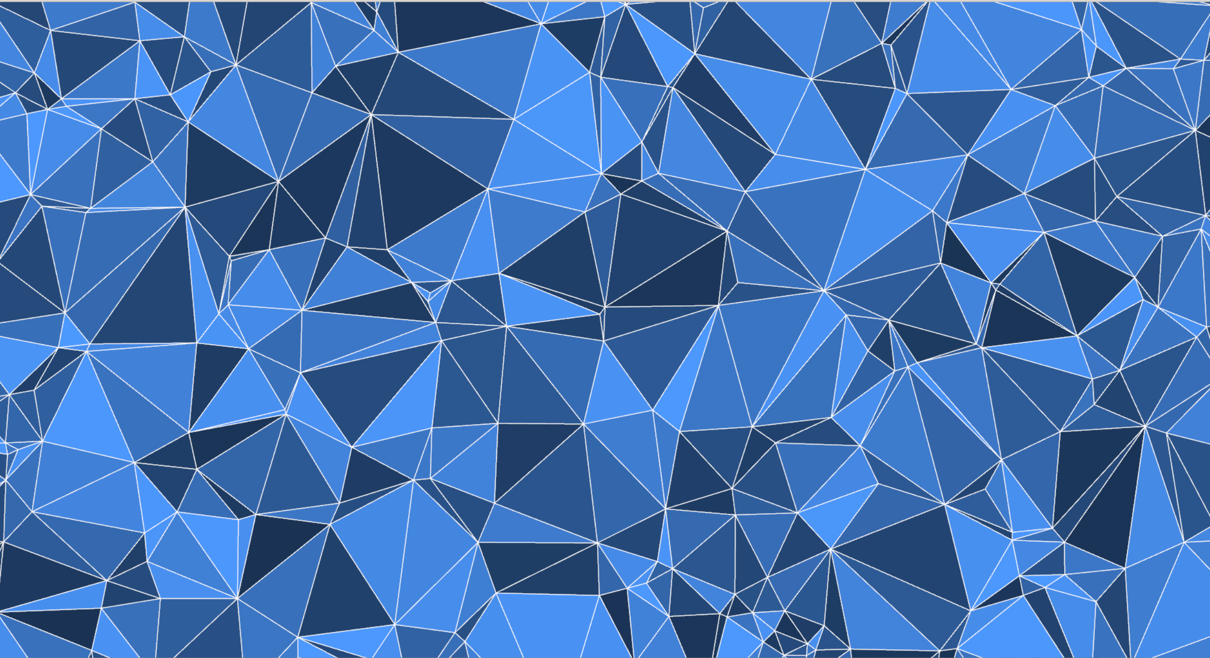
Delaunay Triangulation Mosaic



Algorithmical Geometry: Delaunay Triangulation

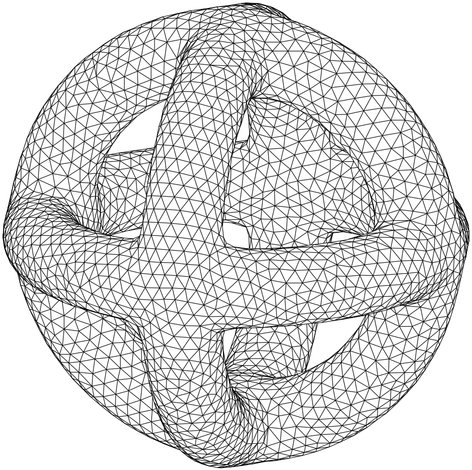
Markus Pawellek

January 16, 2022

Outline

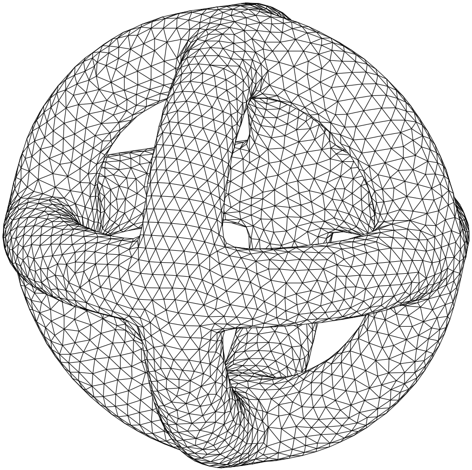
Introduction

Introduction



*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

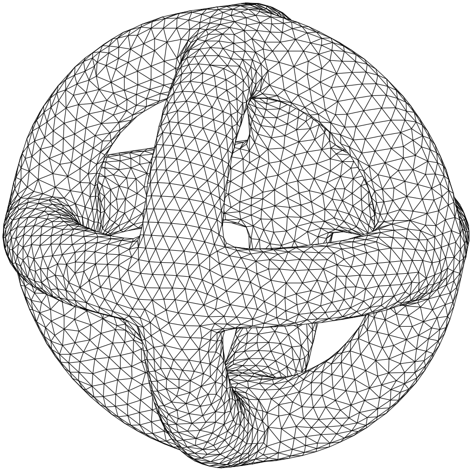
Introduction



Educational Problems:

*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

Introduction

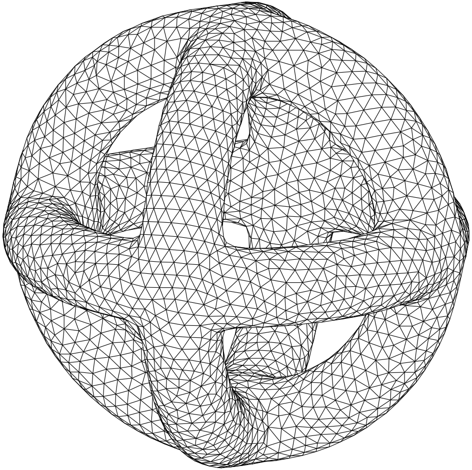


Educational Problems:

- Many Resources

*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

Introduction

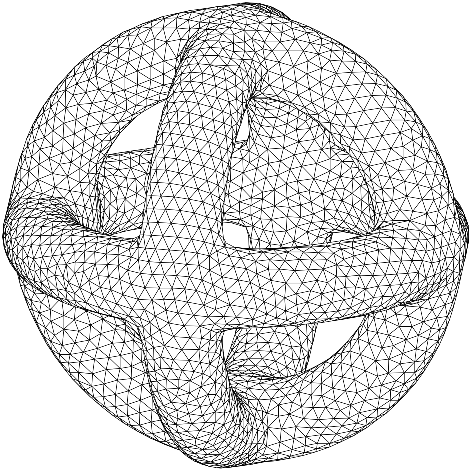


Educational Problems:

- Many Resources
- Duality to Voronoi Diagrams

*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

Introduction

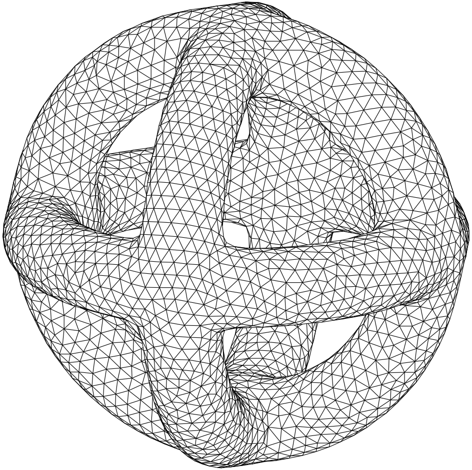


Educational Problems:

- Many Resources
- Duality to Voronoi Diagrams
- Multiple Algorithm Types: Incremental, Sweepline, Divide-and-Conquer

*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

Introduction



Educational Problems:

- Many Resources
- Duality to Voronoi Diagrams
- Multiple Algorithm Types: Incremental, Sweepline, Divide-and-Conquer
- Varying Data Structures

*https://upload.wikimedia.org/wikipedia/commons/b/b8/Approx-3tori.svg, December 29, 2021

Introduction: Previous Work

Introduction: Previous Work

1980 Lee and Schachter, “Two Algorithms for Constructing a Delaunay Triangulation”

Introduction: Previous Work

1980 Lee and Schachter, “Two Algorithms for Constructing a Delaunay Triangulation”

1985 Guibas and Stolfi, “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”

Introduction: Previous Work

1980 Lee and Schachter, "Two Algorithms for Constructing a Delaunay Triangulation"

1985 Guibas and Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams"

1987 Dwyer, "A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations"

Introduction: Previous Work

1980 Lee and Schachter, “Two Algorithms for Constructing a Delaunay Triangulation”

1985 Guibas and Stolfi, “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”

1987 Dwyer, “A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations”

1996 Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”

Introduction: Previous Work

1980 Lee and Schachter, “Two Algorithms for Constructing a Delaunay Triangulation”

1985 Guibas and Stolfi, “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”

1987 Dwyer, “A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations”

1996 Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”

2014 Fuetterling, Lojewski, and Pfreundt, “High-Performance Delaunay Triangulation for Many-Core Computers”

Introduction: Previous Work

1980 Lee and Schachter, “Two Algorithms for Constructing a Delaunay Triangulation”

1985 **Guibas and Stolfi, “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”**

1987 Dwyer, “A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations”

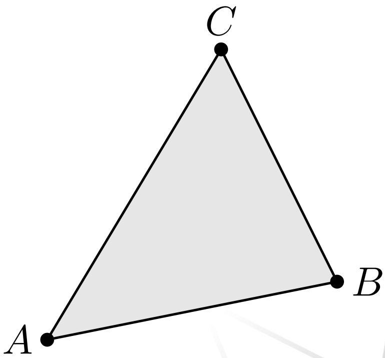
1996 Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”

2014 Fuetterling, Lojewski, and Pfreundt, “High-Performance Delaunay Triangulation for Many-Core Computers”

Mathematical Preliminaries

Triangle

$A, B, C \in \mathbb{R}^2$ affinely independent define vertices of a triangle.



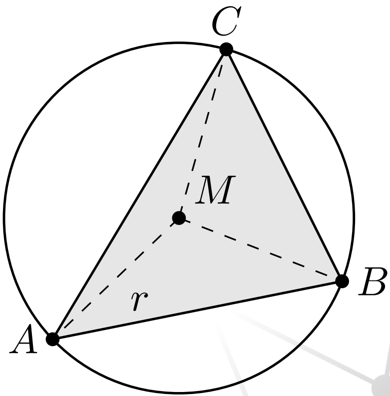
Mathematical Preliminaries: Triangle and Circumcircle

Triangle

$A, B, C \in \mathbb{R}^2$ affinely independent define vertices of a triangle.

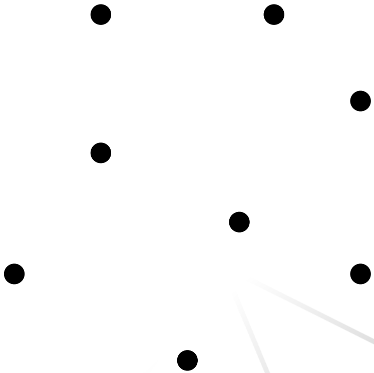
Circumcircle

Circle that intersects with all vertices of the triangle.



Point Set

$\mathcal{V} \subset \mathbb{R}^2$ finite, $\#\mathcal{V} \geq 3$,
affinely span $\mathbb{R}^2$



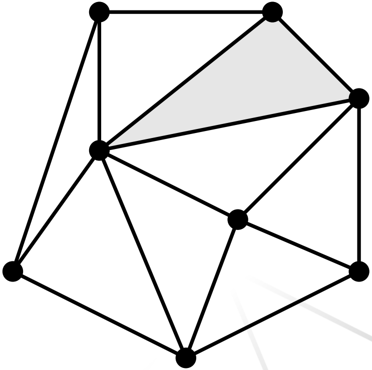
Mathematical Preliminaries: Triangulation

Point Set

$\mathcal{V} \subset \mathbb{R}^2$ finite, $\#\mathcal{V} \geq 3$,
affinely span $\mathbb{R}^2$

Triangulation

Planar straight-line graph over $\mathcal{V}$ such that its edges form a maximal subset of non-crossing edges.



Point Set

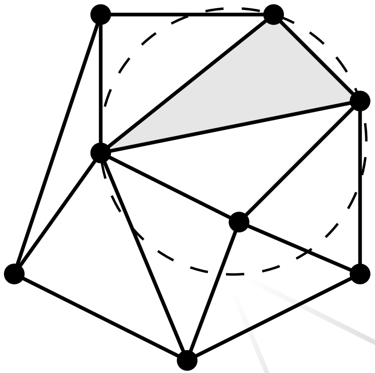
$\mathcal{V} \subset \mathbb{R}^2$ finite, $\#\mathcal{V} \geq 3$,
affinely span $\mathbb{R}^2$

Triangulation

Planar straight-line graph over $\mathcal{V}$ such that its edges form a maximal subset of non-crossing edges.

Delaunay Triangulation

Circumcircle of any triangle contains no other points of $\mathcal{V}$.



Mathematical Preliminaries: Delaunay Triangulation

Point Set

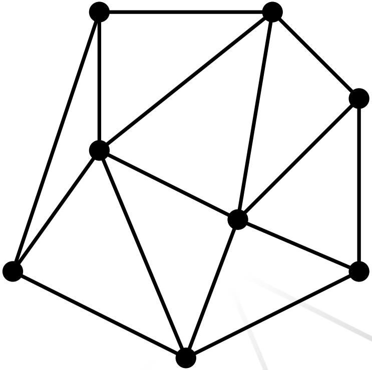
$\mathcal{V} \subset \mathbb{R}^2$ finite, $\#\mathcal{V} \geq 3$,
affinely span $\mathbb{R}^2$

Triangulation

Planar straight-line graph over $\mathcal{V}$ such that its edges form a maximal subset of non-crossing edges.

Delaunay Triangulation

Circumcircle of any triangle contains no other points of $\mathcal{V}$.



Point Set

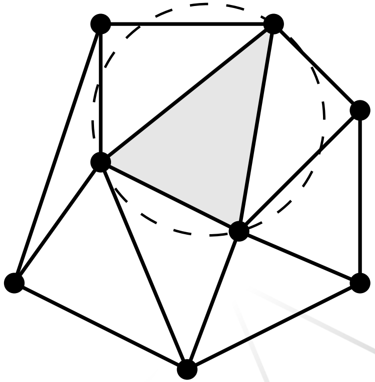
$\mathcal{V} \subset \mathbb{R}^2$ finite, $\#\mathcal{V} \geq 3$,
affinely span $\mathbb{R}^2$

Triangulation

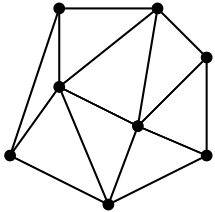
Planar straight-line graph over $\mathcal{V}$ such that its edges form a maximal subset of non-crossing edges.

Delaunay Triangulation

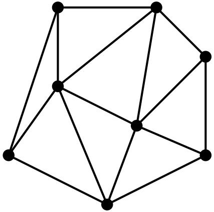
Circumcircle of any triangle contains no other points of $\mathcal{V}$.



Mathematical Preliminaries: Properties

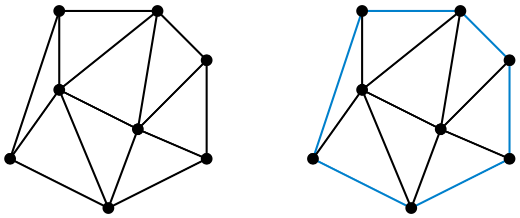


Mathematical Preliminaries: Properties



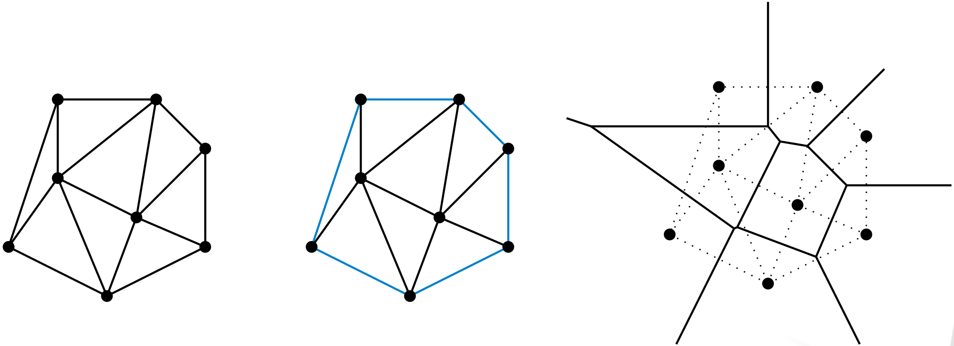
- Optimality: maximization of the minimum angle of all angles

Mathematical Preliminaries: Properties



- Optimality: maximization of the minimum angle of all angles
- Convex hull is contained

Mathematical Preliminaries: Properties



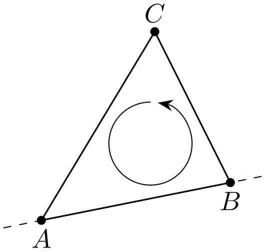
- Optimality: maximization of the minimum angle of all angles
- Convex hull is contained
- Voronoi diagram is the dual

Mathematical Preliminaries: Properties of Delaunay Triangulation

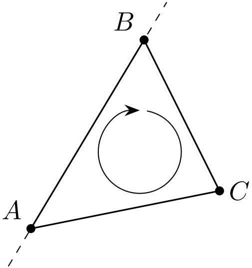
- Duality to Voronoi Diagram (also useful for proofs)
- always exists -> proof
- If no points are cocircular, unique -> proof
- optimality: maximization of the minimum angle of all angles -> proof, reason why delaunay is good
- boundary is convex hull

Geometric Primitives

Geometric Primitives: Counterclockwise

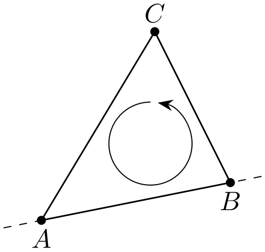


$\longleftrightarrow$

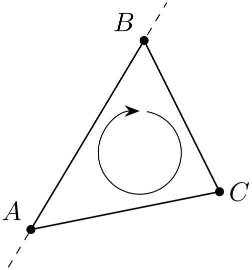


Counterclockwise Order

Geometric Primitives: Counterclockwise

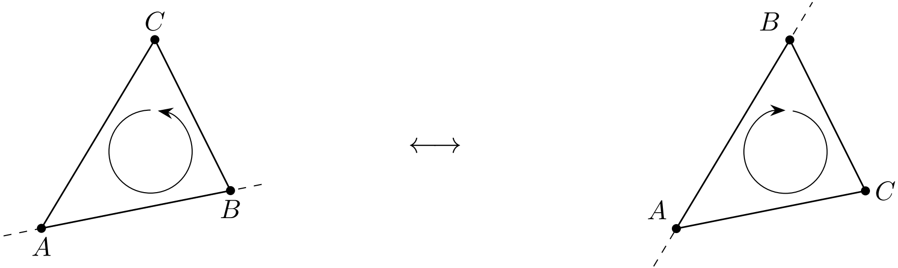


$\longleftrightarrow$



Counterclockwise Order $\iff$ C is left of $\overline{AB}$

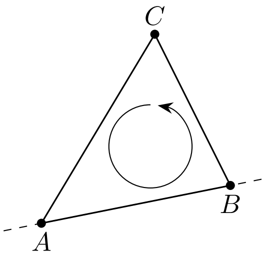
Geometric Primitives: Counterclockwise



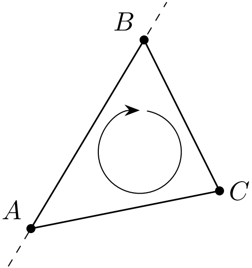
Counterclockwise Order $\iff$ C is left of $\overline{AB}$

$$0 < \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix}$$

Geometric Primitives: Counterclockwise



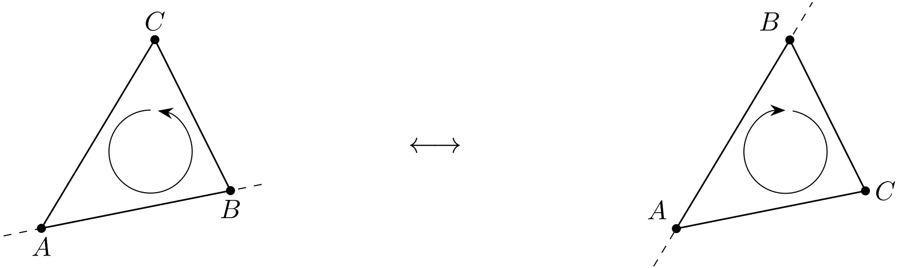
$\longleftrightarrow$



Counterclockwise Order $\iff$ C is left of $\overline{AB}$

$$0 < \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix} = \begin{vmatrix} B_x - A_x & B_y - A_y \\ C_x - A_x & C_y - A_y \end{vmatrix}$$

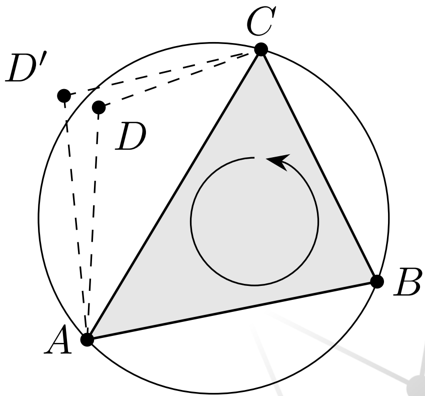
Geometric Primitives: Counterclockwise



Counterclockwise Order $\iff$ C is left of $\overline{AB}$

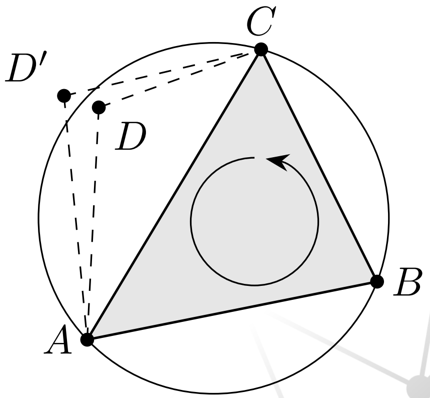
$$0 < \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix} = \begin{vmatrix} B_x - A_x & B_y - A_y \\ C_x - A_x & C_y - A_y \end{vmatrix} = \det \begin{pmatrix} B - A & C - A \end{pmatrix}$$

Geometric Primitives: Inside Circumcircle



Geometric Primitives: Inside Circumcircle

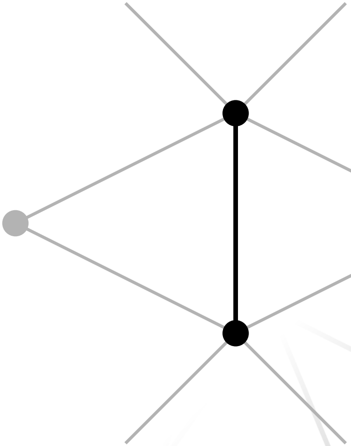
$$0 < \begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{vmatrix}$$



Data Structure

Data Structure: Quad-Edge

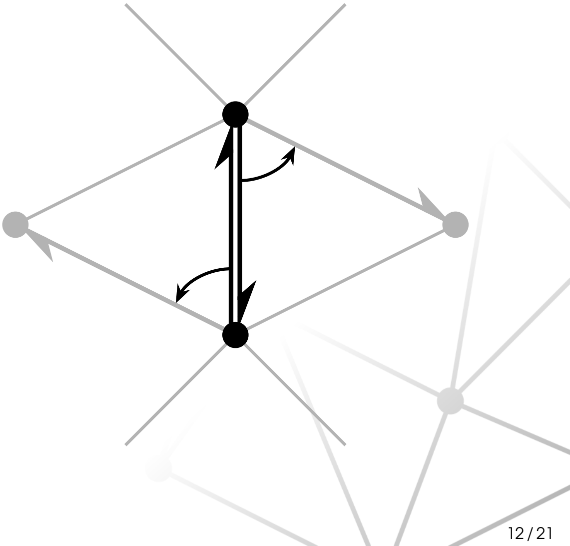
Edge-Based List-Like Data Structure:



Data Structure: Quad-Edge

Edge-Based List-Like Data Structure:

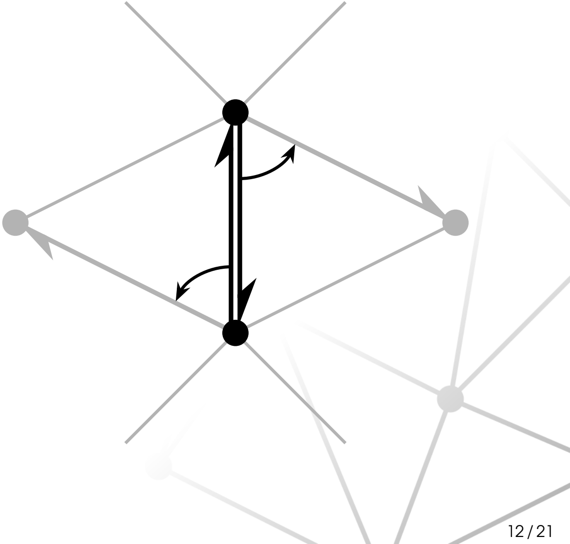
- Directed edges for vertices



Data Structure: Quad-Edge

Edge-Based List-Like Data Structure:

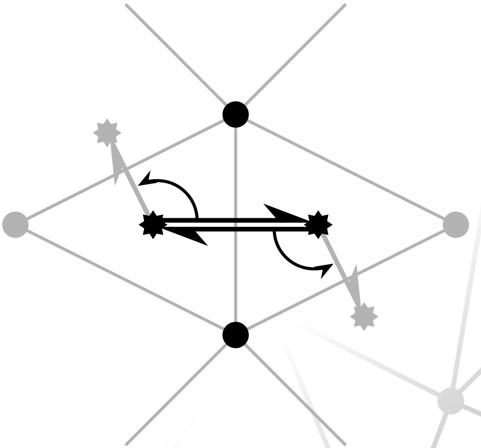
- Directed edges for vertices
- Pointer to ccw. next directed edge with same origin vertex



Data Structure: Quad-Edge

Edge-Based List-Like Data Structure:

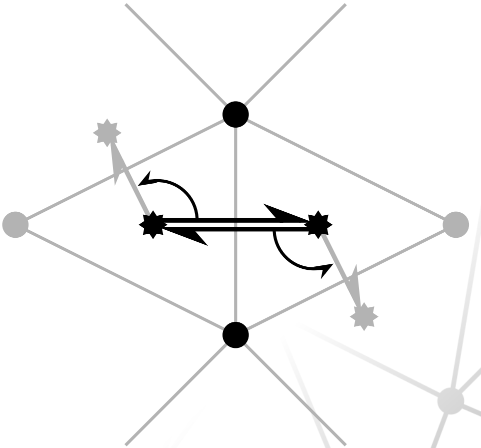
- Directed edges for vertices
- Pointer to ccw. next directed edge with same origin vertex
- Directed dual edges for adjacent faces



Data Structure: Quad-Edge

Edge-Based List-Like Data Structure:

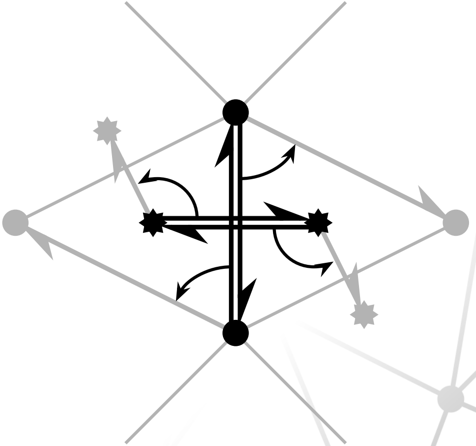
- Directed edges for vertices
- Pointer to ccw. next directed edge with same origin vertex
- Directed dual edges for adjacent faces
- Pointer to ccw. next directed dual edge with same origin face



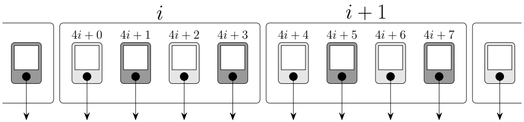
Data Structure: Quad-Edge

Edge-Based List-Like Data Structure:

- Directed edges for vertices
- Pointer to ccw. next directed edge with same origin vertex
- Directed dual edges for adjacent faces
- Pointer to ccw. next directed dual edge with same origin face



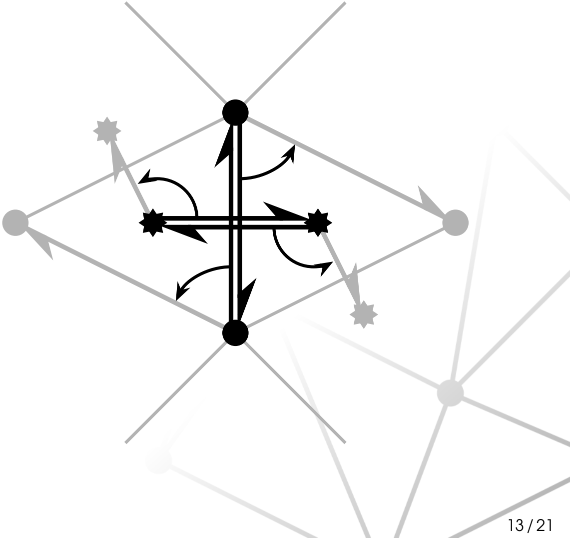
Data Structure: Quad-Edge Implementation



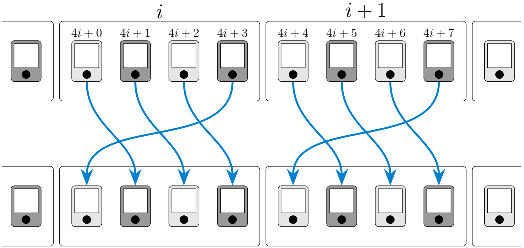

```
struct edge {
  size_t next;
  size_t data;
};

struct quad_edge {
  edge data[4];
};

vector<vertex>    vertices{};
vector<quad_edge> quad_edges{};
vector<size_t>    free_edges{};
```

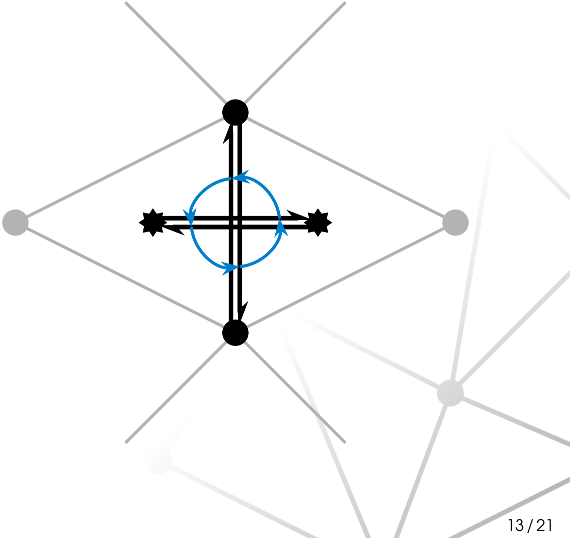


Data Structure: Quad-Edge Implementation

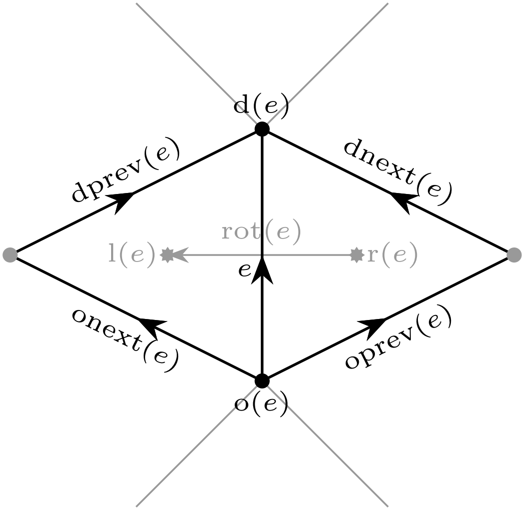


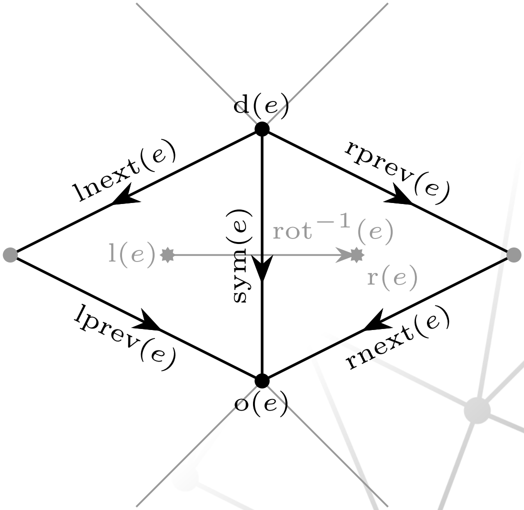
$$\mathrm{rot}\colon \mathbb{N}_0 \to \mathbb{N}_0$$

$$\mathrm{rot}(x) = 4 \cdot \left\lfloor \frac{x}{4} \right\rfloor + (x + 1 \mod 4)$$



Data Structure: Quad-Edge Functions





Data Structure: Quad-Edge Operations

- edge functions
- create edge
- splice
- connect
- delete edge

Algorithm

Algorithm: Overview

Algorithm: Overview

Triangulation Algorithm

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Subroutine: Triangulate

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Subroutine: Triangulate

1. If point count is smaller than four, make edge or triangle and return.

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Subroutine: Triangulate

1. If point count is smaller than four, make edge or triangle and return.
2. Split point set into left and right half.

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Subroutine: Triangulate

1. If point count is smaller than four, make edge or triangle and return.
2. Split point set into left and right half.
3. Triangulate left and right half.

Algorithm: Overview

Triangulation Algorithm

1. Sort the given point set by increasing x coordinate.
2. Triangulate sorted point set.

Subroutine: Triangulate

1. If point count is smaller than four, make edge or triangle and return.
2. Split point set into left and right half.
3. Triangulate left and right half.
4. Merge left and right triangulations.

Algorithm: Merge Triangulations

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

1. Compute and add lower common tangent.

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

1. Compute and add lower common tangent.
2. Use lower common tangent as baseline.

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

1. Compute and add lower common tangent.
2. Use lower common tangent as baseline.
3. Loop until baseline becomes upper common tangent:

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

1. Compute and add lower common tangent.
2. Use lower common tangent as baseline.
3. Loop until baseline becomes upper common tangent:
   3.1 Remove invalid edges adjacent to and above baseline.

Algorithm: Merge Triangulations

Subroutine: Merge Triangulations

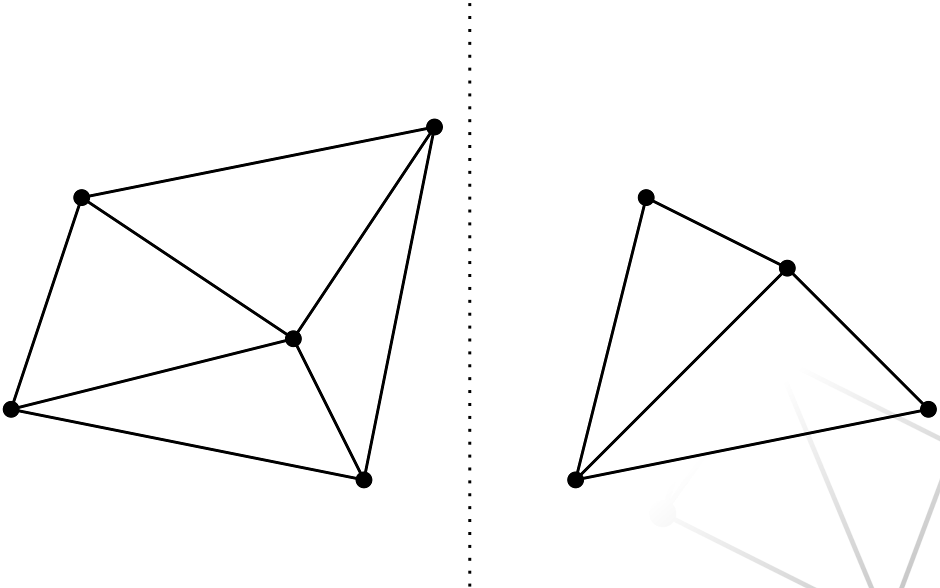
1. Compute and add lower common tangent.
2. Use lower common tangent as baseline.
3. Loop until baseline becomes upper common tangent:
   3.1 Remove invalid edges adjacent to and above baseline.
   3.2 Insert cross edge above baseline.

Algorithm: Merge Triangulations

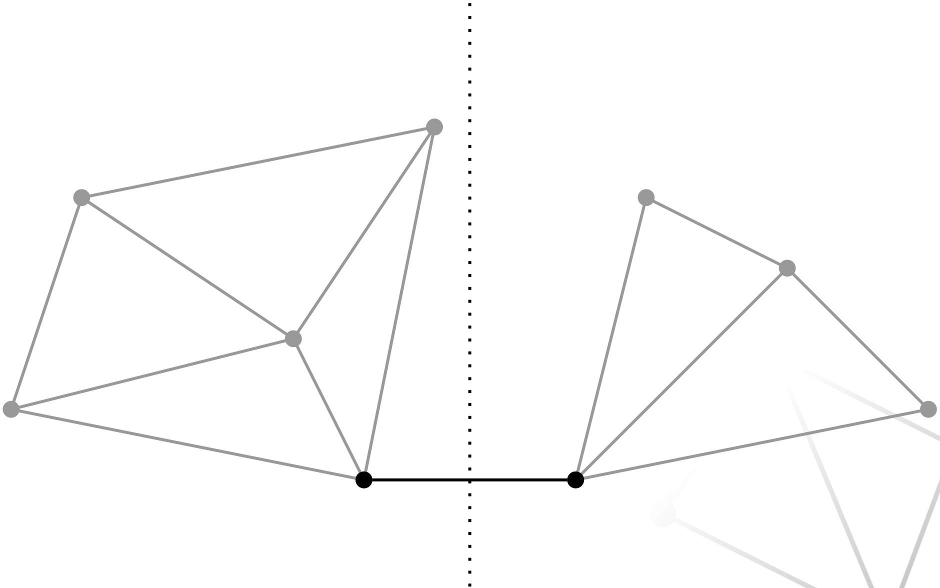
Subroutine: Merge Triangulations

1. Compute and add lower common tangent.
2. Use lower common tangent as baseline.
3. Loop until baseline becomes upper common tangent:
   3.1 Remove invalid edges adjacent to and above baseline.
   3.2 Insert cross edge above baseline.
   3.3 Make this cross edge the new baseline.

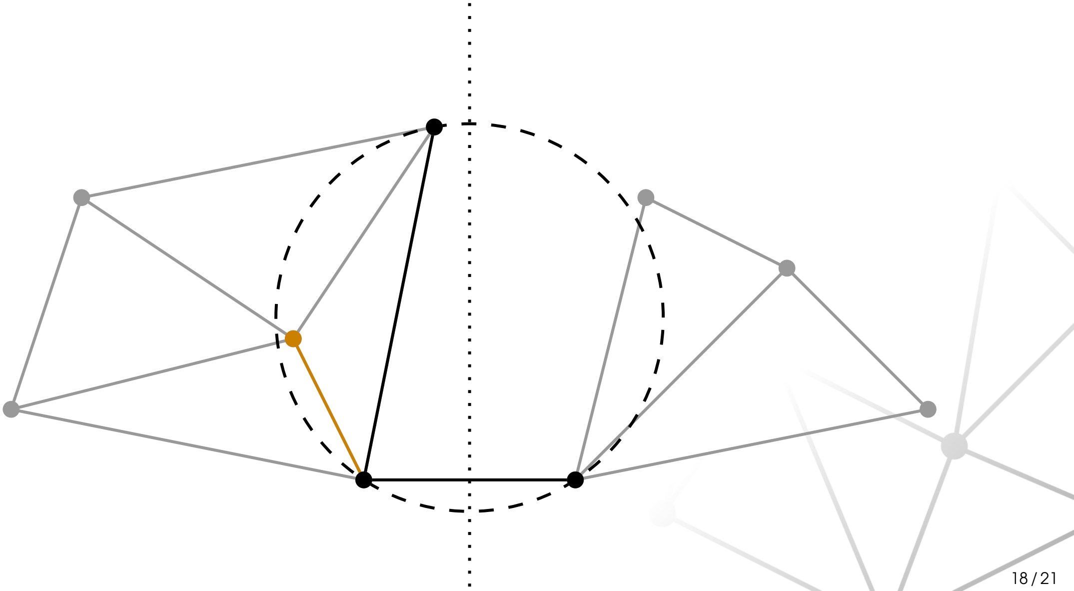
Algorithm: Merge Triangulations Example



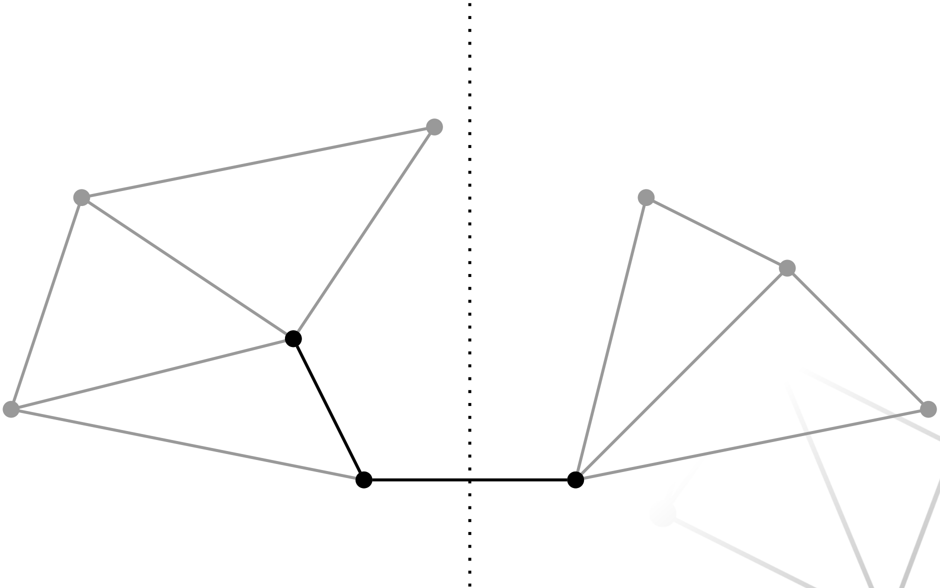
Algorithm: Merge Triangulations Example



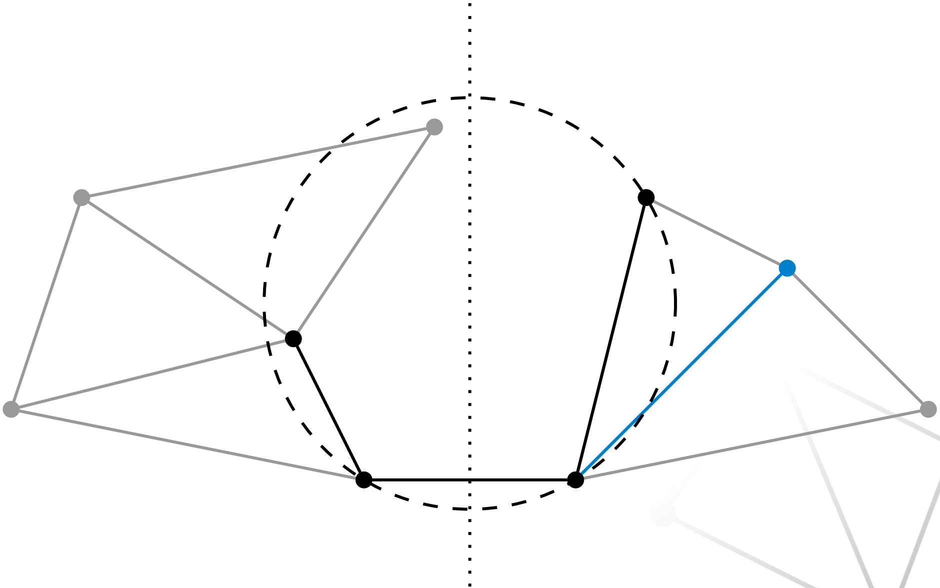
Algorithm: Merge Triangulations Example



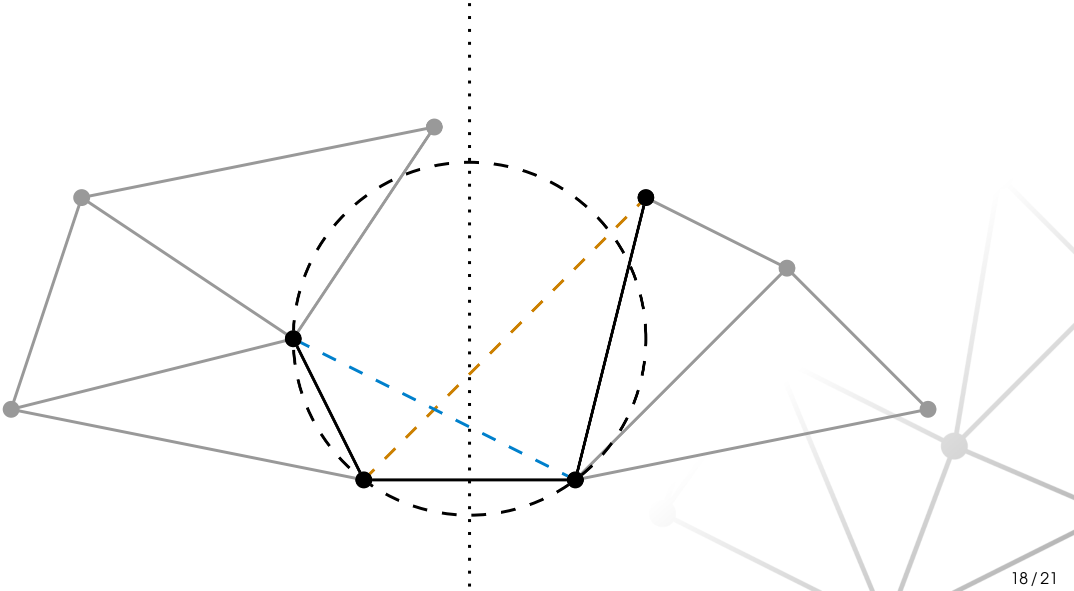
Algorithm: Merge Triangulations Example



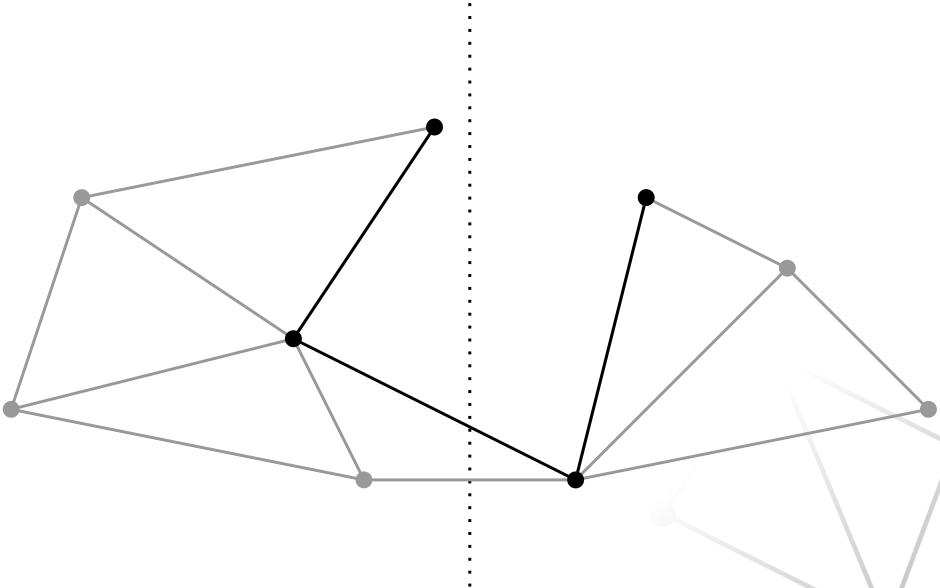
Algorithm: Merge Triangulations Example



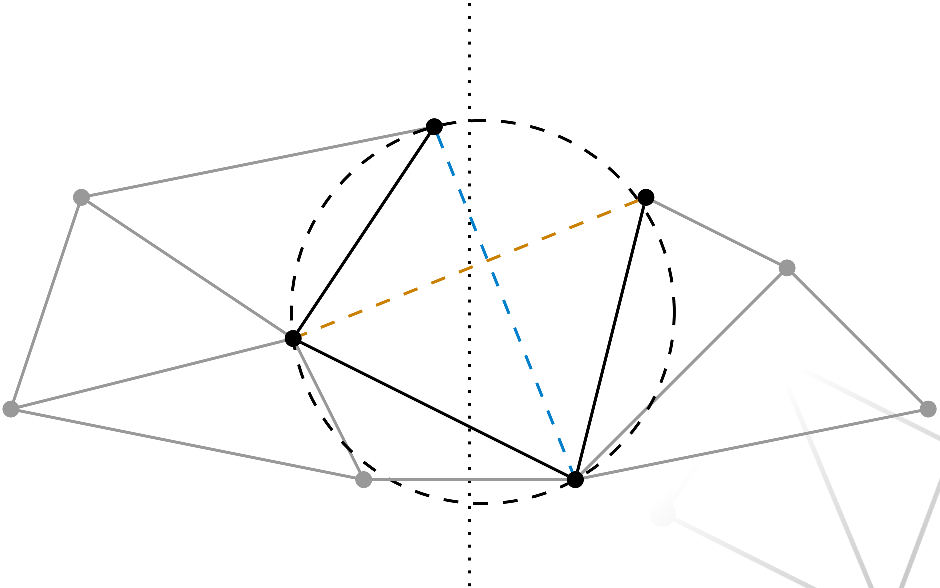
Algorithm: Merge Triangulations Example



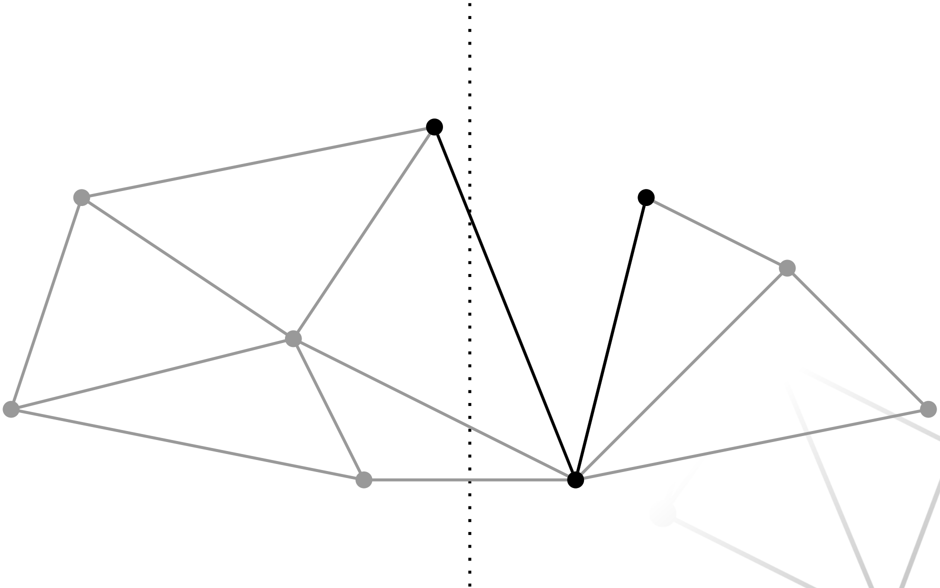
Algorithm: Merge Triangulations Example



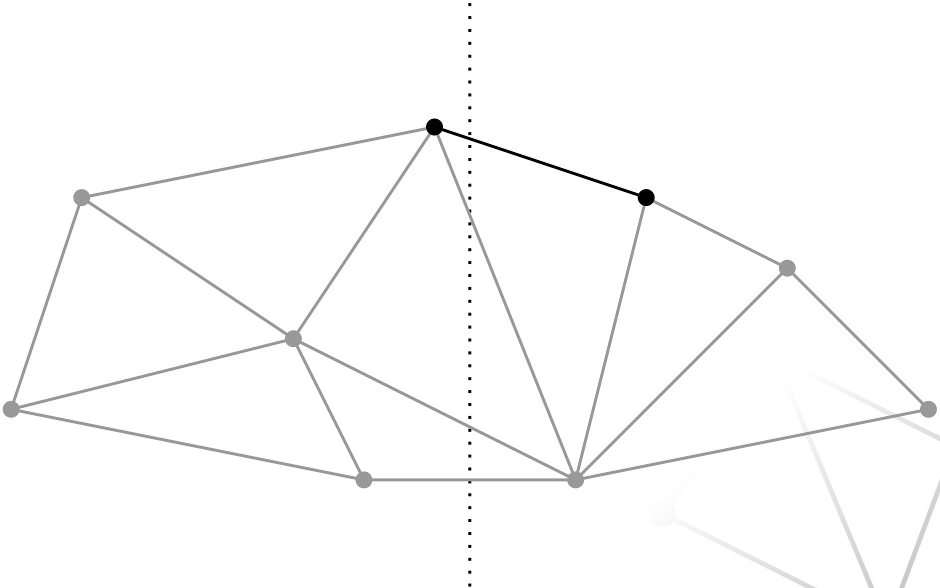
Algorithm: Merge Triangulations Example



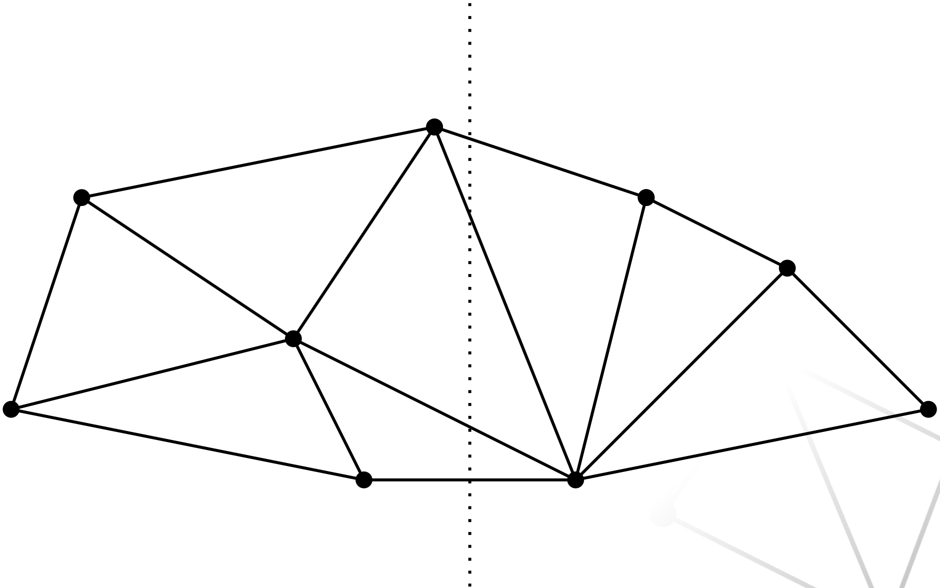
Algorithm: Merge Triangulations Example



Algorithm: Merge Triangulations Example



Algorithm: Merge Triangulations Example



Algorithm: Correctness

Algorithm: Complexity

Implementation

Implementation

- Geometric Primitives need exact computation and therefore arbitrary precision

Applications

Conclusions

Thank you for Your Attention!

References

(1) D. T. Lee and B. J. Schachter. “Two Algorithms for Constructing a Delaunay Triangulation”. In: *International Journal of Computer and Information Sciences* 9 (1980), pp. 219–242. DOI: 10.1007/BF00977785.

(2) Leonidas Guibas and Jorge Stolfi. “Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams”. In: *ACM Transactions on Graphics* 4 (April 1985), pp. 74–123. DOI: 10.1145/282918.282923. URL: http://sccg.sk/~samuelcik/dgs/quad_edge.pdf (visited on 11/07/2020).

(3) Rex A. Dwyer. “A Faster Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations”. In: *Algorithmica* 2 (November 1987), pp. 137–151. DOI: 10.1007/BF01840356.

(4) Jonathan Richard Shewchuk. “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”. In: *Applied Computational Geometry: Towards Geometric Engineering*. Ed. by Ming C. Lin and Dinesh Manocha. Vol. 1148. Lecture Notes in Computer Science. From the First ACM Workshop on Applied Computational Geometry. Springer-Verlag, May 1996, pp. 203–222. URL: https:

(7) D. F. Watson. “Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes”. In: *The Computer Journal* 24 (1981), pp. 167–172. DOI: 10.1093/comjnl/24.2.167.

(8) A. Bowyer. “Computing Dirichlet Tessellations”. In: *The Computer Journal* 24 (1981), pp. 162–166. DOI: 10.1093/comjnl/24.2.162.

(9) Christoph Burnikel. *Delaunay Graphs by Divide and Conquer*. 1998. URL: https://pure.mpg.de/rest/items/item_1819432_4/component/file_2599484/content (visited on 11/07/2020).

(10) P. Cignoni, C. Montani, and R. Scopigno. “DeWall: A Fast Divide-and-Conquer Delaunay Triangulation Algorithm in E^d”. In: *Computer-Aided Design* 30 (1998), pp. 333–341. DOI: 10.1016/S0010-4485(97)00082-1.

(11) Jyrki Katajainen and Markku Koppinen. “Constructing Delaunay Triangulations by Merging Buckets in Quad-Tree Order”. In: *Fundamenta Informaticae* 11 (April 1988), pp. 275–288.