An Automatic Image Segmentation Technique Based on Pseudo-convex Hull

Sanjoy Kumar Saha[1], Amit Kumar Das[2], and Bhabatosh Chanda[3]

[1] Computer Science and Engineering Department
Jadavpur University, Kolkata, India
sks_ju@yahoo.co.in
[2] Computer Science and Technology Department
Bengal Engineering and Science University, Shibpur, Howrah, India
amit@cs.becs.ac.in
[3] Electronics and Communication Science Unit
Indian Statistical Institute, Kolkata, India
chanda@isical.ac.in

Abstract. This paper describes a novel method for image segmentation where image contains a dominant object. The method is applicable to a large class of images including noisy and poor quality images. It is fully automatic and has low computational cost. It may be noted that the proposed segmentation technique may not produce optimal result in some cases but it gives reasonably good result for almost all images of a large class. Hence, the method is found very useful for the applications where accuracy of the segmentation is not very critical, e.g., for global shape feature extraction, second generation coding etc.

1 Introduction

Today it is needless to mention the importance and necessity of image segmentation. Probably it is the most intensively researched topic in the field of image processing and understanding. Some major concepts include feature thresholding [1], region growing [2], change in feature detection [3], facet model [4], active contour [5], watershed [6], etc. In [7], a scheme is presented to find out the semantic objects in an image. But, it is applicable for colour images only. A multilevel hypergraph partitioning method has been discussed in [8]. The scheme suffers from prohibitive computational cost. Depending on the application domain as well as the quality of the image data many variations of these approaches have come up. Thus, hundreds of papers are available in the literature. All these segmentation algorithms may be classified into two groups: (i) Region extraction and (ii) Contour detection. However, these two groups have a strong correspondence between them. That means if region is available, contour can readily be found by applying boundary extraction method [1] and, on the other hand, if contour is available, region can be generated straightaway by filling [9]. Secondly, none of these algorithms are fully automatic; they always need some form of user intervention – in the form of threshold selection or markers selection or contour initialization etc.

In this paper, we present a fully automatic, low cost and robust segmentation algorithm. However, it should be noted that the proposed algorithm may not give the best result in many cases, but it gives reasonably good result for a really large class of images. By the term 'reasonably good result' here we mean that the outer-most contour of the segmented/extracted region approximates the actual contour closely. This kind of segmentation results is good enough in various types of applications, where exact segmentation may not be very crucial. For example, it may be suitable for extracting global shape features (like aspect ratio, circularity, etc.) that are used in CBIR, for second generation compression where different regions are coded differently, for supplying initial contour to snake algorithm, or may be used as a mask for selecting marker in watershed algorithm etc.

The paper is organized as follows. Section 2 elaborates the problem while section 3 presents fast algorithm for computing Pseudo-convex hull. Proposed segmentation algorithm is described in section 4 step by step. Experimental results are presented in section 5 and section 6 contains concluding remarks.

2 Problem Formulation

It is mentioned earlier that the proposed algorithm works for a class of images. So, first, we like to define that class. Depending on the contents, images may be grouped into three classes: (i) class of images containing a single dominant object (Class-1), (ii) class of images containing many objects of more or less equal significance (Class-2), and (iii) class of images containing no objects of specific interest, but their combination appears very picturesque (Class-3). The class-3 is exemplified by outdoor scenery consisting mostly of sky, water body (like, sea, river, lake etc.), grass-field, beach etc. none of which is particularly important, but surely the combination is. Images of a group of people, cluttered objects, busy area (e.g., railway station, departmental store, city street, etc.), business meeting and like belong to Class-2. Finally, Class-1 contains images of our child, friend, relative, home, car, pet, object of our interest (e.g., ancient building, monument, sculpture and statue, biomedical image, animal, bird, etc.), famous personality, and so on. These objects, in the image, occupy the major area mostly at the center and are sharply focused. There could be other objects too in the image, but those are given usually less emphasis while photographed and are treated as background. Hence, we say that Class-1 contains images of single dominant object. In any estimate, number of images belong to Class-1 is by far large than that of Class-2 and Class-3 together. It is also observed that dominant objects or the objects of interest in the Class-1 images are closely convex shaped. However, the term 'closely' is qualitative in nature and introduces ambiguity in decision. So an objective measure is in order.

An object A is said to be *convex* if its intersection with a line having any slope angle θ produces at most one line segment. However, in order to explain the working of our algorithm we describe convex object in a different way. Suppose an image contains an object A. If two distinct line segments, with an angle θ

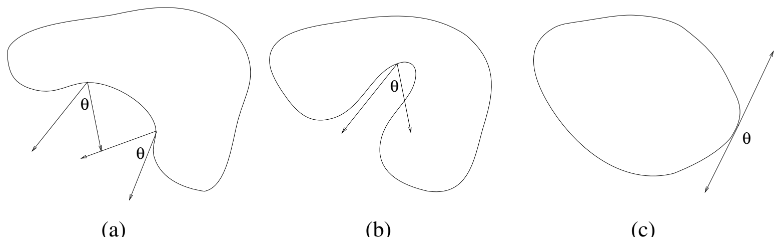

Fig. 1. Different types of objects: (a) pseudo-convex, (b) concave, and (c) convex

between them, starting from every point on the boundary of A can reach the image frame without intersecting any of the interior point of A [see Fig. 1], then we call A is n *pseudo-convex* object with respect to θ; It is readily evident that the objects we mostly deal with are neither strictly convex nor concave, but are of type pseudo-convex. Hence, in this work, we classify 2D objects into three groups: Convex, Pseudo-convex and Concave.

Since we work on discrete domain and it is known that digital straight line segment can uniquely defined only for the slope 0^o, 45^o, 90^o and 135^o [10], we confine our definition of pseudo-convex objects in terms line segment of said orientations only.

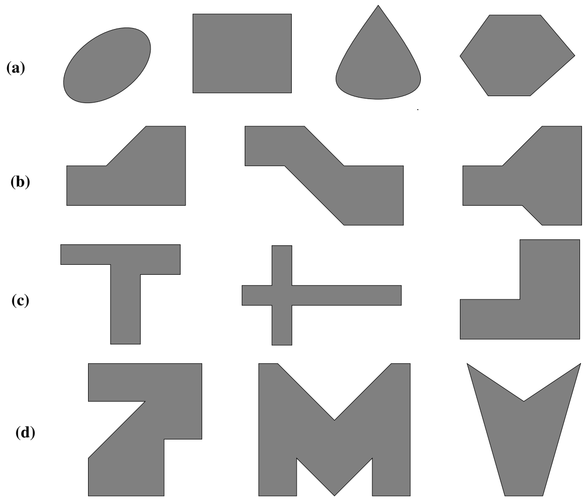

Fig. 2. Different types of objects (a) convex, (b) ramp-convex, (c) ortho-convex and (d) wedge-convex objects

Definition: A digital object A is said to be *pseudo-convex* if two line segments, with an angle θ between them and one of them is either horizontal (slope 0^o) or vertical (slope 90^o), starting from every point on the boundary of A can reach the image frame without intersecting any of the interior point of A.

A is a true convex object for $\theta \geq 180^o$ and it is taken as a concave object if $\theta < 45^o$. Otherwise, if $45^o \leq \theta < 180^o$ then the object is 'closely convex shaped' which can be further classified as follows. If $135^o \leq \theta < 180^o$ then the shape of A is called ramp-convex. It is ortho-convex if $90^o \leq \theta < 135^o$. A is wedge-convex for $45^o \leq \theta < 90^o$. Figure 2 shows some examples of convex, ortho-convex, ramp-convex and wedge-convex objects.

The proposed segmentation algorithm is based on the idea of obtaining a closely convex region corresponding to the dominant object in an image. It may be noted that this region is nothing but the pseudo-convex (ramp, ortho or wedge-convex) hull of the dominant object.

Now suppose a digital graylevel image $I(i,j)$ contains a dominant object and ideal segmentation of I produces a binary image containing a connected component A corresponding to the dominant object. Our segmentation algorithm tries to obtain a closely convex (e.g., convex, ortho-convex, ramp-convex or wedge-convex) region, say, R such that

$$error = \#\{(A \cap R^c) \cup (A^c \cap R)\} < t_a \tag{1}$$

The operator '#' stands for cardinality of a set and t_a is the threshold of tolerance. In the proposed method R is computed as pseudo-convex hull of the set of pixels obtained from initial processing (e.g., edge pixel extraction) of I for a given θ. Detail of the algorithm is described in the next section.

3 Fast Algorithm for Computing Pseudo-convex Hull

Since backbone of the proposed scheme is computing pseudo-convex (i.e., ramp-convex or ortho-convex or wedge-convex) hull. We first present an efficient algorithm for the same. To design the algorithm we adopt the definition of pseudo-convex except that the lines originate from image frame. And then it is examined whether pair of lines with given θ has reached the boundary before meeting any interior pixels.

Suppose a binary image $B(i,j)$ contains a set of points A whose pseudo-convex hull is to be determined. That means $B(i,j)$ may be represented as

$$B(i,j) = \begin{cases} 1 & (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

An example of B is shown in Fig. 3(a)(i). Hence, the steps of the algorithm are:

Step 1: Take four other arrays $H(i,j)$, $V(i,j)$, $D1(i,j)$ and $D2(i,j)$ of same size as that of $B(i,j)$, and initialize them with 1's.

Step 2: Now for each row of $H(i,j)$ [An example is illustrated in Fig. 3(b & c)(i)]

1. Start from first column, change its pixel value to zero and move right until $B(i,j) = 1$ or the last column is reached.
2. If the last column is not reached then start from last column, change pixel values to zero, and move left ward until $B(i,j) = 1$.

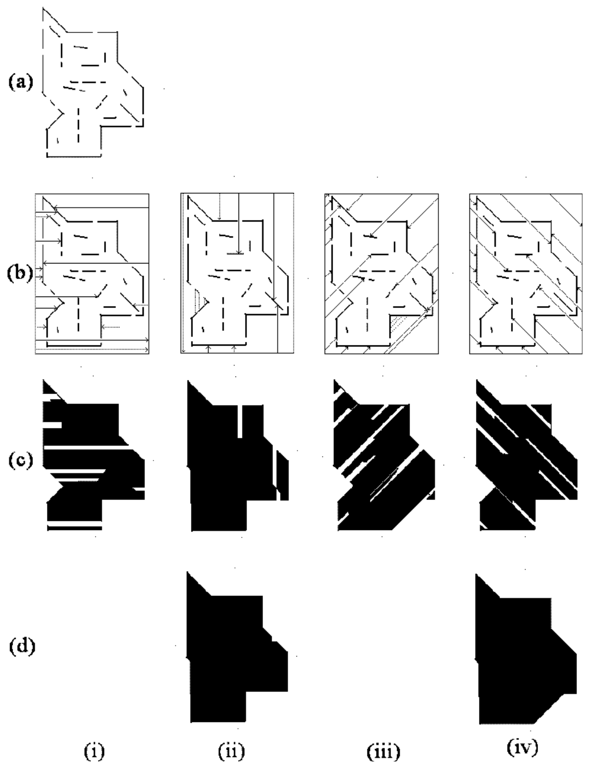

Fig. 3. Scanning directions and output of each step of pseudo-convex hull algorithm

Step 3: Now repeat sub-steps of 2 for $V(i,j)$, $D1(i,j)$ and $D2(i,j)$ with appropriate directions i.e., upward and downward for V and so on. [Results are illustrated in Fig. 3(b & c)(ii)-(iv).]

Step 4: Finally, produce a binary image $P(i,j)$ that contains the pseudo-convex hull of the given point set A as follows:

$$P(i,j) = \begin{cases} 1 & H(i,j) + V(i,j) + D1(i,j) + D2(i,j) \geq th \\ 0 & \text{otherwise} \end{cases}$$

1. $th = 1$ is equivalent to $\theta = 135^o$ and we have ramp-convex hull. [See Fig. 3(d)(iv).]
2. $th = 2$ is equivalent to $\theta = 90^o$ and if only $H(i,j)$ and $V(i,j)$ taken, we have ortho-convex hull. [See Fig. 3(d)(ii).]
3. $th = 3$ is equivalent to $\theta = 45^o$ and we have wedge-convex hull.

Finally, it may be noted that wedge-convex hull ($hull_w$) is the closest estimate of the object as

$$A \subseteq hull_{wedge}(A) \subseteq hull_{ortho}(A) \subseteq hull_{ramp}(A) \subseteq hull(A)$$

As the algorithm involves only the traversal of the pixels along a direction originating from the image frame, computational cost is quite low. The order of such complexity is $o(n)$, where, n is the number of pixels in the image.

4 Description of the Proposed Scheme

In this section we describe the details of the proposed segmentation algorithm. Note that the segmentation algorithm is fully automated and assumes that the image contains only one dominating object and other objects, if present, are small in comparison to the object of interest.

4.1 Segmentation Algorithm in Steps

Input to this step is a gray level image representing the intensity map of the scene. If the original image is in colour we convert it to HLS or HSV or any other similar triplet and take the L or V component as an intensity image. The segmentation is done in three steps, assuming that the image background does not have high contrast texture, as elaborated below:

- I. Noise removal.
- II. Initial Segmentation.
  - (a) Formation of Gradient image.
  - (b) Thresholding.
- III. Final Segmentation.
  - (a) Approximate object area determination.
  - (b) Removal of small objects by component labeling.
  - (c) Final extraction of object region.

Explanation of each steps are in order.

Noise removal. Smoothing filters are, in general, used for noise removal and blurring. Blurring is used as a preprocessing step to remove small details from the image prior to extraction of large objects as well as bridging of small gaps in lines and curves. In our case noise removal gets priority rather than blurring and we would like to keep edge sharpness intact. We used median filtering which is a suitable tool to get the desired effects. A 5×5 window is used over which the median filtering is done to remove noise.

Initial Segmentation a) Formation of Gradient image
Here edges are detected on the basis of gray level discontinuities. To achieve this, gradient (i.e., the maximum rate of change of the gray level within a specified neighborhood) at every point of the filtered image is computed.

The neighborhood around a pixel (i, j) within which the gradient is computed and the weights given to the neighboring pixels are shown below:

a	b	c	1	$\sqrt{2}$	1
d	(i,j)	e	$\sqrt{2}$	(i,j)	$\sqrt{2}$
f	g	h	1	$\sqrt{2}$	1

The gradient at the centre point (i, j) of the 3×3 mask is computed using the difference operator based on the concept of Weber ratio as follows.

$$\begin{aligned} m_0 &= (c + \sqrt{2}e + h)/(2 + \sqrt{2}) \\ m_1 &= (a + \sqrt{2}d + f)/(2 + \sqrt{2}) \\ g_0 &= \mid m_0 - m_1 \mid /(m_0 + m_1 + 1) \end{aligned}$$

Similarly, g_1 is computed considering the elements a, b, c and f, g, h. For g_2, the diagonal elements b, c, e and d, f, g are considered. And, g_3 is computed considering the elements a, b, d and g, h, e. The intensity gradient $g(i, j)$ at the point (i, j) is then defined as

$$g(i, j) = max\{g_0,\ g_1,\ g_2,\ g_3\} \tag{2}$$

The value of $g(i, j)$ ranges from –127 to 127 and is symmetrically distributed with zero mean. It is observed that the distribution of $g(i, j)$ closely follows Laplace density function [11].

$$f(x) = \frac{1}{2\sigma} e^{-|x-\mu|/\sigma} \quad \text{for } -\infty < x < \infty \tag{3}$$

where μ is the mean and σ is the standard deviation of the population. Secondly, $p\%$ of population lie in the range $[\mu - k\sigma,\ \mu + k\sigma]$ where

$$k = -ln(1 - \frac{p}{100}) \tag{4}$$

b) Thresholding
The gradient image is then subject to a threshold operation to get a binary image containing edge pixels. Suppose we assume that less than $q\%$ of all the image pixels are edge pixels and $p = 100 - q$. Then the threshold is $\mu + k\sigma$; μ and σ are computed from the gradient image. Edge image is given by

$$B(i, j) = \begin{cases} 1 & \text{if } |g(i, j)| > \mu + k\sigma \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In our experiment we have taken $q = 20$. Obviously, the threshold computed this way may not give optimum results for all kinds of images. The resulting edge images thus have *thick* or *broken* or *extraneous* edges or their combination. However, these edge pixels are sufficient to generate approximate pseudo-convex hull of the object of interest as will be seen below.

These steps are demonstrated by an example shown in Fig. 4. Fig. 4(a) is the input gray level image and Fig. 4(b) shows its edge pixels.

Final Segmentation a) Approximate object area determination
To find out the object, the edge image obtained in previous step undergoes the pseudo-convex hull algorithm as described in section 3. We determine wedge-convex hull of the set of edge pixels, i.e., use $th = 3$. We choose wedge-convex hull to ensure that the boundary of the computed hull be as close to the concave

region of the object as possible and also that the nearby small objects or extraneous edges due to background texture (if any) do not merge with the dominant object.

The result of approximate object area determination is shown in Fig. 4(c). This process may generate small objects due to presence of extraneous edge pixels as seen in figure Fig. 4(c). They may be removed through connected component analysis as described below.

b) Removal of small objects by component labeling

After computing the wedge-convex hull we get an initial estimate of the dominant object. We also get some small objects arising out of scattered extraneous edge pixels in the background. These can be isolated by component labeling and subsequently removed keeping only the biggest one.

Fig. 4(d) shows the result after removal of small objects by connected component analysis.

c) Final extraction of object region

After removal of small objects we finally determine the dominant object region by applying pseudo-convex hull algorithm with $th = 1$ on the point set of largest connected component obtained from previous step. This time we choose to compute ramp convex hull to remove undesired intrusion into the object region due to broken edges. However, this also fills up concave regions. Result of the final step is shown in Fig. 4(e).

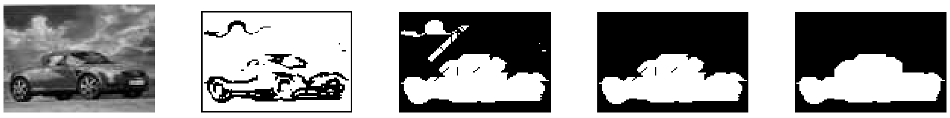
(a) (b) (c) (d) (e)

Fig. 4. Segmentation steps; (left to right) a) Original image; b) After threshold operation; c) After 1st level of segmentation; d) After removal of small component and e) Final segmentation

5 Experimental Results and Discussion

The segmentation algorithm proposed here are implemented on a Alphaserver DS 20E machine with UNIX OS. The average time taken (for image of size 200×320 approx.) is less than 10 msec. The algorithm can be made even faster by readily parallelizing various parts. We have tested the proposed algorithm on a large number (1000 approx.) of images of various types. A few results of the experiment are shown in Fig. 5 where the original image, segmented image and the contour superimposed original image are shown side by side.

To compare the performance of the proposed segmentation scheme, we have implemented the scheme proposed by Siebert [12]. The results are shown in figure 6.

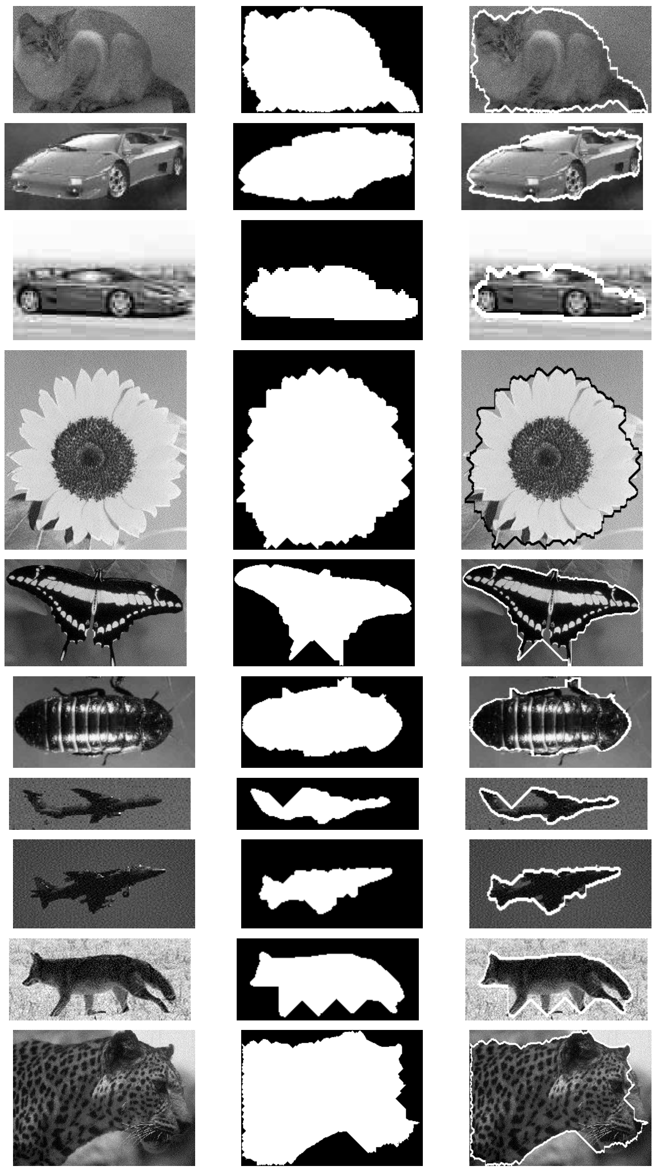

Fig. 5. Segmentation Results (left to right): (a) Original image; (b) Segmented image and (c) Superimposed image taking the original and the segmented image

To get the visually best possible result in case of Siebert's scheme, the parameters are set manually. It appears that Siebert's scheme suffers from various drawbacks. The algorithm depends on a number of parameters like θ_{cc}, *abrupt change* etc. setting of which cannot be automated. θ_{cc} is a fraction of strong point count

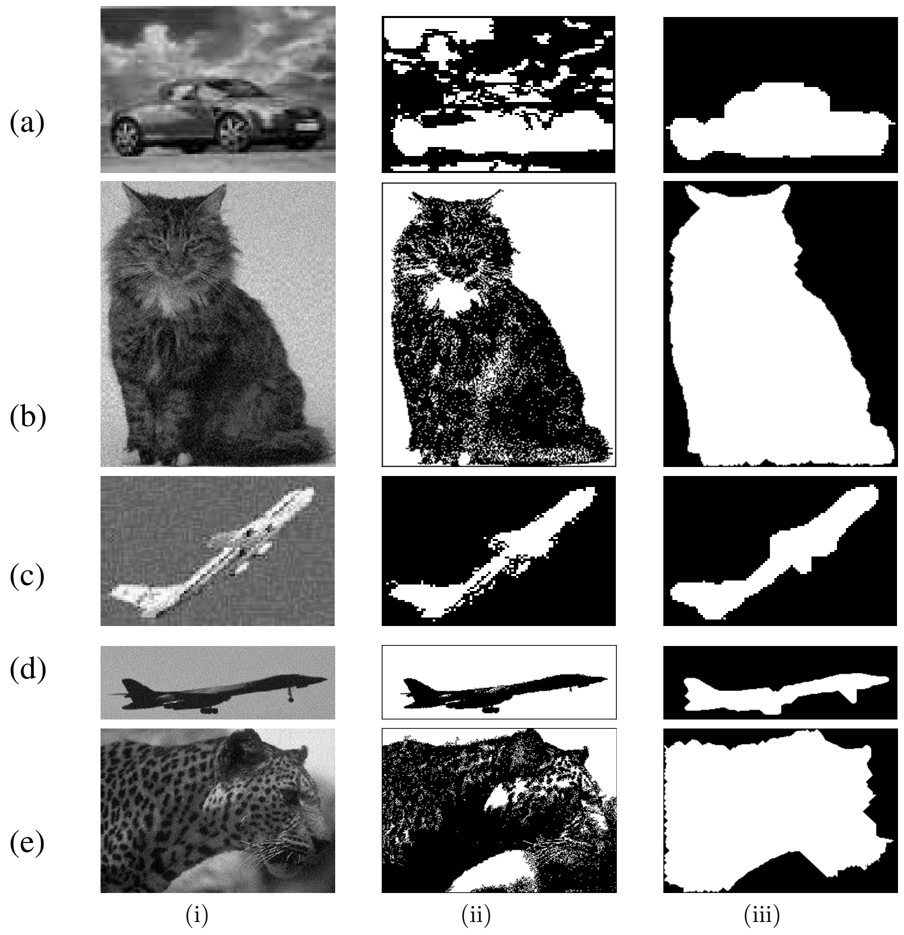

Fig. 6. Column (i) shows original image; (ii) and (iii) show corresponding segmented image by Siebert's scheme and our scheme respectively. In case of Siebert's scheme (θ_{cc}, *abrupt change*) for the five images considered are (0.4, 0.4), (0.4, 0.5), (0.1, 0.3),(0.4, 0.4) and (0.3, 0.4) respectively.

in the image and is a criterion to determine how far the region growing will continue. A region growing may result into overspill. If the overspill results into *abrupt change* in certain parameters (say, region size), then that overspill is discarded. The region growing starts from a seed region which is selected based on the smoothness factor. Hence, sometimes the central object (figure 6(b),(d),(e)) and sometimes the encompassing background (figure 6(a),(c)) appear as output. Moreover, the performance varies from image to image and fails if the background contains texture. As figure 6 shows, the performance of Siebert's algorithm is very good for (d) and moderate in case of (b) and (c). But, it fails in case of (a) and (e). On the other hand, our scheme provides good output for all those cases. Another drawback is that, the algorithm is very slow. Segmenting around 500 images using the two algorithms, it has been observed that on an average the proposed algorithm is almost 10 times faster than Siebert's algorithm.

6 Conclusion

In this paper we have presented a fast and robust image segmentation algorithm based on edge detection and determination of pseudo-convex hull of edge points. The algorithm may be considered fully automatic as the required parameters are set in the algorithm itself and no user intervention is needed during batch operation. Assigned value of the parameters are not claimed to be optimal, they together can do the job reasonably well in almost all the cases. For example, the threshold used to detect edge points is by no means the best, even then the detected edge points provide sufficient clue to estimate the object region through the pseudo-convex hull algorithm. The result of the proposed algorithm is compared with that of of a recent work and is found superior in most of the cases.

Finally, it should again be noted that emphasis is given to design simple and fully automated segmentation method that incurs very low computational cost. The proposed method may not give the best result in all cases, but it surely gives acceptable results in almost all cases. Thus the method is useful where accuracy of segmentation is not very critically demanded. The class for which the method would give good results is also defined and is found really large.

References

1. Rosenfeld, A., Kak, A.C.: Digital Picture Processing. Volume II. Academic Press, N.Y. (1982)
2. Pavlidis, T., Liow, Y.T.: Integrating region growing and edge detection. IEEE Trans. on PAMI **12(3)** (1990) 225–233
3. Canny, J.: A computational approach to edge detection. IEEE Trans. on PAMI **8(6)** (1986)
4. Haralick, R.M., Shapiro, G.L.: Computer and Robot Vision. Volume 2. Addison Wesley, Reading, MA (1992)
5. Williams, D.J., Shah, M.: A fast algorithm for active contours. CVGIP: Image Understanding **55(1)** (1990) 14–26
6. Beucher, S.: Watersheds of functions and picture segmentation. In: Proceedings of IEEE ICASSP-82. (1982)
7. Mezaris, V., Kompatsiaris, I., Strintzis, M.G.: Still image segmentation tools for object-based multimedia applications. Intl. Journal of Pattern Recognition and Artificial Intelligence **18(4)** (2004) 701–725
8. Rital, S., Cherifi, H., Miguet, S.: A segmentation algorithm for noisy images. In: Proceedings of 11th Intl. Conf. on Computer Analysis of Images and Patterns, France (2005)
9. Foley, J.D., Dam, A., Feiner, S.K., Hughes, J.D.: Computer Graphics - Principles and Practices. Addision - Wesley (1993)

10. Rosenfeld, A.: Digital straight line segments. IEEE Trans. on Computer **C-23** (1974) 1264–1269
11. Rao, C.R.: Linear Statistical Inference and Its applications, 2nd ed. Wiley Eastern, New Delhi (1973)
12. Siebert, A.: Segmentation based image retrieval. SPIE 3312 **SRIVD VI** (1998) 14–23