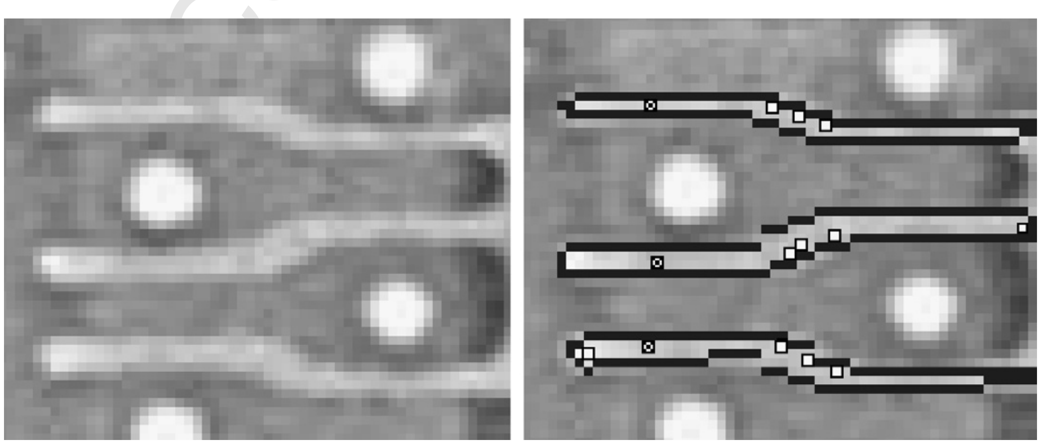

Fig. 3. On the left we show the original image and on the right we illustrates the extension to more general shapes. The white circles mark is the seed selected by the user, and the white squares show the automatically selected seeds for extension.

otherwise impossible using the basic graph cut segmentation algorithm of Boykov and Jolly [5] due to its bias towards shorter boundary.

The basic step of our algorithm is to segment a piece of the object which is compact with relation to the user provided or automatically detected seed. The exact seed placement determines the size of the first piece. If the seed is placed close to the object boundary, then the compact piece segmented may be of smaller size compared to the compact segment produced if the seed was placed closer to the center.

3.7. Our algorithm

We now summarize our algorithm as follows. We assume that the objects are compact in shape or can be divided either vertically or horizontally in compact, roughly collinear parts. We build a graph of size greater than the maximum possible size of the object around the seed. Once the initial segment is obtained, it is likely to contain only a portion of the object that agrees with the compact shape. Then the boundary of the segment is checked to find weak edges, if any. The initial segment is extended in smaller pieces along the direction of the weak edges detected as described above. Thus by iteratively running the graph-cut we can segment the whole object regardless of its length.

Notice that our piecewise segmentation approach may seem similar to the region growing methods. However this similarity is superficial. In region growing, neighboring regions are repeatedly merged, based on some criterion of region similarity. In our approach, the best new region to add to the current segmentation is found by optimization, namely we choose the best region to add out of combinatorially many possible regions.

4. Results

We explored the challenging industrial problem of transistor gate segmentation in the images of integrated chips. It is an important preliminary step for performing intellectual property protection and competitive intelligence analysis in integrated circuitry domain. To obtain the images, the integrated circuit is de-layered and SEM micro-photographed. The images of the upper layers of the chip, that contain the metal wiring, are typically of high quality and can be segmented by automated means. The lower levels contain the dopant, the silicon implementation of the transistors. The images of these layers are typically of low quality and could have substantial variation in brightness and contrast. They occasionally contain artifacts due to the remains of the upper layer left during delayering. Two of the most important parameters in integrated chip circuitry are the length and the width of the transistor gates. They determine the circuitry power characteristics and are crucial for proper modeling and understanding of its functionality, which is essential for determining if the functionality is replicating a patented design. In order to obtain these measurements, accurate segmentation of the transistor gates is essential. Prior to the development of the application described in this paper, the measurements were taken manually by a human operator. It was done by selecting the gate in the image using a computer application, which also involved time consuming operations like zooming and panning across the image. The attempts to use off the shelf segmentation algorithms, such as magic wand or local thresholding, were unsuccessful.

Fig. 4 shows some of the images of integrated chips provided by Semiconductor Insight Inc., which are representative of the images used regularly. The images are in gray scale with 256 intensity values, where 0 represents black and 255 represents white. The transistor gates appear roughly rectangular in shape, and therefore can be well approximated with a compact shape. Notice the large variation in the noise level across the images. Hence accurate estimation of the parameter σ in the boundary term of the energy function is a crucial part in our work in order to accommodate the variation. From Fig. 4, it is also evident that the other challenges are the large variation in contrast and intensity range of the transistor gates. Another challenge is the wide variability in the transistor gate sizes, which range in length from 10 to a few thousands of pixels.

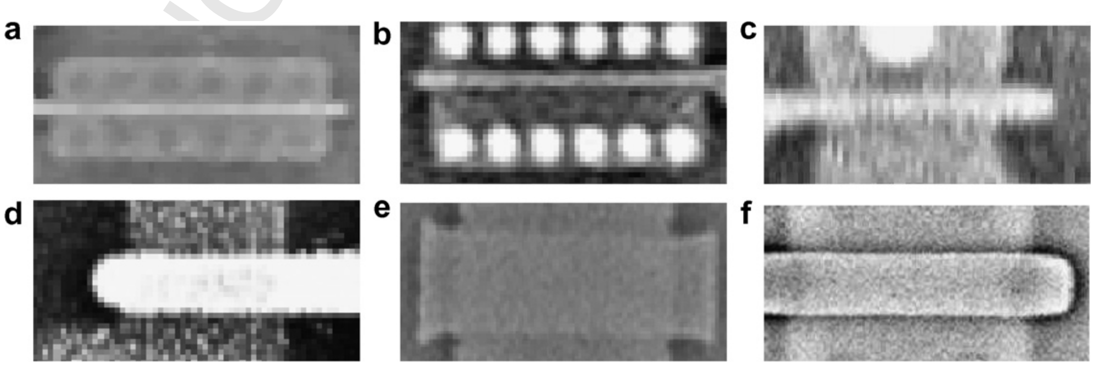

Fig. 4. Sample of the images provided by Semiconductor Insight Inc., showing the variation in image contrast, noise characteristics and the size of the object. Different scales are used for displaying. In each of these images, the transistor gate has horizontal orientation and is at the center of the image. Fig. 6 clearly delineates the gates for each of these images in gray.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

For all the experiments in this section, the parameters were set to the following values: $\alpha = 0.007$, $\gamma = 0.4$. The minimum size for the object was set to be 3 by 3 pixels, and the largest size for the object was set to be 130 by 130. Parameter *bias* is chosen automatically, as discussed in Section 3.4. As discussed in Section 3.4, the parameter σ is computed as the average of the intensity difference between two adjacent pixels using the data collected around the seed and the knowledge of the minimum possible size of the object.

We first compare our results with the results of the algorithm in Boykov and Jolly [5]. Fig. 5(a) shows the result of algorithm Boykov and Jolly [5] using only the boundary term. Only a small part of the transistor gate is segmented, due to the bias to small segments if the regional term is small or completely absent. On including the intensity model for describing the regional property of the segment, the algorithm Boykov and Jolly [5] produces multiple segments with very complex boundaries, most of them being false alarms, shown in Fig. 5(b). This happens because of considerable overlap of the background and object intensity distributions. 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 transis-

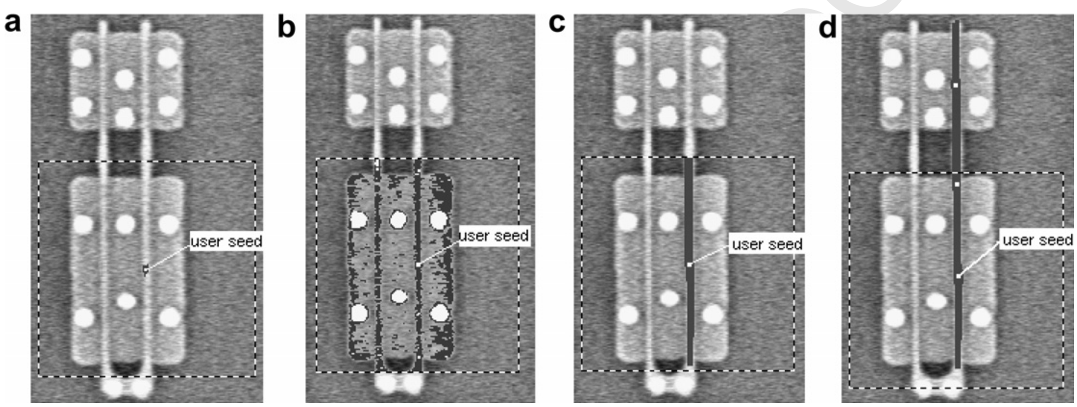

Fig. 5. In (a and b), we show segmentation results obtained using the algorithm of Boykov and Jolly [5]: (a) with the boundary term only; (b) with intensity model as regional term along with the boundary term. In (c and d), we show segmentation results obtained with our algorithm: (c) initial segment obtained with an automatically determined value of $bias > 0$; (d) 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 labeled. The large dotted square shows the maximum allowed segment size. The gray color shows the segmented object. Please note that in (b), the gray color which indicates the object is perceived to be much darker than the gray color in (a, c, and d).

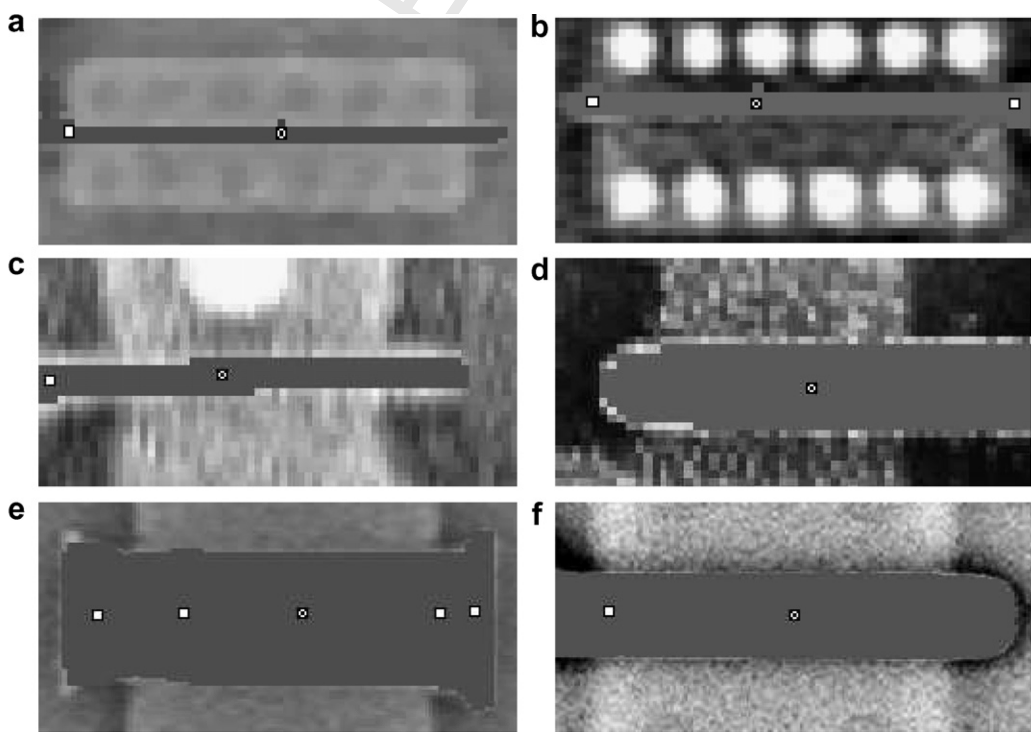

Fig. 6. Shows the segmentation results obtained using our algorithm on the images in Fig. 4. The segmented transistor gate is shown in gray.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

tor gate shown in Fig. 5(c). The results in Fig. 5(c) correspond to an acceptable initial segment, which is then extended iteratively to the whole object as shown in Fig. 5(d). Every time a segment is extended, a new seed point, marked with a white square in Fig. 5(d), is located.

Fig. 6 shows the segmentation results obtained using our algorithm on the images in Fig. 4. The user provided seed pixel is marked with a white circle and the automatically detected extension seeds are marked with white squares. Despite large variation in size, intensity distribution, noise type, shape and contrast, each transistor gate is accurately segmented.

To evaluate the performance of our system, we considered 10 images, each containing dozens of transistor gates and segmented 100 transistor gates chosen at random. In 91 cases the transistor gates are segmented accurately, giving the overall accuracy of 91%. In 6 cases the initial segments were segmented accurately but the extension failed. In 3 cases the segmentation boundaries aligned with the wrong but stronger intensity edges. Figs. 7 and 8 show some failure cases.

Fig. 9 shows more results which illustrate the challenges our system can deal with. In the first two rows, the transistor gate is very narrow, and in addition, in the first row its shape is far from compact. In the third row, the transistor gate includes large white circular spots and the seed is placed far from the center. In the last row, the transistor gate has a noticeable artifact (on the left) which is inconsistent with the overall intensity histogram and creates strong intensity edges inside the gate. In all these cases, our segmentation system gave an accurate segmentation.

Fig. 10 shows the results of segmentation of a long thin transistor gate which has nearly horizontal orientation but is rotated several degrees, and therefore is not compact. Our system has no problem extracting it in several compact pieces. In (b) and (c), we show the results under very different seed placements. Results are essentially identical, which shows the insensitivity of our system to the exact seed placement.

The application is implemented using C++ on a P4 2.8 GHz computer. The time varies with the size of the object. For small gates, it only takes a fraction of a second. Larger gates, such as of size 120×3000 pixels, are segmented in less than 2 s. It is currently being used by Semiconductor Insight Inc., to upgrade their existing manual segmentation system to a semiautomatic one.

We also applied our algorithm to segment objects in other types of images. Note that we chose to segment objects which are well approximated with several nearly collinear pieces of compact shape. The results are in shown in Fig. 11. The tool in the Fig. 11(a) is very far from a compact shape, but we were able to extract it by extending it in compact pieces. The roof in Fig. 11(b) is also not compact and was extracted in several pieces from a complex background.

We also applied our algorithm to segment the eye sockets in a 2D slice of MR brain image, which is required as a first step in the process of cortex segmentation. The eye sockets are elliptical in shape and follow the convex shape assumption. Fig. 12(a) shows the eye socket segmented with our semiautomatic algorithm. Fig. 12(b) shows the result obtained with the basic graph cut algorithm, which requires more interaction, yet unable to segment the whole sockets.

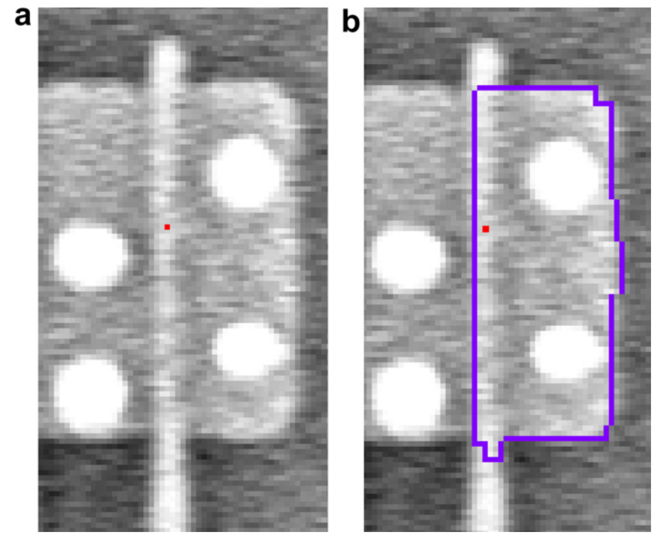

Fig. 8. (a) Original image of the transistor along with the user marked object seed. (b)The true boundary of the transistor is too weak, so the segmentation boundary sticks to the stronger but wrong boundary.

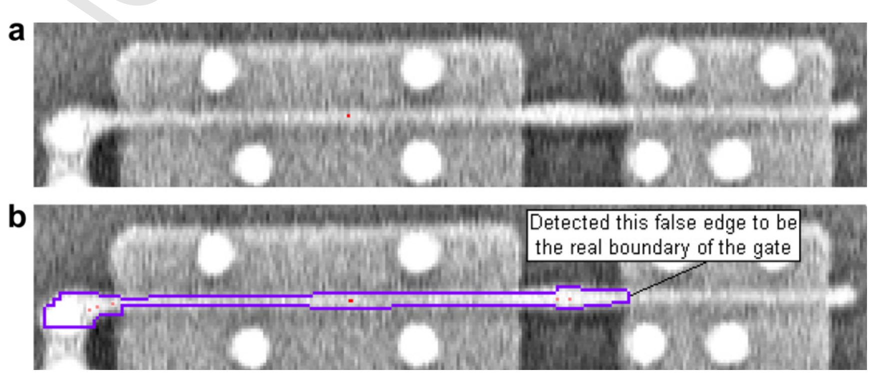

Fig. 7. (a) Original image of the transistor. (b) The transistor gate is segmented and is 4 pixels wide. The extension fails when the segmentation boundary stops at the wrong edge in the image.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

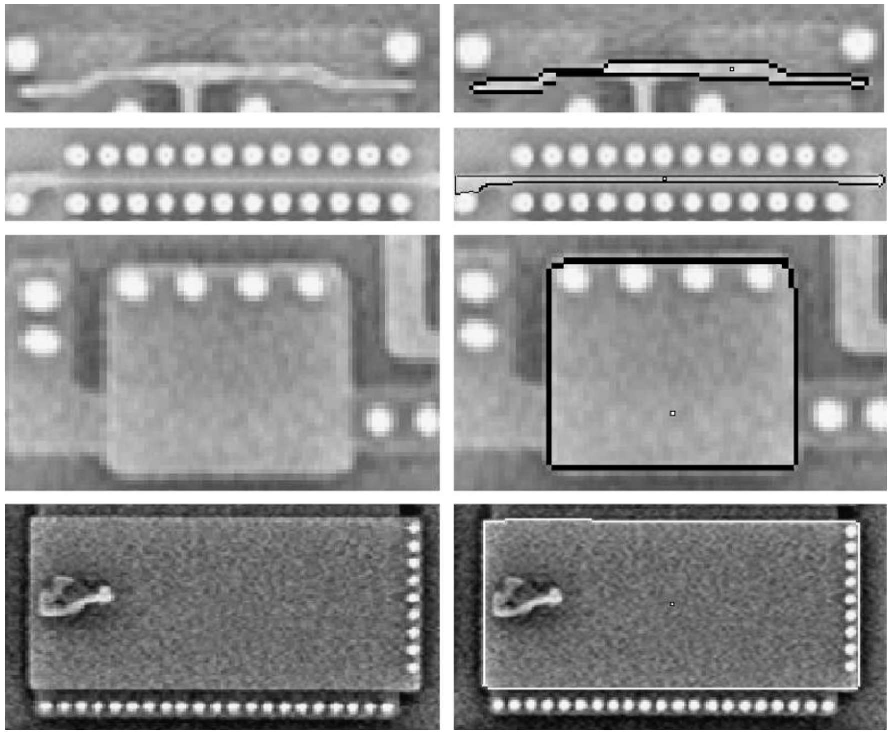

Fig. 9. The left column shows the original images, the right column shows the segmentation results. We used black color in the first three rows and white color in the last row to outline the segment boundary. White dot outlined in black is used to show the seed provided by the user.

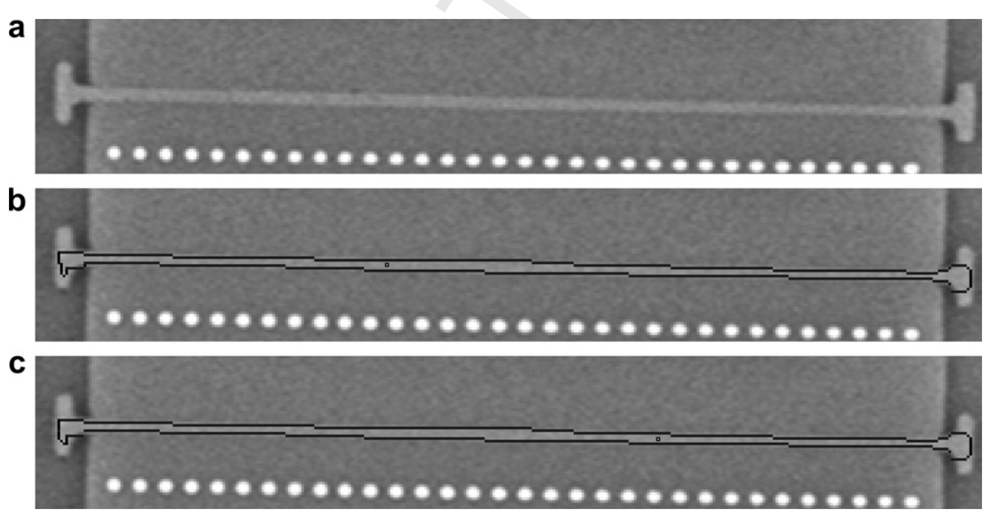

Fig. 10. (a) Shows the original image; (b and c) show the result of segmentation with different seed placement. The results of segmentation are outlined in black.

5. Discussion

In this paper, we presented a semiautomatic segmentation algorithm developed by modifying the basic graph cut segmentation algorithm of Boykov and Jolly [5]. We showed how problem specific assumptions and constraints can be well utilized to reduce the user interaction and also the complexity of the problem. The main contribution of our work is the introduction of the compact shape prior into the graph cut segmentation, which adds robustness to the algorithm. An additional benefit of using the compact shape prior is that we are able to introduce a parameter *bias* into the framework. This parameter biases the graph cut algorithm to segment objects with longer boundaries. We also showed how an application specific "quality check" for segmentation can be used to automatically select the appropriate parameters in graph cut segmentation.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006

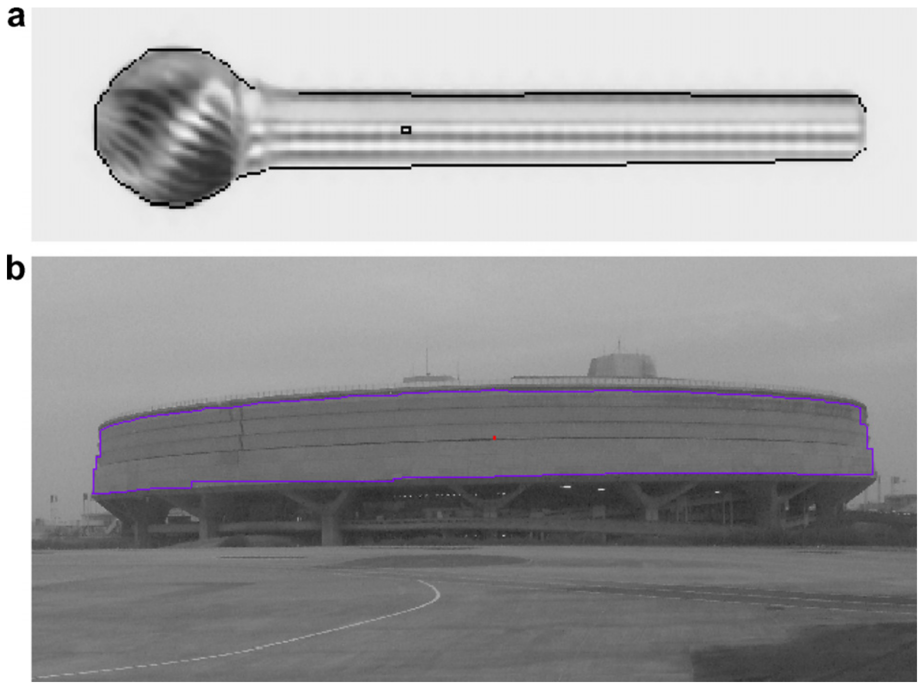

Fig. 11. (a) The tool has been segmented well in spite of the variation of width and shading within the object. (b) The roof part of the building has been segmented as several pieces of compact shape.

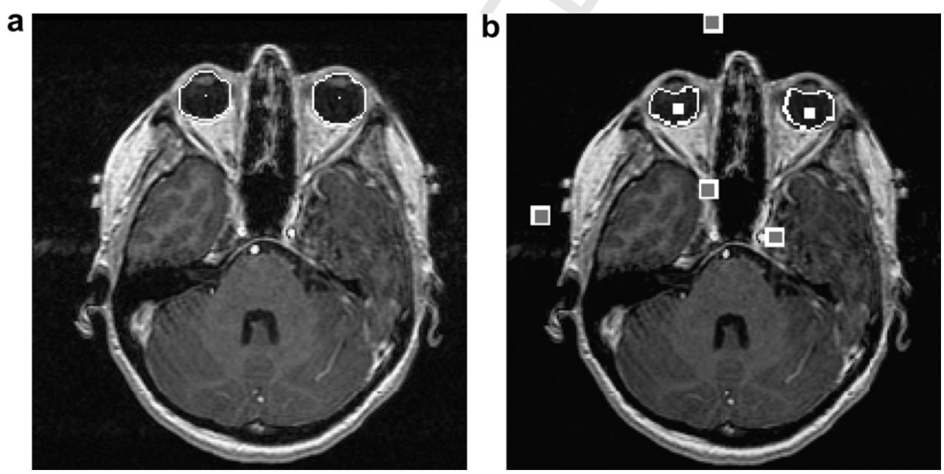

Fig. 12. (a) Segmentation of the eye socket using our algorithm. The seeds are marked as the white pixels inside the eye socket. Note that one eye socket is segmented at a time. (b) Segmentation of eye socket using basic graph cut algorithm. The object seeds are shown as white boxes, each 5×5 pixels, the background pixels are shown as gray boxes outlined with white, each 7×7 pixels big.

A weakness of the compact shape prior is that it is not rotationally invariant as it is defined with respect to the vertical and horizontal axes. If an object is compact with respect to a rotated set of axis, it can still be segmented in compact pieces as long at the rotation is not too large, in experiments we have found that we can tolerate rotations of about 6–7 degrees.

Acknowledgements

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

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, M. Cohen, Interactive digital photomontage, in: ACM Transactions on Graphics, vol. 23, 2004, pp. 294–302.
[2] A. Amini, T. Weymouth, R. Jain, Using dynamic programming for solving variational problems in vision 12 (9) (1990) 855–867, September.
[3] A. Blake, C. Rother, M. Brown, P. Perez, P. Torr, Interactive image segmentation using an adaptive gmmrf model. in: European Conference on Computer vision, vol. I, 2004, pp. 428–441.
[4] Y. Boykov, G. Funka Lea, Graph cuts and efficient n-d image segmentation, International Journal of Computer Vision 69 (2) (2006) 109–131, September.

[5] Y. Boykov, M.P. Jolly, Interactive graph cuts for optimal boundary and region segmentation, in: International Conference on Computer Vision (ICCV), vol. I, 2001, pp. 105–112.
[6] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Transaction on PAMI 26 (9) (2004) 1124–1137, September.
[7] L. Cohen, On active contour models and ballons, Computer Vision, Graphics and Image Processing 53 (2) (1991) 211–218.
[8] L.D. Cohen, R. Kimmel, Global minimum for active contour models: a minimal path approach, in: IEEE Conference on Computer Vision and Pattern Recognition, 1996, pp. 666–673.
[9] D. Cremers, Nonlinear dynamical shape priors for level set segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–7.
[10] D. Cremers, S. Osher, S. Soatto, Kernel density estimation and intrinsic alignment for shape priors in level set segmentation, International Journal of Computer Vision 69 (3) (2006) 335–351, September.
[11] A.X. Falaco, J. Udupa, S. Samarasekara, S. Sharma, User-steered image segmentation paradigms: live wire and live lane, in: Graphical Models and Image Processing, vol. 60, 1998, pp. 233–260.
[12] P. Felzenszwalb, D. Huttenlocher, Efficient graph-based image segmentation, International Journal of Computer Vision 59 (2) (2004) 167–181, September.
[13] L. Ford, D. Fulkerson, Flows in Networks, Princeton University Press, 1962.
[14] D. Freedman, T. Zhang, Interactive graph cut based segmentation with shape priors, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. I, 2005, pp. 755–762.
[15] Gonzalez, Woods, Digital Image Processing, second ed., Prentice Hall, Berlin, Heidelberg, New York, 1996.
[16] L. Grady, E.L. Schwartz, Isoperimetric graph partitioning for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (3) (2006) 469–475, March.
[17] I. Jermyn, H. Ishikawa, Globally optimal regions and boundaries as minimum ratio weight cycles, IEEE Transaction on PAMI 23 (10) (2001) 1075–1088, October.
[18] M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contour models, International Journal of Computer Vision 2 (1998) 321–331.
[19] V. Kolmogorov, R. Zabih, What energy function can be minimized via graph cuts? IEEE Transaction on PAMI 26 (2) (2004) 147–159, February.
[20] M. Kumar, P. Torr, A. Zisserman, Obj cut, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. I, 2005, pp. 18–25.
[21] S. Lee, J. Seo, Level set-based bimodal segmentation with stationary global minimum 15 (9) (2006) 2843–2852. August.
[22] M. Leventon, W. Grimson, O. Faugeras, Statistical shape influence in geodesic active contours, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. I, 2000, pp. 316–323.
[23] Y. Li, J. Sun, C. Tang, H. Shum, Lazy snapping, ACM Transactions on Graphics 23 (3) (2004) 309–314.
[24] E.N. Mortensen, W.A. Barrett, Interactive segmentation with intelligent scissors, in: Graphical Models and Image Processing (GMIP), vol. 60, 1998, pp. 349–384.
[25] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: Algorithm based on hamilton jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[26] M. Rousson, N. Paragios, Shape priors for level set representations, in: European Conference on Computer Vision, vol. II, 2002, p. 78 ff.
[27] J. Shi, J. Malik, Normalized cuts and image segmentation, in: IEEE Conference on Computer Vision(ICCV), 1997, pp. 731–737.
[28] G. Slabaugh, G. Unal, Graph cuts segmentation using an elliptical shape prior, in: International Conference on Image Processing (ICIP), vol. II, 2005, pp. 1222–1225.
[29] A. Tsai, A. Yezzi Jr., W. Wells III, C. Tempany, D. Tucker, A. Fan, W. Grimson, A. Willsky, Model-based curve evolution technique for image segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. I, 2001, pp. 463–468.
[30] O. Veksler, Image segmentation by nested cuts, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. I, 2000, pp. 339–344.
[31] O. Veksler, Stereo correspondence with compact windows via minimum ratio cycle, IEEE Transaction on PAMI 24 (12) (2001) 1654–1660, December.
[32] S. Wang, T. Kubota, J. Siskind, J. Wang, Salient closed boundary extraction with ratio contour, IEEE Transaction on PAMI 27 (4) (2005) 546–561. April.
[33] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its application to image segmentation, IEEE Transaction on PAMI 15 (11) (1993) 1101–1113, November.

Please cite this article in press as: P. Das et al., Semiautomatic segmentation with compact shape prior, Image Vis. Comput. (2008), doi:10.1016/j.imavis.2008.02.006