Reinforcement Learning for Remeshing

Final Projection – EEC 289A (Fall 2017)

By: Ahmed Mahmoud

UCDAVIS
ELECTRICAL AND COMPUTER ENGINEERING

Idea based on

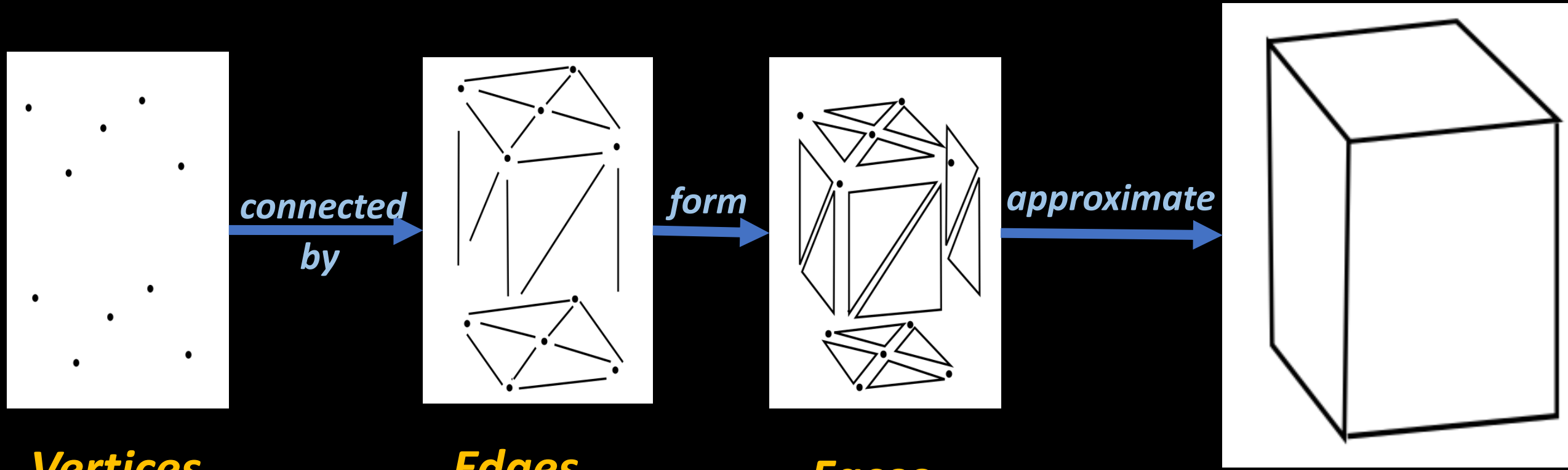
“A Constrained Resampling Strategy for Mesh Improvement” **in Symposium on Computational Processing (SGP) – July 2017**

Agenda

- What is a *mesh* and *remeshing*? **(1~2 mins)**
- Problem statement **(3 mins)**
- Solution for the problem **(3 mins)**
- Where *RL* is needed the solution **(1~2 mins)**

Mesh

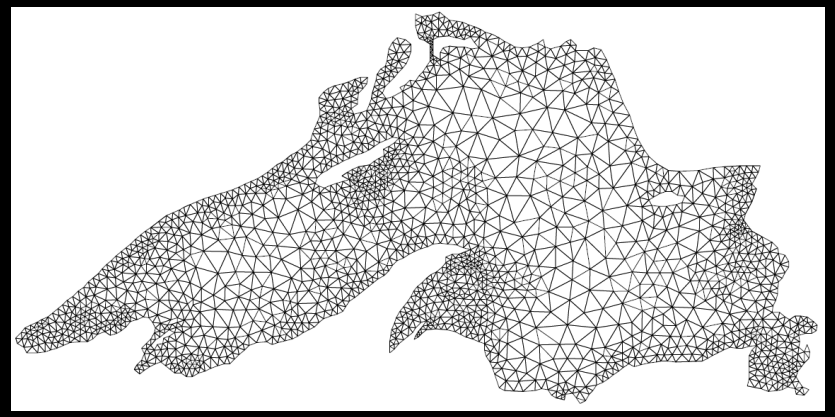
- A mesh is vertices connected by edges to compose faces that decomposed some a subset of a domain



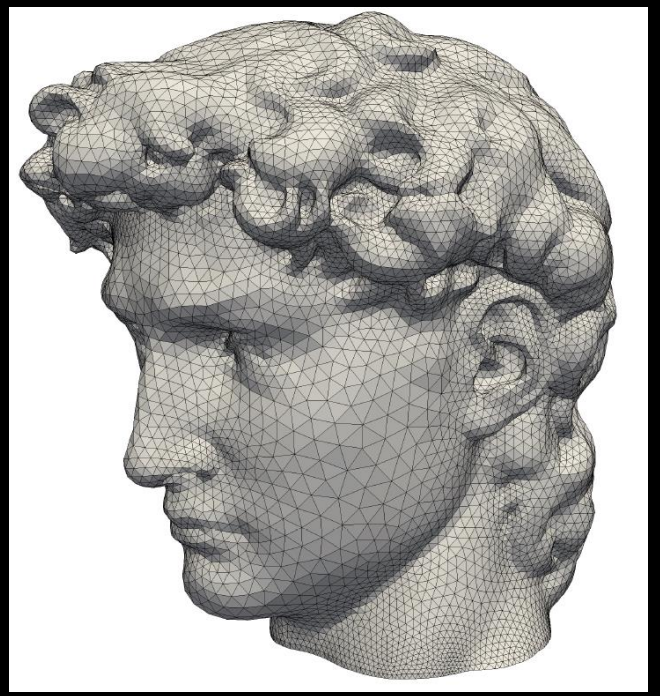
Credits: Wikipedia

Mesh

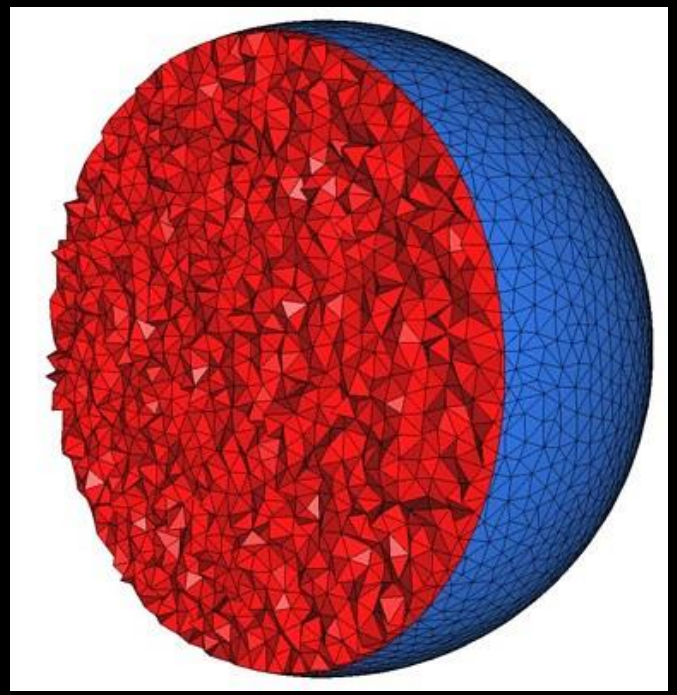
- The domain subset we are trying to approximate could be 2D domain, 2D domain embedded in 3D or 3D domain



2D Domain



Curved surface
2D embedded in 3D

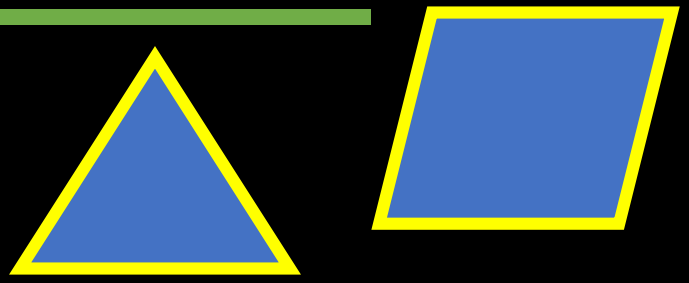


3D Domain

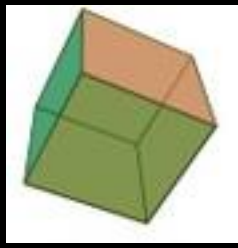
Credits: CGAL

Mesh

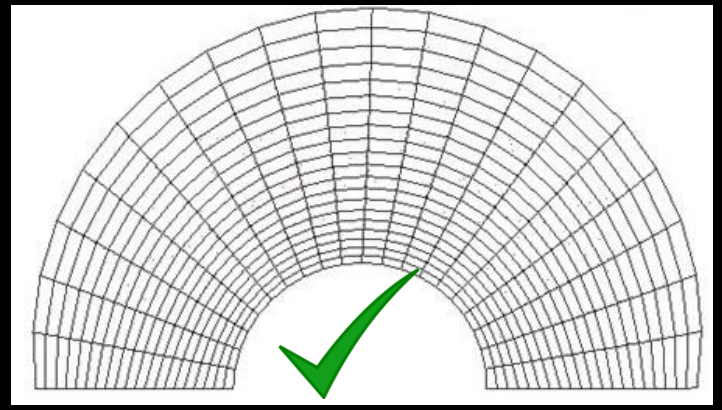
- Faces are most commonly triangles or quad

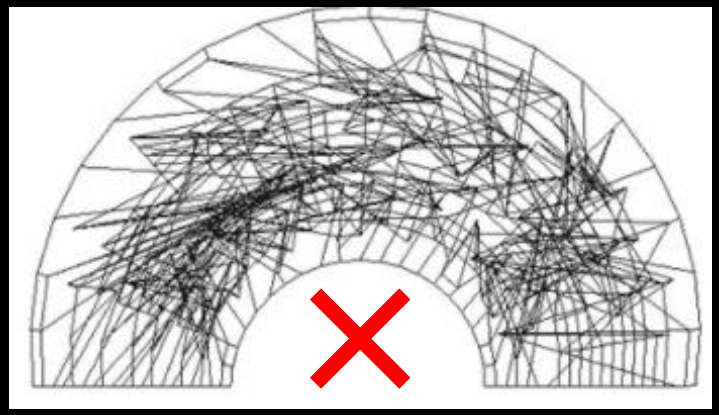


- In 3D, they becomes tetrahedron and hexahedrons



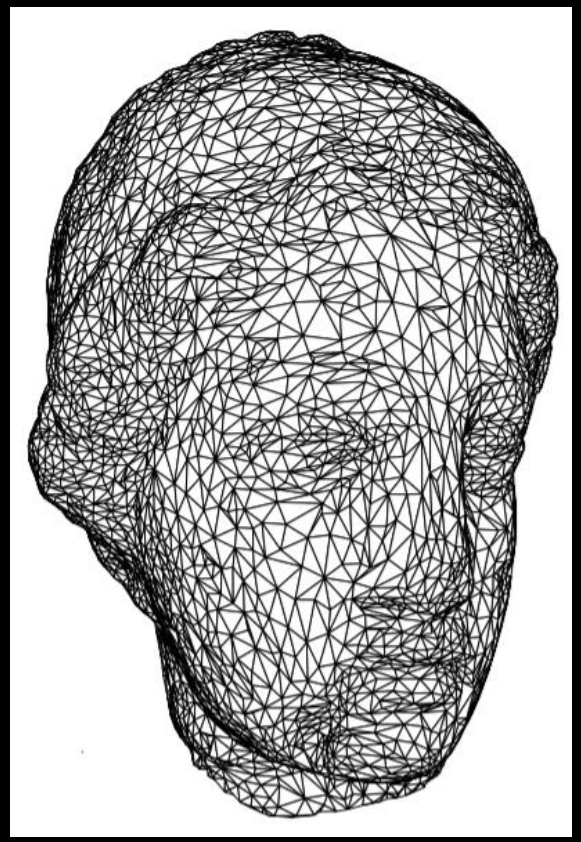
- Do not allow overlapping faces or intersecting elements.

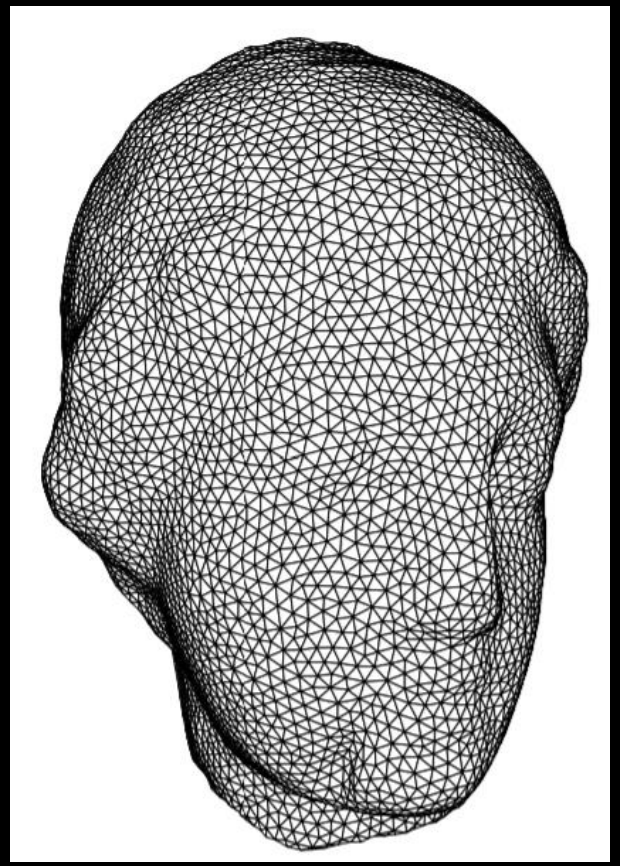




Remeshing

- Turning an input low-quality mesh into a high-quality one
- Quality is based on the application
- Common desirable quality: *regularity*

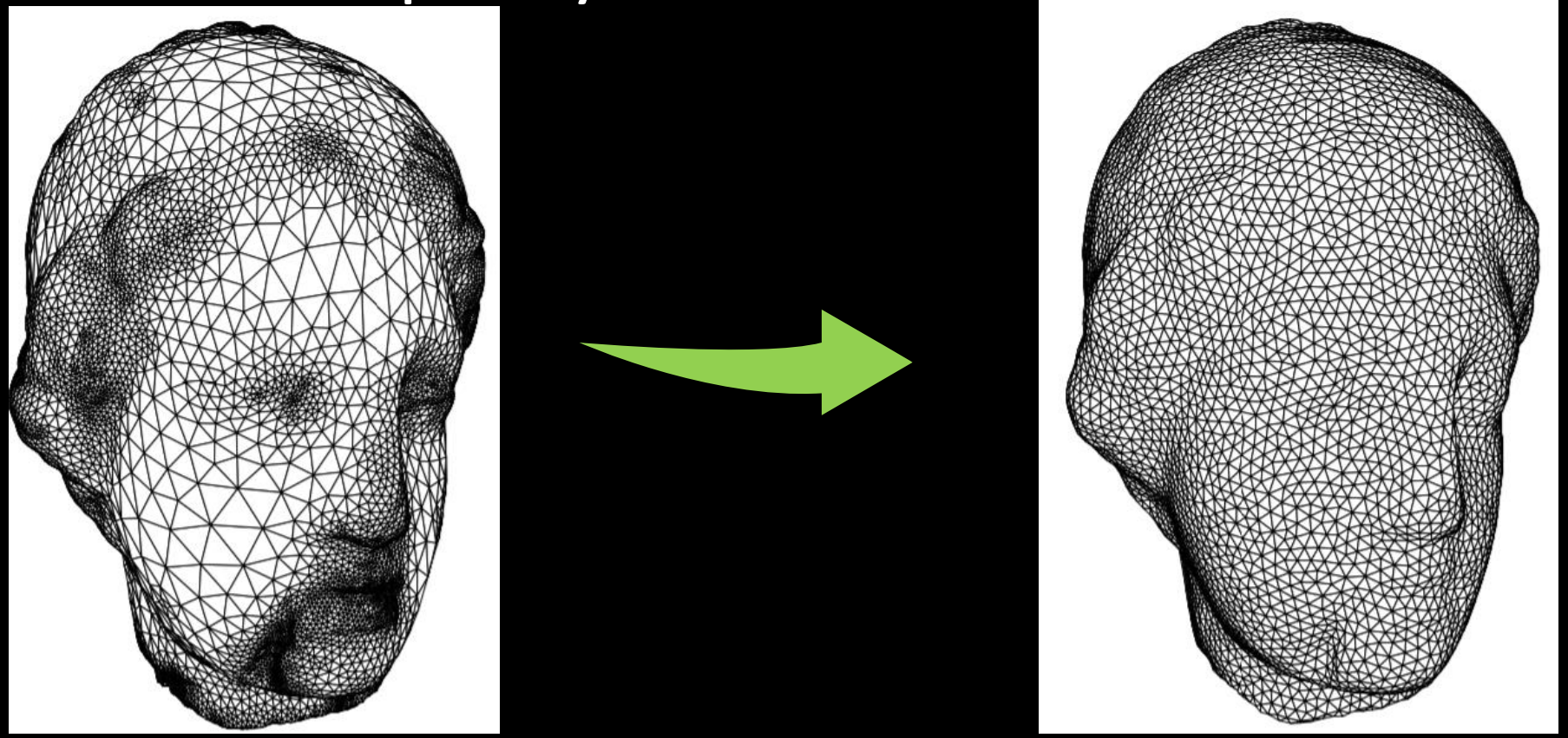




Interactive Geometry Remeshing – Alliez ,et al. 2002

Remeshing

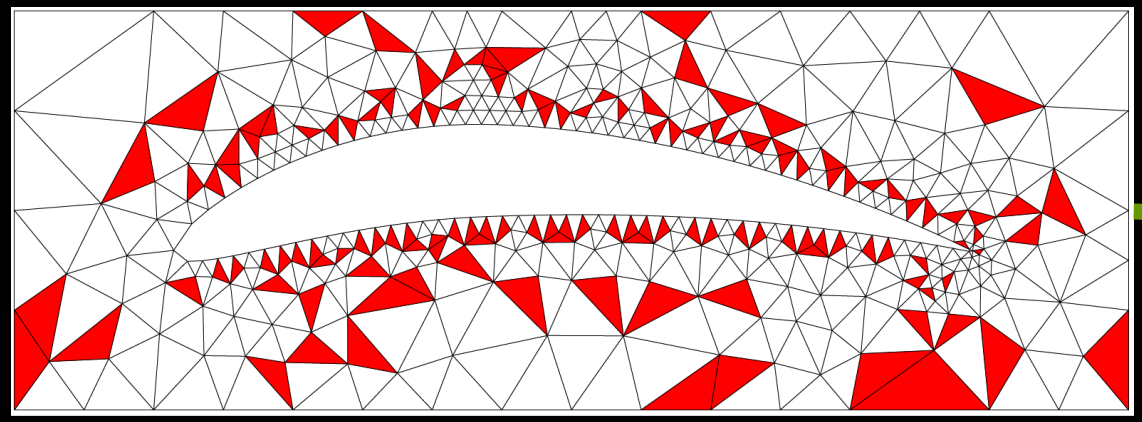
- Turning an input low-quality mesh into a high-quality one
- Quality is based on the application
- Common desirable quality: *vertex count*



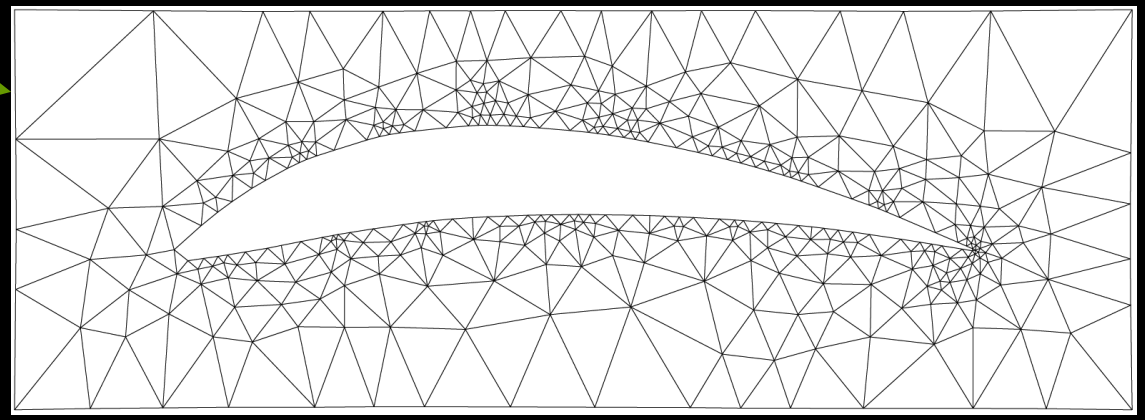
Interactive Geometry Remeshing – Alliez ,et al. 2002

Remeshing

- Turning an input low-quality mesh into a high-quality one
- Quality is based on the application
- Common desirable quality: *non-obtuse triangles*



Red = obtuse triangle



Problem Statement

Given an input mesh, modify it such that no angle is greater than 90^{o}. It is allowed to move vertices, add vertices and remove vertices. Additionally, try to use as few vertices as possible.

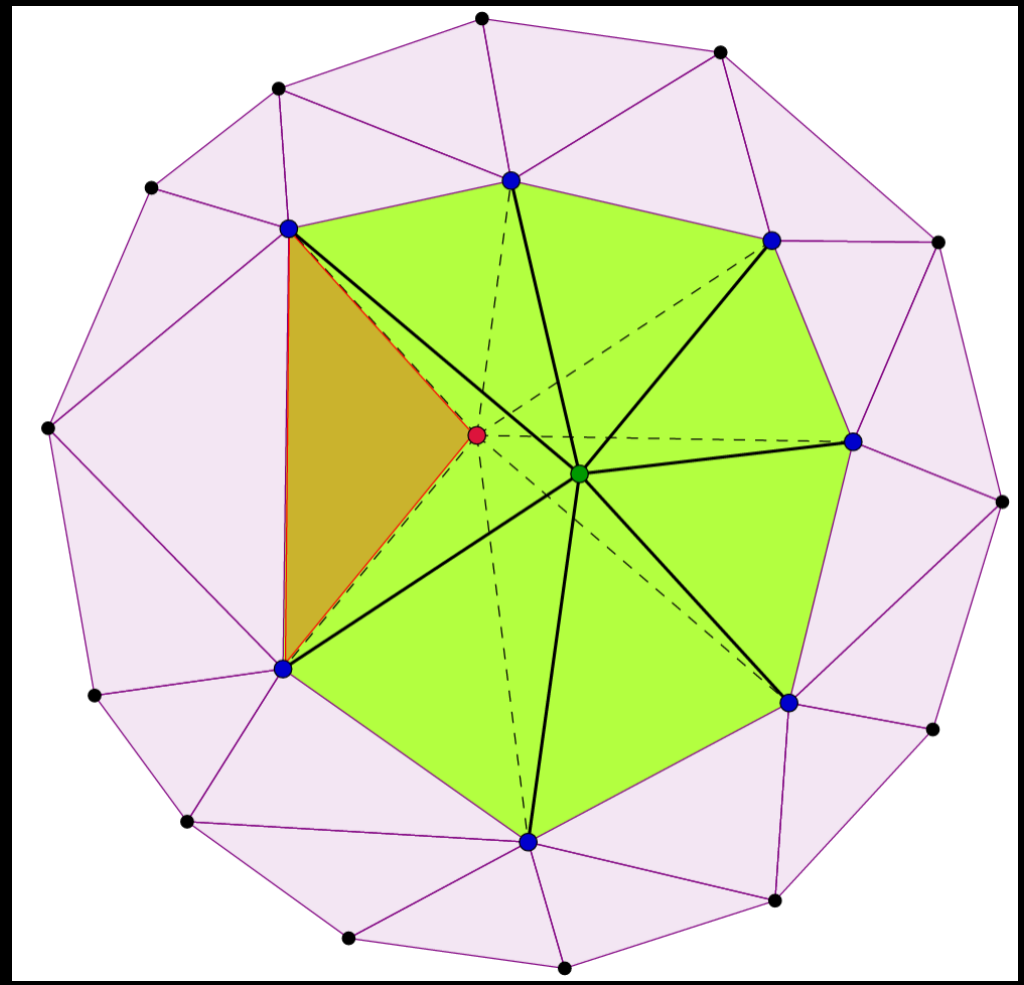
Solution

1) Iterate over all mesh elements
2) Select an element with low quality *(obtuse triangle)*
3) Apply one of the remeshing operator
    - guarantee to not create another bad element
    - will either fix this element or leave it as it is
4) Move on to another bad elements and apply 3)
5) Keep iterating until all elements are fixed or reach maximum number of iterations

Solution

Remeshing operator

1) Relocation

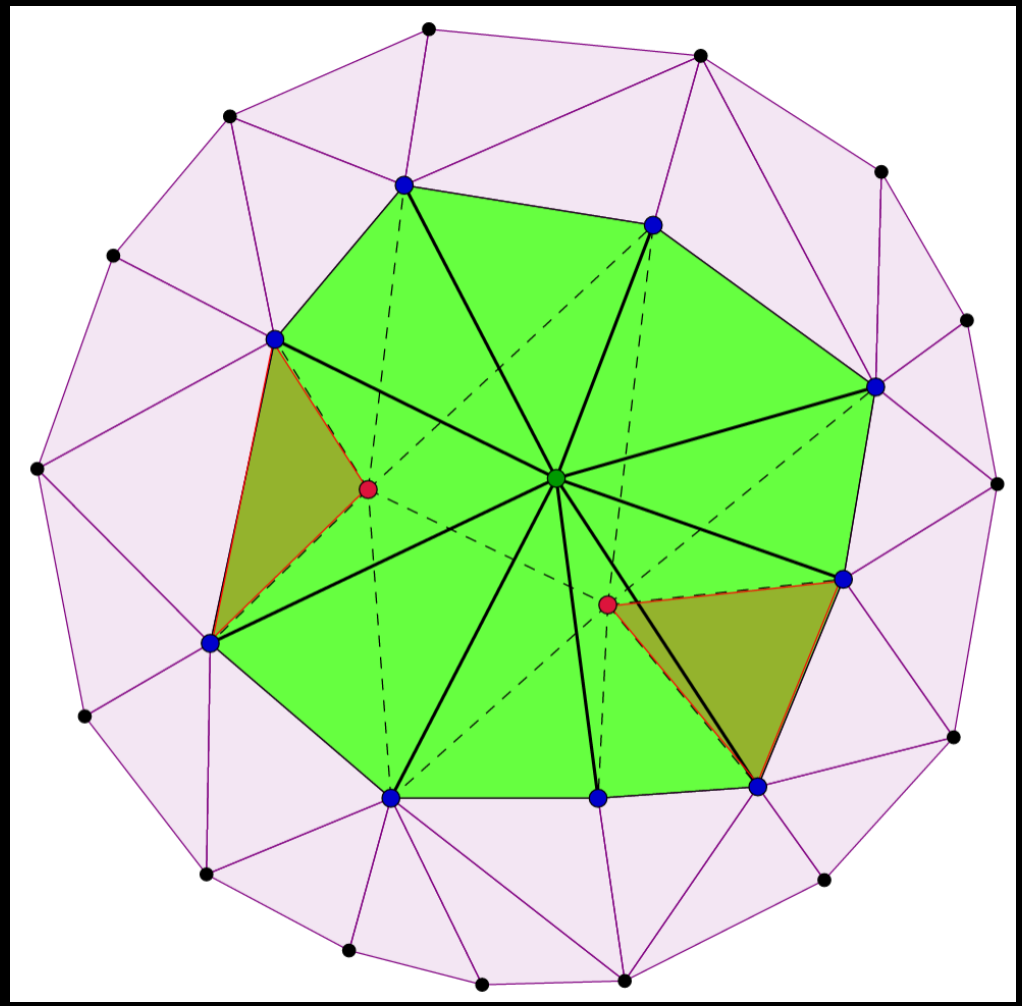


Red = bad element
Dotted edges = before
Solid edges = after

Solution

Remeshing operator

2) Ejection

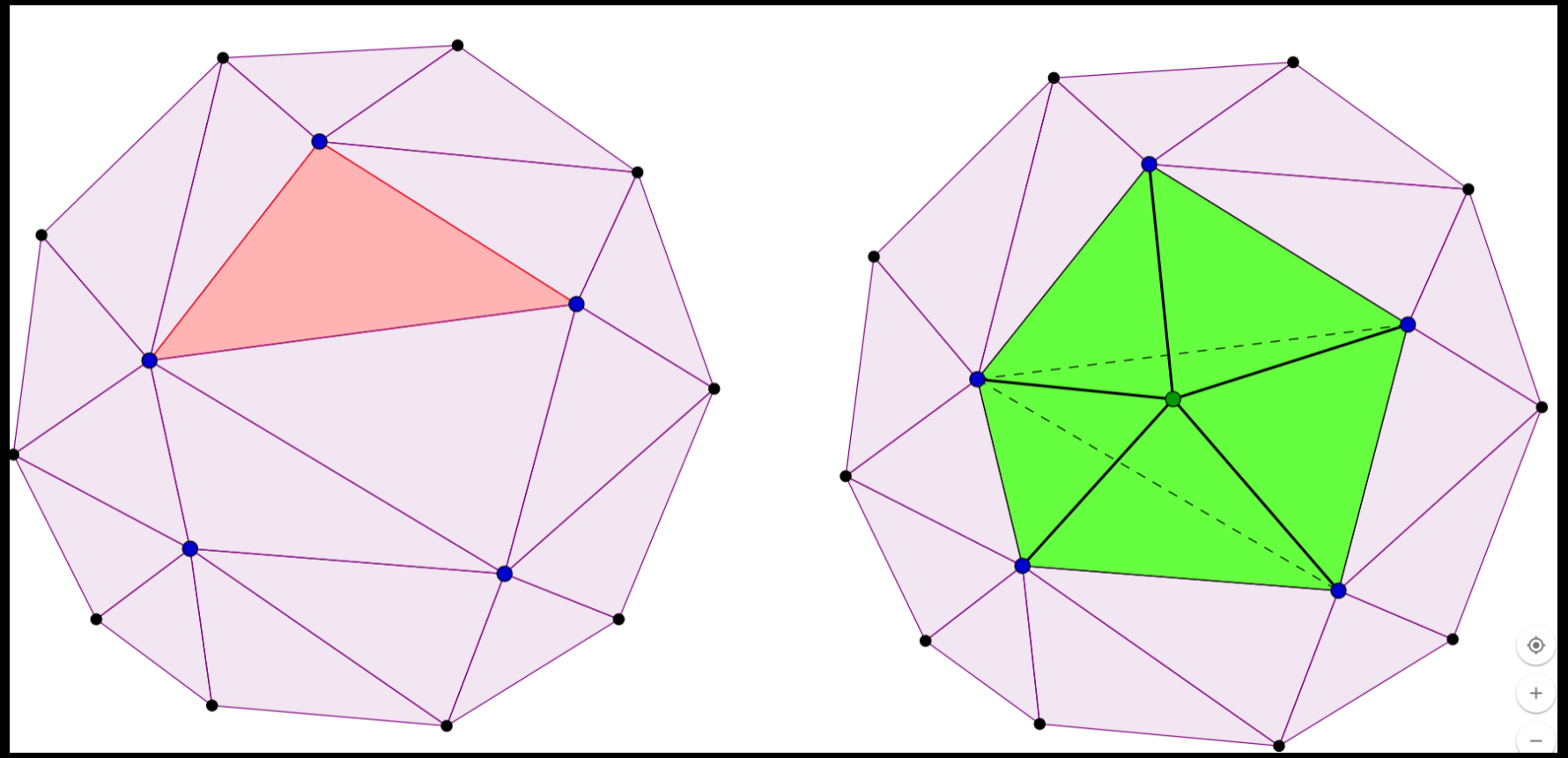


Red = bad element
Dotted edges = before
Solid edges = after

Solution

Remeshing operator

3) Injection



Red = bad element
Dotted edges = before
Solid edges = after

RL for Remeshing: idea

- Operators scheduling is done *statically* such that one operator is applied to all low-quality elements before trying another operator

- Dynamic scheduling via RL:
    - Environment -> input mesh
    - Agent -> picks operators
    - Reward -> 1 for non-obtuse everywhere, 0 otherwise

Thank You!