Computational Geometry Project Proposal
INFO-F420

Compatible Point Set Triangulations

Muthi Dorel-Adrian 000 427 829
Perret Romain 000 427 613
Université Libre de Bruxelles
MA-INFO, Academic Year 2020 - 2021

3 December 2020

1 Introduction

The *triangulation* in the case of a point set is the process of connecting as many points from this set as possible such that the resulting edges don't intersect each other. We thus end up with a set of triangles, contained in the convex-hull of the set. A *compatible triangulation* corresponds to a triangulation of a point set that has the same combinatorial structure as another one, where the points delimiting their triangles can be linked by a bijection, implying that their vertices represent two isomorphic sets. Given two sets of n points S_1 and S_2, a triangulation T_1 of S_1 and a triangulation T_2 of S_2, are considered as compatible triangulations if there is a bijection ϕ such that ijk is a triangle of T_1 with no points of S_1 inside iff $\phi(i)\phi(j)\phi(k)$ is a triangle of T_2 with no points of S_2 inside. However, this relationship is not ensured for every two sets of two dimensional points (or planar points). In particular, determining compatible triangulations for two such sets in general position, where no three points are aligned, and for which the number of extreme points is the same, remains an *open problem* [8].

The problem is considered to be true for some [1], while also being considered false for points which are not in general position. Furthermore, compatible triangulations do not always exist if the bijection between the points is given and fixed [9]. When the bijection is not given, the conjecture is proven only for point sets with at most three points interior to the hull [1]. Nonetheless, compatible triangulations are always possible if considering the addition of a bounded linear number of interior points called *Steiner points*. A list of articles related to the topic: [3, 7, 6].

2 Implementation

In order to expose the topic, we will aim at letting the user place points in an arbitrary manner for two separate figures, which will then be used to build planar graphs contained in the respective convex hull of their point set. The user will then be able to connect points in these sets to create polygons (simple at first), before querying the application to detect if these have compatible triangulations. Because finding compatible triangulations requires the two point sets to exhibit the same number of points, and similarly for the number of extreme points defining the convex hull, we will only allow the user to place points for the second set once having finished the first one. This way, the user will be limited and will only be able to place the required number of points in the second figure. Once two such point sets will be drawn, we will then compute the triangulations of both and find if bijective mappings can be made between them. Of course, listing all the different triangulations will require us to implement some kind of data structure to store segments or triangles of each triangulation in order to quickly determine if a segment or triangle contained in one triangulation is also included in the second one. These mappings could then be highlighted graphically by the means of different colors.

3 Work division

Work done:

- Romain: Input of the point set.
  - Creation of two canvas.
  - Input of a point set.
  - Convex Hull computation of the point sets
  - Validation of the input of the first point set in order to unlock the input for the second point set.
  - Validation of the second input convex hull size according to the first one.
  - Verification of general position in both point sets.
- Dorel: Enumerating the triangulation of the point sets by brute force.
  - Search all the inner edges of the point set convex hull.
  - Compute the inner edges number of a point set triangulation using the point set theorem about the edges and the faces of its triangulation.
  - Search all the combinations of the inner edges with the same size as the computed number of inner edges.
  - Reject all the combinations with intersecting edges.
  - Extract the triangles(faces) of the valid combinations.
- Romain: Searching other methods to enumerate the triangulations.

Remaining work:

- Romain: Find a compatible triangulation between the two sets if there exists one.
- Dorel: Generate different colors for a set of triangles.
- Romain: Dislpay with colors the compatible triangulations.

4 Input of point sets

The webpage contains two P5 canvas, one for each point set. The user must insert first a point set in the left canvas and by clicking on the validation button, he will be allowed to insert the second point set in the right canvas. The convex hull of both point sets are computed using the Graham Scan algorithm and if the convex hull of the second point set does not have the same size as the first one the user will have to insert a again the second point set. General position of

the points is checked at each insertion of a point set in both canvas. Once the both point sets are validated the search of a compatible triangulation between the two point sets begins.

5 Enumerating triangulations of a point set

We can enumerate by brute force the triangulations of a point set. The first step is to search all the inner edges of the point set convex hull found at the input of the point sets. Second step is to search the different combinations of those inner edges. We compute the edges number of any point set triangulation using the formula of the theorem 9.1 from *Computational Geometry : Algorithms and Applications 3rd* [4]. Let P be a point set with n points and k by the points number of the convex hull boundary. Then for any triangulation of P the formula for the edges number is: 3n-3-k and the formula for the number of triangles (faces) is: 2n-2-k. We implemented also the last formula to make verifications, if needed. We subtract from the edges number of the triangulation, the edges number of the convex hull in order to find the triangulation inner edges number. This number is used to find all the combinations of the inner edges with the size of that number. The third step is to reject all the combinations where the edges are intersecting. As we used the edges number of the theorem, we assume that the combinations without intersections are valid triangulations as their edges number correspond to the one specified by the theorem. The fourth step consists in the extraction of the triangles from the combination of edges and the convex hull edges. We browse all the 3-tuples of the triangulation edges and we save the 3-tuple in a list if it is forming a triangle. Three edges are forming a triangle if each pair of edges has a common vertex and if each common vertex is different from the other. This last step is needed in order to search and display in colors a compatible triangulation between the two given points set.

We tried implementing a DCEL (doubly connected edge list) data structure to store each inner edges combination in order to browse the triangles (faces) of the triangulation but the structure is not working always due to a mistake not yet identified. We would initialize first the DCEL with the edges of the convex hull and then inserting one by one the inner edges we would verify if there is an intersection and decide then to reject the combination. We found complicated to manage the case where we insert an edge not connected to any existing vertex and we decided to insert first the inner edges which have one vertex or two vertices already in the DCEL. In some cases an error is occurring while trying to insert the inner edges.

We tried understanding the algorithm from the article *An efficient algorithm for enumeration of triangulations* [2] but we couldn't understand well enough the algorithm in order to implement it. They use the flip process of Delaunay triangulation and a lexicographical order for the points, edges and triangulations in order to build a tree containing all the triangulations which is a spanning tree

of the graph containing all the point set triangulations. The root of the tree is the largest lexicographical triangulation. Starting from any triangulation the algorithm enumerates all the triangulations of the point set. Their algorithm use several data structures. They used a non specified data structure for the triangulation. They used a DCEL and 4 trees (3 blanaced search trees and 1 static search tree) to store divers data during the process of enumerating the triangulations. *Parallel Enumeration of Triangulations* [5] is another article which is explaining an parallel algorithm to enumerate the triangulation but we didn't have the time to explore it more and in addition of that Javascript is a single threaded language.

References

[1] Oswin Aichholzera et al. "Towards compatible triangulations". In: (2003). URL: https://core.ac.uk/download/pdf/82242222.pdf.

[2] Sergei Bespamyatnikh. "An efficient algorithm for enumeration of triangulations". In: (2002). URL: https://www.sciencedirect.com/science/article/pii/S0925772102001116.

[3] Jeff Danciger, Satyan L. Devadoss, and Don Sheeny. "Compatible triangulations and point partitions by series-triangular graphs". In: (2005). URL: https://arxiv.org/pdf/cs/0502043.pdf.

[4] Mark de Berg et al. *Computational Geometry : Algorithms and Applications*. 3rd. Springer-Verlag Berlin and Heidelberg GmbH & Co. K (7 mars 2008), 2008. ISBN: 978-3540779735.

[5] Charles Jordan, Michael Joswig, and Lars Kastner. "Parallel Enumeration of Triangulations". In: (2018). URL: https://arxiv.org/abs/1709.04746.

[6] Zhiguang Liu et al. "High-quality Compatible Triangulations and their Application in Interactive Animation". In: (2018). URL: http://nrl.northumbria.ac.uk/id/eprint/35739/1/Accepted%20version.pdf.

[7] Anna Lubiw and Debajyoti Mondal. "On Compatible Triangulations with a Minimum Number of Steiner Points". In: (2018). URL: https://arxiv.org/pdf/1706.09086.pdf.

[8] J. O'Rourke. *Problem 38: Compatible Triangulations*. URL: http://www.science.smith.edu/~jorourke/TOPP/P38.html#Problem.38. (accessed: 09.10.2020).

[9] Alan Saalfeld. "Joint triangulations and triangulation maps". In: (1987). URL: https://www.census.gov/srd/papers/pdf/rr87-23.pdf.