Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Crawl through Neighbors: A Simple Curve Reconstruction Algorithm

Amal Dev Parakkat and Ramanathan Muthuganapathy

Advanced Geometric Computing Lab, Department of Engineering Design, Indian Institute of Technology Madras, India

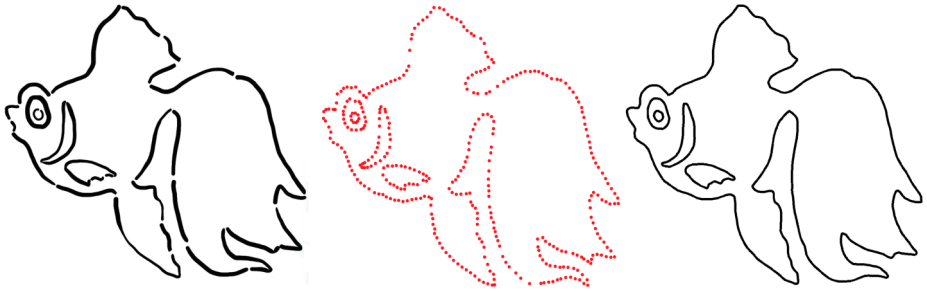

Figure 1: *A sample user drawn sketch, Sampled point set and Connected sketch*

Abstract

Given a planar point set sampled from an object boundary, the process of approximating the original shape is called curve reconstruction. In this paper, a novel non-parametric curve reconstruction algorithm based on Delaunay triangulation has been proposed and it has been theoretically proved that the proposed method reconstructs the original curve under ε-sampling. Starting from an initial Delaunay seed edge, the algorithm proceeds by finding an appropriate neighbouring point and adding an edge between them. Experimental results show that the proposed algorithm is capable of reconstructing curves with different features like sharp corners, outliers, multiple objects, objects with holes, etc. The proposed method also works for open curves. Based on a study by a few users, the paper also discusses an application of the proposed algorithm for reconstructing hand drawn skip stroke sketches, which will be useful in various sketch based interfaces.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Given a point set $S \in R^2$ (where $S = \{v_1, ..., v_n\}$) sampled from an unknown curve $\sum$, the curve reconstruction problem is to construct a polygonal chain from S which best approximates $\sum$. It is a challenging problem because of the fact that the problem is ill-posed [Ede98]. It is a relevant problem because of its wide applications in various fields like computer graphics, computational geometry, computer vision, image processing etc.

1.1. Related Work

One of the earliest works in this area is α-shape [EKS83], proposed as a generalisation of the convex hull of a set of points. The shape of the point set is characterised using its Delaunay triangulation. The output depends on the parameter α.

Figueiredo and Gomes [FMG] proposed an Euclidean minimum spanning tree to approximate the curve. The algorithm initially finds an approximate single open curve. It then uses a heuristic-based approach for reconstructing disconnected and closed curves, if any.

'Crust' is a family of algorithms that uses both the Voronoi diagram as well as the Delaunay triangulation of the input point set S, either independently or in tandem. The crust [ABE98] contains an edge if and only if there exists an edge in Delaunay triangulation of $S \bigcup V$, where V are the vertices of the Voronoi diagram, both of whose endpoints are in S. Nearest Neighbor crust (nn-

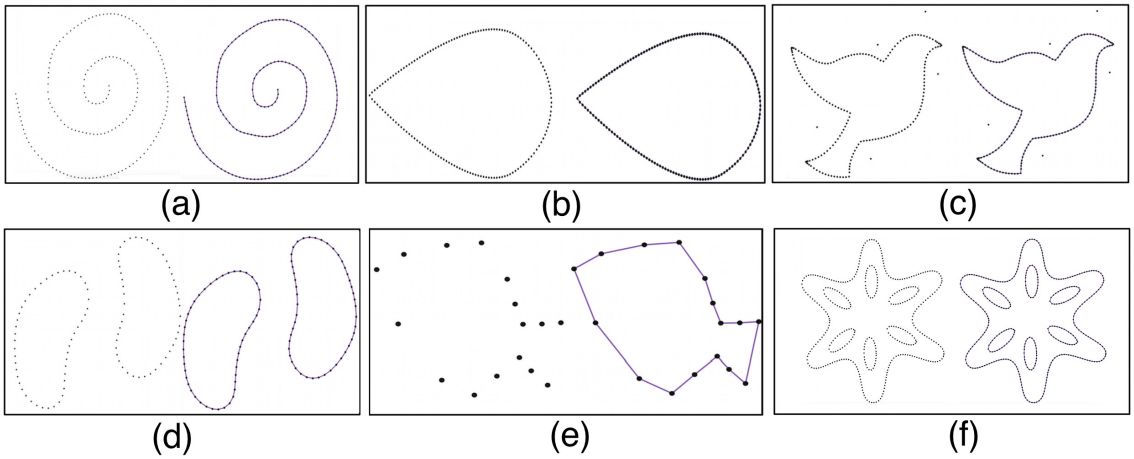

Figure 2: *Point set with different features along with our output , (a) Open curve, (b) Sharp corner, (c) Object with outliers, (d) Multiple objects, (e) Sparse point set, (f) Object with holes.*

crust) [DK99] contains edges between all nearest neighbours in the input point set along with the shortest edge from vertices of degree one which makes an angle of more than $\pi/2$. The algorithm is non-parametric and works in any number of dimensions. It has been proved that the nn-crust will approximate the input curve under certain sampling conditions. Conservative crust [DMR00], a parametric approach, contains edges after removing certain edges of the Gabriel graph generated from the Delaunay triangulation whose B-disk is a Voronoi-disk.

Giesen [Gie99] proposed a travelling salesman sroblem (TSP) based approach to generate a closed polygon. The algorithm is designed assuming the input is a single closed curve. The algorithm was further improved by Dey and Wenger [DW01] to handle open curves with corners and end points based on normal and angle information.

de Goes et.al. [dGCSAD11] proposed an optimal transport based approach that generates the final curve by greedily relocating vertices and collapsing edges of the Delaunay triangulation of the input point set for optimizing the local assignment and for minimizing the increase of total transport cost between the point set and triangulation respectively. The main advantage of the algorithm is that it can reconstruct the approximate curve from a noisy point set.

There are quite a few approaches viz. [DKWG08, GDJ*11, PM15, MPM15] that can be classified as Delaunay sculpting methods which remove edges in a particular order. χ-shape [DKWG08] generates a simple polygon by successively removing Delaunay edges based on a user given parameter χ. In simple shape [GDJ*11], the edges are removed if they satisfy a set of parameters and their selection criteria. It also ensures a simple polygon as an output. Shape hull [PM15], a non-parametric algorithm removes a triangle if its circumcenter lies outside the shape and reconstructs a single curved boundary. The *ec*-shape algorithm is a non-parametric algorithm for reconstructing a simple polygon. In *ec*-shape [MPM15], an exterior edge gets removed if it is longer than the local neighbouring edges.

Water Distribution Model-crust (WDM-crust) [PPM15] is based on a water flow analogy on the Voronoi diagram of S. The algorithm is non-parametric and can handle outliers in the input point set.

Most of the existing curve reconstruction algorithms depend on tuning a parameter, and at times, require more than one parameter to be tuned. The process of tuning even a single parameter is very tedious. There is no well-defined approach, and a trial and error method is usually needed to find parameter(s) that might generate the desired output. For example α-shape [EKS83], χ-shape [DKWG08] etc. require a single parameter, Gathan [DW01] requires two parameters and simple shape [GDJ*11] requires three parameters to be tuned. Other approaches that also use parameter(s) are conservative-crust [DMR00] and optimal transport [dGCSAD11].

Some of the algorithms are feature-specific, i.e., they are mainly designed for reconstructing objects with some particular features. The algorithm in [FMG] has been designed to work for reconstructing a single open curve, whereas heuristics have been employed for disconnected or closed curves. These heuristics are not guaranteed to work for some inputs as the Euclidean minimum spanning tree may not always give a branching point between disconnected curves. The crust algorithm [ABE98], though a non-parametric one, has been designed for handling smooth curves. The algorithm in [Gie99] cannot handle a set of curves or open curves. NN-crust [DK99] and conservative-crust [DMR00] algorithms appear to have been designed for reconstructing only smooth curves. The Shape hull algorithm [PM15] only works for specific case of curves called divergent-concave shapes (please see [PM15] for more details). The *ec*-shape [MPM15] may fail to remove non-boundary edges in some particular cases where all circles are empty and the edge is still not a part of the shape. χ-shape, simple shape, shape hull and ec-shape algorithms have been designed such that the result will always be a single simple polygon and hence they cannot handle objects with holes, open curves, and multiple objects. The limitation of WDM-crust [PPM15] is that it cannot handle open curves as well as objects with holes.

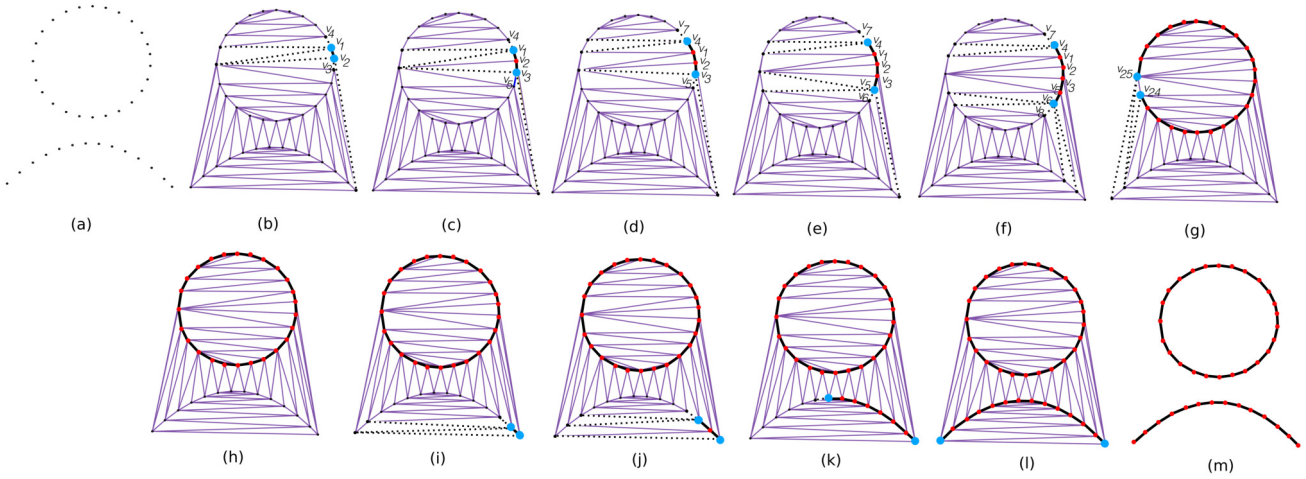

Figure 3: *Illustration of our algorithm. Dotted edges are linked edges and extremity vertices are in blue - (a) Point set (b) Delaunay triangulation. The algorithm starts with the shortest (seed) edge e_{12} between vertices v_1 and v_2. All the linked edges from the vertices are shown (dotted lines). (c) Shortest from the linked edges is then picked - e_{23}. Linked edges from extremity vertices v_1 and v_3 are also shown. (d)-(f) Algorithm continues (g) The linked edges from v_{24} and v_{25} are shown. (h) As the distance between them is less than all the distances of the linked edges, the edge between v_{24} and v_{25} is added ('closable' condition). (i)-(l) - The algorithm starts again from unvisited vertices with a seed edge and continues. (m) Output curves.*

1.2. Our work

In this paper, an algorithm that is non-parametric (i.e., that does not require a user parameter for reconstruction) and can also handle variants in inputs such as closed curves, open curves, multiple objects and curves having features such as sharp corners is presented. The proposed algorithm is based on the Delaunay triangulation of the set of input points. Though such an algorithm has many applications, in this paper, it has been employed for sketch completion.

Various sketch based interfaces, including 2D to 3D conversion techniques [OSJ11] need the user to give a single stroke (continuous tracing) closed sketch as input, a hard one for a novice user or requiring processing of the sketch such as stroke grouping [LWH15]. Based on a user study, we found that it is easier to follow image-assisted skip stroke sketching (user leaves small gaps between each strokes [GJ12] and s/he can start/end anywhere, i.e., unordered). Hence, our paper also discusses the usage of the proposed curve reconstruction algorithm for connecting such skip stroke sketches appropriately. The algorithm can be embedded in an image-assisted sketch based interfaces (such as [OSJ11]) to facilitate automatic connection of skip strokes in sketches.

The following are the major contributions of the developed algorithm for curve reconstruction:

- The algorithm is non-parametric and hence avoids the need to specify a user-defined parameter.
- The algorithm is not feature-specific and hence can handle open curves (Figure 2(a)), sharp corners (Figure 2(b)), outliers (Figure 2(c)), multiple objects (Figure 2(d)), sparse data (Figure 2(e)), and objects with holes (Figure 2(f)).
- Theoretically, the algorithm is justified using ε-sampling.
- Extensive qualitative comparison demonstrates that the developed algorithm performs better than or is comparable to various existing algorithms.
- An application of the proposed method to an image-assisted sketch completion has also been demonstrated.

2. Algorithm

Let DT denote the Delaunay Triangulation [BCKO08] of the set of points S (the terms points and vertices are used interchangeably). Let e_{ij} represent the edge between the vertices v_i and v_j and d_{ij} represent the straight-line Euclidean distance between them. [ABE98] has shown that, for smooth curves (that includes connected components but not branches/self-intersections, see [ABE98] for more details), under the ε-sampling [ABE98] condition on the input point set, the piecewise linear approximation of the reconstruction of the input is a set of connected subgraphs of the DT of S. Using this, the basic idea of our algorithm is to build connected subgraphs from DT. Nevertheless, we use a seed edge and the concept of extremity vertices for 'crawling through neighbours' as opposed to using DT in conjunction with Voronoi diagram as in [ABE98].

DEFINITION 1 Medial axis of a curve $\sum$ is the closure of the set of points which have more than one closest points on $\sum$ [ABE98]

DEFINITION 2 Local feature size $LFS(p)$ at a point $p \in \sum$ is the least distance of p to the medial axis of $\sum$ [ABE98]

DEFINITION 3 A point set is called an ε-sample of a curve $\sum$, if every point $p \in \sum$ has a sample within a distance $\varepsilon * LFS(p)$, where $0 < \varepsilon < 1$ [ABE98]

Initially, DT of S is computed and all the vertices are marked as *unvisited*. It is also assumed that there are more than three vertices in each of the connected subgraphs. Once the DT of the point set is computed, the algorithm broadly consists of the following steps:

- Computing a seed edge from the unvisited vertices of DT.
- Crawling using the seed edge to build a connected subgraph.
- Termination of each connected subgraph.
- Finding subsequent subgraphs, if any.

2.1. Computing a seed edge from the unvisited vertices of DT

THEOREM 2.1 Let e_{ab} be the shortest edge in DT. Then v_a and v_b are adjacent points in the polygonal representation C of $\sum$.

Proof Suppose v_a and v_b are not adjacent points in C, then based on ε-sampling there exists another point v_k adjacent to v_a such that $d_{ak} < d_{ab}$ which contradicts that e_{ab} is the shortest edge in DT. □

Theorem 2.1 can also be argued using Theorem 12 in [ABE98].

DEFINITION 4 A seed edge e_{ij} is the shortest edge in DT between two unvisited vertices v_i and v_j if v_i and v_j are the nearest neighbours of each other.

Figure 3(a) shows a set of points and its DT in Figure 3(b). A seed edge from DT is computed (e_{12} (in black) between the vertices v_1 and v_2 in the Figure 3(b)). Both vertices have now been marked as *visited*. Let G_1 be a connected subgraph with vertices v_1 and v_2 and the edge e_{12}.

DEFINITION 5 A vertex in a connected subgraph is said to be an extremity vertex if it has degree (i.e. number of edges connected to a vertex) one.

The vertices v_1 and v_2 of G_1 are extremity vertices in Figure 3(b).

2.2. Crawling using the seed edge

At this juncture, both the vertices in G_1 are extremity points. To identify the next edge starting from either vertex v_1 or v_2, the edges in the DT attached to the vertices v_1 or v_2 in the DT are looked at.

DEFINITION 6 Let v_m and v_n be the extremity vertices in G_1. All the unvisited vertices having edges with either v_m or v_n in DT are termed as linked vertices. The corresponding edges are called linked edges (for example, dotted edges in in Figure 3(b)) .

DEFINITION 7 Of all the linked edges, the minimum distance edge is called a candidate edge and the corresponding unvisited vertex is called a candidate vertex.

v_3 in Figure 3(c)) is the candidate vertex. G_1 is updated with the identified candidate vertex v_3 and edge e_{23}. This procedure can be considered as 'crawling through neighbours' as it crawls from an extremity vertex to the candidate vertex by looking only at the neighbours of the extremity vertex. The candidate vertex v_3 is then marked as visited. Now, the extremity vertices of G_1 are v_1 and v_3 and the 'crawling through neighbours' is repeated for the extremity vertices v_1 and v_3.

The linked edges from v_1 and v_3 are shown in dotted lines in Figure 3(c)). However, before adding another candidate vertex (say v_4) and its corresponding edge to G_1 , the distance between the vertices of the candidate edge is compared with the distance between extremity vertices. The candidate edge is added to the subgraph only if the distance between the vertices of the candidate edge is lesser than the distance between extremity vertices. The graph G_1 consists of the edges in black (and the corresponding vertices) in Figure 3(d)) and the extremity vertices are v_4 and v_3. The process of identifying candidate edges is then continued from the vertices v_4 and v_3 and the crawling continues thereafter (Figures 3(d),(e),(f))

Algorithm 1: *Reconstruct*(S)

Input: Input point set, S.
Output: Reconstructed Curve C.

```
Construct Delaunay triangulation, DT(S).
Mark all the vertices as unvisited.
repeat
    Identify the seed edge. Initiate a connected subgraph with
    the extremity vertices and the edge. Mark the extremity
    vertices as visited.
    repeat
        Find the linked vertices and edges, and subsequently the
        candidate vertex and edge (the distance between its
        vertices is d).
        Update the connected subgraph with the candidate vertex
        and edge. Mark the candidate vertex as visited.
        if number of edges in the subgraph is greater than two
        then
            Let d1 = distance between the extremity vertices.
            if d > d1 OR no candidate vertex is available then
                Update the subgraph with the edge between the
                extremity vertices if the subgraph is 'closable'.
                Terminate the identification of the current
                connected subgraph.
            end if
        end if
    until
    A connected subgraph from DT is created
until All the vertices in DT are marked visited.
return All the connected subgraphs.
```

2.3. Termination of the connected subgraph

An edge is no longer added to G_1 if it satisfies one of the following conditions:

- if the distance between the vertices of the candidate edge is greater than the distance between the extremity vertices or
- when there are no linked vertices available.

The first termination condition implies that an edge can be added between the extremity vertices and the connected subgraph can be updated with the added edge if G_1 is 'closable'. A subgraph having more than two edges is said to be 'closable' if one of its extremity point is second nearest neighbour of the other. Figure 3(g) shows the extremity vertices v_{24} and v_{25}. The distance between the vertices of the candidate edge is greater than the distance between the two extremity vertices. Hence, G_1 is updated with the edge between vertices v_{24} and v_{25}. The process of the computation of the connected subgraph G_1 is then terminated (Figure 3(h)).

2.4. Finding subsequent subgraphs

If there are any unvisited vertices in DT, the algorithm then proceeds to find the next seed edge, if any (Figure 3(i)) and the process continues by identifying linked edges and then candidate edges as mentioned in the previous sections (refer Figures 3(j),(k),(l)). The extremity vertices in (Figure 3(l)) do not have any linked edges and thereby no candidate edges are available. In Figure 3(l), the 'closable' condition is not satisfied and hence the edges of the subgraph form an open polygonal chain. Figure 3(m) shows the reconstructed curves.

The pseudo code for the method is given in Algorithm 1. The Delaunay triangulation computation takes $O(n\,log\,n)$ time. A priority queue has been used to efficiently find seed edges and it takes $O(n\,log\,n)$ for insertion of all edges. Other checks can be efficiently done in $O(n)$ complexity. Hence, the overall time complexity of the algorithm is $O(n\,log\,n)$.

2.5. Theoretical guarantee

Assume a simple closed curve $\sum$ has been sampled using ε-sampling defined by Amenta et.al. [ABE98]. Suppose C be the polygonal chain representation of $\sum$. Based on Theorem 2.1, we start with the smallest edge in DT.

THEOREM 2.2 Let v_c be a candidate vertex and v_e be the corresponding extremity vertex in a connected subgraph C_{inter} which is not closable, then the vertices of e_{ce} are adjacent samples in C.

Proof The proof is by mathematical induction.
Base case: In the base case we consider that C_{inter} has only one edge e_{ee^*} which is the smallest edge in DT by Theorem 2.1. Since C is closed every vertex has two linked edges in C. Therefore there exists another vertex v_i such that $e_{ei} \in C$. However, by our algorithm $d_{ce} \leq d_{ei}$, thus $v_i = v_c$ because otherwise the ε-sampling assumption is violated, therefore $e_{ce} \in C$.
Induction step: Assuming that all edges in C_{inter} are part of C we prove that the next candidate edge e_{ce} will also be a part of C. By a similar argument as above, we get a contradiction on the ε-sampling assumption which proves $e_{ce} \in C$. □

By recursively following Theorem 2.2 it can be ensured that after termination, all edges in C_{inter} are a part of C.

THEOREM 2.3 Let C_{inter} be a closable curve with extremity points v_e and v_{e^*} then $e_{ee^*} \in C$.

Proof Suppose $e_{ee^*} \notin C$. Because C is closed, v_e has degree two in C, i.e. there exists some $v_i \notin C_{inter}$ such that $e_{ei} \in C$. By ε-sampling, we have $d_{ei} < d_{ee^*}$ which contradicts the fact that C_{inter} is closable. □

Under ε-sampling, the 'closable' condition of adding an edge between extremity vertices is also valid and hence the algorithm guarantees exact reconstruction of a set of simple closed curves.

3. Results and discussion

Algorithm 1 has been implemented using the geometric kernel CGAL [cga]. Figure 4 shows a few of the implementation results. The figure indicates that the algorithm can perform well on a variety of objects. Reconstruction of objects with holes has been demonstrated on objects such as the coffee mug, teapot, windows in a bus etc. The algorithm has also captured point sets having multiple objects, such as the EG logo. The logo also illustrates that the algorithm has captured sharp corners quite well. Single closed curves such as a horse, a dog etc. have also been captured well. Other features such as small holes in the piston ring have also been reconstructed well.

3.1. Qualitative Comparison

To assess the performance of the proposed algorithm, we perform an extensive comparison of our algorithm with various reconstruction algorithms viz. α-shape [EKS83], crust [ABE98], nn-crust [DK99], χ-shape [DKWG08], simple-shape [GDJ*11], de Goes et al. [dGCSAD11], ec-shape [MPM15] and wdm-crust [PPM15]. It can be noted that our algorithm is non-parametric, whereas many of the algorithms require tuning of at least one parameter. Hence, the best perceived output has been presented for approaches that require parameter tuning. Figures 5 and 6 present the compared results (do note that both figures are in landscape mode), where the nature of the input is indicated at the left extreme (in landscape mode), and the approaches used for comparison at the top of the table. The input point sets used for testing have a wide variety of characteristics, viz., simple objects, multiple objects, objects with holes, sparse data, open curves, objects with sharp corners and outliers. It can be noted that there is no database of 2D point set (unlike that of in 3D). The pink ellipses in the figures underline some of the places where an algorithm has not been able to make the right 'connections' for reconstruction. We would like to mention that, for testing purposes, the input point sets used are generic and do not follow any sampling criteria (ε-sampling is used for only providing theoretical guarantee).

Clearly, our algorithm outperforms most of the algorithms in most cases and is comparable in few other cases. For example, for the 'simple curve1' (Figure 5), α-shape does not capture the dolphin's fin correctly, crust gives disconnected objects as the output and nn-crust results in a non-simple shape. Other approaches also fail to capture the fin correctly, whereas our approach returns the required output.

For 'simple curve2', α-shape adds triangles between tentacles to the shape before taking the body, crust results in an object with open edges and extra edges between tentacles, nn-crust gives two disconnected objects. χ-shape returns the exact shape for this input, though other approaches appear to have captured 'wrong connections' (regions indicated in pink ellipse) as can be seen from the respective figures. Our approach once again performs better than most and as good as χ-shape in this test point set.

Not many of the algorithms can reconstruct a collection of curves. Especially Delaunay sculpting based algorithms like *ec*-shape, χ-shape, simple shape etc. cannot reconstruct a collection of curves because of the regularity constraint (see [DKWG08] for details). Alphabets 'SGP' in Figure 5 illustrate the case for 'multiple objects' case, where our algorithm captures the shape better than other algorithms.

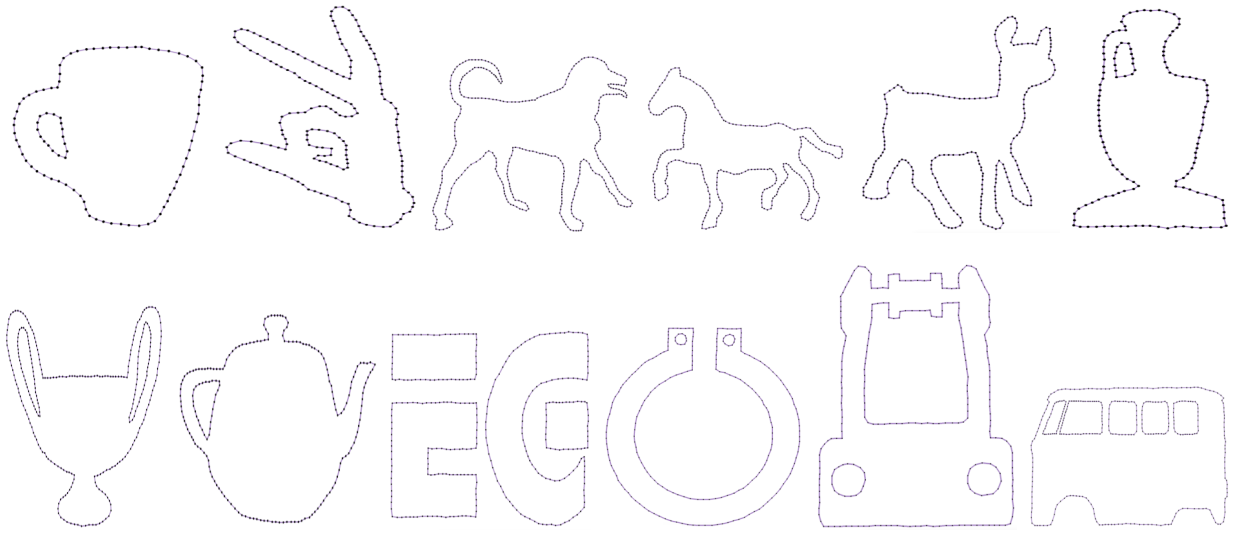

Figure 4: *Results of our algorithm on random shapes. The shapes have a wide variety of features and our algorithm has captured them.*

Many of the algorithms (Delaunay sculpting based algorithms) are designed to extract only outer boundaries whereas objects might have inner boundaries as well. Some of the algorithms output false open curves instead of holes. Our algorithm outperforms other algorithms as it does not detect any false open curves, as can be seen in 'object with hole1' and 'object with hole2' (last but two rows in Figure 5). For 'sparse point set', most of the algorithms have been found to be working reasonably well (see the last row of Figure 5).

Curve reconstruction algorithms are mainly focused on reconstructing closed curves, and hence many algorithms fail to distinguish between closed and open curves. Our algorithm is capable of detecting open curves as well (see 'open curves1' and 'open curves2' in Figure 6). Among the other algorithms, crust captures better than nn-crust and optimal transport based approach.

Sharp corners are one of the important features in a curve. 'share corner1' and 'sharp corner2' in Figure 6 show the results of various algorithms for a shape with sharp corners. Our approach has performed equally well or better than most other approaches. The optimal transport based approach also appears to capture sharp corners better than other approaches.

Handling of outliers has also become crucial for reconstruction algorithms. In this aspect, we have tested our algorithm by adding a certain amount of points as outliers. 'outlier 1 (16%)' and 'outlier 2 (37%)' in Figure 6 illustrate that our approach can handle outliers as well.

3.2. Discussion with other Delaunay-based approaches

Though our algorithm is based on DT, it differs from the existing ones in the following aspects: (a) Unlike the sculpting approaches, it does not remove the triangles only from outside and hence is not restricted only to certain shapes (such as the one in [PM15]). (b) Algorithms such as crust [ABE98] use a combined Delaunay/Voronoi diagram, ours uses only the edges in the Delaunay. (c) Moreover, since our approach crawls from the vertex of an identified edge, a direct ordering of the vertices/edge becomes possible using simple 'insertion'. (d) As already emphasized, our approach is independent of any input parameter(s) unlike many other Delaunay-based approaches. Though a user input parameter(s) adds flexibility, it is often tedious to identify the right set of parameter(s) for a desired output.

4. Sketch Completion

Sketch based interfaces have been shown to be useful in various applications in fields of computer graphics, entertainment etc. 2D to 3D conversion is an interesting and fascinating application using sketch based interfaces, where a user will sketch in 2D to make a 3D model.

4.1. User Study

We conducted a user study to find out the difficulties faced while sketching. The main findings are that the users found it difficult to draw without a reference image. We also found a big improvement when sketching was assisted using images (such as the one in [dPI13]). Even in image-assisted sketching, the main problems faced by users while drawing single stroke sketches are:

- Extreme difficulty in drawing complex sketches.
- Difficulty in drawing acute angled sharp corners which leads to self intersecting curves.
- Difficulty in tracking long smooth curves, unless the user has experience with sketch pads.
- Accidentally, users take their pen off while sketching.
- Users lose control of the pen after some time if the sketch is too complex or big, because of pain.

It needs expertise to draw 2D sketches in a single stroke as well as resuming from the exact point where s/he left off. Figure 7(a)

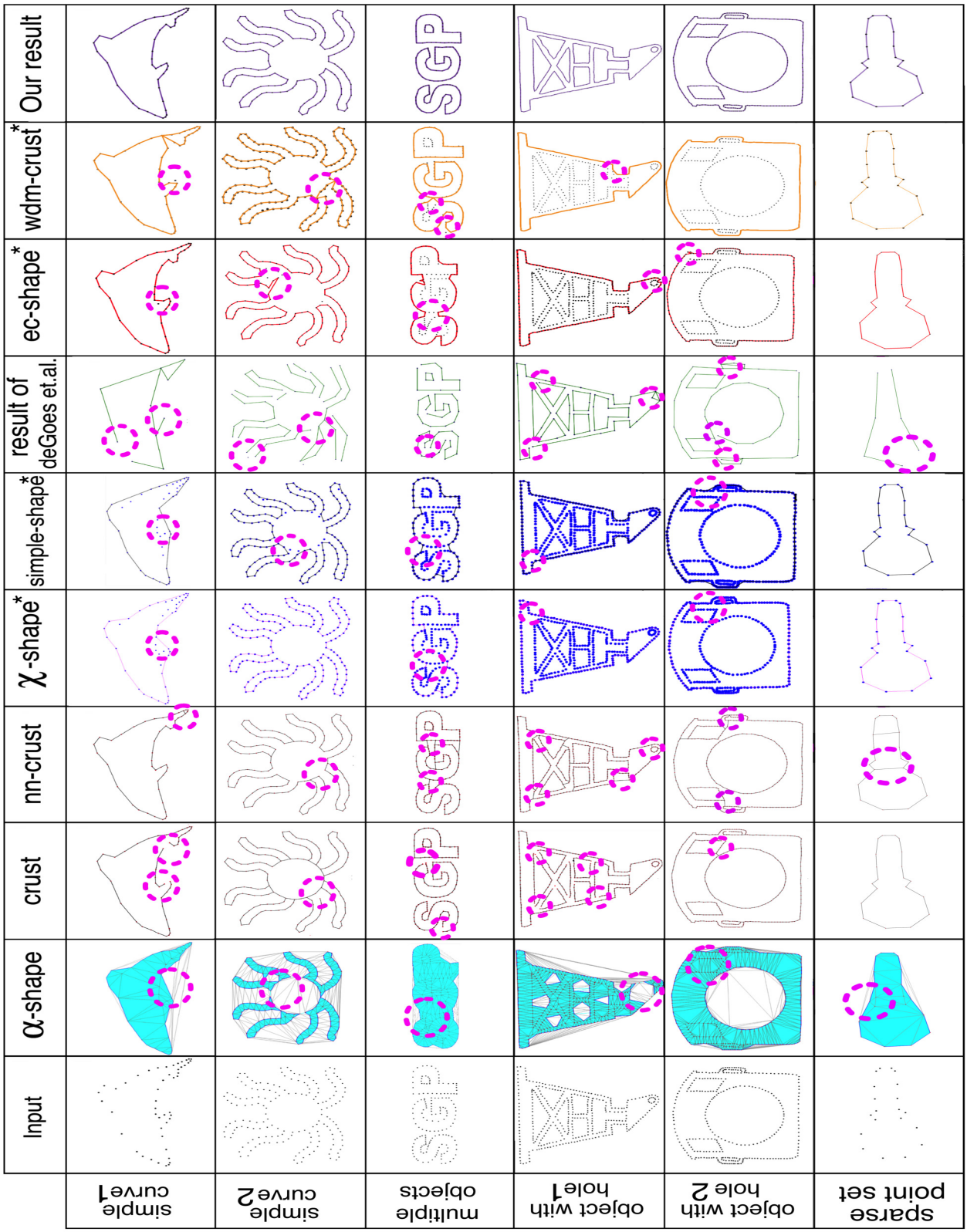

Figure 5: *Comparison of our algorithm with other approaches. Left extreme in the figure indicate the characteristics of the input and the top row shows the approach employed (The algorithms which output simple polygons without holes have been marked with an asterisk).*

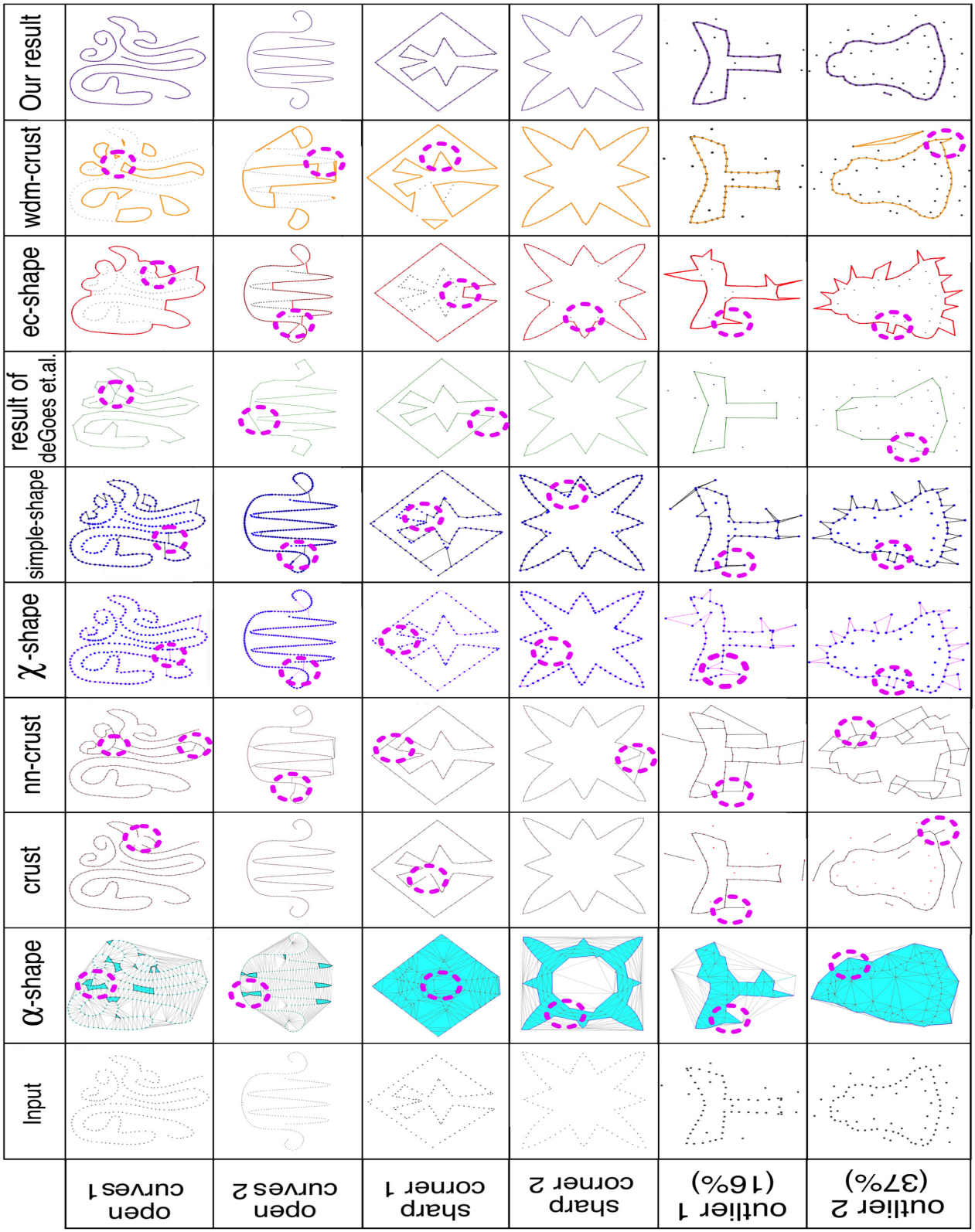

Figure 6: *Comparison of our algorithm with other approaches. Left extreme in the figure indicate the characteristics of the input and the top row shows the approach employed.*

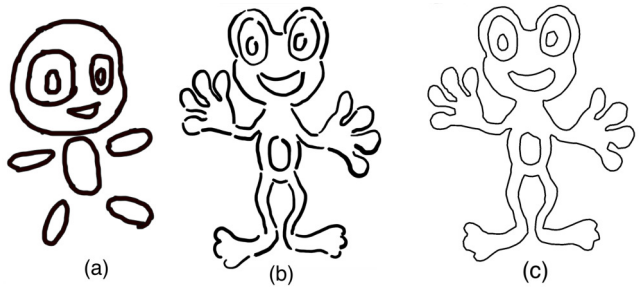

Figure 7: *(a) Multistroke sketch with continuity between consecutive strokes. (b) A sample sketch (c) Our sketch completion result*

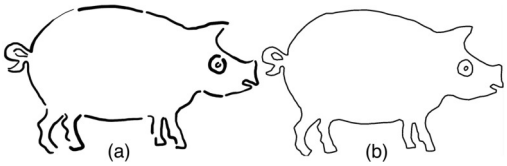

Figure 8: *Sketch completion results: (a) A sample sketch (b) Our result*

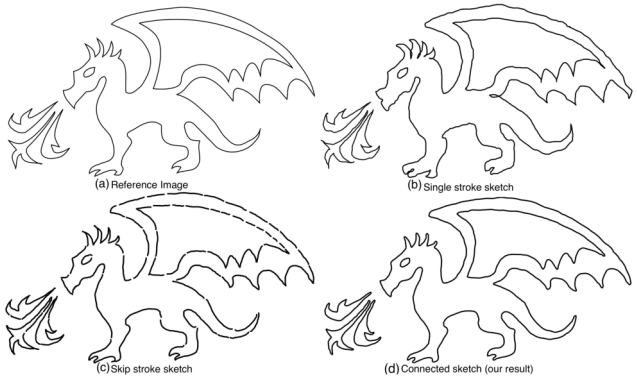

Figure 9: *Image assisted sketching experiment 1: (a) Reference Image (b) Single stroke sketch and (c) Sketch with skip strokes drawn by a novice user (d) Connected sketch returned by our algorithm on (c)*

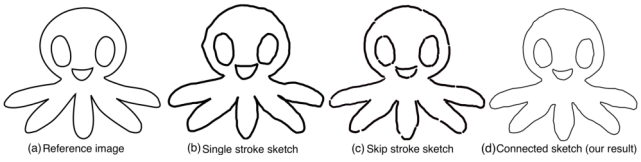

Figure 10: *Image assisted sketching experiment 2: (a) Reference Image (b) Single stroke sketch and (c) Sketch with skip strokes drawn by an intermediate user (d) Connected sketch returned by our algorithm on (c)*

shows an example of a user drawing in which the user is allowed to draw using multiple strokes, with an additional condition that there should not be any discontinuities between the lines. It is clear from the figure that the user could not draw from the exact end point of the previous stroke. It is usually easier to draw disconnected curves which approximate the shape. Such disconnected curves are shown in Figures 7(b) and 8(a).

4.2. Application of our algorithm to sketch completion

Figure 9(a) shows a reference image of a dragon we provided as part of our user study to a selected group of candidates, most of whom were novices to this task. Figure 9(b) is an image assisted set of single stroke sketches drawn by a novice user. The users could not track the correct boundary and all the users found it difficult to draw the wings and fire flames because of its non-convex nature and sharp features. Figure 9(c) is the dragon sketch drawn by the same user using skip stroke style. To reconstruct the original shape from the skip sketch, morphological thinning has been initially applied on the input sketch and a point set is generated using software like [Roh14]. The generated point set is fed to our algorithm to reconstruct the sketch. It is evident from Figure 9(d) that our approach has performed the sketch completion task quite well.

Figure 10(a) is another reference image of an octopus we provided to the group of users. Even though the reference image is quite simple to trace, because of the absence of minute features, users found it difficult to do a single stroke (Figure 10(b)). Instead, it was easier to follow skip stroke sketching pattern because of the flexibility of taking out the pen out in between. Users found it very helpful that they do not have to start from the end point of the previous stroke (Figure 10(c) shows the skip stroked sketch). Figure 10(d) shows the completed sketch using our reconstruction algorithm.

All the users showed an ability to track the boundary, and sketch more easily and comfortably using skip strokes. These disconnected lines can be connected using our algorithm to make the sketch completed (the results for Figures 7(b), 8(a) are shown in Figures 7(c), 8(b) respectively). Hence, we believe that our algorithm will be extremely useful in sketch completion work, apart from other areas where reconstruction is required. Figure 11 shows the comparison of sketch completion with other reconstruction algorithms, which demonstrates that our algorithm yields much superior result (Figure 9(d)).

4.3. Limitations

Though our approach is simple to implement and has been demonstrated on a wide variety of objects having different features such as objects with holes, open curves, sharp corners and even objects with outliers, the major limitation is that the approach cannot handle objects with noise (Figure 12(a)) for which the reconstructed output (Figure 12(b)) is not a desired one. Even if the objects have a vertex with a degree more than two (T-junction is one such example, Figure 12(c), self-intersecting curve is another), the algorithm fails to capture the intended shape (Figure 12(d)) as it leaves the curves open near the junction. For a set of open curves that are too close to each other, the termination condition may not return all of them as open curves. The proposed sketch completion is applicable

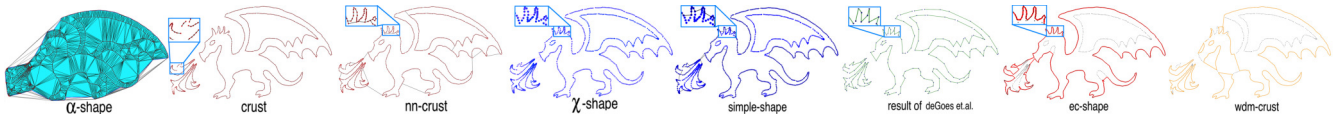

Figure 11: *Sketch comparison with other reconstruction results. Other approaches either have unwanted connection, missing connections or generated only outer boundary. Our algorithm's result is shown in Figure 9(d).*

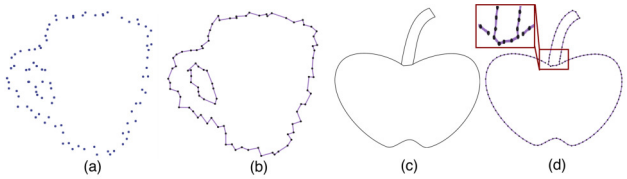

Figure 12: *Limitaions (a) Noisy input (b) Our algorithm fails (c) T-junctions (d) Fails to capture the 'closed' T-junctions*

only for image-assisted skip stroke sketching and not for free hand sketching.

5. Conclusion

A simple Delaunay based algorithm for curve reconstruction has been proposed in this paper. The proposed algorithm is capable of reconstructing inputs with multiple objects, objects with holes, sharp corners, objects with outliers, open curves and sparse point set. Qualitatively, it has been demonstrated that our approach performs better (or in some cases, comparable) with other approaches. Future work involves modifying the algorithm to alleviate the limitations as well as extension to 3D models.

References

[ABE98] AMENTA N., BERN M., EPPSTEIN D.: The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing 60*, 2 (1998), 125 – 135. 1, 2, 3, 4, 5, 6

[BCKO08] BERG M. D., CHEONG O., KREVELD M. V., OVERMARS M.: *Computational Geometry: Algorithms and Applications*, 3rd ed. ed. Springer-Verlag TELOS, Santa Clara, CA, USA, 2008. 3

[cga] "CGAL, Computational Geometry Algorithms Library. "http://www.cgal.org. 5

[dGCSAD11] DE GOES F., COHEN-STEINER D., ALLIEZ P., DESBRUN M.: An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum 30*, 5 (2011), 1593–1602. 2, 5

[DK99] DEY T. K., KUMAR P.: A simple provable algorithm for curve reconstruction. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 1999), SODA '99, Society for Industrial and Applied Mathematics, pp. 893–894. 2, 5

[DKWG08] DUCKHAM M., KULIK L., WORBOYS M., GALTON A.: Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition 41*, 10 (2008), 3224 – 3236. 2, 5

[DMR00] DEY T. K., MEHLHORN K., RAMOS E. A.: Curve reconstruction: Connecting dots with good reason. *Computational Geometry 15*, 4 (2000), 229 – 244. 2

[dPI13] DOS PASSOS V. A., IGARASHI T.: Landsketch: A first person point-of-view example-based terrain modeling approach. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2013), SBIM '13, ACM, pp. 61–68. 6

[DW01] DEY T. K., WENGER R.: Reconstructing curves with sharp corners. *Computational Geometry 19*, 2âĂŞ3 (2001), 89 – 99. Combinatorial Curves and Surfaces. 2

[Ede98] EDELSBRUNNER H.: Shape reconstruction with delaunay complex. In *Proceedings of the Third Latin American Symposium on Theoretical Informatics* (London, UK, UK, 1998), LATIN '98, Springer-Verlag, pp. 119–132. 1

[EKS83] EDELSBRUNNER H., KIRKPATRICK D., SEIDEL R.: On the shape of a set of points in the plane. *IEEE Transactions on Information Theory 29*, 4 (Jul 1983), 551–559. 1, 2, 5

[FMG] FIGUEIREDO L. H., MIRANDA GOMES J.: Computational morphology of curves. *The Visual Computer 11*, 2, 105–112. 1, 2

[GDJ*11] GHEIBI A., DAVOODI M., JAVAD A., PANAHI F., AGHDAM M. M., ASGARIPOUR M., MOHADES A.: Polygonal shape reconstruction in the plane. *IET Computer Vision 5*, 2 (March 2011), 97–106. 2, 5

[Gie99] GIESEN J.: Curve reconstruction, the traveling salesman problem and menger's theorem on length. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry* (New York, NY, USA, 1999), SCG '99, ACM, pp. 207–216. 2

[GJ12] GRIMM C., JOSHI P.: Just drawit: A 3d sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (Aire-la-Ville, Switzerland, Switzerland, 2012), SBIM '12, Eurographics Association, pp. 121–130. 3

[LWH15] LIU X., WONG T.-T., HENG P.-A.: Closure-aware sketch simplification. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 168:1–168:10. 3

[MPM15] METHIRUMANGALATH S., PARAKKAT A. D., MUTHUGANAPATHY R.: A unified approach towards reconstruction of a planar point set. *Computers & Graphics 51* (2015), 90 – 97. International Conference Shape Modeling International. 2, 5

[OSJ11] OLSEN L., SAMAVATI F., JORGE J.: Naturasketch: Modeling from images and natural sketches. *IEEE Comput. Graph. Appl. 31*, 6 (Nov. 2011), 24–34. 3

[PM15] PEETHAMBARAN J., MUTHUGANAPATHY R.: A non-parametric approach to shape reconstruction from planar point sets through delaunay filtering. *Computer-Aided Design 62* (2015), 164 – 175. 2, 6

[PPM15] PEETHAMBARAN J., PARAKKAT A. D., MUTHUGANAPATHY R.: A Voronoi based Labeling Approach to Curve Reconstruction and Medial Axis Approximation. In *Pacific Graphics Short Papers* (2015), Stam J., Mitra N. J., Xu K., (Eds.), The Eurographics Association. 2, 5

[Roh14] ROHATGI A.: Webplotdigitizer 3.5. URL: http://arohatgi.info/WebPlotDigitizer. 9