Hierarchical Distance-based Aggregation

Mukulika Ghosh mghosh@cse.tamu.edu
Nancy M. Amato amato@cse.tamu.edu

Technical Report TR14-006
Parasol Lab.
Department of Computer Science and Engineering
Texas A&M University

April 28, 2014

Abstract

Approximation is one of the key techniques used in manipulating geometric objects. Aggregation is a form of approximation that joins objects that are likely to be grouped together as a single unit. This can improve efficiency, reduce complexity and generate levels of detail, and has application in rendering, cartography and molecular structure design. In this work, we present a general framework to aggregate nearby objects together. The proximity of the objects is determined based on distances between them. Grouped objects are approximated into shapes similar to alpha shapes. This reduces volume as compared to convex hull approximation. However, typical alpha shape approximation, which uses a constant alpha value, fails to adapt to non-uniform inputs. To address this, we use a function to adaptively determine the value of alpha based on the properties of the objects to be aggregated. Varying the threshold distance that determines the group of nearby objects, we can create a hierarchy of aggregation for any environment. We evaluate our method using two dimensional objects. The quality of the aggregated objects is evaluated using shape metrics including area, perimeter and circularity and is compared with convex hull and alpha shape approximations. Our method produces aggregates which are closer to the original objects than alpha shapes and convex hulls. We also demonstrate how the levels in aggregation can be used to solve motion planning problems more efficiently in complex environments. We have also extended our implementation to three dimensional objects with results similar to two dimensional environments.

1 Introduction

Approximation is one of the key techniques in manipulating geometric objects for efficient processing in a variety of applications in computer graphics [34], geology [7] and computational biology [31]. Popular approximation techniques include simplification [18], clustering [29] and decomposition [28, 19]. The objective of all these techniques is to obtain a simpler version of the original model(s) with certain properties such as convexity, number of components and number of primitives that expedite further processing.

Object aggregation approximates a set of objects that are likely to be grouped together as a single entity. The structure of the aggregated object should be such that the approximation error does not have a negative impact on the results used for future processing. Aggregation is often used in cartography, where a hierarchy of maps is created by aggregating neighboring buildings, trees and other geological objects to form nuclear colonies [44, 21]. It is also used in crystal structure design where geometric simplices like spheres are aggregated to form the three dimensional structure of the lattice [8, 33].

In applications such as motion planning, a feasible collison-free path is computed in an environment with a set of obstacles. Often complex geometric models are used as obstacles. To improve efficiency, the complex objects can be approximated by simpler, and often convex shapes [27, 30, 23]. Sometimes, the large numbers of obstacles adversely affects the efficiency, especially if they are likely to be considered as a group. For example, a group of trees in an environment may be grouped together as a single obstacle if the passage between the trees is not important and alternate routes around them exist. However, the convex hull approximation of the obstacles to be grouped might significanctly reduce free space and can cause narrowing of existing passages, making the passage challenging to navigate or even impassable as shown in Fig. 1.

In this work, we propose a new method to identify and aggregate a set of nearby objects. More specifically, a set of objects is aggregated if the distance between them does not exceed a user-defined value, δ. By varying δ, we define an aggregation hierarchy in which the lowest level is the original environment and the highest level contains a single aggregated object. We use methods similar to alpha shapes [14, 15] to generate the approximate shapes of the grouped objects. An

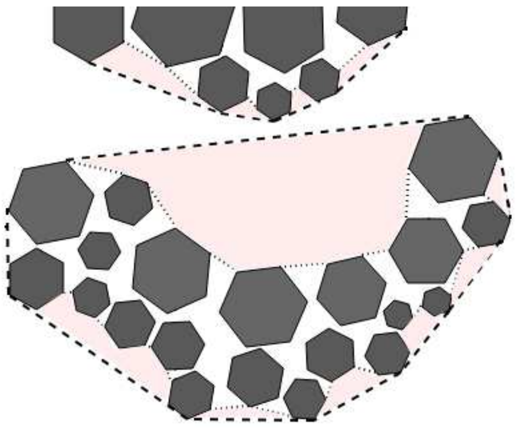


Figure 1: The convex hull envelope (dashed line) wastes space as compared to the concave hull envelope (dotted line). This wasted space (shown in pink) can make finding a solution path challenging or even impossible.

alpha shape [14, 15] is a generalization of the convex hull; a high enough alpha value generates the convex hull. Aggregation using alpha shapes with a constant alpha is used in GIS applications such as PostGIS [2] and ArcGIS [1]. But a constant alpha value can fail to capture topological variation that can occur in a heterogeneous environments. Hence, we use a function to adaptively determine the value of alpha based on the topological features of the objects that need to be aggregated. If the function returns a constant value, our aggregation method behaves similarly to the alpha shape approximation.

Our main contributions are:

- A general framework to aggregate a set of disjoint objects such that:
  - objects are grouped based on the minimum distance between them, and
  - the approximate shape of the aggregated objects is determined by a function of the properties of the objects and threshold distance.
- An aggregation hierarchy for an environment with a set of objects.

We analyzed the quality of our aggregation using shape metrics including area, perimeter and circularity and compared the quality with convex hull and alpha shape approximation. Our results demonstrate that our method creates shapes closer to the original objects with less error in approximation (as measured in terms of the volume of free space consumed) than convex hull and alpha shape approximations. Our results also demonstrate an application of the aggregation hierarchy to a motion planning problem in a complex environment where the hierarchy improves the efficiency of the planning as compared to planning in the original environment. We have also extended our implementation to three dimensional objects with results similar to two dimensional environments.

2 Related Work

Approximations of original models are computed for efficient processing. Approximation hierarchies [4] are used to decide the trade-off between approximation error and efficiency. Aggregation is used to generalize a large number of data into smaller number of groups. Clustering algorithms are popular in determining the groups for a variety of objectives [13] depending on the input data set and the nature of the groups. K-means and fuzzy clustering form the underlying clustering approach to define various form of approximations, like decomposition [38, 26].

Most of these algorithms do not use disjoint polygons/polyhedra as an input model, and hence clustering is used only to define the boundaries of the groups. As in Geo-spatial applications, the polygons are clustered to describe regions based on area, neighborhood, etc [25]. Zhang et.al [43] provide a generalized framework for clustering polygons based on three defined input data boundaries, features and spatial events. The distance function is also user-defined, depending on the defined attributes of the input polygon. Applications like ArcGIS and PostGIS aggregate polygons based on distance. They rasterize the input polygons and join the features if they are close with respect to a threshold distance. In this work, we extended this idea in determining the groups that will form the aggregates.

Aggregation of a polyhedral mesh is often used as an union operator as mesh generator [40]. In most applications, the mesh used for union are intersecting or an intersecting boundary is generated to join the mesh along the intersecting boundary [40, 12]. Disjoint three dimensional points are clustered to form a simplified or new polyhderal structure [6]. Alpha shapes [17], skin surfaces [10] and other methods are used to determine the polyhedral mesh cover. Disjoint meshes are joined using these shapes from polyhderal simplices to create new complex structures [8, 33]. Most of these methods use homogeneous objects, and hence traditional alpha shapes [14, 17] are enough to serve the purpose. However, to handle non-uniformity in inputs, weighted alpha shapes [16] and conformal alpha shapes [9] are defined, which are used in applications like mesh reconstruction and feature detection. Instead of determing the weights on the point set, we determine the alpha adaptively based on the topological features.

Hagbi and El-Sana describe a mesh simplification strategy using carving of Constrained Delaunay Tetrahdralization [22]. Although their objective is to simplify the mesh, the algorithm can be applied to disjoint meshes to create an aggregated structure. In this work, we use the carving technique from [22] to define the aggregated structure where the distance between them determines whether they are to be aggregated or not.

In this work we combined the concept of polygon aggregation based on a distance threshold and approximate surface generation in three dimensional meshes to aggregate a set of objects if they are close to each other and the surface of the aggregates is approximated based on topological attributes of the objects. Our work provides a general framework to aggregate group of nearby objects such that joining of nearby features in GIS applications forms one of the specialized version of the proposed aggregation method. Also, the aggregation method does not assume the objects to be intersecting, as in polyhedral aggregation.

3 Our Approach

Aggregation clusters nearby objects into a single group while reducing the approximation error. In this work, we assume the objects to be disjoint (or objects which have zero distance between them are already merged as a single object). We will refer to the objects as models or objects interchangeably.

Aggregating objects by their distance involves two major steps: (1) identifying the groups of nearby objects and (2) creating approximate shapes that cover each group of objects. To find the group of nearby objects, we abstract the environment in a graph called the *neighborhood graph*. The vertices in the neighborhood graph correspond to the models in the environment. Neighboring models are connected by weighted edges in the graph, with the weight being the minimum distance between the objects. In order to identify the group of models to be aggregated, we reduce the neighborhood graph of the original environment to that of the aggregated environment. This reduction involves collapsing the edges with weights less than the threshold δ. Varying δ creates a hierarchy of neighborhood graphs for different levels of aggregation. Each of the vertices in the resulting neighborhood graph represents an aggregated group of objects. Depending on the function used to approximate the aggregated objects, intermediate levels might also be introduced in the hierarchy of aggregation. The sketch of the algorithm is provided in Algorithm 1.

Algorithm 1 Distance Aggregation(M,δ)

```
Input: A set of models $M$ and threshold distance $\delta$.
Output: An aggregated set of models $AM$.
 1: Initialize the Neighbor Graph $NG(V, E)$ with each $m \in M$ as $v$ in $V$.
 2: for each $e \in E$ do
 3:    if $w(e) < \delta$ then
 4:       Group the models at the end vertices of $e$, $M_e$.
 5:       Collapse the edge $e$ to a single vertex $v$ with $M_e$ as the group of models it represents.
 6:    end if
 7: end for
 8: for each $v \in V$ do
 9:    Create the approximate shape, $am$ of the group of models from which $v$ is evolved or the
       model it represent.
10:    $AM \leftarrow AM \cup am$.
11: end for
12: return $AM$
```

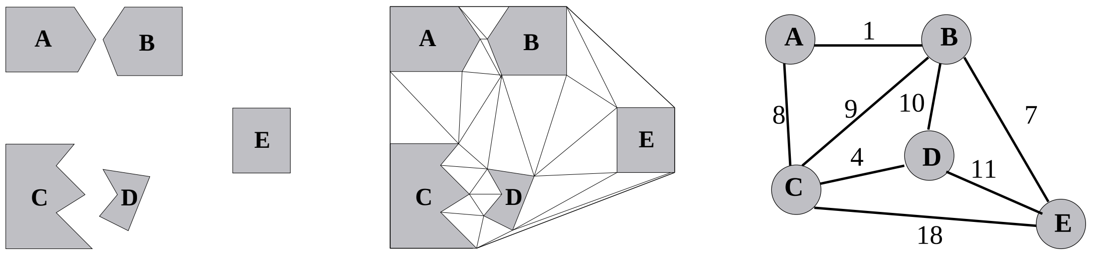
3.1 Initial Neighborhood Graph Construction

To identify the groups of models to be aggregated, we need to abstract both the neighborhood of the models and the distance between the neighboring models. A dual graph of the environment, known as the neighborhood graph, is constructed with models as vertices and the neighborhood relationship among the models as edges. The initial (the lowest level) graph is constructed with the objects in the original environment as individual vertices. The neighboring objects are then connected by edges in the graph, with edge weight as the minimum distance between the neighboring objects.

To find the neighborhood relationship of objects in the environment, we need to construct an approximate Voronoi diagram or identify the Voronoi regions where models are sites instead of points. For our implementation, we use Constrained Delaunay Triangulation (CDT) [37, 11], to identify these Voronoi regions. CDT is similar to Delaunay triangulation with an added constraint of including predefined edges to be included in the output. Various other approaches that define the neighborhood, like Voronoi diagram and/or Delaunay Triangulation [5, 20], Gabriel Graphs [32], Relative Neighborhood Graph [24, 3] and others can similarly be used to encode the neighborhood relationship of the objects in neighborhood graph.

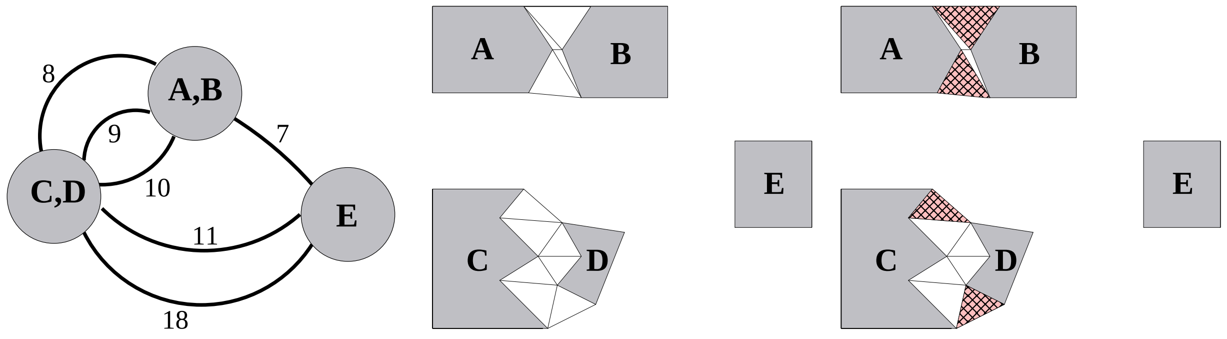
We construct the CDT of the convex hull of all the objects in the environment, excluding the objects, i.e., the convex hull of all the objects with the objects as holes. The constrained edges are that of the objects and the convex hull. This creates a boundary or region for the triangulation. Fig. 2(b) shows the CDT of the environment in Fig. 2(a).

After construction of the CDT, every edge in the CDT is checked. If the end-vertices of the edge



(a) Environment (b) CDT (c) Neighborhood Graph

Figure 2: CDT of an environment and corresponding neighborhood graph.



(a) Resulting Neighborhood Graph (b) CDT after removal of edges (c) Exposed triangles

Figure 3: Resulting neighborhood graph with $\delta = 5$ and the resultant CDT structure after removal of edges corresponding to the edge connections in the neighborhood graph.

belong to different model boundaries then, an edge is created in the initial neighborhood graph connecting the vertex representative of the models in the graph. The weight is updated according to the shortest length CDT edge found between the models. Fig. 2(c) is the initial graph constructed using the CDT in Fig. 2(b).

3.2 Identifying Models To Be Aggregated

With the representation of the models and their neighborhood in the form of the intial neighborhood graph, it is easier to identify the groups of objects that need to be aggregated. The neighborhood graph of the aggregated environment is constructed by collapsing the edges in the initial neighborhood graph that has weight less than the user-defined threshold distance δ.

The edge-collapse operation in the neighborhood graph is defined as merging the end-vertices as a single vertex. The vertices in the initial neighborhood graph that are reduced to a single vertex in the resulting neighborhood graph represent a group of objects to be merged. Hence, by tracing back the merging history of every vertex in the resulting neighborhood graph to the initial neighborhood graph, we generate the final groups of models to be merged. As shown in Fig 3(a), the edge between vertex A and B is merged to a single vertex with A and B in its merging history.

Two edges from the graph in Fig. 2(c) are merged to form the neighborhood graph in Fig. 3(a) when δ is set to 5. In our implementation, the triangles between the objects in the CDT that are connected in the neighborhood graph by an edge are also removed such that the CDT corresponds exactly to the neighborhood graph, as shown in Fig. 3(b).

3.3 Aggregated Approximation of the Models

The resulting neighborhood graph generated by collapsing the edges with weights greater than the threshold distance δ from initial neighborhood graph represents the number of aggregated groups and their neighborhood relation. The sets of objects represented by the vertices in the final neighborhood graph are the groups of objects that need to be approximated into individual shapes. A variety of shape descriptors can be used to generate the approximate boundary of the aggregated groups. For more details refer to [42, 41].

In this work, we use approximates similar to alpha shapes to construct the boundary of the aggregated models. In our implementation, the resulting CDT, after the removal of the edges corresponding to the edges in the latest neighborhood graph, generates a set of initial approximations of the aggregated models. These initial approximations are further sculpted to reduce error or wastage of space. The sculpting is done by iteratively removing *exposed* triangles from the CDT. An *exposed* triangle has at least one edge with no neighboring triangle in CDT. Fig. 3(c) shows the exposed triangles as checquered pink triangles. As the CDT contains triangles in the free space outside the objects boundary in the original environment, the approximation process does not sculpt beyond the original object boundaries.

An exposed triangle is removed if one of its edges exceeds a value which is determined by an *aggregate* function F, of the models at either end of the edges in the triangle. Below are two aggregate functions studied in this work:

$$F(m_1, m_2) = \delta \tag{1}$$

$$F(m_1, m_2) = 0.5 * (\min(m_1, m_2) + \sigma(m_1, m_2) + \delta) \tag{2}$$

where m_1 and m_2 are the two models at either end of the edges of the triangle, $\min(m_1, m_2)$ is the minimum distance between the models m_1 and m_2, and $\sigma(m_1, m_2)$ is the standard deviation of the length of all connections.

Equation 1 reduces the aggregation framework similar to alpha shape aggregation in GIS applications in which features within the input threshold distance are aggregated. If the equation returns a constant value for a given environment, it creates alpha shapes of the aggregated objects. Equation 2 generates the approximated boundary closer to minimum distance connections based on the variation in the lengths of the connection and the threshold distance. The minimum distance and the standard deviation incorporate the average topological variation between the objects to the minimum distance connection. Averaging the value with the threshold distance δ balances δ with minimum distance connection and topological variations. Other functions, like the average of the threshold distance and the average distance, can also be used.

4 Results

We evaluated our approach in terms of efficiency and shape descriptor quality metrics including total area, total perimeter and a circularity measure that state how close the resulting model is to a circle. The aggregation using the aggregate function stated in Equation 1 (alpha approximation) is refered as alpha shape and Equation 2 is refered as our method in this section. We used 5 different environments, as shown in Fig. 4 (ordered by their total number of vertices). The implementation is done in C++, and the experiments are conducted on a PC with Pentium CPU and 2GB RAM. We used the library Triangle [35] to construct the CDT.

Fig. 4 shows the agggregated models using the same threshold for both aggregation methods (alpha shapes and aggregate function) in 5 different environments. The threshold distance δ for each environment is stated in the caption. The choice of δ used in the experiments is random in the range from minimum and maximum distance connection in the neighborhood graph. For the same threshold, our method results in an aggregation closer to the original environment than alpha shape approximation. This is due to balancing of the value of aggregate function with respect to the minimum distance of the models, variation in distances between the model, and the threshold distance δ. Sometimes the aggregation using our method for a higher threshold resembles aggregation in alpha shape approximation for a lower threshold, as shown in Fig. 5. This reduces the height of the aggregation hierarchy using our method as compared to alpha shape method, eliminating levels with large approximated areas, which are less important than tighter and more intuitive approximations. Also, the connections formed by our method are narrower than alpha shape approximation.

4.1 Time

Fig. 6(a) shows the time efficiency of the aggregation for each environment. The result shows an increase in time with an increase in the number of total vertices in the environment. A constant time is observed for any given environment at different levels of aggregation. This is because the time is heavily dependent on the complexity of the CDT construction algorithm. As both the approximations using alpha shapes or our method are a constant time operation, difference in this function is not reflected in the time efficiency. However, if an aggregate function required computation specific to the state models at the levels in aggregation, the complexity will increase or decrease accordingly with the level of aggregation.

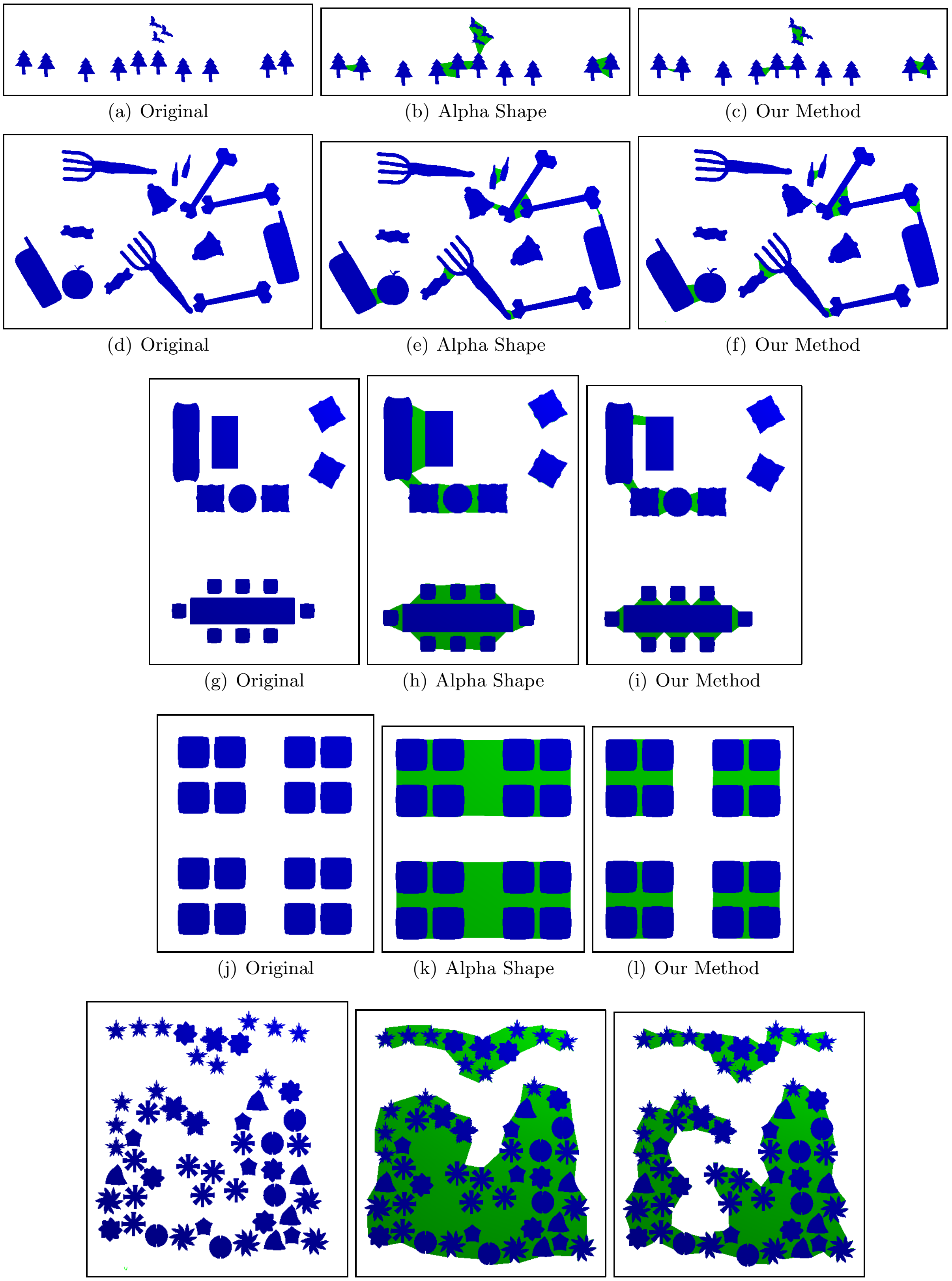
4.2 Shape Quality Metrics

Aggregation introduces changes in the shape of the objects that are aggregated together. In this subsection, we analyze the shape quality of the aggregated shapes in terms of the metrics like area, perimeter and a circularity measure $C(E)$ for an environment E as defined in Equation 3:

$$C(E) = \frac{4\pi A(E)}{P(E)^2} \tag{3}$$

where $A(E)$ is the total area and $P(E)$ is the total perimeter of all the objects in E. Fig. 6(b) gives the circularity measure for the 5 environments at 4 different levels. The values are normalized to those of the original environment. The figure demonstrates that using alpha shapes there is a sharp increase in circularity, whereas using our method the increase in circularity is slow with respect to the original model. The increase in circularity with an increase in tolerance states that the aggregated models are closer in resemblance to the original models at a low tolerance level than to a circle at high tolerance level. The circularity measure for convex hull cover is not shown, as it is the highest achievable circularity for the aggregated components and is higher than any of the aggregation methods, as expected.

The total area and the total perimeter of the aggregated models at different threshold levels for each of the 5 environments are shown in Fig. 6(c) and Fig. 6(d) respectively. All the measures are normalized to that of original environment. There is an increase in area and decrease in perimeter with increase in tolerance. The change in area and perimeter is more drastic for alpha shape than in our aggregation. This is due to the fact that our aggregation uses an average of threshold and minimum distance with standard deviation as the threshold for removal of edges from the approximate of the aggregated models, which results in approximations closer to the original models. Hence, our method incurs less approximation error than alpha shapes and convex hulls.


(a) Original (b) Alpha Shape (c) Our Method

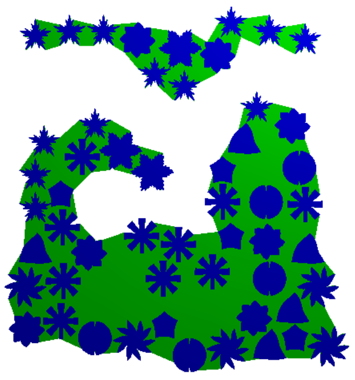
(d) Original (e) Alpha Shape (f) Our Method

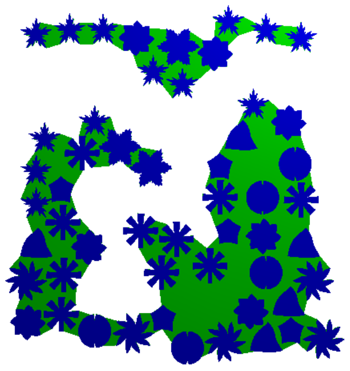
(g) Original (h) Alpha Shape (i) Our Method

(j) Original (k) Alpha Shape (l) Our Method

(m) Original (n) Alpha Shape (o) Our Method

Figure 4: Original and aggregated environments using alpha shape and our method of approximation with the same delta value for each environment. (a)-(c) are Tree environments ($\delta = 0.8$), (d)-(f) are Scatter environments ($\delta = 4$), (g)-(i) are Room environments ($\delta = 20$), (j)-(l) are Colony environments ($\delta = 28$), and (m)-(o) are Forest environments ($\delta = 40$).


(a) Alpha Shape (d=30)


(b) Alpha Shape (d=20)

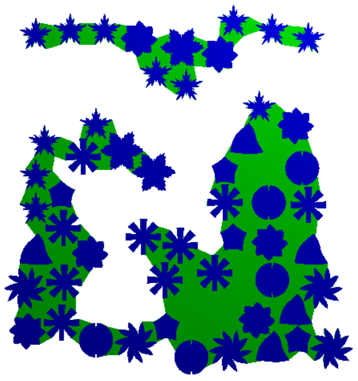
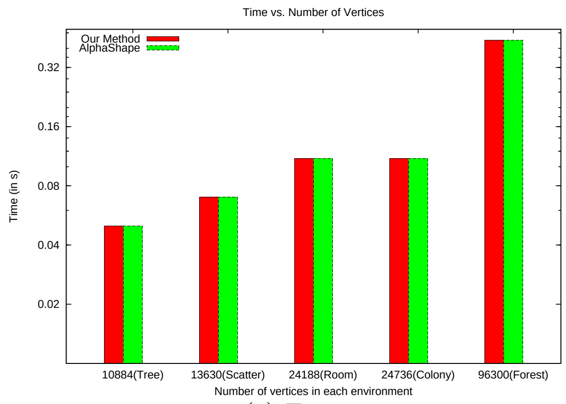
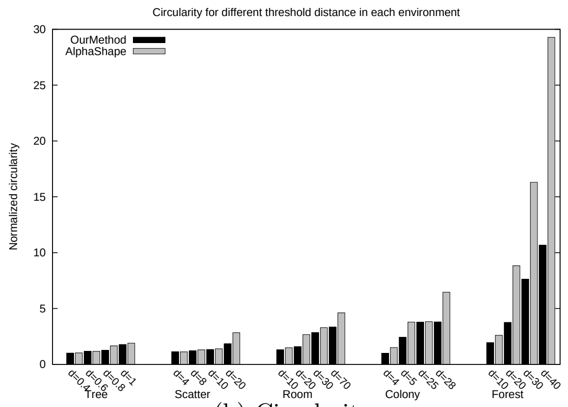

(c) Our Method (d=30)

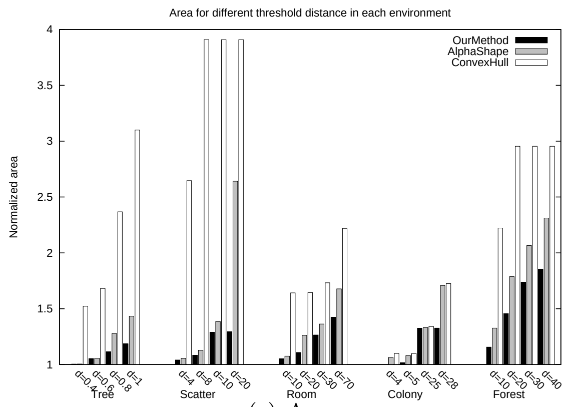
Figure 5: Alpha shape aggregation at lower threshold resembles our aggregation at a higher threshold. (a) Alpha shape aggregation using $\delta = 30$, (b) alpha shape aggregation using $\delta = 20$ resembles, (c) our aggregation using $\delta = 30$



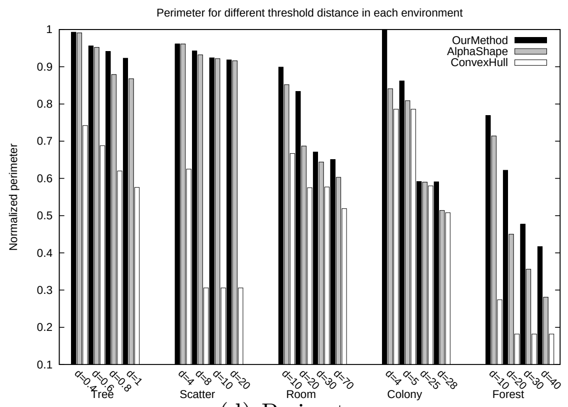
(a) Time



(b) Circularity



(c) Area



(d) Perimeter

Figure 6: (a) Time efficiency of aggregation in 5 environments. (b)-(d) Area, Perimeter and circularity of the objects in the aggregated environments using envelope described by alpha shape, our method and convex hull. The measures are ordered with increase in tolerance for each environment. All the measure are normalized to that of the objects in original environment.

4.3 Application to Motion Planning

With the decrease in tolerance, the free space in the original environment is evolved to generate a hierarchy of maps for the environment. To find a path from a start to a goal position in the environment, we can use the hierarchy of the aggregated environment to guide the planning towards the goal. The idea is to start from a coarser level (high δ) as shown in Fig. 7(a). If the roadmap is unable to solve the query, we use the next level environment and extend the map in the region absent in the previous level used. In Fig. 7(b), the map is only extended in the region not present in Fig. 7(a). The process is continued until the query is solved (See Fig. 7(c)).

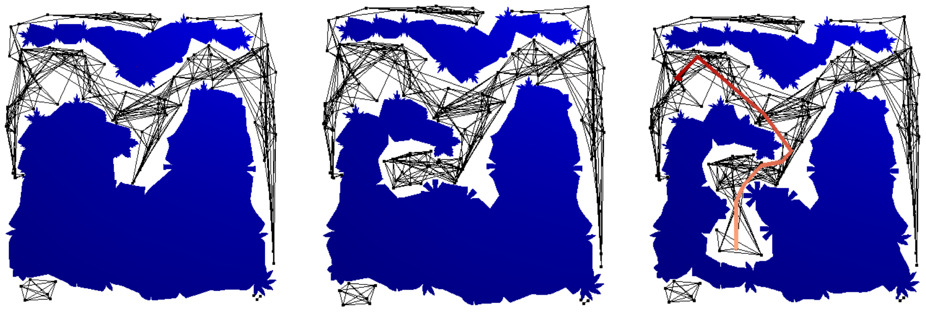

(a) Map at level 1 (b) Map at level 2 (c) Map at level 3 with path

Figure 7: Roadmap using different level of aggregation till the query is solved.

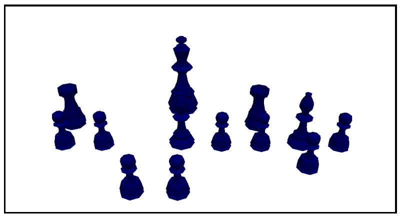
4.4 Extension to three dimension

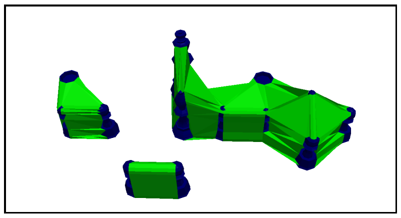
We extended our implementation to three dimensional objects. Instead of using Constrained Delaunay Triangulation, we use the external package Tetgen [39] to construct the Constrained Delaunay Tetrahedralization [36]. As exposed edges are removed in 2D, we remove exposed faces in 3D. A face in CDT is removed only if length of its edges exceeds the alpha returned by the aggregate function.

Fig. 8 shows the aggregated shapes generated in a heterogeneous chess board environment using alpha shape approximation and our method. Similar to the results for 2D environment, alpha shapes generate wider connections with more approximation error (in terms of volume used) than our method. Fig. 9 demonstrates different levels in the aggregation hierarchy using our method on a homogeneous 15-spheres environment. However, the approximation shape can be further improved by using aggregate function on other properties of the face like area and height instead of using length of its edges.

5 Conclusion

This paper provides a general framework to aggregate a set of nearby disjoint objects. The algorithm includes finding the group of nearby objects based on a threshold distance and approximating the shape of the aggregated models as determined by a function of the objects to be merged and the threshold distance. Varying the threshold distance, generates a hierarchy of aggregated models. The hierarchy is further detailed by the shape descriptor function used to generate the approximates of the aggregated models. Our implementation validates the method for 2D polygonal objects. The quality of the aggregation corresponds closely to that of the original model as the threshold


(a) Original


(b) Alpha Shape

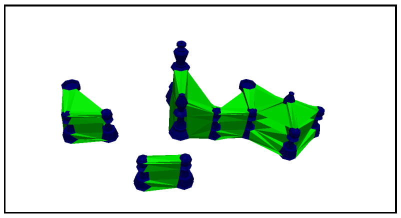
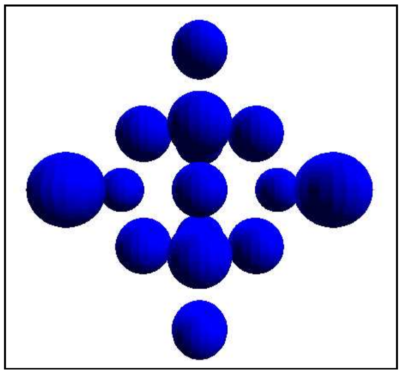
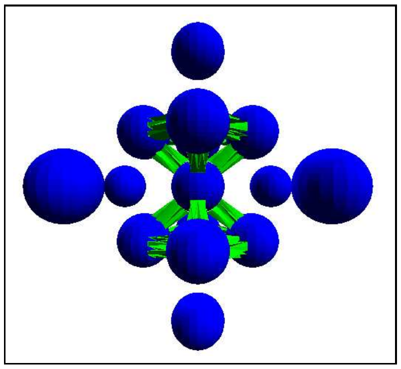
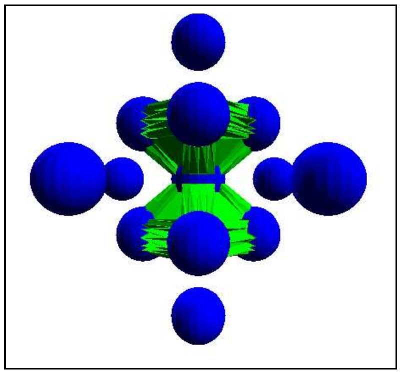
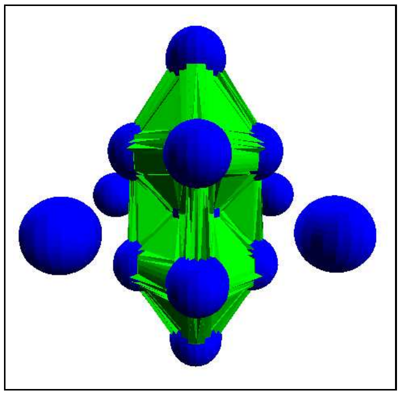

(c) Our Method

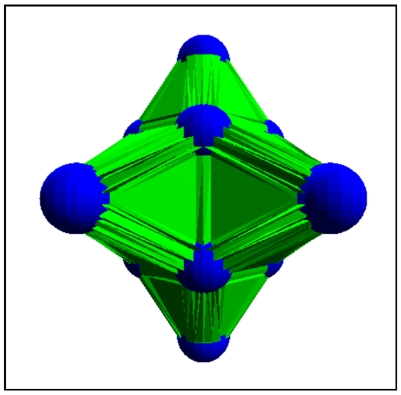
Figure 8: Aggregation in a heterogeneous chess board environment(a) using alpha shapes(b) and our method(c)


(a) Original


(b) Our Method at d=1


(c) Our Method at d=1.5


(d) Our Method at d=2


(e) Our Method at d=3

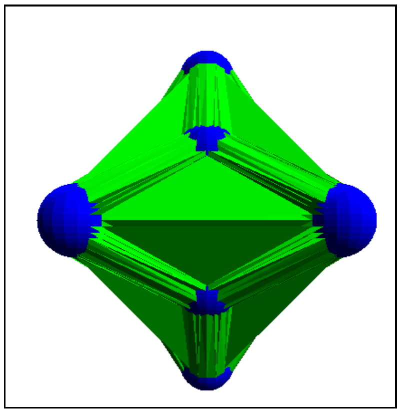

(f) Our Method at d=4

Figure 9: Hierarchical aggregation in a homogeneous three dimensional 15-spheres environment varying the value of δ using our method.

is decreased. Also using a function that can represent the variation in topology generates an intuitive approximate of the original objects even at high threshold. We demonstrated how the levels in approximation can be utilized in a motion planning application. We have extended our implementation to three dimensional objects with results similar to two dimensional objects.

References

[1] *Arcgis,http://www.arcgis.com/features/.*

[2] *Postgis,http://www.postgis.us/.*

[3] Pankaj K. Agarwal and Jiří Matoušek, *Relative neighborhood graphs in three dimensions*, Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '92, Society for Industrial and Applied Mathematics, 1992, pp. 58–65.

[4] Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno, *Hierarchical convex approximation of 3d shapes for fast region selection*, Computer graphics forum, vol. 27, Wiley Online Library, 2008, pp. 1323–1332.

[5] Franz Aurenhammer, *Voronoi diagrams&mdash;a survey of a fundamental geometric data structure*, ACM Comput. Surv. **23** (1991), no. 3, 345–405.

[6] Fausto Bernardini and Chandrajit L Bajaj, *Sampling and reconstructing manifolds using alpha-shapes*, (1997).

[7] Thomas Brinkhoff and Hans-Peter Kriegel, *Approximations for a multi-step processing of spatial joins*, IGIS'94: Geographic Information Systems, Springer, 1994, pp. 25–34.

[8] James C. Caendish, David A. Field, and William H. Frey, *An apporach to automatic three-dimensional finite element mesh generation*, International Journal for Numerical Methods in Engineering **21** (1985), no. 2, 329–347.

[9] Frederic Cazals, Joachim Giesen, Mark Pauly, and Afra Zomorodian, *Conformal alpha shapes*, Proceedings of the Second Eurographics/IEEE VGTC conference on Point-Based Graphics, Eurographics Association, 2005, pp. 55–61.

[10] Ho-Lun Cheng and Xinwei Shi, *Quality mesh generation for molecular skin surfaces using restricted union of balls*, Computational Geometry **42** (2009), no. 3, 196 – 206.

[11] L. P. Chew, *Constrained delaunay triangulations*, Proceedings of the Third Annual Symposium on Computational Geometry (New York, NY, USA), SCG '87, ACM, 1987, pp. 215–222.

[12] R. Chouadria and P. Vron, *Identifying and re-meshing contact interfaces in a polyhedral assembly for digital mock-up*, Engineering with Computers **22** (2006), no. 1, 47–58 (English).

[13] F Dehne and H Noltemeier, *Clustering geometrie objects and applications to layout problems'*.

[14] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, *On the shape of a set of points in the plane*, Information Theory, IEEE Transactions on **29** (1983), no. 4, 551–559.

[15] H. Edelsbrunner and E. P. Mücke, *Three-dimensional alpha shapes*, ACM Trans. Graph. **13** (1994), no. 1, 43–72.

[16] Herbert Edelsbrunner, *Weighted alpha shapes*, University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

[17] Herbert Edelsbrunner and Ernst P. Mücke, *Three-dimensional alpha shapes*, ACM Trans. Graph. **13** (1994), no. 1, 43–72.

[18] Michael Garland and Paul S. Heckbert, *Surface simplification using quadric error metrics*, Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216.

[19] Aleksey Golovinskiy and Thomas Funkhouser, *Randomized cuts for 3d mesh analysis*, ACM SIGGRAPH Asia 2008 papers (SIGGRAPH Asia 2008), 2008, pp. 145:1–145:12.

[20] LeonidasJ. Guibas, DonaldE. Knuth, and Micha Sharir, *Randomized incremental construction of delaunay and voronoi diagrams*, Algorithmica **7** (1992), no. 1-6, 381–413 (English).

[21] Renzhong GUO and Tinghua AI, *Simplification and aggregation of building polygon in automatic map generalization [j]*, JOURNAL OF WUHAN TECHNICAL UNIVERSITY OF SURVEYING AND MAPPING (WTUSM) **1** (2000), 004.

[22] Nate Hagbi and Jihad El-Sana, *Carving for topology simplification of polygonal meshes*, Computer-Aided Design **42** (2010), no. 1, 67 – 75, Advances in Geometric Modelling and Processing.

[23] Philip M. Hubbard, *Approximating polyhedra with spheres for time-critical collision detection*, ACM Trans. Graph. **15** (1996), no. 3, 179–210.

[24] J.W. Jaromczyk and G.T. Toussaint, *Relative neighborhood graphs and their relatives*, Proceedings of the IEEE **80** (1992), no. 9, 1502–1517.

[25] D. Joshi, A.K. Samal, and Leen-Kiat Soh, *Density-based clustering of polygons*, Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on, 2009, pp. 171–178.

[26] Sagi Katz and Ayellet Tal, *Hierarchical mesh decomposition using fuzzy clustering and cuts*, ACM Trans. Graph. **22** (2003), no. 3, 954–961.

[27] Jyh-Ming Lien, *Approximate convex decomposition and its application*, Ph.D. thesis, Texas A& M University, 2006.

[28] Jyh-Ming Lien and Nancy M. Amato, *Approximate convex decomposition of polyhedra and its applications*, Computer Aided Geometric Design **25** (2008), no. 7, 503 – 522, Selected papers from the Solid and Physical Modeling and Applications Symposium 2007 (SPM 2007).

[29] Rong Liu and Hao Zhang, *Segmentation of 3d meshes through spectral clustering*, Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, 2004, pp. 298–305.

[30] Rong Liu, Hao Zhang, and James Busby, *Convex hull covering of polygonal scenes for accurate collision detection in games*, Proceedings of graphics interface 2008 (Toronto, Ont., Canada, Canada), GI '08, Canadian Information Processing Society, 2008, pp. 203–210.

[31] Kasra Manavi, Alan Kuntz, and Lydia Tapia, *Geometrical insights into the process of antibody aggregation*, (2013).

[32] David W. Matula and Robert R. Sokal, *Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane*, Geographical Analysis **12** (1980), no. 3, 205–222.

[33] D.C. Rapaport, J.E. Johnson, and J. Skolnick, *Supramolecular self-assembly: molecular dynamics modeling of polyhedral shell formation*, Computer Physics Communications **121122** (1999), no. 0, 231 – 235, Proceedings of the Europhysics Conference on Computational Physics.

[34] Jarek Rossignac and Paul Borrel, *Multi-resolution 3d approximations for rendering complex scenes*, Springer, 1993.

[35] Jonathan Richard Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, Applied Computational Geometry: Towards Geometric Engineering (Ming C. Lin and Dinesh Manocha, eds.), Lecture Notes in Computer Science, vol. 1148, Springer-Verlag, May 1996, From the First ACM Workshop on Applied Computational Geometry, pp. 203–222.

[36] ______, *Constrained delaunay tetrahedralizations and provably good boundary recovery*, In Eleventh International Meshing Roundtable, 2002, pp. 193–204.

[37] ______, *General-dimensional constrained delaunay and constrained regular triangulations i: Combinatorial properties*, Discrete and Computational Geometry, 2005.

[38] Shymon Shlafman, Ayellet Tal, and Sagi Katz, *Metamorphosis of polyhedral surfaces using decomposition*, Computer Graphics Forum, vol. 21, Wiley Online Library, 2002, pp. 219–228.

[39] Hang Si, *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, http://tetgen.berlios.de/.

[40] Charlie C L Wang, *Approximate boolean operations on large polyhedral solids with partial mesh reconstruction*, Visualization and Computer Graphics, IEEE Transactions on **17** (2011), no. 6, 836–849.

[41] Jiann-Shing Wu and Jin-Jang Leou, *New polygonal approximation schemes for object shape representation*, Pattern Recognition **26** (1993), no. 4, 471 – 484.

[42] Dengsheng Zhang and Guojun Lu, *Review of shape representation and description techniques*, Pattern recognition **37** (2004), no. 1, 1–19.

[43] J Zhang, A Samal, and LK Soh, *Polygon-based spatial clustering.*

[44] Xie Zhong, Ma Lina, and Wu Liang, *Polygon aggregation algorithm using the simple feature model*, Electrical and Control Engineering (ICECE), 2010 International Conference on, 2010, pp. 2900–2904.