www.cadanda.com
Computer-Aided Design AND APPLICATIONS™

A Graph based Geometric Approach to Contour Extraction from Noisy Binary Images

Amal Dev Parakkat[1*], Jiju Peethambaran[2*], Philumon Joseph[1+] and Ramanathan Muthuganapathy[2+]
[1]Government Engineering College Idukki, Kerala, India, *adp.upasana@gmail.com, +philumon@gmail.com
[2]Indian Institute of Technology Madras, India, *jijupnair2000@gmail.com, +emry01@gmail.com

ABSTRACT

We present a method to extract the contour of geometric objects embedded in binary digital images using techniques in computational geometry. Rather than directly dealing with pixels as in traditional contour extraction methods, we process on object point set extracted from the image. The proposed algorithm works in four phases: point extraction, Euclidean graph construction, point linking and contour simplification. In point extraction phase, all pixels that represent the object pattern are extracted as a point set from the input image. We use the color segmentation to distinguish the object pixels from the background pixels. In the second phase, a geometric graph G=(V,E) is constructed, where V consists of the extracted object point set and E consists of all possible edges whose Euclidean distance is less than a threshold parameter, l ; which can be derived from the available information from the point set. In point linking phase, all border points are connected to generate the contour using the orientation information inferred from the clockwise turn angle at each border point. Finally, the extracted contour is simplified using collinearity check. Experiments on various standard binary images show that the algorithm is capable of constructing contours with high accuracy and achieves high compression ratio in noisy and non-noisy binary images.

Keywords: Contour extraction, Binary images, Edge detectors, Noisy images
DOI: 10.3722/cadaps.2013.xxx-yyy

1 INTRODUCTION

The boundary lines of geometric objects embedded within an image is referred to as a contour [15]; see Fig. 1. In computer vision, contour features are used to identify, localize and analyze objects in digital images. So, contour extraction from images, particularly images with complex shapes and noise, is a topic of interest to researchers in the field of computer vision, graphics and pattern recognition.

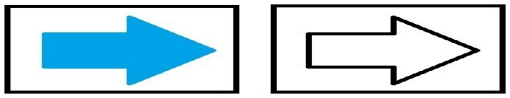
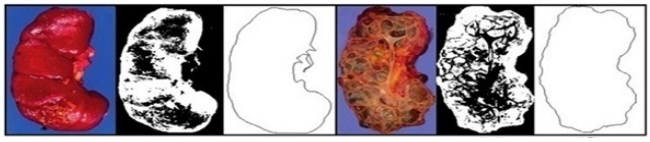


Fig. 1. An image and corresponding contour



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

Fig. 2.(a) Human kidney, (b) Binary image of (a), (c) Contour of (b) extracted by our algorithm, (d) Human kidney with polycystic kidney syndrome, (e) Binary image of (d), (f) Contour of (e) extracted by our algorithm [Image courtesy:[7]]

Contour extraction finds many applications in different domains. For instance, in the development of computer aided design (CAD) system for detecting breast cancers, region of interest (ROI) segmentation is extremely important. In [21], a local binary image along with contour extraction has been proposed for ROI segmentation from mammogram patches. In automated medical diagnosis, anatomical abnormalities of internal organs due to tumors or particular syndromes can be determined by matching the contours of corresponding infected and uninfected body parts (Please see Fig. 2). Further, contours extracted from binary images of optical coherence tomography (OCT) images are used for automated OCT segmentation in the clinical diagnosis of coronary arterial lumen [23]. Contours are widely applied in pattern analysis of digital images through feature extraction [14]. Contour consists of a small subset of the pixels representing the geometric objects in digital image and also shares many important features with the original object pattern. As a consequence, the computation time will be considerably reduced if the feature extraction algorithms are applied on the contour rather than on the original pattern [13, 14]. In machine vision, contour detection plays a significant role especially when dealing with the inspection of manufactured parts. Contour extraction along with matching can be applied in unsupervised inspection of machine parts for geometric irregularity [18].

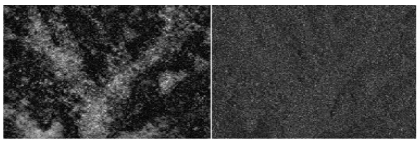


Fig. 3. Satellite image **Left:** before and **Right:** after transmission.

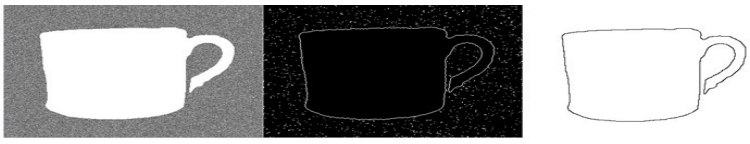


Fig. 4. Left: Noisy input image **Middle:** Output of Sobel edge detector **Right:** Output of our algorithm.

In applications such as tracking of earth resources, geographical mapping, weather forecasting, prediction of agricultural crops etc., images transmitted by satellites or remote sensors are often contaminated by noisy components [12] (Fig. 3 shows an example of satellite image before and after transmission). Consequently, edge detection or contour extraction in such images poses a great challenge as the noise can change the contrast along edges, and even lead to local contrast reversals. Smoothening the image (e.g., with a Gaussian filter [6]) reduces the noise, but it may also weaken the contrast across the edges and blend adjacent edges [1]. So, a perfect contour detection in a noisy image using common edge detectors is extremely difficult. This is more evident from Fig. 4, which shows the result of Sobel edge detection [5] algorithm over noisy image. In this paper, we propose a geometry-based contour extraction approach that works well with noisy binary images (see Fig. 4). Further, we demonstrate the efficiency of the proposed algorithm through various experimental studies on noisy and non-noisy images. Rest of this paper is organized as follows: In Section 2, we present some closely related work. In Section 3, we discuss the contour extraction algorithm. Section 4 illustrates empirical studies with results and discussions. We conclude the paper in section 5.

2 RELATED WORK

Several computational models have been reported in the literature for contour or edge detection in images. Some of the earlier works include Robert [20], Sobel [5] and Prewitt [17] edge detectors, all of which use local derivative filters to do the boundary detection. D. Marr et.al [9] exploits intensity

changes to detect the edges by finding zero values of Laplacian of Gaussian distribution. Canny edge detector [4] is one of the well-known edge detection algorithm that uses sharp discontinuities in the brightness channel to detect edges.

Contours extracted by geometric algorithms are often found to have high compression ratio and smooth representation compared to pixel based methods. Despite this fact, very few geometry/graph based algorithms address contour extraction form digital images. One such algorithm is proposed by Pedro J. et al. [15], which makes use of geometric techniques such as clustering, linking, and simplification to find contours in $O(n^2 \log n)$ time where n is the size of extracted feature points from the image. Q. Zhu [24] describes a graph based approach for boundary detection in 2D image with clutters which uses cycle tracing in the output of an external graph cut. There are also some methods proposed to capture discontinuity in detected edges [19]. For example, X. Ren et al. [19] uses constrained Delaunay triangulation over the set of contours detected by local edge detectors to remove gaps and clutters present in the extracted edges.

In general, computational approaches proposed for contour extraction can be classified depending on whether they do local, regional or global processing [6]. In local processing, adjacent pixels in the neighbourhood are linked if they satisfy some criteria. Some examples of local contour tracing algorithms are: square tracing algorithm, Moore-neighbourhood algorithm [22], radial sweep and bug follower's algorithm [13, 14, 16]. A recent local approach include multi scale version of Pb detector [2], where a Pb detector [10] is an edge detection algorithm that predict the posterior probability of a boundary with a particular orientation at each image pixels by measuring the local image brightness, colour, and texture channels. In regional methods, starting from a seed point, regions grow by adding pixels that are previously known to be a part of the same region or contour. Global processing techniques try to find pixels which lie on curves of specific shapes. However, all these three traditional methods possess some drawbacks: local methods ignore everything other than those pixels within the neighbourhood and hence does not take into account the valuable global information about the geometric proximity of pixels; regional methods needs some prior knowledge about which pixels lie on which contour; and usage of global processing techniques such as Hough transform are only applicable to some specific types of shapes [15]. As opposed to this, our approach exploits the proximity and orientation of points inferred through a threshold length parameter and turn angle at each boundary pixel, requires no prior knowledge on the regional belongings of pixels and is not restricted to any specific shape.

3 ALGORITHM

A bi-level image (binary image) is a digital image in which each pixel can have one of 2 values: 0 or 1. Pixel value 0 is used for denoting background pixels and 1 for denoting object pixels. Our algorithm is designed to generate closed contours in binary images and uses some simple geometric techniques for the contour construction. Approach consists of four phases: point extraction, graph construction, point linking and contour simplification as shown in Fig. 5. In point extraction phases, a set of points that represent the object pattern in the image, are extracted. Then a geometric graph is constructed on these extracted points in the second phase. The boundary edges are linked together to construct the contour in the third phase. Finally, the extracted contour is simplified using collinearity checking. We use the following few notations while describing the algorithm. Let p_1, p_2 are two points having coordinates (x_1,y_1) and (x_2,y_2) respectively, then (p_1,p_2) denotes the edge connecting points p_1 and p_2 and $d(p_1, p_2)$ denotes the Euclidean distance between points p_1 and p_2.

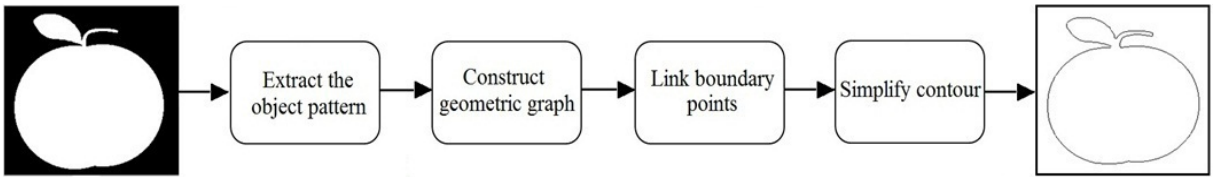
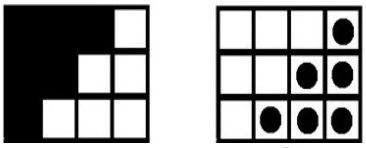


Fig.5. Overview of the Approach.

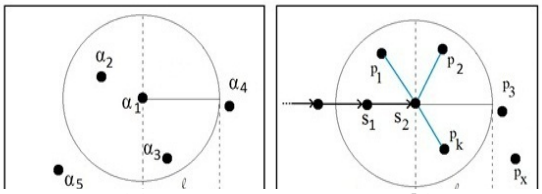
3.1 Point Extraction

In image processing, the basic processing unit is a pixel whereas the basic processing unit in discrete geometry is a point. In this step, the algorithm deals with the mapping and extraction of pixels that represent the object pattern in the given digital image to its corresponding object point set for further processing. Once the point set is extracted, the problem of contour detection can be reinterpreted as a geometric problem that deals with the construction of the shape border for the point set in 2D. Since the inputs to the algorithm are binary digital images that have only two color components, the color value at each pixel can be exploited to decide on whether it belongs to the object pattern or background. We employ a color segmentation to extract the object pattern from the image. Let the image I={F, G} where F={f_1, f_2, f_3…, f_n} denotes the set of foreground pixels (or object pattern) with color value c and B={b_1, b_2,…, b_k} denotes the set of background pixels. For each pixel f_i=(x_i, y_i) in I, the algorithm adds a point p_i= (x_i, y_i) to P, i.e. the point set P= {p_1, p_2,…, p_n} is constructed where p_i=f_i, 0<i<n+1. Fig. 6 shows an example in which the foreground pixels are transformed into a set of points.



(a) (b)

Fig. 6. (a) Sampled part of an input image with object Pattern in white color, (b) Corresponding points (black filled circles)



(a) (b)

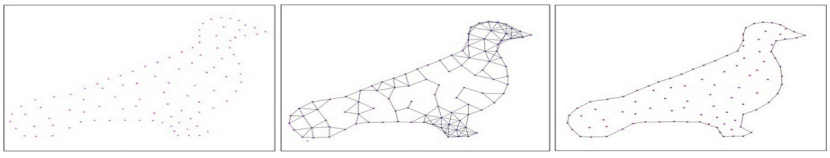
Fig. 7. Illustration of (a) using parameter value l and (b) linking a point from edge (s_1, s_2).

3.2 Geometric Graph Construction

3.2.1 Graph construction

This step constructs a geometric graph G= (V, E) from the point set P, where V is the set of all points in P and E is the set of all edges (p_i,p_j) such that p_i, p_j in P and d(p_i, p_j) < threshold value l. The threshold parameter can be statically defined and provided to the algorithm and hence eliminates the hassles of parameter tuning for the construction of a connected graph. Static value of threshold parameter can be derived through a pixel grid based analysis of the image which is presented in Section 3.2.2. In order to realize the graph, for each point p_i in P, algorithm checks the length to all other p_j in P, if it is less than the threshold l, then it adds the edge (p_i, p_j) to E.

Formally, from a point α_1 in P an edge is created to α_i, if α_i belongs to the interior of a circle centered at α_1 with radius l. In Fig. 7(a), an edge is created from α_1 to α_2 and α_3 as they belong to the interior of circle. Fig. 8(a) and 8(b) show an example point set of bird and its corresponding geometric graph with appropriate length parameter l respectively.



(a) (b) (c)

Fig.8. Left: Input point set, **Middle:** Corresponding geometric graph with appropriate value of l, **Right:** Contour extracted by our algorithm.

3.2.2 Threshold parameter

Now we analyse the pixel grid of a portion of the image in order to derive the value for threshold parameter. Consider the mask shown in Fig. 9 which gives the all possible neighbourhoods of a pixel at position (x, y). Please note that distance between the centres of two adjacent pixels (squares in the Fig.

9) is 1. So, the Euclidean distance needed to connect a point at position (*x, y*) to any points in its horizontal, vertical and diagonal neighbours are 1, 1 and $\sqrt{2}$ respectively.

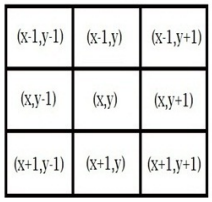


Fig.9. Neighbourhoods of a pixel (*x,y*).

Diagonal length $= \sqrt{(x-(x-1))^2+(y-(y-1))^2} = \sqrt{(x-(x+1))^2+(y-(y-1))^2}$
$= \sqrt{(x-(x-1))^2+(y-(y+1))^2} = \sqrt{(x-(x+1))^2+(y-(y+1))^2} = \sqrt{2}$

As a general observation, in the case of objects with a closed loop, at least one point in a pixel should have a transition to the point in its diagonal neighbour. Moreover, the length to connect a diagonal neighbour is greater than that of its horizontal or vertical neighbours. So providing threshold value as diagonal length will also account for the horizontal and vertical lengths but not vice versa. So in order to get the contour, we have to construct the graph with threshold parameter of at least $\sqrt{2}$ (1.41421356237).

3.3 Point Linking

In this phase, the algorithm extracts the contour from the geometric graph constructed in the previous phase. A pseudo-code of the procedure is presented in the algorithm 1.

The algorithm has been designed for a coordinate system whose origin point, (*0, 0*) lies at the top left which is always the case with images. Degree of a vertex v, is the number of edges in G incident to v. Algorithm starts by selecting the leftmost valid vertex using the function Left_Most_Valid_Vertex(). The leftmost valid vertex is the vertex in G, with least x value and degree greater than 1. If there are multiple leftmost valid vertices, then the one with least y value is selected. The function, Find_Second_Valid_Candidate(v_1) returns the vertex v_2=(x_2, y_2) with respect to vertex v_1=(x_1, y_1) such that $x_2 \geq x_1$ and $y_2 > y_1$. Starting from the initial edge (v_1, v_2) in G, the algorithm successively adds points v_q that make largest turn angle with the boundary edge under consideration((s_2, v_q)) with the constraint that v_q has a degree greater than 1. The function Find_Valid_Candidate(s_1, s_2) returns a vertex v_q such that $\angle(s_1 s_2 v_q) > \angle(s_1 s_2 v_k), \forall (s_2, v_k)$ belong to the graph G. If the degree of vertex v_q is 1 then v_q gets removed from G and the same procedure is repeated to find the next valid candidate v_q.

Algorithm 1 Contour extraction Algorithm

```
1:  procedure EXTRACT_CONTOUR(Geometric Graph G)
2:      v1 = Left_Most_Valid_Vertex()
3:      v2 = Find_Second_Valid_Candidate(v1)
4:      edge = (v1, v2), s1 = v1, s2 = v2, C = {(v1, v2)}
5:      while s2 ≠ v1 do
6:          vq = Find_Valid_Candidate(s1, s2)
7:          s1 = s2, s2 = vq
8:          C = C ⋃ (s1, s2)
9:      end while
10:     return C
11: end procedure
```

The linking process proceeds in anticlockwise order and the turn angle is taken in clockwise direction. In Fig. 7(b), with respect to the edge (s_1, s_2), s_2 is connected with p_1, p_2 and p_k shown in blue color as these three points lie within the circle centered at s_2 having radius l (threshold). It is apparent from the Fig. 7(b) that (s_2,p_k) makes the largest clockwise angle with (s_1,s_2) and hence (s_2,p_k) will be added as the next contour edge. The process stops when the linking process reaches the starting edge. A vertex is considered to be valid iff it has degree greater than 1. Since the contour is a cycle and if a

vertex with degree 1 is selected as a candidate point then there will not be an outgoing path. The presence of such invalid candidates is considered as a noise and hence will be eliminated from the graph.

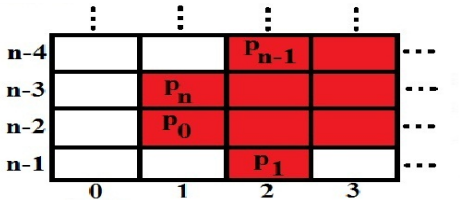


Fig. 10. Part of an image *I*.

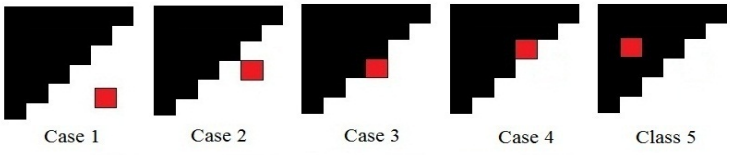


Fig. 11. Possible positions for the occurrence of noise.

It is important to show that the algorithm always terminates by returning closed contours which captures the geometric features of the object boundary. Fig. 10 shows a part of random synthetic image *I*. We argue that the algorithm returns a contour which contains an edge between the last point (p_n) and the first point (p_0). The cyclic property of the contour is ensured by the degree criteria, which says that all the candidate points p_k selected for linking have degree greater than or equal to two. Degree ≥ 2 for a selected candidate point implies that there always exists an outgoing path from each selected candidate point p_k. Moreover, if the selected candidate point has degree > 2, the algorithm will select the one with largest clockwise turn angle as the next candidate point and hence make sure that the set of selected points reflects the geometric features of the object boundary. For instance, in Fig. 10, p_n has 5 neighbor object pixels (or points) each within a distance of 1.415. In the last iteration of point linking phase, the algorithm will select the next candidate point as p_0 because p_0 is the point that makes the largest clockwise turn angle with respect to edge (p_{n-1}, p_n) while processing in anti clock wise direction. If not p_n, there would be any other point lies in the neighborhood of p_0 apart from p_1 as p_0 has a degree>1 (The function Left_Most_Valid_Vertex() make sure this at start of the algorithm). Once the processing reaches the first point, the algorithm terminates. For non-noisy images with single objects, the algorithm always terminates by returning a closed contour.

Fig. 11 shows five possible places for the occurrence of a noisy point. The red point in the Fig. 11 shows noisy data point and black portion shows lower right part of an object. In the first case (noisy point is detached from object), since the noisy point cannot be connected by parameter *l* the point has no impact on extracted contour. In the second case (the noisy point is in diagonal neighbor of exactly one object point), the noisy point is not a valid candidate point because it has only one neighbor and hence degree 1. The noisy point will marginally affect the contour in third and fourth cases by adding a non-object point to contour (false positive) and removing an object point from contour (true negative) respectively. It is to be noted that, as the object size increases, the visual impact of false positives and true negatives on the extracted contour becomes more negligible. The fifth case illustrates the occurrence of a noisy point within the object, and it has no impact on contour since it is not part of contour.

The main idea behind this simple linking procedure is that, if we add an edge other than that makes maximum turn angle, then it will make a fissure in the contour. For non-uniformly distributed points, the value of parameter *l* determines the shape of contour. Moreover, it is not necessary to start with the leftmost valid point, any extreme valid point will generate the same contour where "extreme point" refers to the point with minimum or maximum x value or minimum or maximum y value. Fig. 8(c) shows a sample output of a point linking process.

Fig. 12. Left: A sample contour, **Middle:** Points selected after contour simplification, **Right:** Simplified contour.

3.4 Contour Simplification

In most of the cases the contour extracted in the previous step will contain large number of pixel most of which are redundant. So in order to reduce the space needed to store the contour, it is necessary to eliminate all the least relevant points from the extracted contour. Simplification is based on the fact of the high probability for the existence of collinear points. Suppose 'k' adjacent points are lying on a line, i.e. 'k' points are collinear then all 'k-2' points other than the end points are not required to reconstruct the line. So a compression ratio of k/2 can be achieved by remembering only the end points. In this phase, a walk is done over the extracted contour to remove irrelevant points from it. Fig. 12 shows an example of how a contour is simplified.

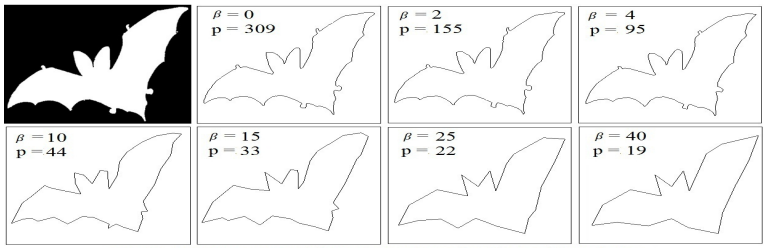


Fig. 13. Contour extracted for various values of 'β' from a binary image (p denotes number of pixels needed to represent the contour).

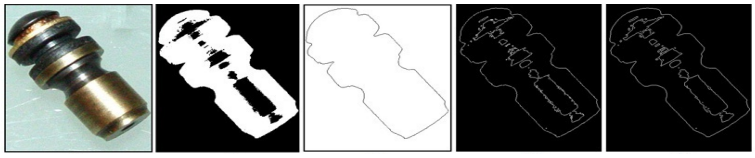
There is also a chance for the presence of small irregularities or projections from the exact contour due to the interference of noise. Those points which make this kind of irregularities can be simply eliminated using an external parameter 'β'. Suppose p_i, p_j, p_k are three adjacent points in contour then p_j is considered as an irregularity when the area of $\Delta p_i p_j p_k$ is less than parameter 'β'. Fig. 13 shows the change in contour with respect to the tuning of parameter 'β'. This is an optional part that can be used if the user is giving more priority to compression than the accuracy.

3.5 Complexity

Let the size of extracted point set be n. The Euclidean graph construction needs $O(n^2)$ time, because while creating the graph, each point may be connected to all other $(n-1)$ points. In point linking phase, from each candidate point, choosing the next candidate point among a set of size $(n-1)$ costs $O(n^2)$ time. The contour simplification phase requires $O(n)$ time. So the overall asymptotic time complexity is $O(n^2)$.

4 EMPIRICAL STUDY

The algorithm was implemented using MATLAB, C++ and the results were displayed using OpenGL.



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

Fig. 14. (a) Original image of a machine part, (b) Binary image of machine part, (c) Contour of machine part extracted by our algorithm, (d) Output of Sobel edge detector, (e) Output of Canny edge detector.

4.1 Qualitative Results

We have used binary images taken from MPEG7 CE Shape–1 Part B database [11] that contains 1400 images in 70 categories, to test the proposed approach. Each category consists of 20 samples where each sample varies from one another with respect to rotation, size, position and image resolution. Experiments were conducted on few random images as well. Fig. 14 shows the contour of a machine part extracted from an illuminated colour image having visual cues like texture and brightness. It is to

be noted that, when dealing with colour or grey-scale images, the image has to be converted to binary before applying our algorithm as shown in Fig. 14.

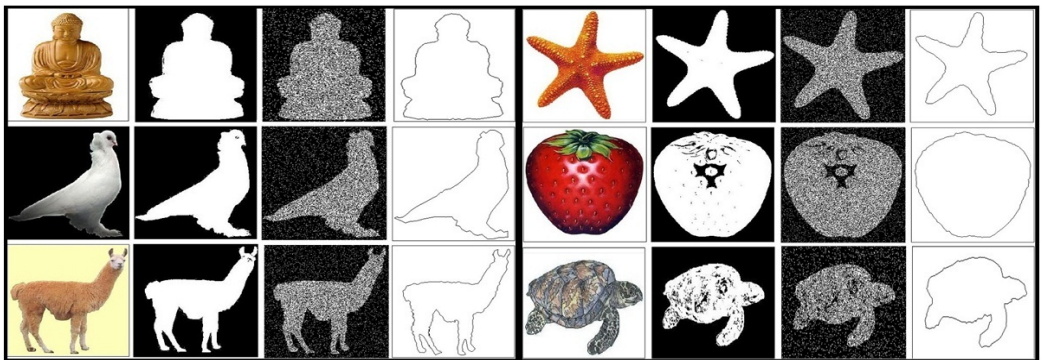


Fig. 15. Results of the proposed algorithm for some random color images taken from Caltech database [3] **Left to Right:** Input image, Binarized image, Image with Gaussian noise, Output of our algorithm

4.2 Robustness to Noise

To evaluate the performance of the approach on noisy images, we injected appropriate amount of Gaussian noise into the images taken from MPEG7 CE Shape–1 Part B [11] and Caltech [3] databases using some well-defined functions in MATLAB. Object contours extracted by the proposed algorithm from a few random color images are shown in Fig. 15. All these images are first binarized and then injected with Gaussian noises before fed into our contour extraction algorithm as illustrated in Fig. 15.

A qualitative comparison of the results generated by our algorithm with the results of Sobel and Canny edge detectors reveals the strength of the proposed approach, especially when applied to noisy binary images as shown in Fig. 16. For all the images in Fig. 16, our approach returns the contour with a clean background whereas Sobel and Canny detectors fail to return contours without background noise. The reason behind the failure of traditional edge detectors is that the noise makes a sharp transition in black background which will be misinterpreted as an edge. So, in the result of edge detectors [4, 5] there will be the presence of misinterpreted edges due to background noise and hence the methods become comparatively ineffective.

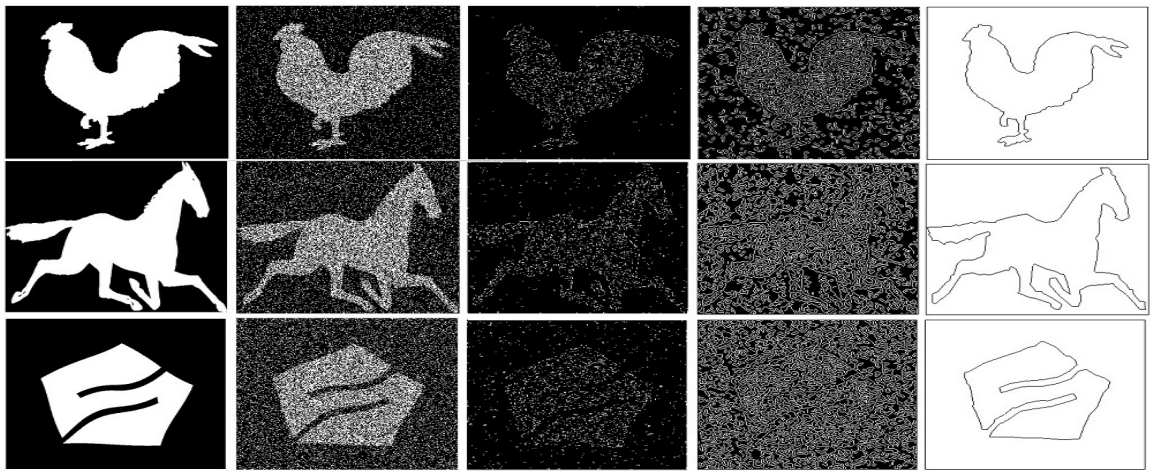


Fig. 16. Comparison of our method with Sobel and Canny edge detectors in noisy images. **Left to Right:** Original image, Input image with 30%, 50% and 70% Gaussian noise respectively in each rows, Output of Sobel edge detector, Output of Canny edge detector, Result of our algorithm.

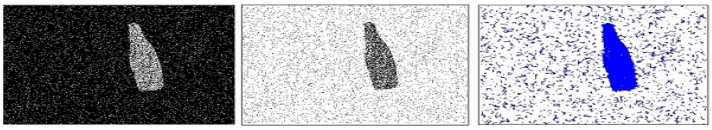


Fig. 17. Illustration of geometric graph construction in the presence of noise. **Left to Right:** Image with background noise, Extracted point set, Geometric graph constructed (blue color).

As opposed to the edge detectors, our algorithm completely rely on proximity and orientation of points and that is one of the major reasons why our results are noise free. Since we are dealing with the binary images, we restrict the intensity values of the injected synthetic Gaussian noise to either 0 or 255. The extracted pixels with intensity value 255 contain object pixels along with some noisy pixels which have the color value 255. As a consequence, few portion of the noise is already eliminated in the point extraction phase. Fig. 17 shows an example of points extracted from a noisy image. While constructing graph, edges are created only if its length is less than or equal to the parameter length l (1.415). As shown in Fig. 17, there does not exist edges between any noisy points or between noisy and object points because they lie a distance greater than 1.415 apart. However, from our experiments, we observed that once the Gaussian noise goes above 70%, edges between noisy points are created and this will affect the subsequent contour extraction.

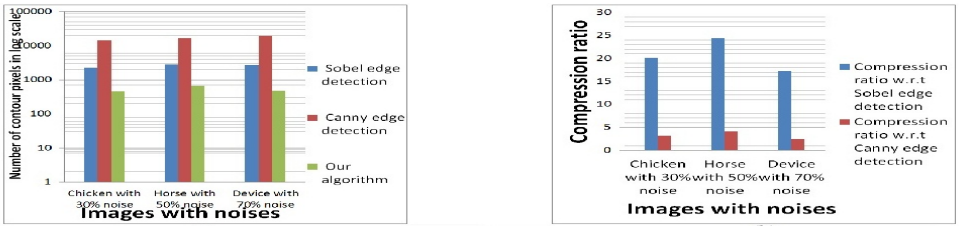
Overall, one can observe that the contours generated by the proposed approach preserve fine details and features of the object pattern, especially in the case of horse, device (Fig. 16), llama and star fish (Fig. 15). Further, fine quality contours were extracted from images with 70% synthetic Gaussian noise (device in Fig. 16) which is much better when compared to the automatic scale selection based method that has a capacity to cope with 10% Gaussian noise [8].

4.3 Compression

Compression is computed by counting the number of pixels in the contours detected by each algorithm. Graphs plotting the comparison with Sobel and Canny edge detection techniques are shown in Fig. 18. In Fig. 18(a), one can observe that the number pixels extracted as the contour are relatively low by our method. Even for noisy images such as device which has 70% synthetic Gaussian noise, our method extracted minimal number of contour pixels as opposed to Canny and Sobel edge detectors. The compression ratio is computed by taking the fraction of number of pixels in the contour extracted by our algorithm and the same in the contour detected by other edge detectors as follows:

$$\text{Compression ratio} = \frac{\text{number of contour pixels in the proposed algorithm}}{\text{number of contour pixels in edge detector}}$$

Indeed, it's a comparative compression ratio that we have presented here. Compression ratio gives a measure of the level of compression achieved by our method which is found to be very high even for the noisy images. On average, the number of pixels reduced to 2.489% - 24.394% as shown in Fig. 18(b).



(a) (b)

Fig. 18. Comparison of contour extraction algorithms for various noisy images in terms of (a) number of contour pixels and (b) compression ratio.

4.4 Comparison

We compare our method with two other contour extraction techniques: a geometry based technique (GBA) [15] and a pixel based contour tracing method called bug followers algorithm (BFA) [13, 14, 16] that deals with binary images.

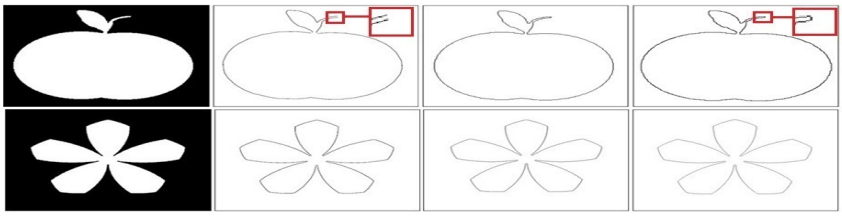


Fig 19. A qualitative comparison of contours extracted by different algorithms for binary images, apple and device **Left to Right:** Input image, Outputs of Geometry based algorithm, Bug followers algorithm and Our algorithm.

The geometry based algorithm [15] makes use of techniques such as clustering, linking and path simplification for contour detection and can handle color/grey scale pictures. Contrastingly, though geometry based approach that makes use of proximity and orientation information of points, our algorithm can extract only closed contours in binary images. However, GBA requires $O(n^2 log\ n)$ time for the entire computation whereas our algorithm runs in $O(n^2)$ which is comparatively better. For feature point extraction, GBA completely rely on Sobel edge detector, whereas our method uses color value at each pixel for the point extraction. Geometric algorithm needs parameters at each stage and hence heavily dependent on parameter tuning. As opposed to this, our algorithm requires parameters such as color value c which doesn't require any tuning (object color is evident in most of the cases and the user can supply the color value directly), threshold parameter l is statically defined and an optional simplification parameter 'β'. In some cases, the output of geometric algorithm may not give the required contour as shown in the first row of Fig. 19. In Fig. 19, for apple image, GBA generates an open contour (as highlighted in the red box) whereas both BFA and our method constructs the required closed contour.

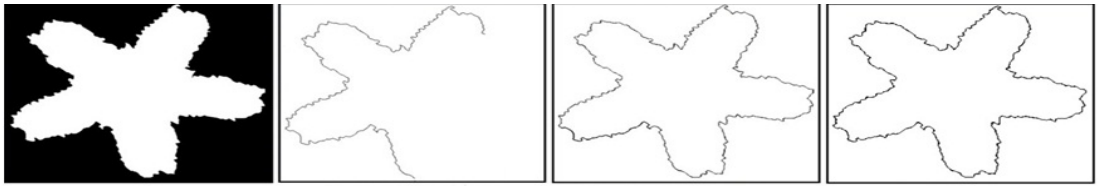


Fig. 20. Contours extracted by BFA and our approach **Left to Right:** Input image, Output of BFA without backtracking, Output of BFA with backtracking, Output of our algorithm.

The proposed contour extraction technique differs from bug followers algorithm (BFA) [13, 14, 16] mainly in the fact that the former is a geometry based approach whereas BFA is a traditional pixel based technique. In several instances, BFA requires backtracking to produce the exact contours, which increases the complexity of the entire process. If backtracking is not applied, it constructs only partial contours as illustrated in the second column of Fig. 20. Table 1 reports the size of the contours (in terms of pixels) extracted by the proposed algorithm and BFA for a few binary images. Compared to BFA, our algorithm generates contours which are more compact for all the tested images.

Image	Object size	Contour size in terms of pixels	
		BFA	Our Algorithm
Apple-2	34534	681	299
Classic-1	28974	873	384
Cup-1	32682	950	431
Device0-5	58526	1838	692

Table 1. Comparison of proposed algorithm and Bug followers algorithm in terms of contour pixels for images taken from MPEG7 CE Shape-1 Part B database [11].

4.5 Limitations & Future Work

One of the major limitations of the proposed algorithm is that it is designed to work with binary images with single object embedded in it. When dealing with colour or grey scale images, one has to convert these images to its corresponding binary images. However, in some of the cases, this conversion ends up in binary images with undesired loops or regions on the objects. In such cases, contour extraction by our algorithm is difficult because false positive object points will get extracted in the extraction phase and subsequently, this affects the graph construction and point linking phases.
Currently, we focus on extending this algorithm to incorporate the following objectives:
Contour extraction from grey scale and colour images with multiple objects.
Extraction of open contours and hole boundaries from images.
Development of an automated medical diagnosis system using contour matching is also under our consideration.

5 CONCLUSION

We have used a geometric approach for contour extraction of digital binary images that is found to be very effective and has the advantage that the method can be applied to both vector and raster graphics. Unlike pixel based contour extraction, our algorithm found to extract more compact and smooth contours and is also capable of generating contours for images with considerable noise, where the standard edge detecting algorithms like Sobel edge detector and Canny edge detector fail. Robustness to background noise, especially without applying any preprocessing noise filtering methods, makes the proposed approach unique and hence several image segmentation applications dealing with input binary images having background noise (MRI scans or satellite images) will find our method very appealing and useful. The compression ratio achieved for noisy images by our algorithm is very high. On average, the number of pixels reduced to 2.489% - 24.394% (Fig. 18(b)). In our experimental study, the default value of the threshold is found to be 1.415, which is theoretically supported by a pixel grid analysis.

REFERENCES

[1] Alpert, S.; Meirav, G.; Nadler, B.; Ronen, B.: Detecting Faint Curved Edges in Noisy Images, Computer Vision – ECCV 2010, Lecture Notes in Computer Science, Volume 6314, 2010, pp 750-763, http://dx.doi.org/10.1007/978-3-642-15561-1_54.

[2] Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J.: Contour Detection and Hierarchical Image Segmentation, IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011, http://dx.doi.org/10.1109/TPAMI.2010.161.

[3] Caltech 101 dataset, http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[4] Canny J.-F.: A computational approach to edge detection, IEEE Transaction on Pattern Analysis and Machine Intelligence 8 (6) (1986) 679–698, http://dx.doi.org/10.1109/TPAMI.1986.4767851.

[5] Duda, R.-O.; Hart, P.-E.: Pattern Classification and Scene Analysis. New York: Wiley, 1973, http://dx.doi.org/10.2307/1573081.

[6] Gonzalez, R.-C. ; Woods, R.E.: Digital Image Processing, 3rd edition. Pearson Prentice Hall, 2008, http://dx.doi.org/10.1117/1.3115362.

[7] Gregory, T.-M.; Cheng, L.: Atlas of Genitourinary Pathology, Springer; 2011 edition, http://dx.doi.org/10.1007/978-1-84882-395-2.

[8] Lindeberg, T.: Edge detection and ridge detection with automatic scale selection, Proceedings of Computer Vision and Pattern Recognition, 1996, http://dx.doi.org/10.1109/cvpr.1996.517113.

[9] Marr, D.; Hildreth, E.: “Theory of edge detection,” Proceedings of the Royal Society of London, 1980, http://dx.doi.org/10.1098/rspb.1980.0020.

[10] Martin, D.; Fowlkes, C.; Malik, J.: Learning to detect natural image boundaries using local brightness, colour and texture cues, PAMI, 2004, http://dx.doi.org/10.1109/TPAMI.2004.1273918.

[11] MPEG7 core experiment CE Shape1- http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz.

[12] Mukesh, C.; Mukesh M.-C.; Gadiya, R.-C.; Frederick M.-C.-H., Jr.: Survey of Image Denoising Techniques, Proc. of GSPx, Santa Clara Convention Center, Santa Clara, CA, pp. 27-30, 2004.
[13] Pavlidis, T.; Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, Maryland, 1982, http://dx.doi.org/10.1007/978-3-642-93208-3.
[14] Pavlidis, T.: Contour Filling in Raster Graphics, Proc. ACM-SIGGRAPH, Dallas, Aug. 5-7, 1981, pp. 29-36, http://dx.doi.org/10.1145/965161.806786.
[15] Pedro, J.; Tejada; Qi, X.; Jiang, M.: Computational Geometry of Contour Extraction, In Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG'09), pages 25-28, August 17-19, 2009, http://dx.doi.org/10.1007/978-3-642-22619-9_2.
[16] Pratt, W.-K.: Digital Image Processing 4th edition, Wiley publications, 2007, http://dx.doi.org/10.1117/1.2744044.
[17] Prewitt, J.-M.-S.: Object enhancement and extraction, In Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld. Eds. Academic Press, New York, 1970.
[18] Rajalingappa, S.; Ramamoorthy, B.; Ramanathan, M.: Unsupervised shape classification of convexly touching coated parts with different geometries, Computer Aided Design and Applications, Volume 11, Issue 3, 2014, pp. 312-317, http://dx.doi.org/10.1080/16864360.2014.863501.
[19] Ren, X.; Fowlkes, C.; Malik, J.: Scale-invariant contour completion using conditional random fields, ICCV, 2005, http://dx.doi.org/10.1109/iccv.2005.213.
[20] Roberts, L.-G.: Machine perception of three-dimensional solids, In Optical and Electro-Optical Information Processing, J. T. Tippett et al. Eds. Cambridge, MA: MIT Press, 1965.
[21] Sharma, S.; Khanna, P.: ROI Segmentation using Local Binary Image, IEEE International Conference on Control System, Computing and Engineering, pp. 136-141, 2013, http://dx.doi.org/10.1109/ICCSCE.2013.6719947.
[22] Toussaint, G.-T.; Course Notes: Grids, connectivity and contour tracing, http://dx.doi.org/10.1.1.55.3197.
[23] Yiannis, S.-C.; Vassilis, G.-K.; Toutouzas, K.; Giannopoulos, A.; Chouvarda, I.; Riga, M.; Antoniadis, A.-P.; Cheimariotis, G.; Doulaverakis, C.; Tsampoulatidis, I.; Bouki, K.; Kompatsiaris, I.; Stefanadis, C.; Maglaveras, N.; Giannoglou, G.-D.: Clinical Validation of an Algorithm for Rapid and Accurate Automated Segmentation of Intracoronary Optical Coherence Tomography Images, International Journal of Cardiology (Elsevier), 2014, http://dx.doi.org/10.1007/978-3-319-00846-2_93.
[24] Zhu, Q.; Song, G.; Shi, J.: Untangling cycles for contour grouping, ICCV, 2007, http://dx.doi.org/10.1109/ICCV.2007.4408929.