CT-shape: Coordinated triangle based reconstruction from dot patterns and boundary samples

Safeer Babu Thayyil, Amal Dev Parakkat, Ramanathan Muthuganapathy

CT-shape: Coordinated triangle based reconstruction from dot patterns and boundary samples

Safeer Babu Thayyil [1], Amal Dev Parakkat [2], and Ramanathan Muthuganapathy [1]

[1] *Advanced Geometric Computing Lab., Department of Engineering Design, Indian Institute of Technology Madras, India*
[2] *Computer Science Laboratory of Ecole Polytechnique (LIX), Ecole Polytechnique CNRS, Paris, France*

Abstract

Given a set of points $S \in \mathbb{R}^2$, reconstruction is a process of identifying the boundary edges that best approximates the set of points. In this paper, we propose a unified algorithm for reconstruction that works for both dot patterns as well as boundary samples. The algorithm starts with computing the Delaunay triangulation of the given point set and edges are iteratively removed based on the structure of a pair of triangles. Further, we also propose additional criteria for removing edges based on characterizing a triangle and using degree constraint. Unlike the existing algorithms, the proposed approach requires only a single pass to capture both inner and outer boundaries irrespective of the number of objects/holes. Moreover, the same criterion has been employed for both inner and outer boundary detection. The experiments show that our approach works well for different kinds of inputs. We have done extensive comparisons with state-of-the-art methods for various kinds of point sets including varying the sampling density and distribution and found to perform better or on par with them.

1 Introduction and related works

Given a set of points S lying on a plane, sampled from an object, the reconstruction is a task of embodying the boundary edges (inner and outer boundaries) that best approximates its geometrical identity. In this paper, the point samples are assumed to be derived from a smooth closed curve(s). When the sample points are derived only from the boundaries of the curve(s) (Figure 1(a)), termed as boundary samples, then the reconstruction is generally called as *curve reconstruction* (Figure 1(b)). On the other hand, sample points, in addition to boundaries, can be acquired from the interior to the curve(s) (Figure 1(c)). This sampling is termed as dot pattern and the corresponding reconstruction is called *shape reconstruction* (Figure 1(d)). Devising a unified algorithm that works for both types of input point sets again increases the level of hardness.

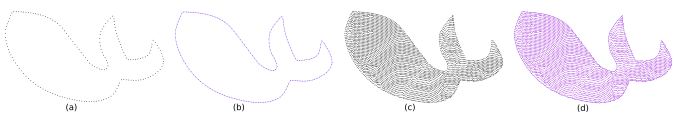

Figure 1: (a) Boundary sample (b) Reconstruction from boundary sample (c) Dot pattern (d) Reconstruction from dot pattern.

Reconstruction algorithms are taxonomised in different ways. It can be based on Delaunay triangulation and non-Delaunay triangulation or curve/shape/unified reconstruction. Some algorithms are designed only for outer boundary detection while others are designed for both outer boundary as well as inner boundary.

Table 1: Strengths and weaknesses of different reconstruction algorithms

Algorithm	Unified	Hole	# Parameters	Multiple Object	Unstructured Hole
α-shape [4]	Y	Y	1	Y	Y
Crust [1]	N	Y	0	Y	Y
nn-crust [2]	N	Y	0	Y	Y
χ-shape [3]	Y	N	1	N	NA
simple-shape [5]	Y	N	3	N	NA
deGoes et. al [6]	N	Y	1	Y	Y
RGG [11]	Y	Y	0	N	N
WDM-crust [12]	N	N	0	Y	N
ec-shape [7]	Y	Y	0	N	Y
HNN-crust [8]	N	Y	0	Y	N
Crawl [10]	N	Y	0	Y	Y
Peel [9]	N	Y	0	Y	Y
Our Algo	Y	Y	0	Y	Y

Table 1 summarizes the strengths and weaknesses of a few of the reconstruction algorithms. Even though there are a lot of works in the area of reconstruction, not all algorithms can handle both types of input - dot patterns and boundary samples. Examples for unified algorithms are α-shape, RGG and ec-shape. Though χ-shape and simple-shape can handle both dot patterns and boundary samples, it can generate only a simple closed curve as output and cannot handle holes. Though unified algorithms such as RGG, *ec*-shape can capture holes, RGG can work only for restricted hole structures and *ec*-shape uses different strategies to capture holes. Crawl and peel can capture different shaped holes but can work only for boundary samples.

α-shape can work for both types of input and independent of the hole structure but requires a parameter α to be tuned. Other approaches such as χ-shape, simple-shape are also parametric algorithms whereas RGG, *ec*-shape, crawl and peel are non-parametric algorithms. It is quite a tedious task to tune the parameter(s) to get the desired output. RGG, *ec*-shape, and χ-shape cannot handle multiple objects.

In this paper, we propose a unified algorithm for the reconstruction of outer boundaries as well as inner boundaries without any user intervention. The following are our **major contributions**: (a) A unified approach for dot pattern and boundary samples with and without holes. (b) The algorithm uses the same strategy for capturing both hole boundary as

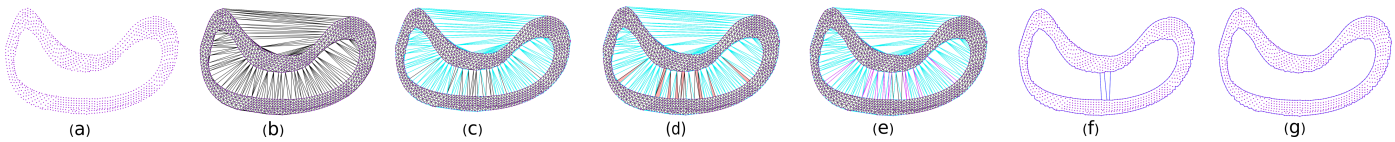

Figure 2: (a) A point set. (b) DT of the point set. (c) marked shared edges of all CT (in cyan). (d) Skinny triangles (in red). (e) marked edges of skinny triangles (in magenta). (f) Graph formed after removing marked edges. (g) Graph after the application of degree constraint.

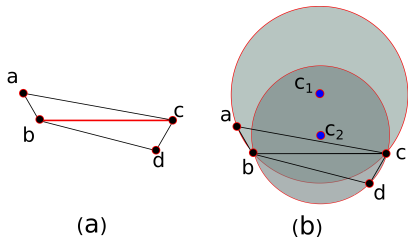

Figure 3: (a) Neighboring triangles $\triangle_{abc}$ & $\triangle_{bdc}$ with shared edge bc. (b) Coordinated triangles $\triangle_{abc}$ & $\triangle_{bdc}$ with circumcenters c_1 & c_2 lying on the same side of the shared edge bc.

well as outer boundary. (c) Our algorithm needs only a single pass irrespective of the number of holes/objects.

2 Definitions

DEFINITION 1 *Neighboring triangles: Two triangles are said to be neighboring triangles if they share an edge (Figure 3(a)).*

DEFINITION 2 *Coordinated triangles: Neighboring triangles are termed as coordinated triangles if their circumcenters lie on the same side of the shared edge (Figure 3(b)).*

DEFINITION 3 *Skinny triangle: A skinny triangle is a thin acute triangle whose base is much smaller than its height.*

DEFINITION 4 *Degree constraint: Only two shorter edges are retained from a vertex (point) and all other edges are removed from that vertex (point).*

3 Algorithm

For a given point set (Figure 2(a)), the algorithm starts with computing the DT of the given point set (Figure 2(b)).

3.1 Marking a shared edge in CT

For each triangle T ∈ DT, the algorithm checks for CT with respect to T. If CT exist, the shared edge between those two triangles is marked (cyan edges in (Figure 2(c)).

3.2 Marking edges from a skinny triangle

Using an angle of 9° for the smallest angle in skinny triangles (shown in red in Figure 2(d)), their two long edges are marked. Figure 2(e) shows all the marked edges so far, in cyan and magenta.

3.3 Applying degree constraint

A graph G is formed from the set of unmarked edges from DT (Figure 2(f)). Since the edges in DT are removed arbitrarily based on CT and skinny triangles, there are possibilities of the presence of non-manifold edges. In order to maintain the output as manifold (for e.g., if there exists only outer boundary, then it should be topologically equivalent to a circle), we impose a degree constraint (Definition 4) on each vertex. Figure 2(g) shows the graph, which is the final reconstructed boundary (in blue) after checking for degree constraint for the point set shown in Figure 2(a).

Algorithm 1 *Complete_Reconstruct*(S)

Input: Input point set, S.
Output: Reconstructed Output R.
1: Construct Delaunay triangulation, $DT(S)$.
2: **for** each triangle T **do**
3: Take all three neighboring triangles and check whether they constitute coordinated triangles.
4: Mark the shared edges for all coordinated triangles.
5: Identify the skinny triangles having less than 10° and mark the two longest edges.
6: **end for**
7: Create a graph G with all unmarked edges of DT (if all three edges are unmarked, they are not considered).
8: Apply degree constraint on all vertices of G.
9: **return** G as CT-shape

The pseudo-code for the algorithm for reconstruction, given a set of points S is delineated in Algorithm 1. The running time complexity of the algorithm can be shown to be $O(n \log n)$ where n is the number of points in S.

4 Results & Discussions

Our algorithm (Algorithm 1) is implemented in C++ with CGAL (Version: 4.6) libraries and visualized in OpenGL and tested in MacOS 10.12.3. The input point sets (dot patterns and boundary samples) consist of points from simple objects, objects with multiple holes, objects with multiple components, objects with non-divergent concavities etc. The algorithm has also been tested with different sampling densities and distributions. Figure 4 shows some of the results of our algorithm for various dot patterns and boundary samples. The figure shows that our algorithm can generate good results for both kinds of inputs with divergent features.

4.1 Comparison with existing algorithms

Here we considered five algorithms (α-shape, χ-shape, simple-shape, RGG, ec-shape with ours) for dot pattern and nine algorithms (α-shape, χ-shape, simple-shape, RGG, ec-shape, Crawl, HNN-crust, Peel, WDM-crust with ours) for boundary samples for the sake of comparison. It may be noted that HNN-crust, WDM-crust, Crawl and Peel do not work for dot patterns and hence they have been included only for the comparison of results for boundary samples as input. In all the comparison results, we use circles to denote regions where boundaries are not well-approximated.

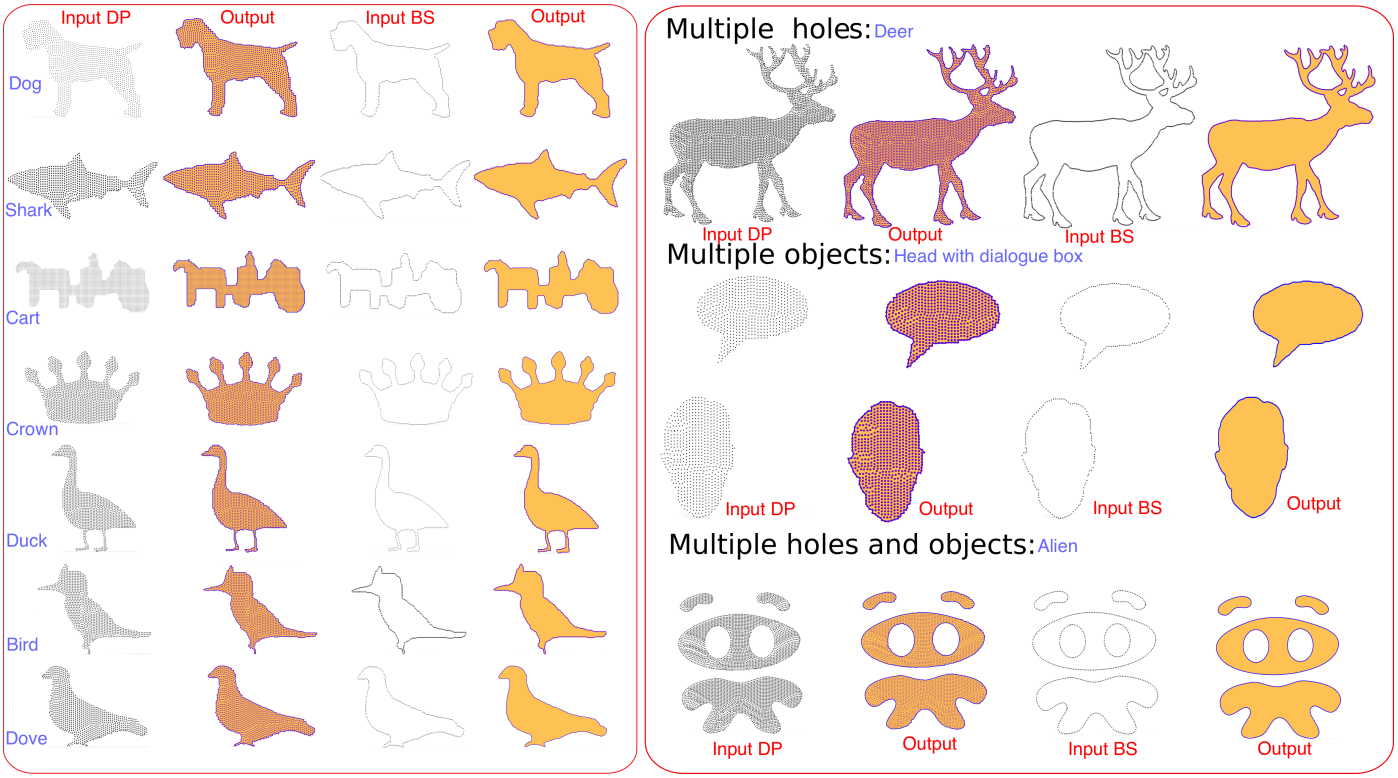

Figure 4: Results of our algorithm (CT-shape) for various features like concavity, multiple holes (deer has two holes), multiple components etc. Input DP = Input dot pattern, Input BS = Input boundary sample, Output = Output of our algorithm.

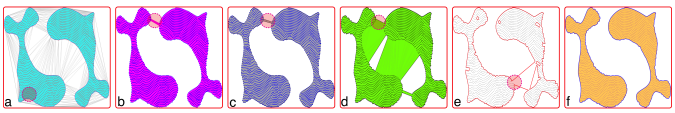

Figure 5: Multiple objects (dot pattern): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) our result (CT-shape). Some algorithms (as indicated in circles) have resulted in single object even when multiple objects are present.

4.1.1 Qualitative Comparison

Capturing multiple objects is a challenge for many algorithm as they work only for single component only. Figures 5 and 6 show the comparison results for multiple objects. Figures 7 and 8 show the result of various algorithms for a point set sampled from an object with multiple holes. Our algorithm has captured all the holes reasonably well.

4.1.2 Quantitative Comparison

We made use of L^2-error norm [3] to compare the results, which is defined as (C and P are the ground truth and reconstructed result respectively):

$$L^2 = \frac{area((C-P)\bigcup(P-C))}{area(C)} \tag{4.1}$$

Figure 9 shows the point density versus L^2 error plots of our experimentation on various point sets extracted from the boundary of the country shapes (Paraguay and Spain) for both dot patterns and boundary samples. From Figure 9, it is clear that our algorithm works better or on par with various unified reconstruction algorithms.

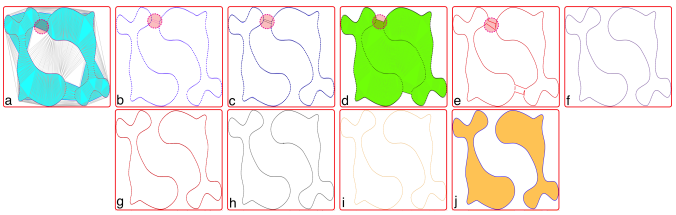

Figure 6: Multiple objects (boundary sample): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) Crawl (g) HNN-crust (h) Peel (i) WDM-crust (j) our result (CT-shape). A few of the algorithms (as indicated in circles) have resulted in single object even when multiple objects are present.

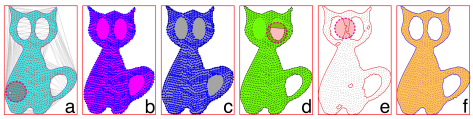

Figure 7: Object with holes (dot pattern): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) our result (CT-shape). χ-shape and simple-shape capture only outer boundaries. RGG works only if the holes are body-arm structured. ec-shape overdigs the holes.

4.1.3 Varying point distributions

Figure 10 shows the comparison of our results with other unified algorithms for various point distributions. The four instances of point distributions we used for experimentation are: Dense Boundary Dense Internal (DBDI), Dense Boundary Sparse In-

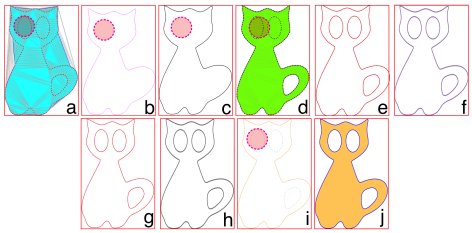

Figure 8: Object with holes (boundary sample): Results of (a) α-shape (b) χ-shape (c) simple-shape (d) RGG (e) ec-shape (f) Crawl (g) HNN-crust (h) Peel (i) WDM-crust (j) our result (CT-shape).

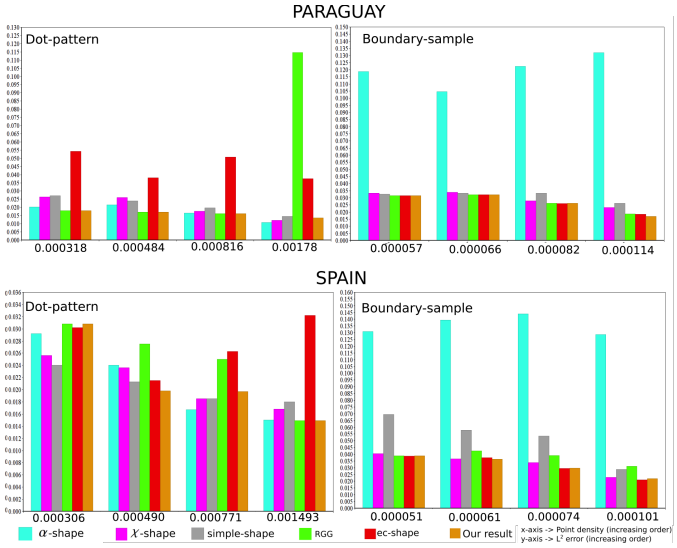

Figure 9: Exemplification of performance comparison on different point densities of different country shapes.

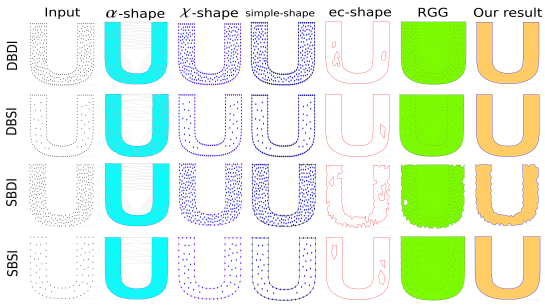

Figure 10: Results showing the performance of different algorithms on various point distributions.

ternal (DBSI), Sparse Boundary Dense Internal (SBDI) and Sparse Boundary Sparse Internal (SBSI). Our algorithm has captured the details quite well except in the case of SPDI.

5 Conclusion

In this paper, we devised a unified reconstruction algorithm and showed it works irrespective of the type of point set (dot pattern or boundary sample). The algorithm is easy to implement and proven to give good results under various point densities and distributions. In contrast to other unified algorithms, our algorithm needs only a single pass to detect the boundaries (both inner and outer) irrespective of the number of holes/objects. As a future work, we would like to extend the algorithm to look into various other challenging tasks like handling point sets with noise and outliers. We are also working on the extension of the proposed algorithm to higher dimensions.

References

[1] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. In *Graphical Models and Image Processing*, pages 125–135, 1998.

[2] Tamal K. Dey and Piyush Kumar. A simple provable algorithm for curve reconstruction. In *SODA'99*, pages 893–894, 1999.

[3] Matt Duckham, Lars Kulik, Michael F. Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41(10):3224–3236, 2008.

[4] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–558, 1983.

[5] Amin Gheibi, Mansoor Davoodi, Ahmad Javad, Fatemeh Panahi, Mohammad M Aghdam, Mohammad Asgaripour, and Ali Mohades. Polygonal shape reconstruction in the plane. *IET computer vision*, 5(2):97–106, 2011.

[6] Fernando de Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. An Optimal Transport Approach to Robust Reconstruction and Simplification of 2D Shapes. *Computer Graphics Forum*, pages 1593–1602, 2011.

[7] Subhasree Methirumangalath, Shyam Sundar Kannan, Amal Dev Parakkat, and Ramanathan Muthuganapathy. Hole detection in a planar point set: An empty disk approach. *Computers & Graphics*, 66:124–134, 2017.

[8] Stefan Ohrhallinger, Scott A. Mitchell, and Michael Wimmer. Curve reconstruction with many fewer samples. *Computer Graphics Forum*, 35(5):167–176, 2016.

[9] Amal Dev Parakkat, Subhasree Methirumangalath, and Ramanathan Muthuganapathy. Peeling the longest: A simple generalized curve reconstruction algorithm. *Computers & Graphics*, 2018.

[10] Amal Dev Parakkat and Ramanathan Muthuganapathy. Crawl through Neighbors: A Simple Curve Reconstruction Algorithm. *Computer Graphics Forum*, pages 177–186, 2016.

[11] Jiju Peethambaran and Ramanathan Muthuganapathy. A non-parametric approach to shape reconstruction from planar point sets through Delaunay filtering. *Computer-Aided Design*, 62:164 – 175, 2015.

[12] Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. A Voronoi based labeling approach to curve reconstruction and medial axis approximation, 2015.