



TriWild: Robust Triangulation with Curved Constraints

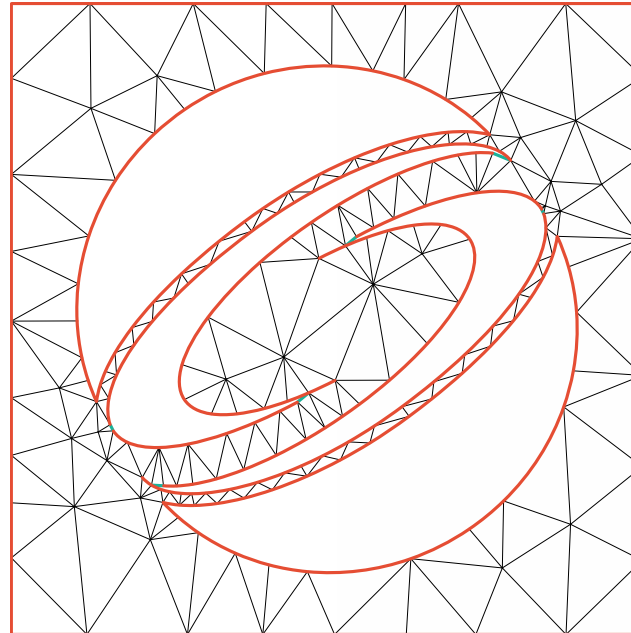
Yixin Hu,¹
Teseo Schneider,¹
Xifeng Gao,²
Qingnan Zhou,³
Alec Jacobson,⁴
Denis Zorin,¹
Daniele Panozzo.¹



2D Triangulation



Input 2D Boundary



Output Triangle Mesh

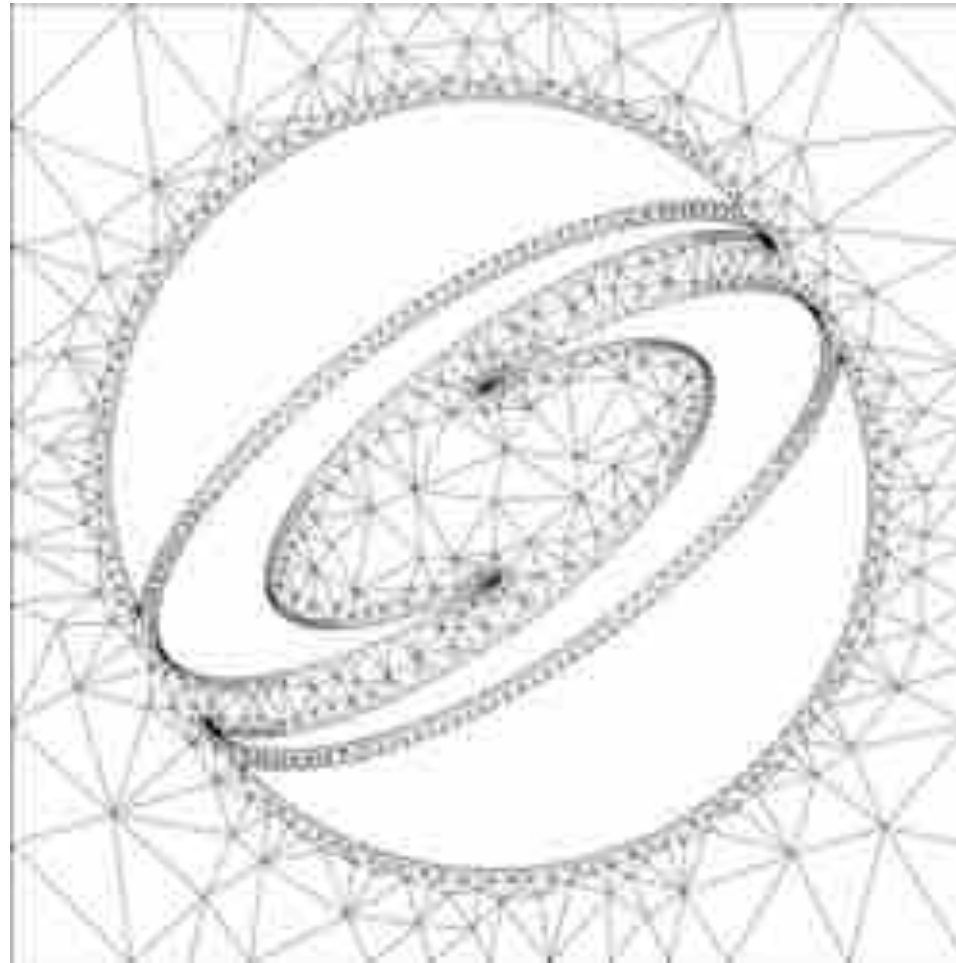


Physical Simulation

Linear Triangulation

#V = 3013, #F = 4149

*want a
smaller mesh?*



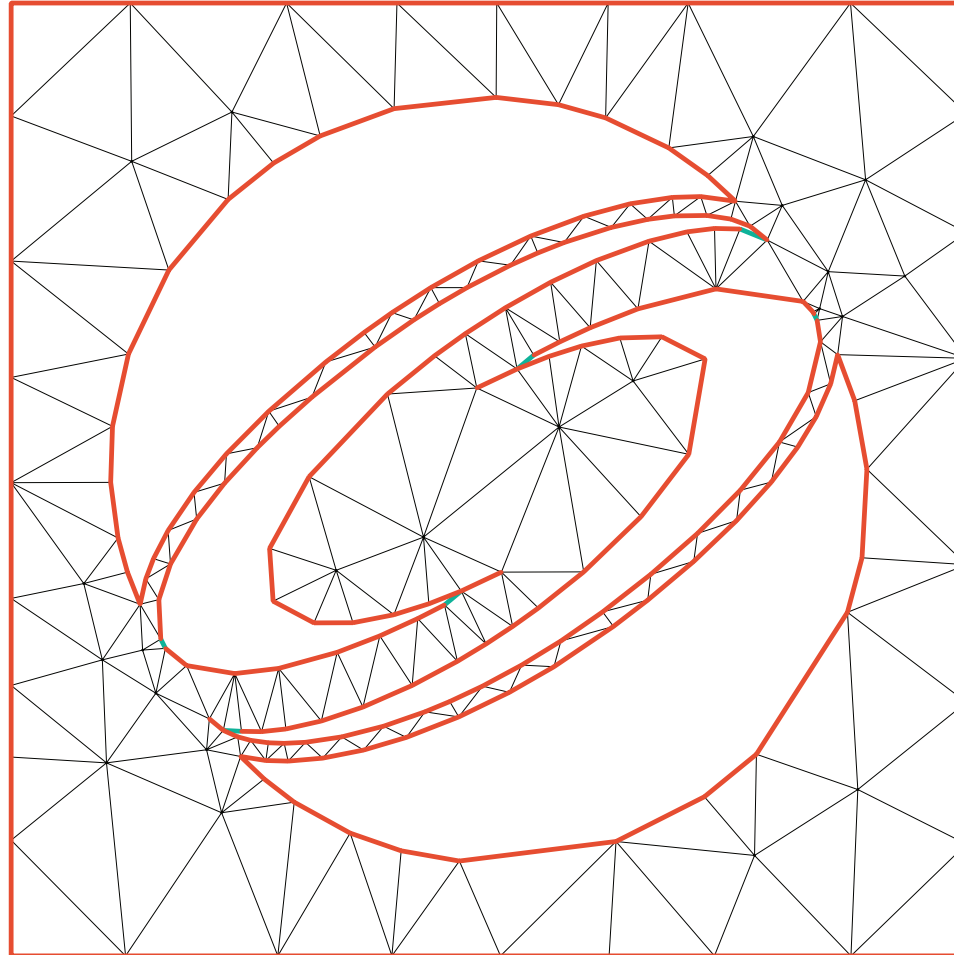
(Generated by Triangle)

Linear Triangulation

~~#V = 3013, #F = 4149~~

#V = 1503

#F = 304

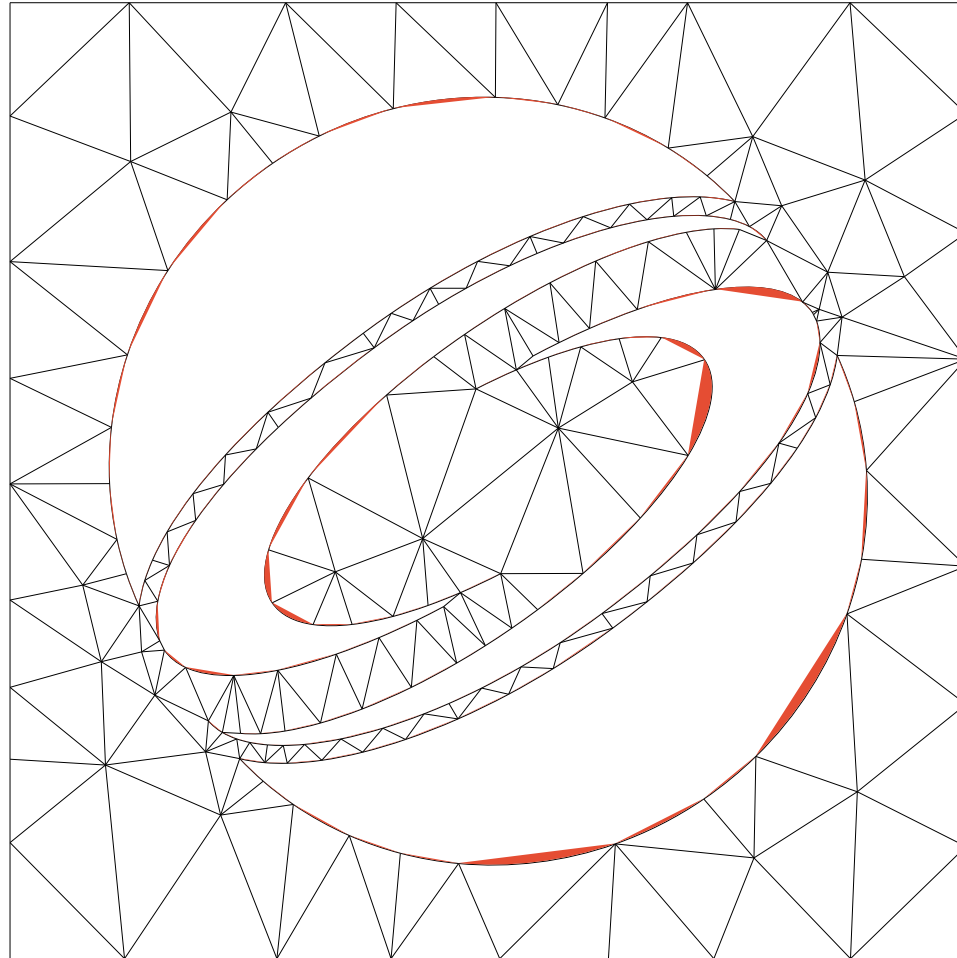


Linear Triangulation

~~#V = 3013, #F = 4149~~

#V = 1503

#F = 304

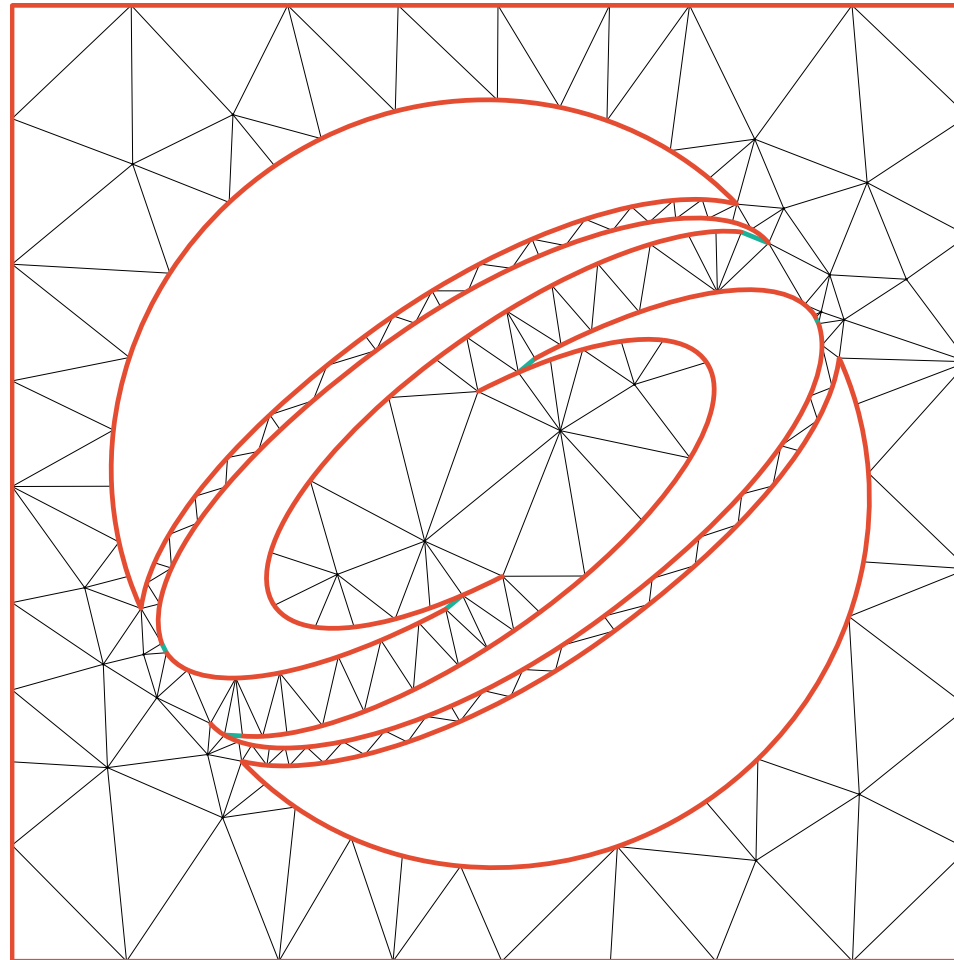


Curved Triangulation

~~#V = 3013, #F = 4149~~

#V = 1503

#F = 304



(Generated by TriWild)

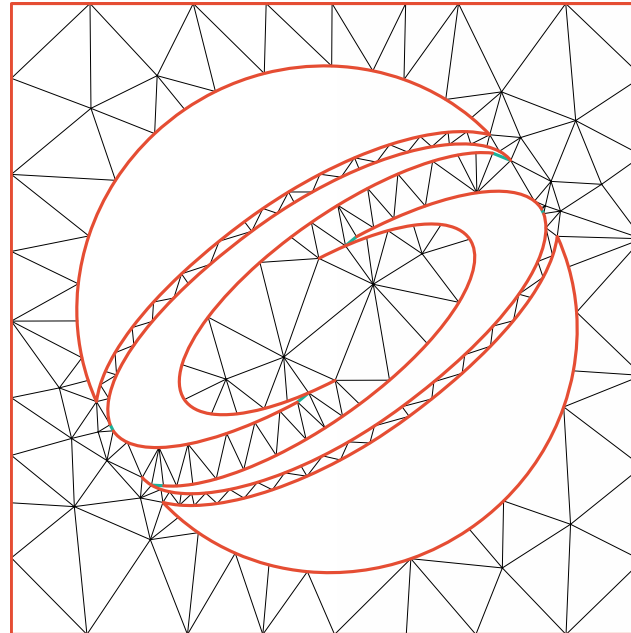


Curved mesh!

2D Triangulation



Input 2D Boundary



Output Triangle Mesh



Physical Simulation

Coarser



Faster

Conforming



Accurate

Existing Methods

- **Curved triangle mesh is not a new concept.**
 - Introduced in late 80s.
 - Haven't been tested on large-scale dataset.
 - Not widely used because no robust tools.

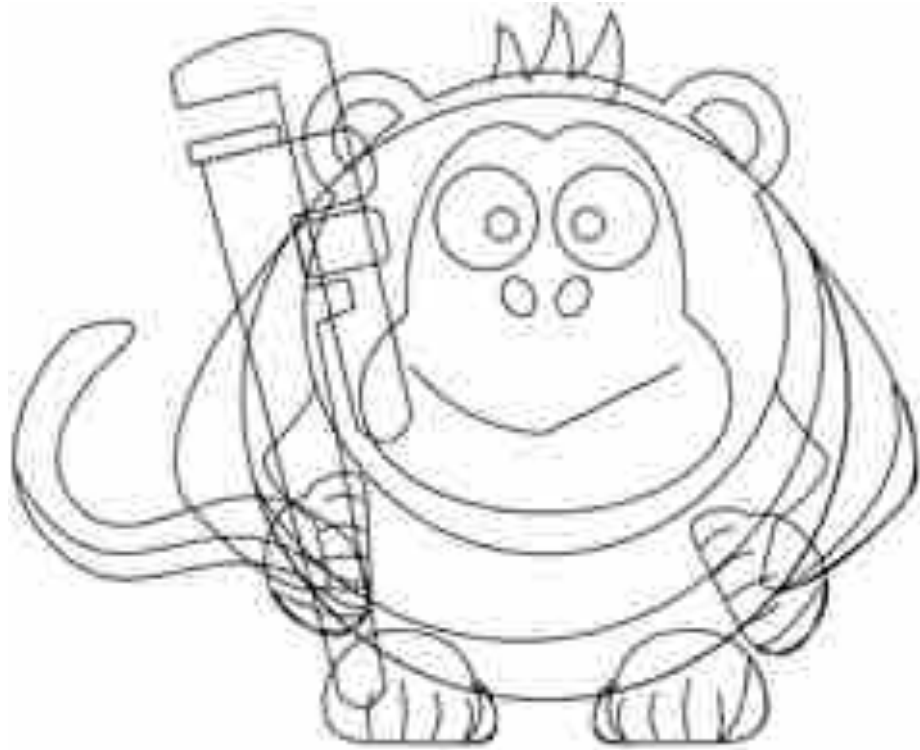


Tangled Curved Constraints



Superposed Curves

Tangled Curved Constraints

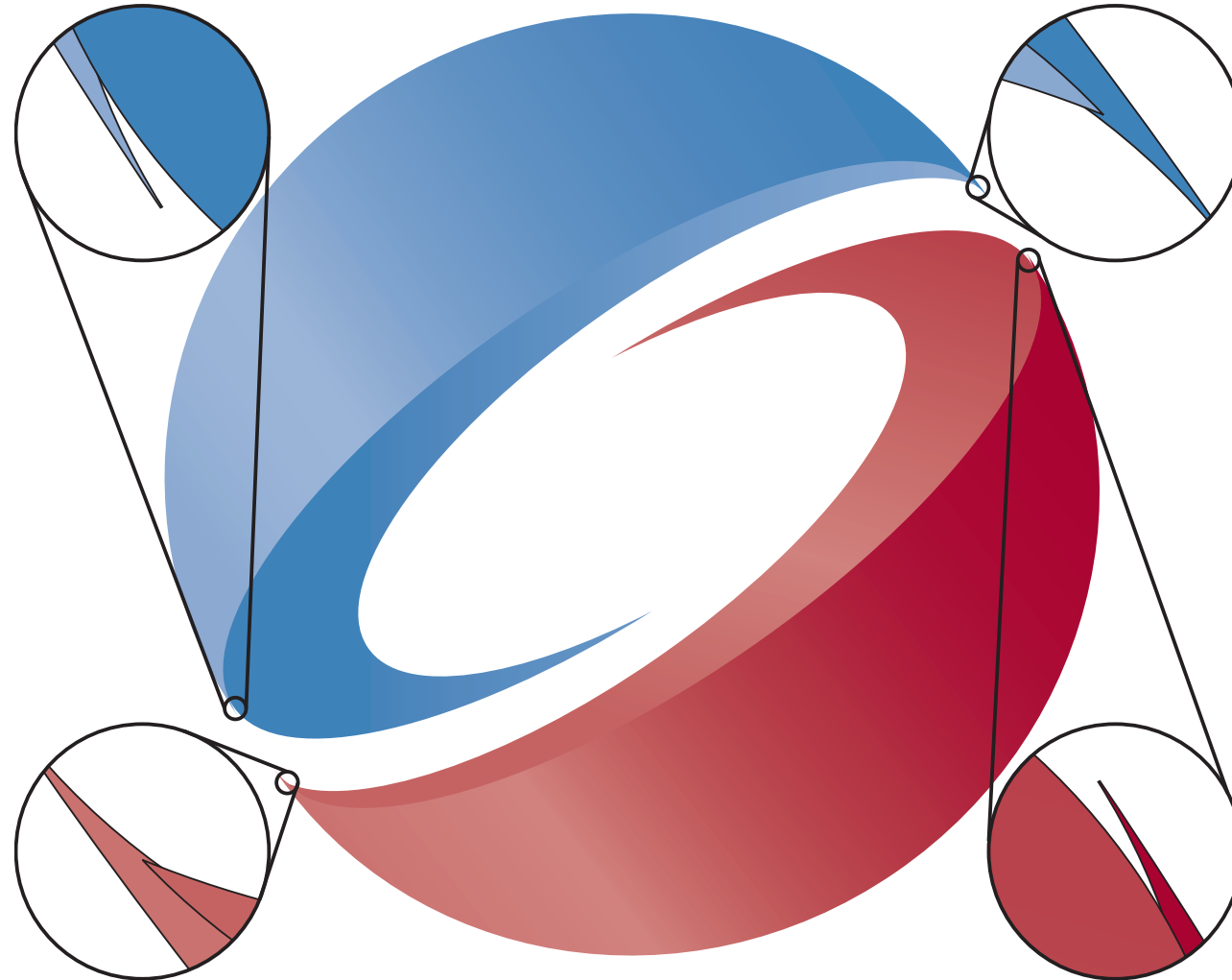


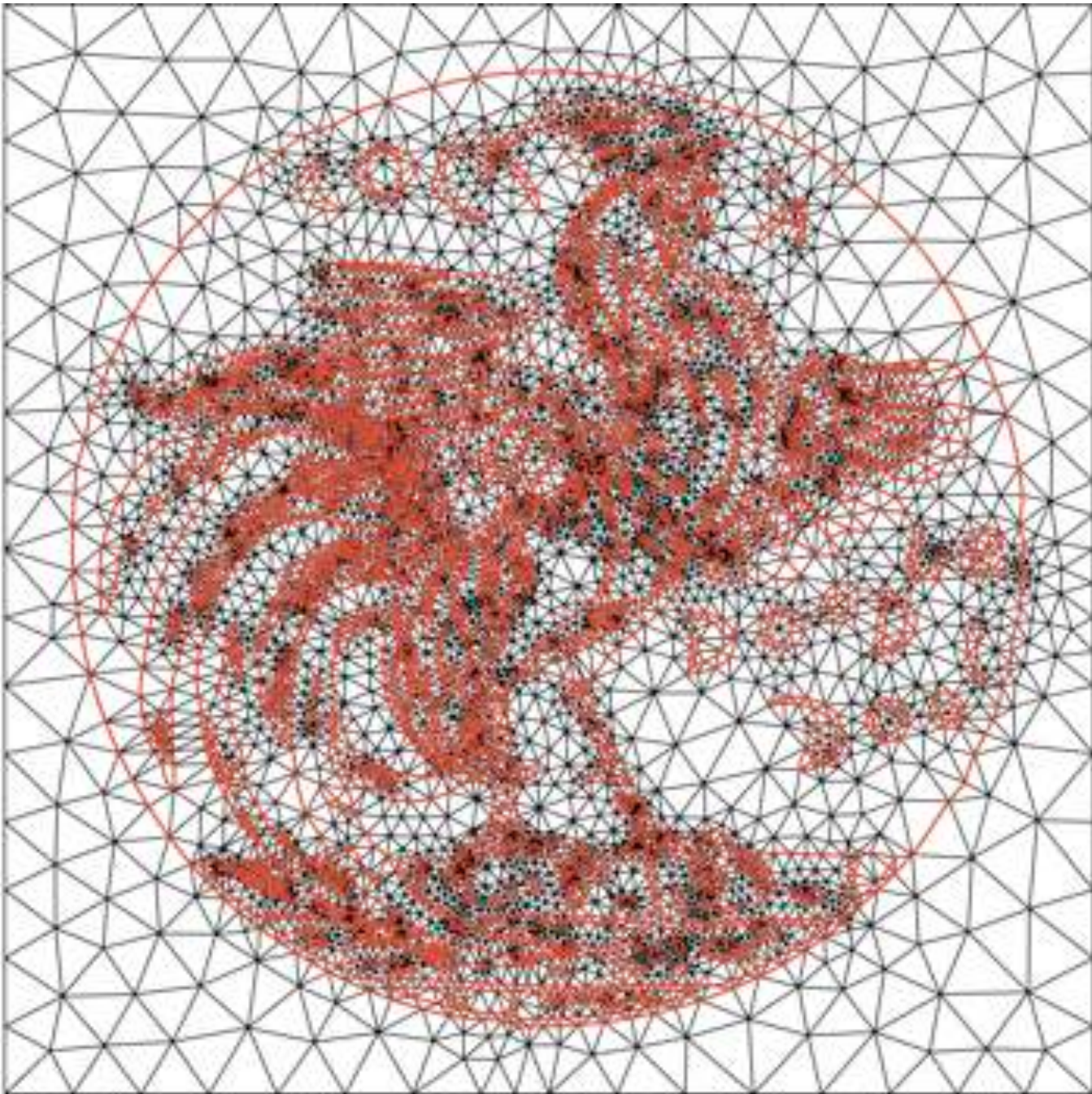
Intersected Curves

Tangled Curved Constraints



Tangled Curved Constraints



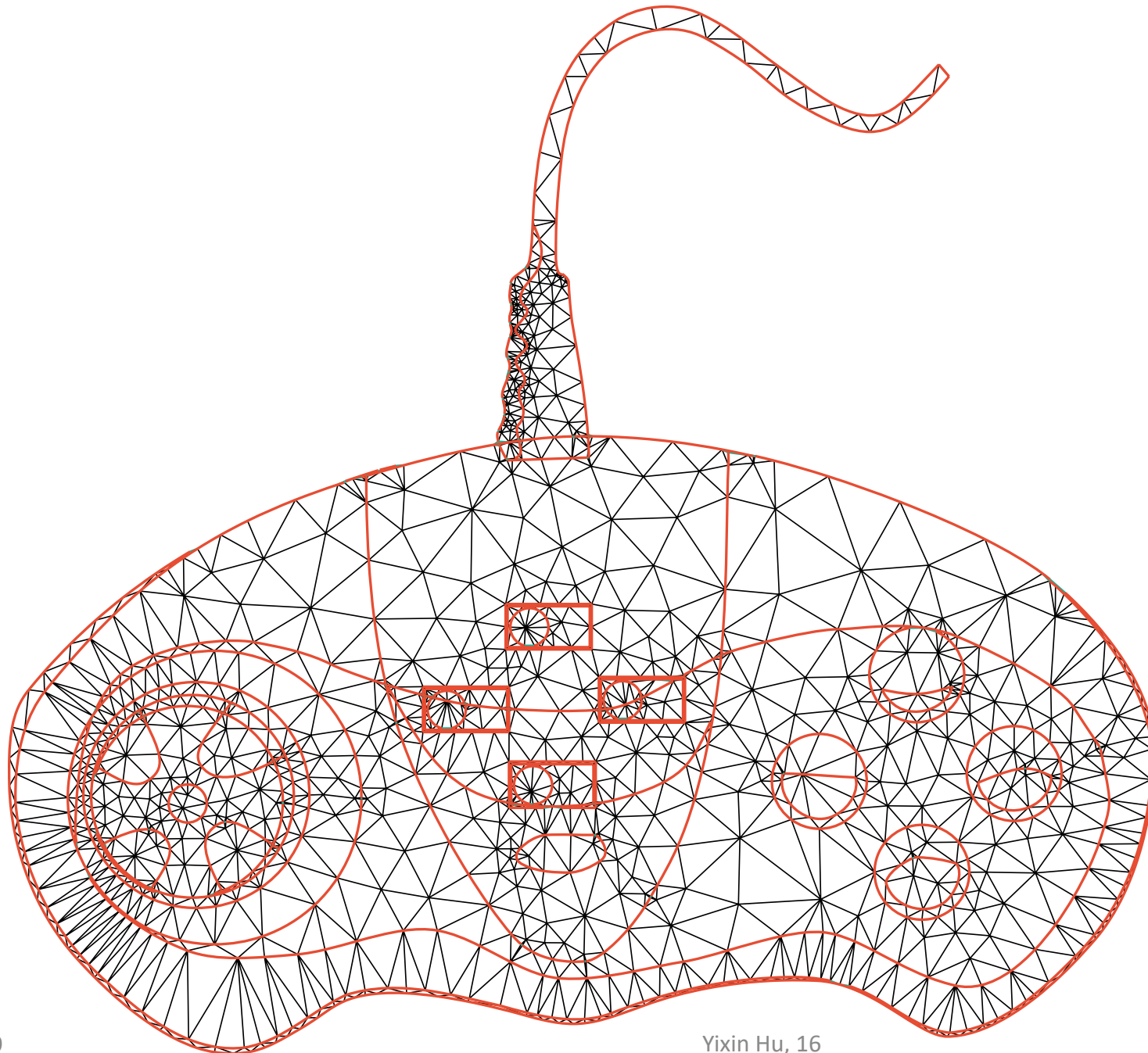


Input:



(Generated by TriWild)



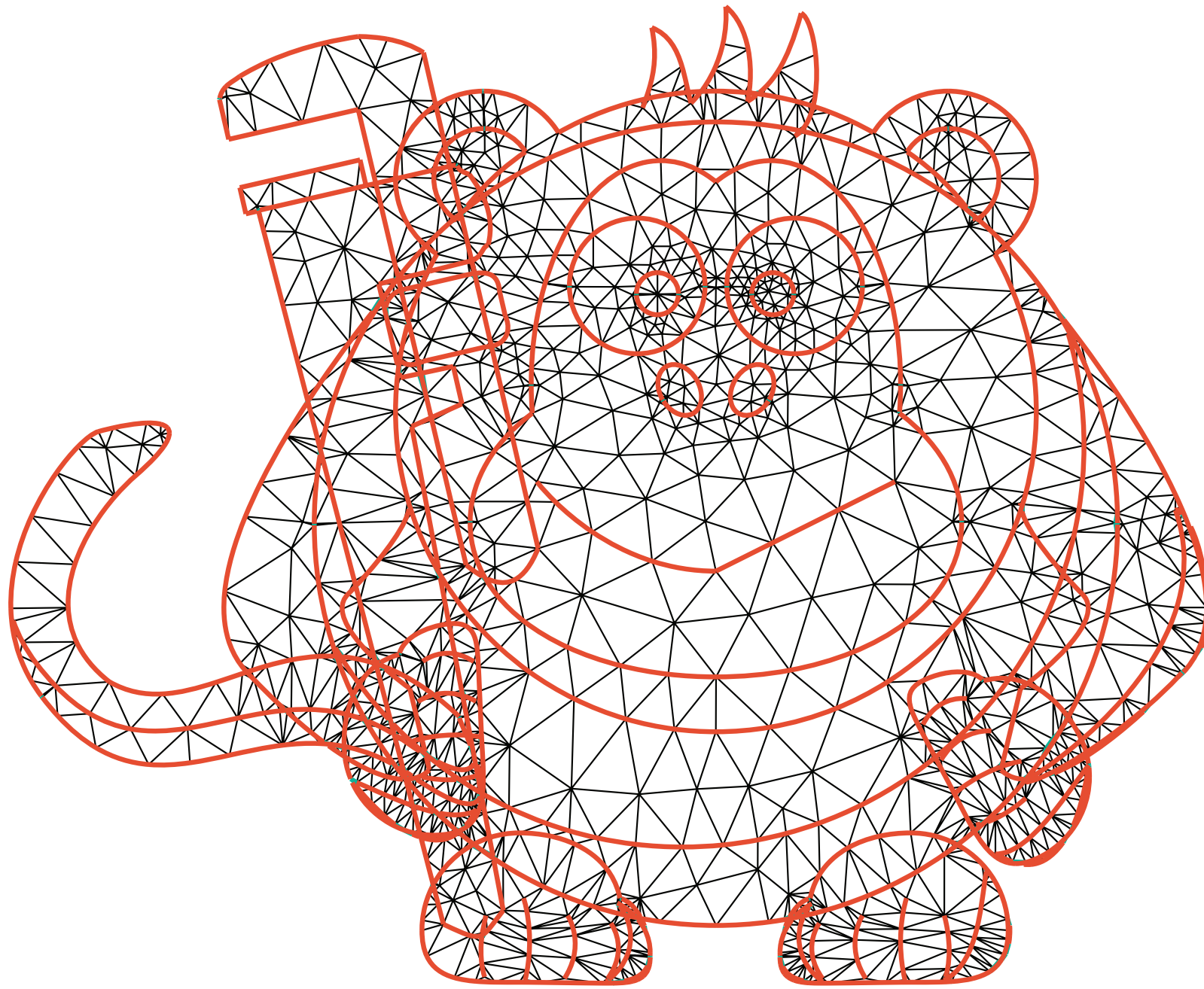


Input:

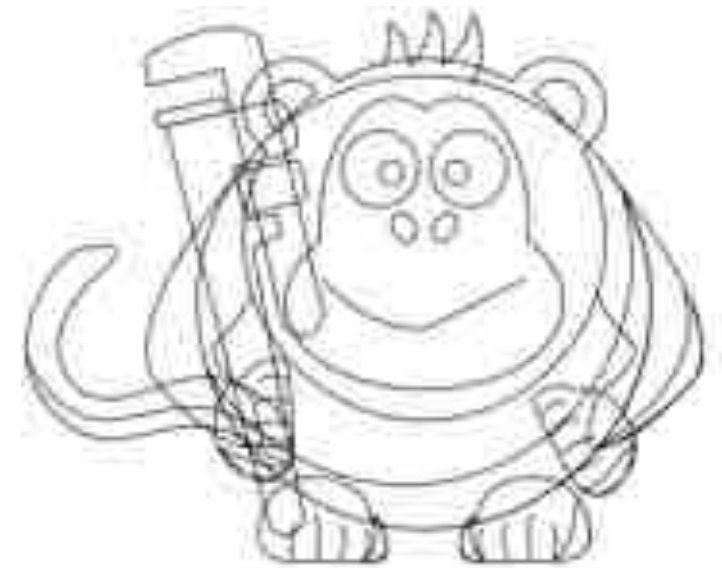


(Generated by TriWild)



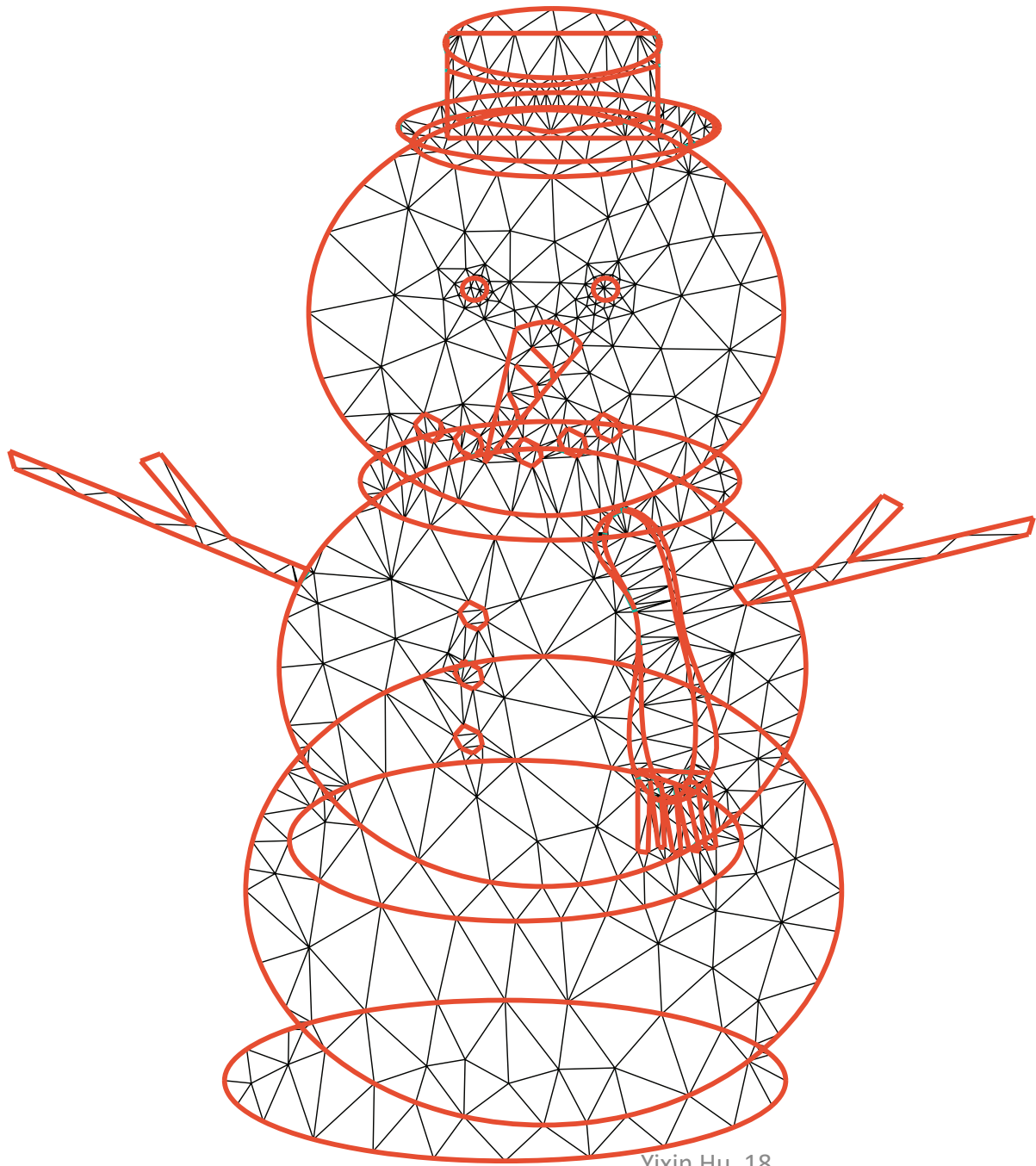


Input:



(Generated by TriWild)





Input:

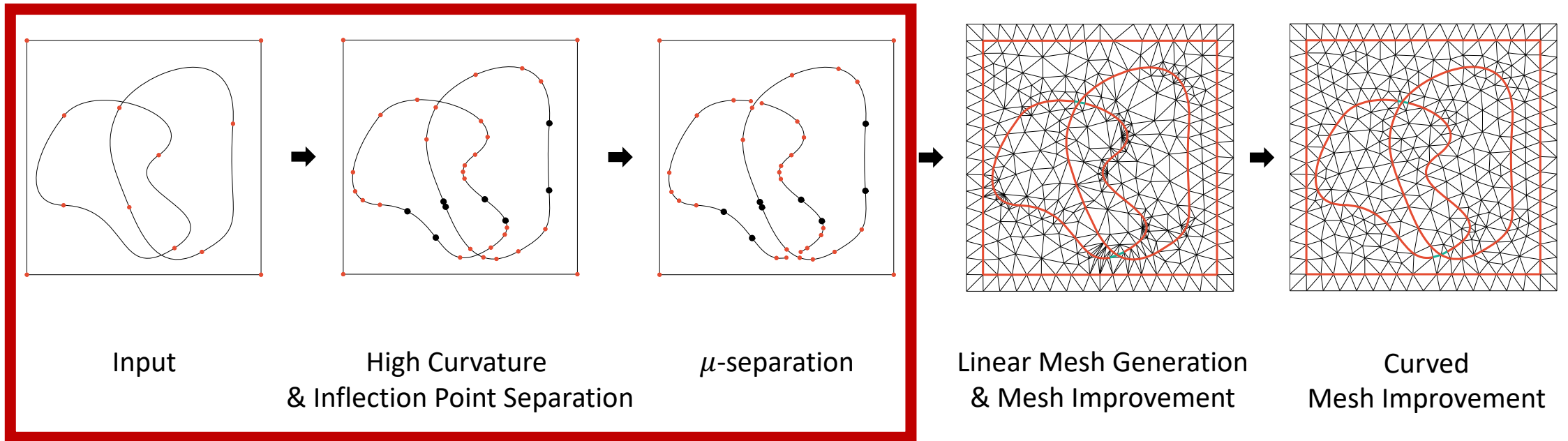


(Generated by TriWild)

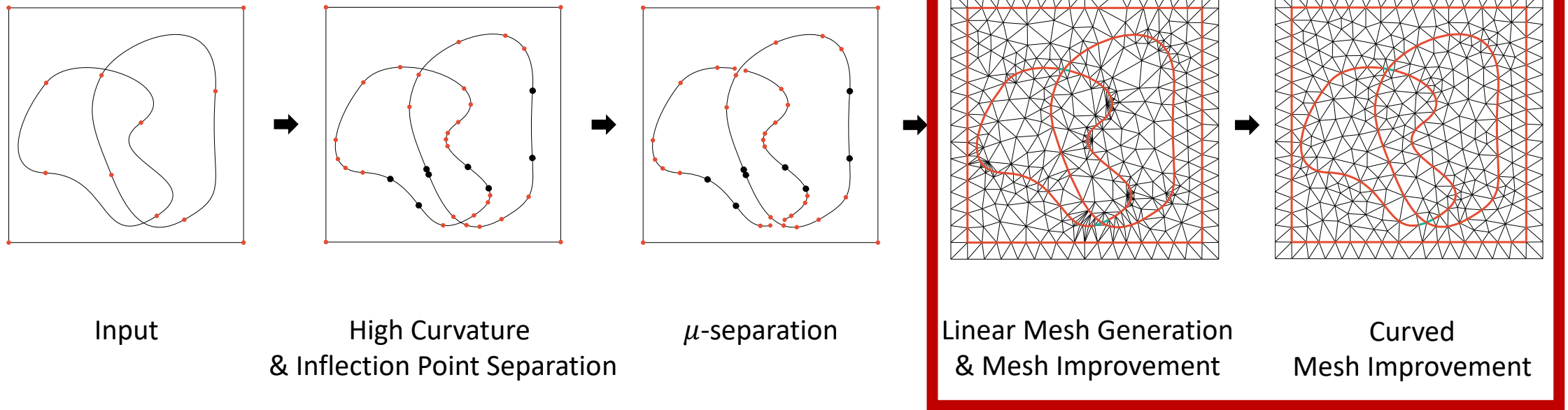


Why TriWild?

"Cleanup" the input curves.



Why TriWild?



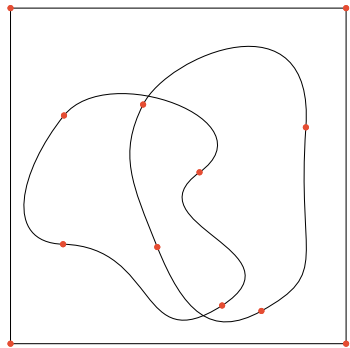
Pipeline - Input & Output

- Input:
 - Bezier curves. (SVG standard, degree 3)
- Output:
 - Curved triangle mesh
 - Conforms to **primary features**.
 - Approximates **secondary features** up to ϵ .

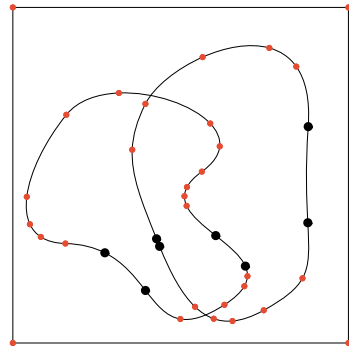
Primary features
Secondary features



Pipeline - Feature Preprocessing

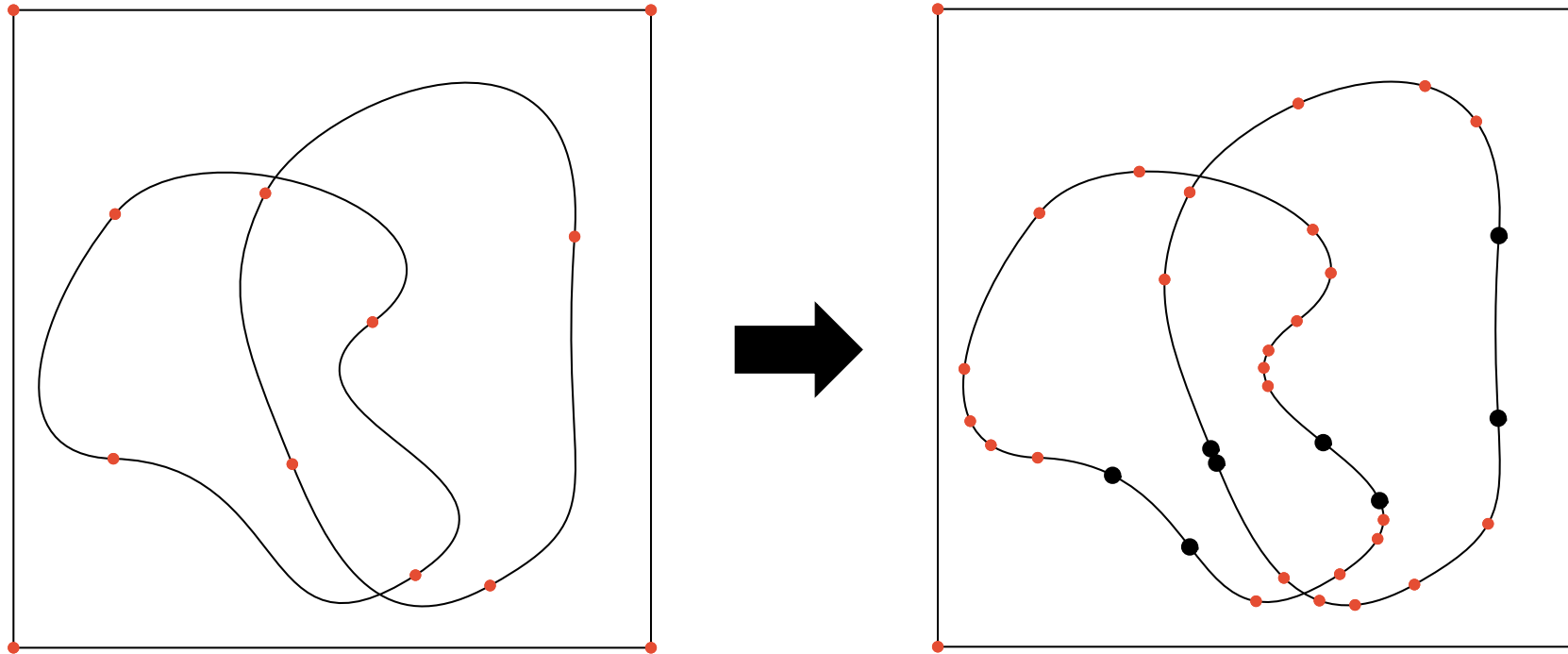


Input



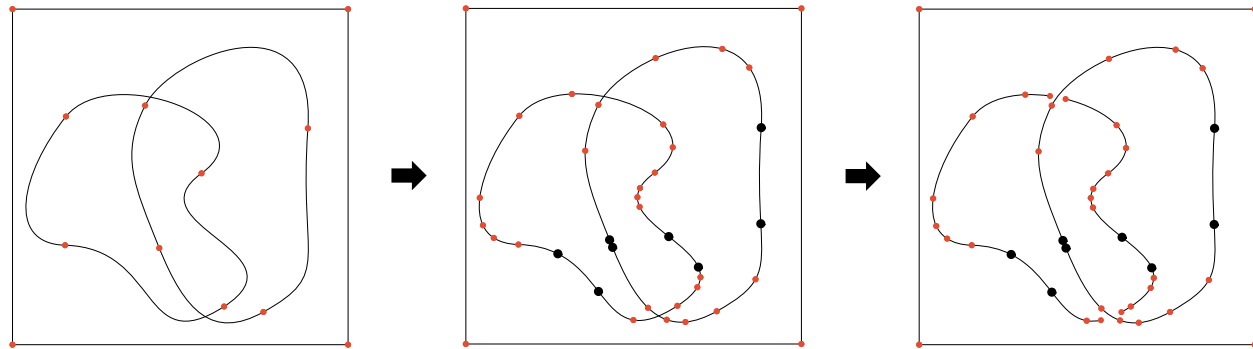
High Curvature
& Inflection Point Separation

Pipeline - Feature Preprocessing



High Curvature & Inflection Point Separation

Pipeline - Feature Preprocessing

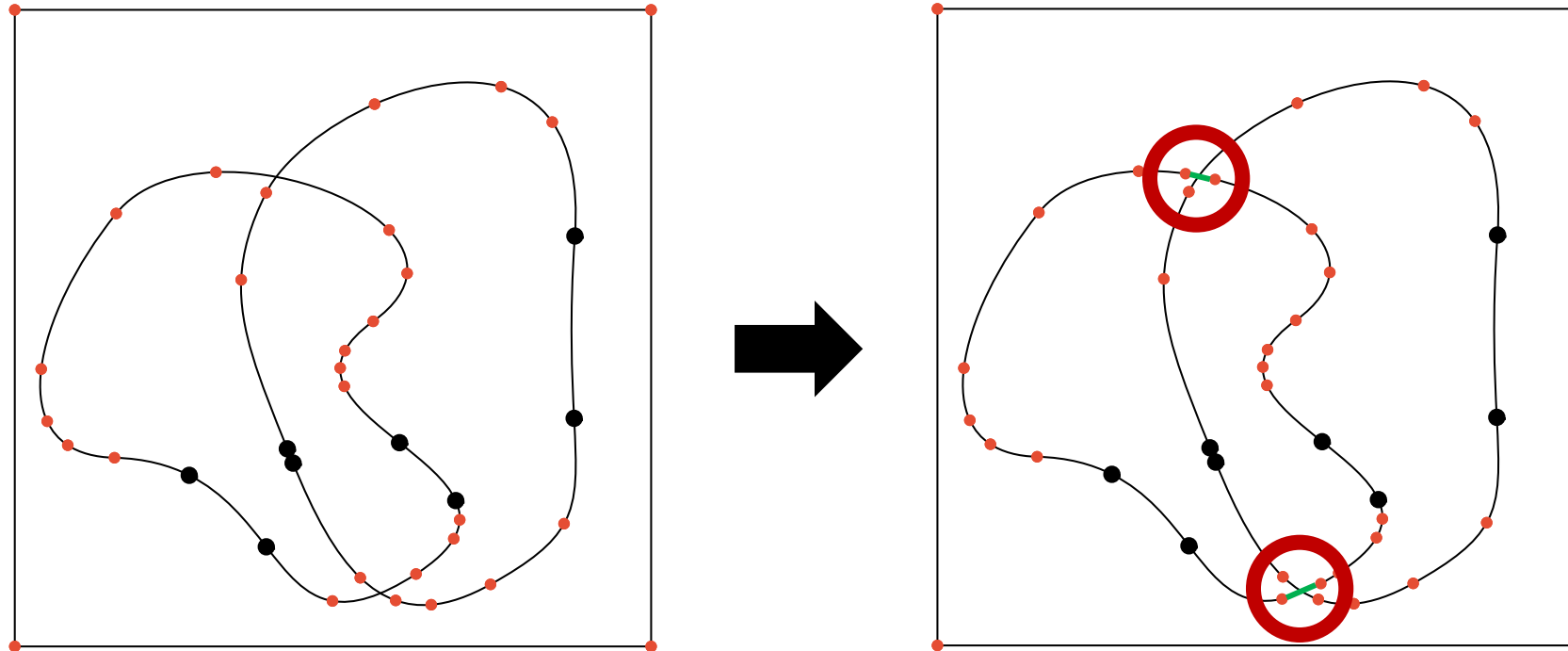


Input

High Curvature
& Inflection Point Separation

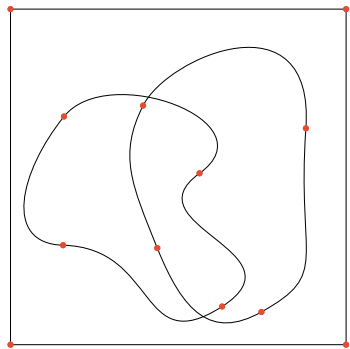
μ -separation

Pipeline - Feature Preprocessing



μ -separation

Pipeline



Input



& I

Tetrahedral Meshing in the Wild

YIXIN HU, New York University
QINGNAN ZHOU, Adobe Research
XIFENG GAO, New York University
ALEC JACOBSON, University of Toronto
DENIS ZORIN, New York University
DANIELE PANOZZO, New York University



Fig. 1. A selection of the real-world meshes in the wild tetrahedralized by our novel tetrahedral meshing technique.

We propose a novel tetrahedral meshing technique that is unconditionally robust, requires no user interaction, and can directly convert a triangle soup into an analysis-ready volumetric mesh. The approach is based on several core principles: (1) initial mesh construction based on a fully robust, yet efficient, signed exact comparison; (2) robust (anisotropic or user-defined) coarsening of the mesh relative to the surface input; (3) iterative mesh improvement with guarantees, at every step, of the output quality. The quality of the resulting mesh is a direct function of the target mesh size and allowed tolerance: increasing allowed deviation from the initial mesh and decreasing the target edge length both lead to higher mesh quality.

Our approach enables “black-box” analysis, i.e. it allows to automatically solve partial differential equations on geometrical models available in the wild, offering a robustness and reliability comparable to, e.g., image processing algorithms operating the data in automatic, large-scale processing of real-world geometry data.

ACM Reference Format:

Xixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 46 (August 2018), 14 pages. <https://doi.org/10.1145/3197117.3201701>

1 INTRODUCTION

Triangulating the interior of a shape is a fundamental subproblem in 2D and 3D geometric computation.

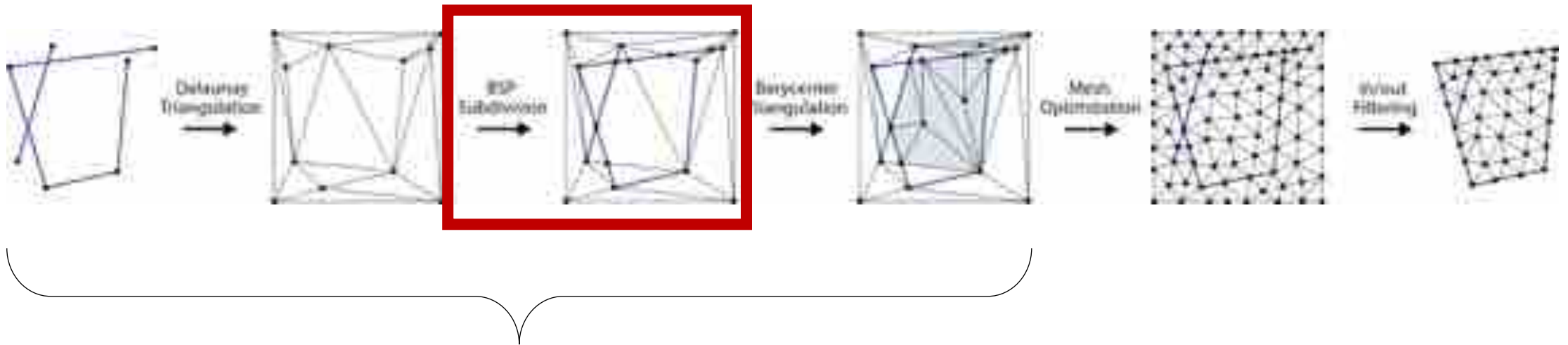
For two-dimensional problems requiring meshing a domain, robust and efficient software for constrained Delaunay triangulation problem has been a tremendous boon to the development of robust and efficient automatic computational pipelines, in particular ones requiring solving PDEs. Robust 2D triangulations inside a given polygonal boundary are also an essential spatial partitioning useful for many applications, such as numerical and feature extraction.



Recap - Linear Mesh Generation (TetWild)

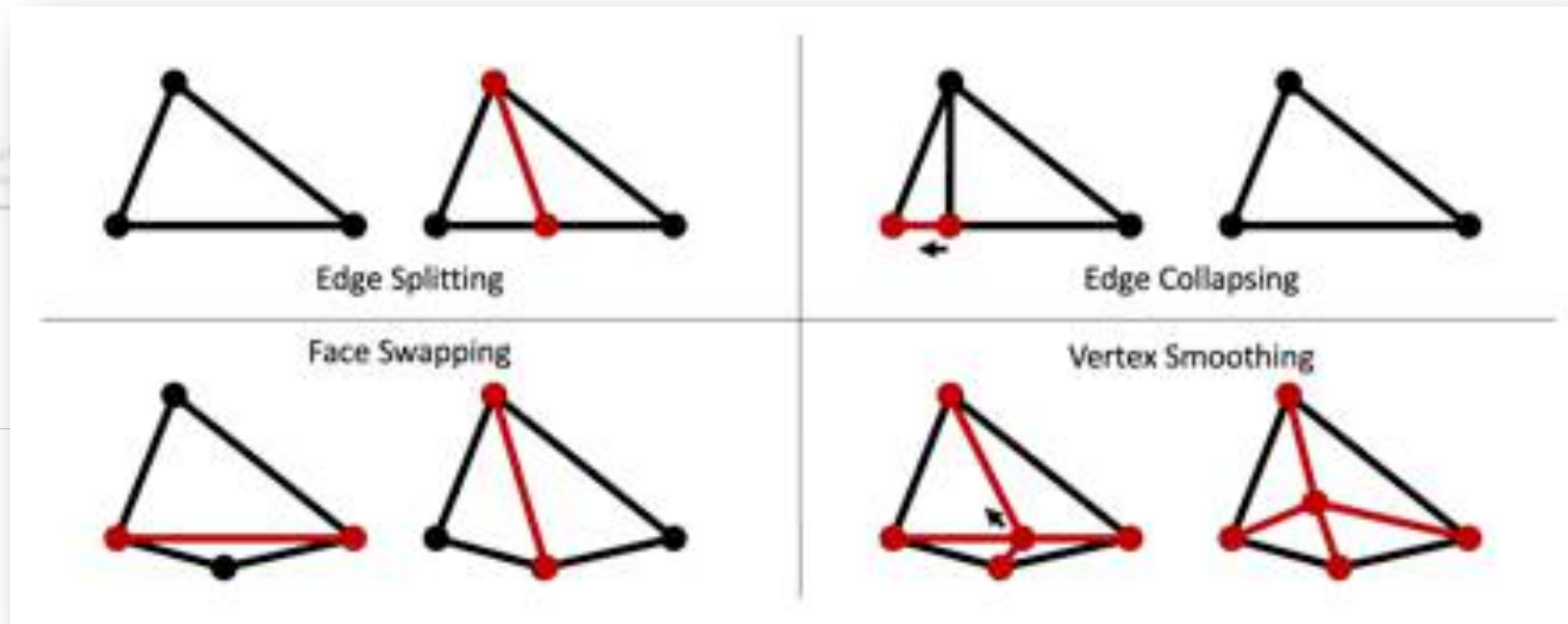
Segment Soup

Linear Triangle Mesh



Stage I: Valid Mesh Generation
(Rational)

Recap - Linear Mesh Generation (TetWild)

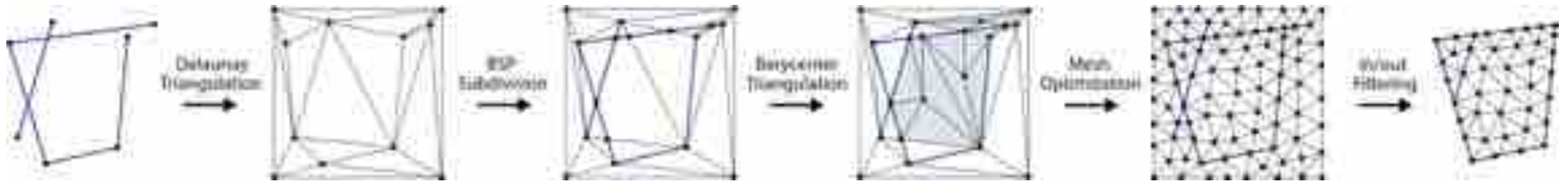


(Rational)

(Mixed \rightarrow Double)

Recap - Linear Mesh Generation (TetWild)

100% success rate on 10,000 real-world models.



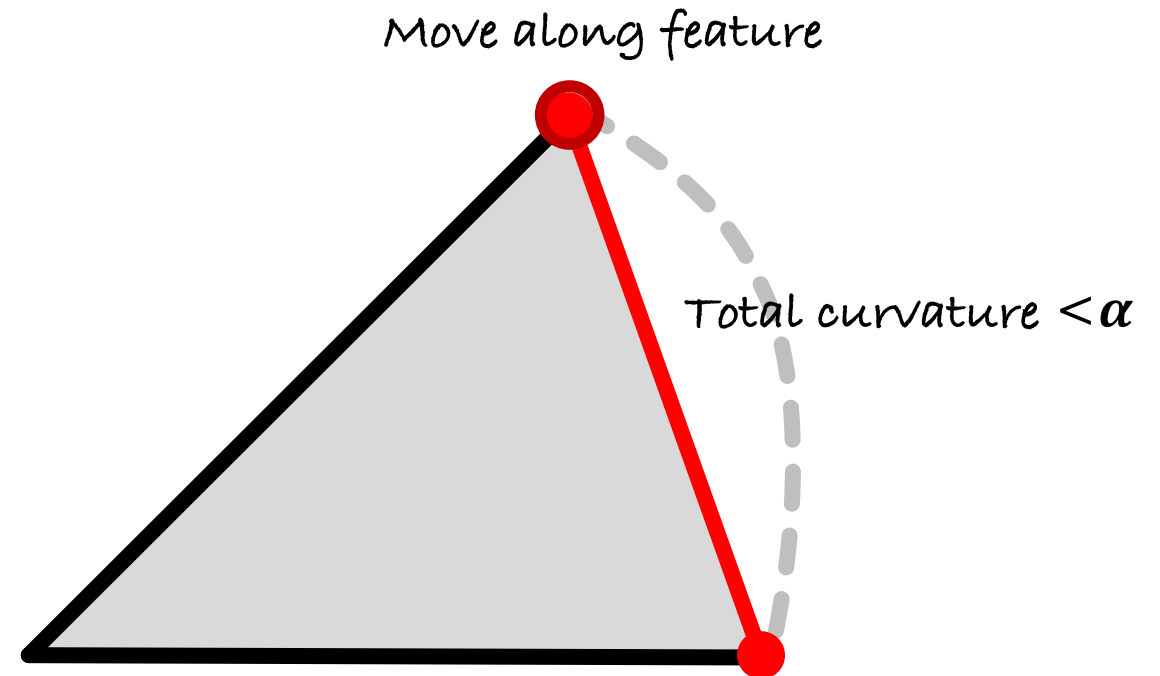
Stage I: Valid Mesh Generation
(Rational)

Stage II: Mesh Improvement
(Mixed \rightarrow Double)

Pipeline - Linear Mesh Generation

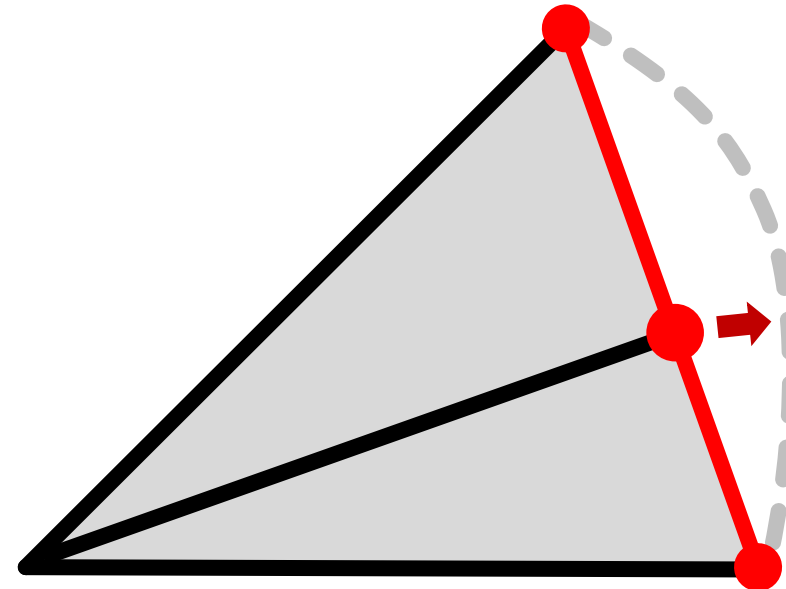
- Feature Invariant:
 - Feature edges' associated curve has total curvature $< \alpha$.
 - Feature vertices can only move along feature curves.

“feature” == “primary feature”



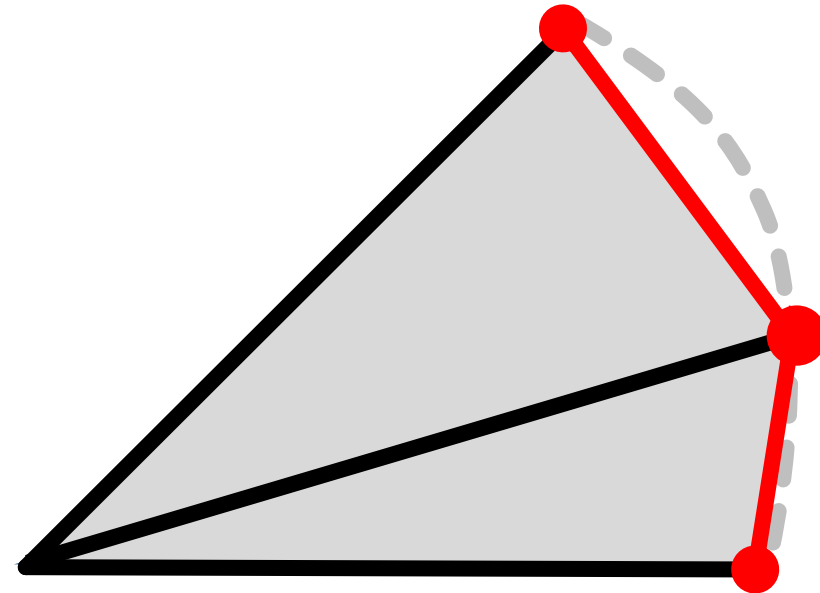
Pipeline - Linear Mesh Generation

- Feature Invariant:
 - Feature edges' associated curve has total curvature $< \alpha$.
 - Feature vertices can only move along feature curves.
- Vertex Projection

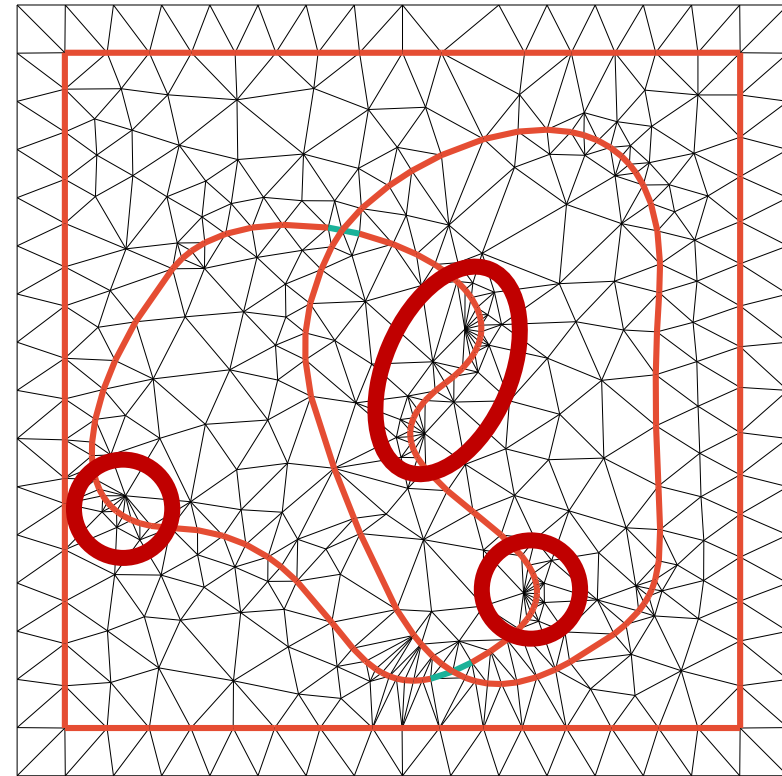
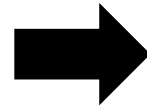
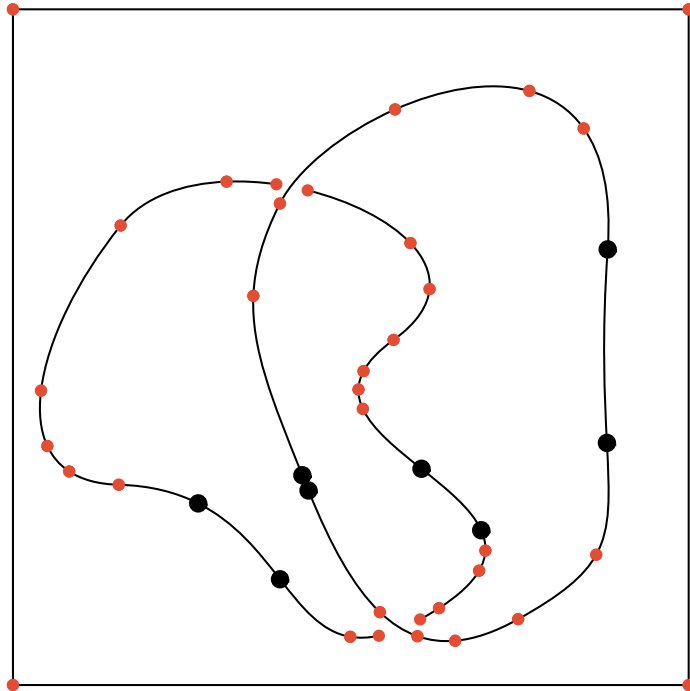


Pipeline - Linear Mesh Generation

- Feature Invariant:
 - Feature edges' associated curve has total curvature $< \alpha$.
 - Feature vertices can only move along feature curves.
- Vertex Projection

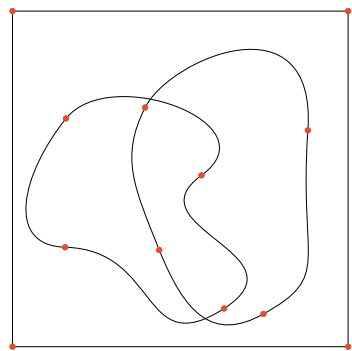


Pipeline - Curved Mesh Generation

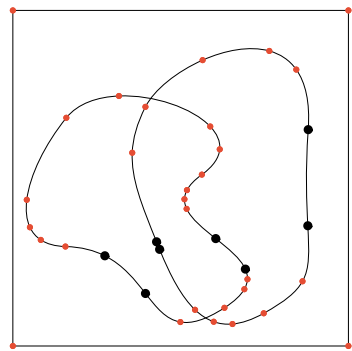


Linear Mesh Generation & Mesh Improvement

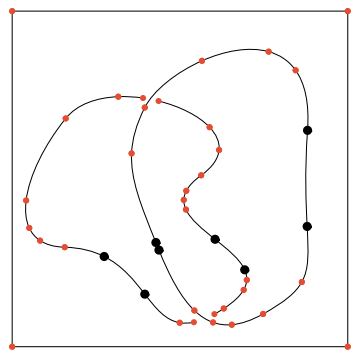
Pipeline - Curved Mesh Generation



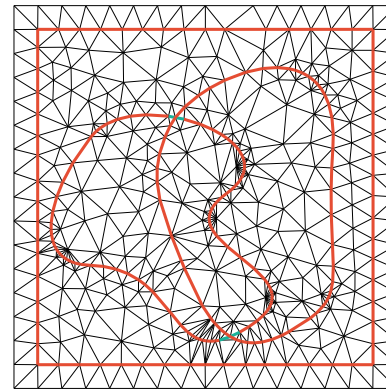
Input



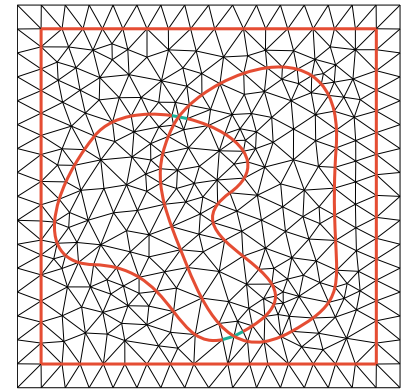
High Curvature
& Inflection Point Separation



μ -separation



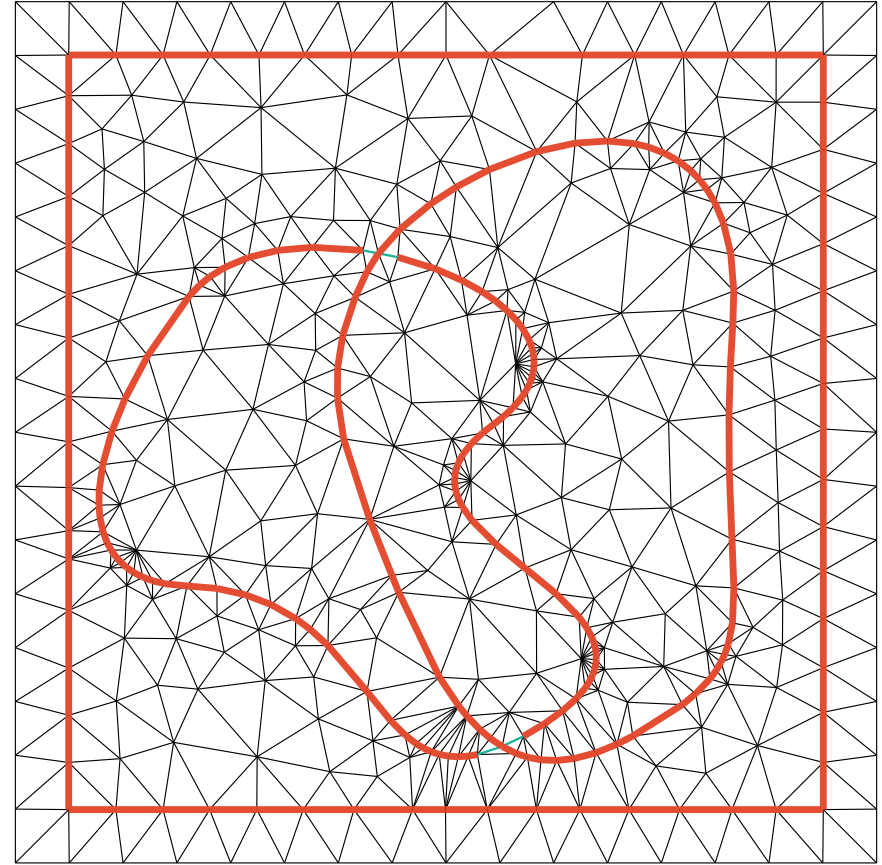
Linear Mesh Generation
& Mesh Improvement



Curved
Mesh Improvement

Pipeline - Curving

- Curving:
 - All feature elements
 - With primary feature edges (thickened in red)



Pipeline

- Inversion

The Generation of Valid Curvilinear Meshes

Christophe Geuzaine¹, Amaury Johnson¹,
Jonathan Lambrechts^{2,3}, Jean-François Remacle², Thomas Toulorge^{2,3}

¹ Université de Liège, Dept. of Electrical Engineering and Computer Sciences, B-4000
Liège, Belgium

{a.johnson, c.geuzaine}@ulg.ac.be,

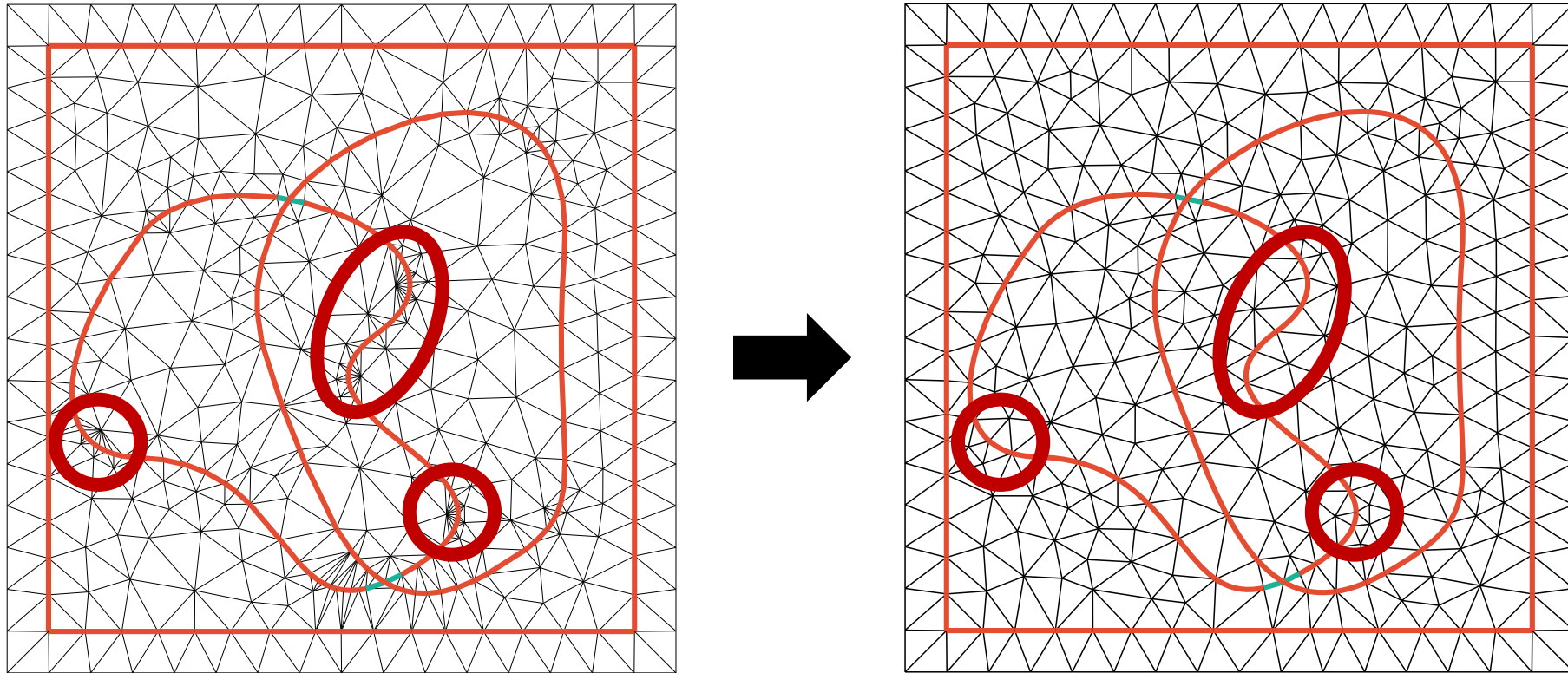
² Université catholique de Louvain, Institute of Mechanics, Materials and Civil
Engineering (MMC), B-1348 Louvain-la-Neuve, Belgium
{thomas.toulorge, jonathan.lambrechts, jean-francois.remacle}@uclouvain.be,
³ Fonds National de la Recherche Scientifique, B-1000 Brussels, Belgium

Abstract. It is now well-known that a curvilinear discretization of the geometry is most often required to benefit from the computational efficiency of high-order numerical schemes in simulations. In this article, we explain how appropriate curvilinear meshes can be generated. We pay particular attention to the problem of invalid (tangled) mesh parts created by curving the domain boundaries. An efficient technique that computes provable bounds on the element Jacobian determinant is used to characterize the mesh validity, and we describe fast and robust techniques to regularize the mesh. The methods presented in this article are thoroughly discussed in Ref. [1,2], and implemented in the free mesh generation software Gmsh [3,4].

Keywords: High-order mesh, curvilinear mesh, geometry discretization, mesh validity, element Jacobian

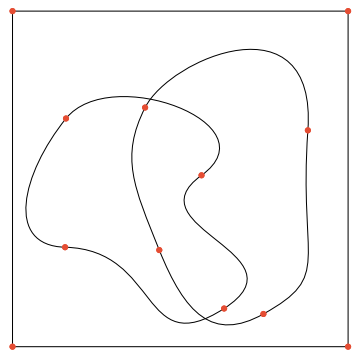


Pipeline - Curved Mesh Generation

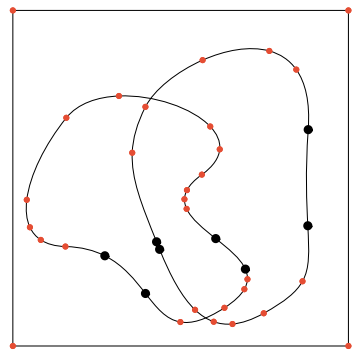


Curved Mesh Improvement

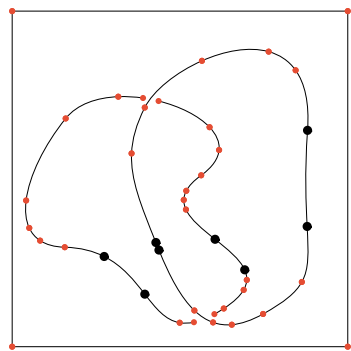
Pipeline - Curved Mesh Generation



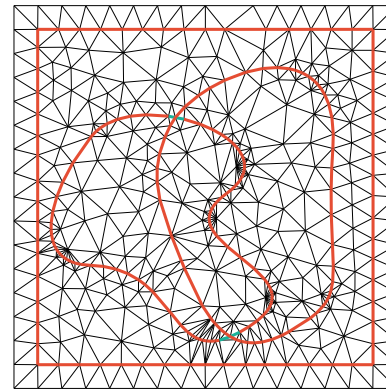
Input



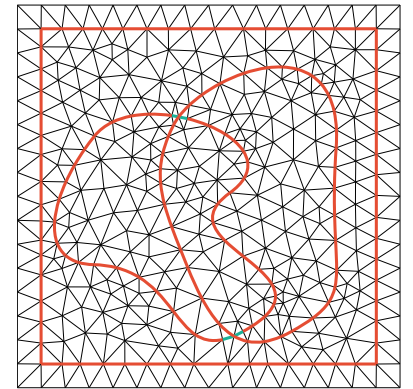
High Curvature
& Inflection Point Separation



μ -separation



Linear Mesh Generation
& Mesh Improvement



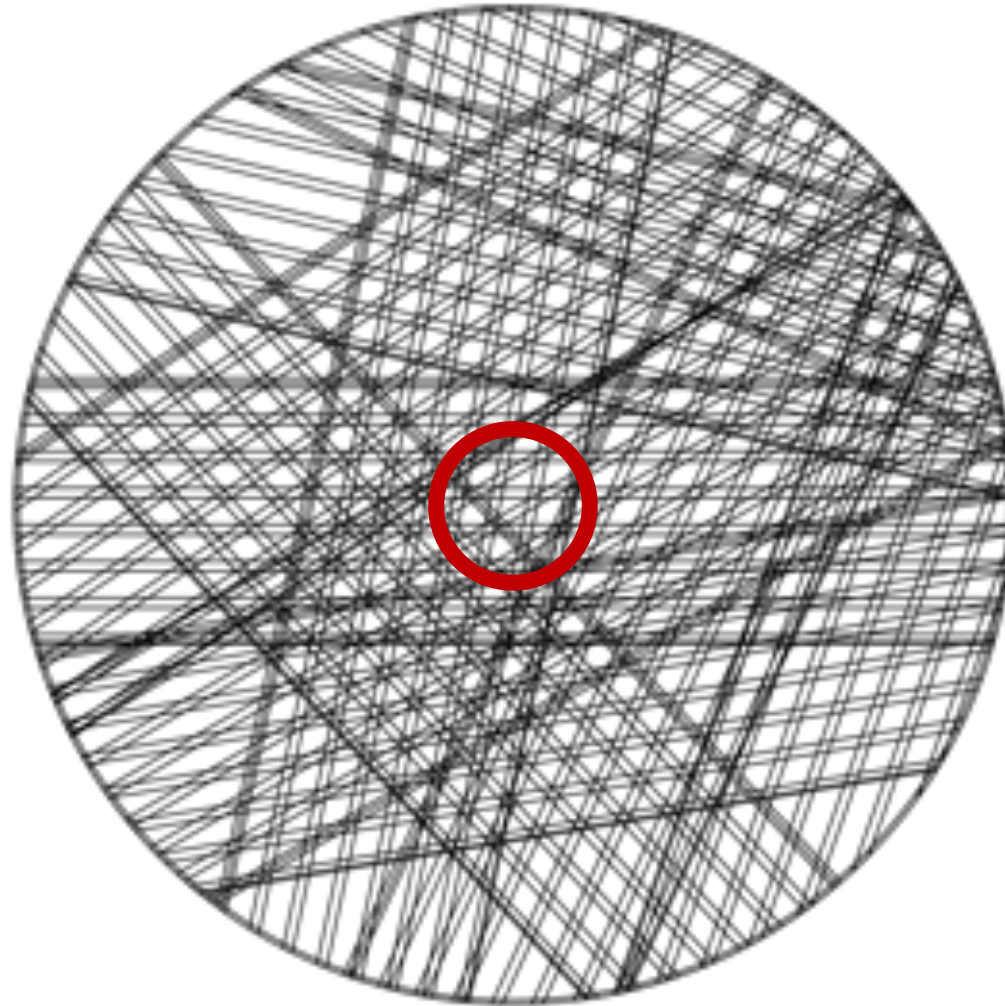
Curved
Mesh Improvement

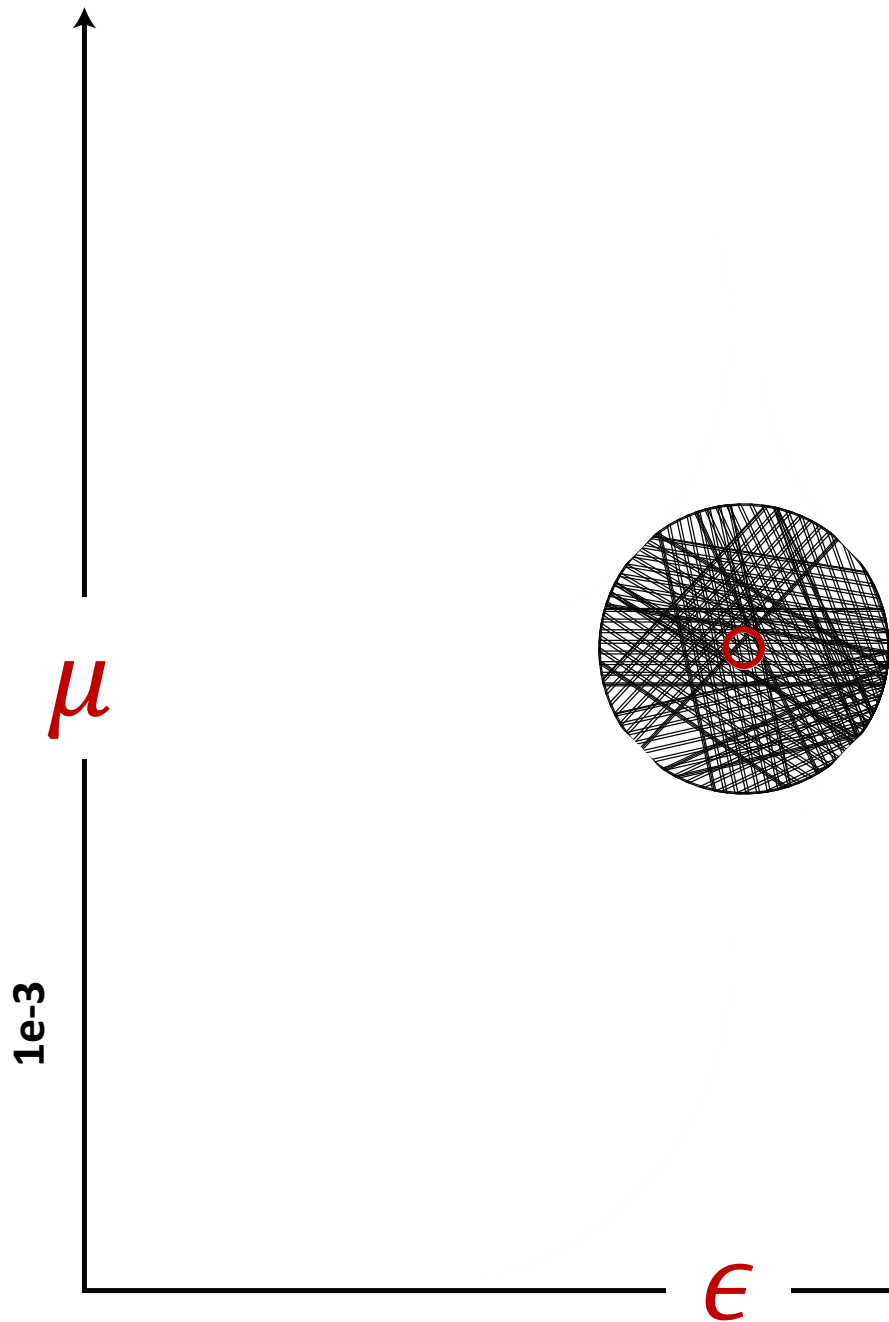
Parameters – Envelope Size

- μ
 - Used in μ -separation for separating primary features and secondary features.
- ϵ (same parameter used in TetWild)
 - Maximum distant that secondary feature edges can deviate from input.



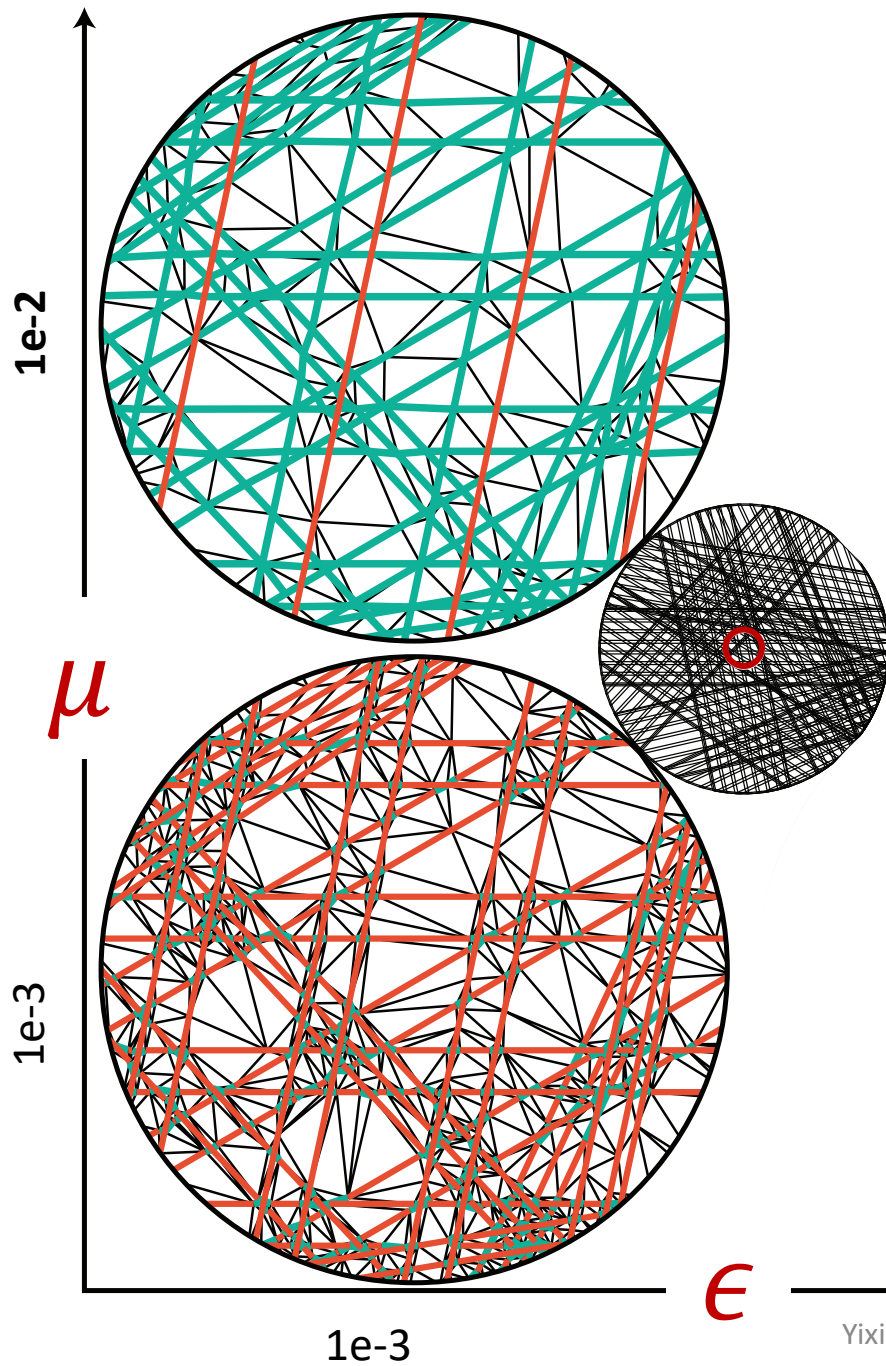
Parameters – Envelope Size





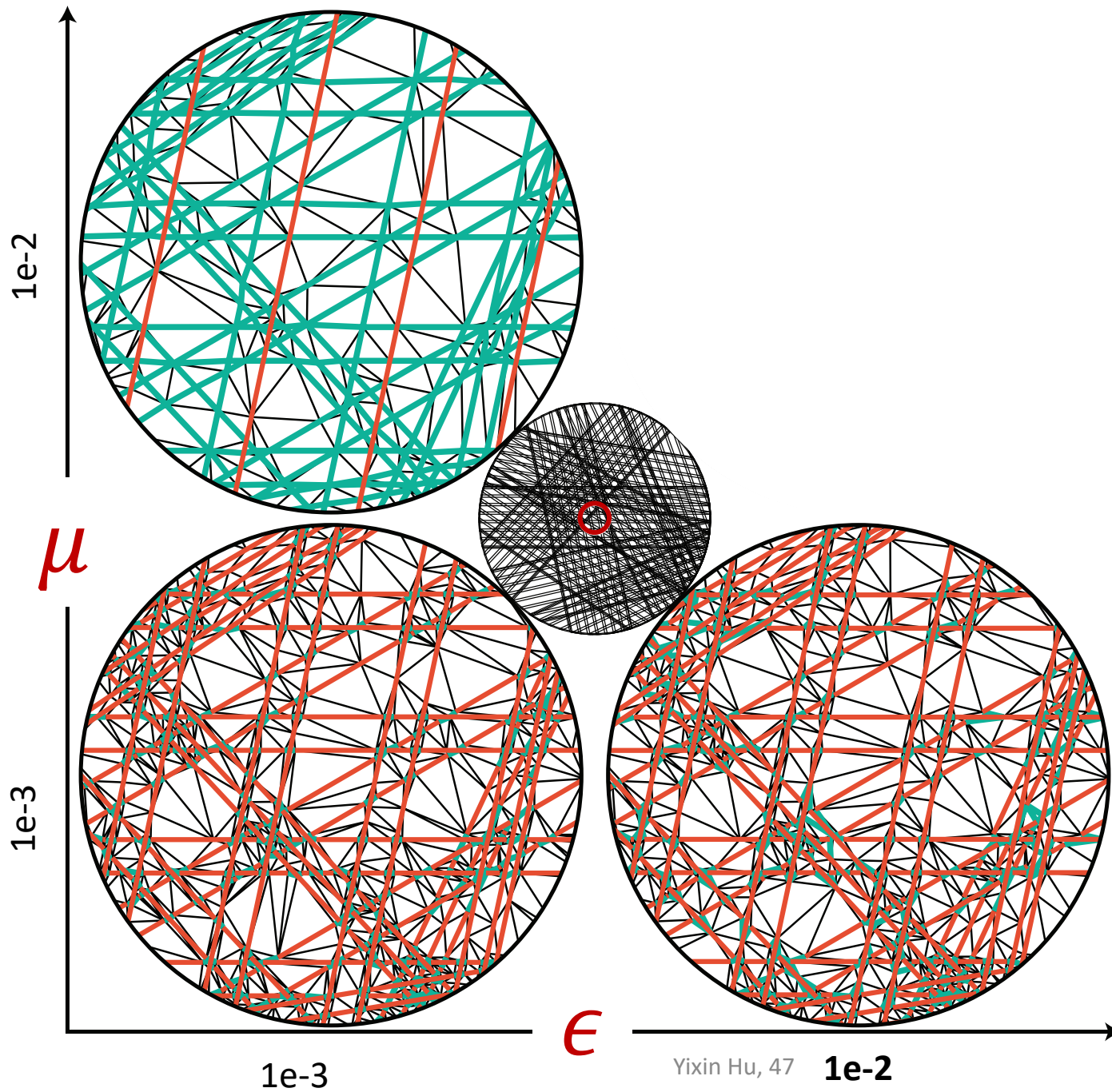
Red:
primary feature edges

Green:
secondary feature edges



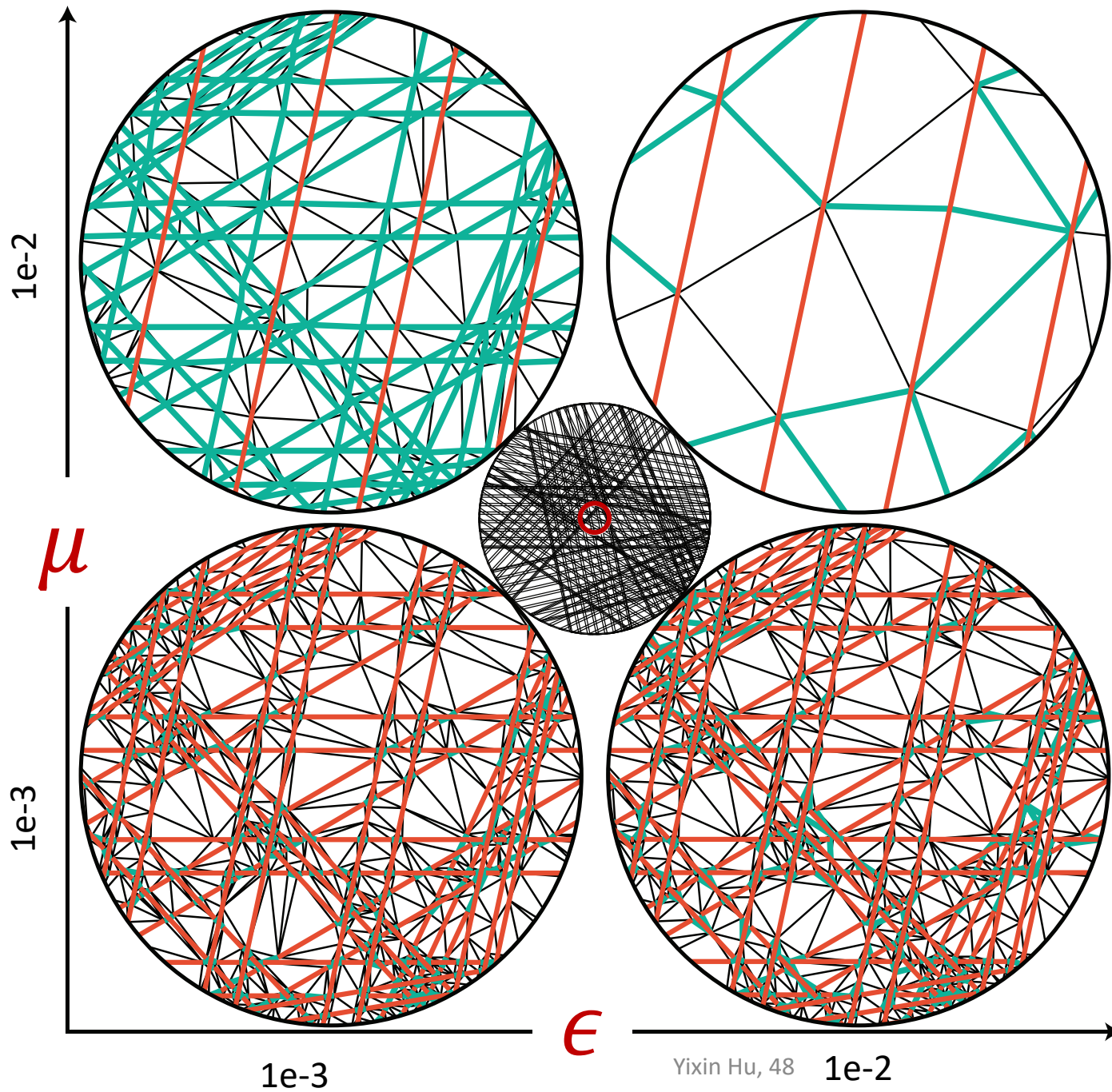
Red:
primary feature edges

Green:
secondary feature edges



Red:
primary feature edges

Green:
secondary feature edges

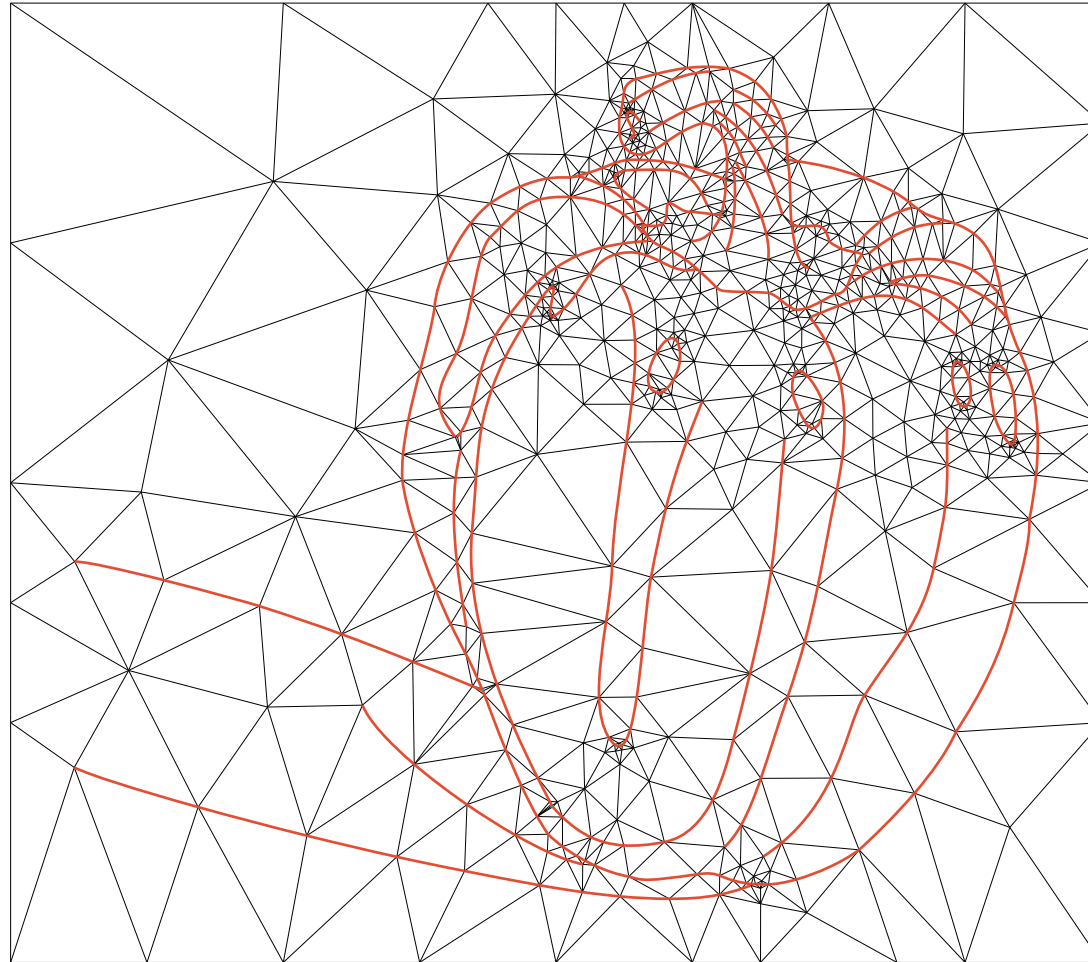


Red:
primary feature edges

Green:
secondary feature edges

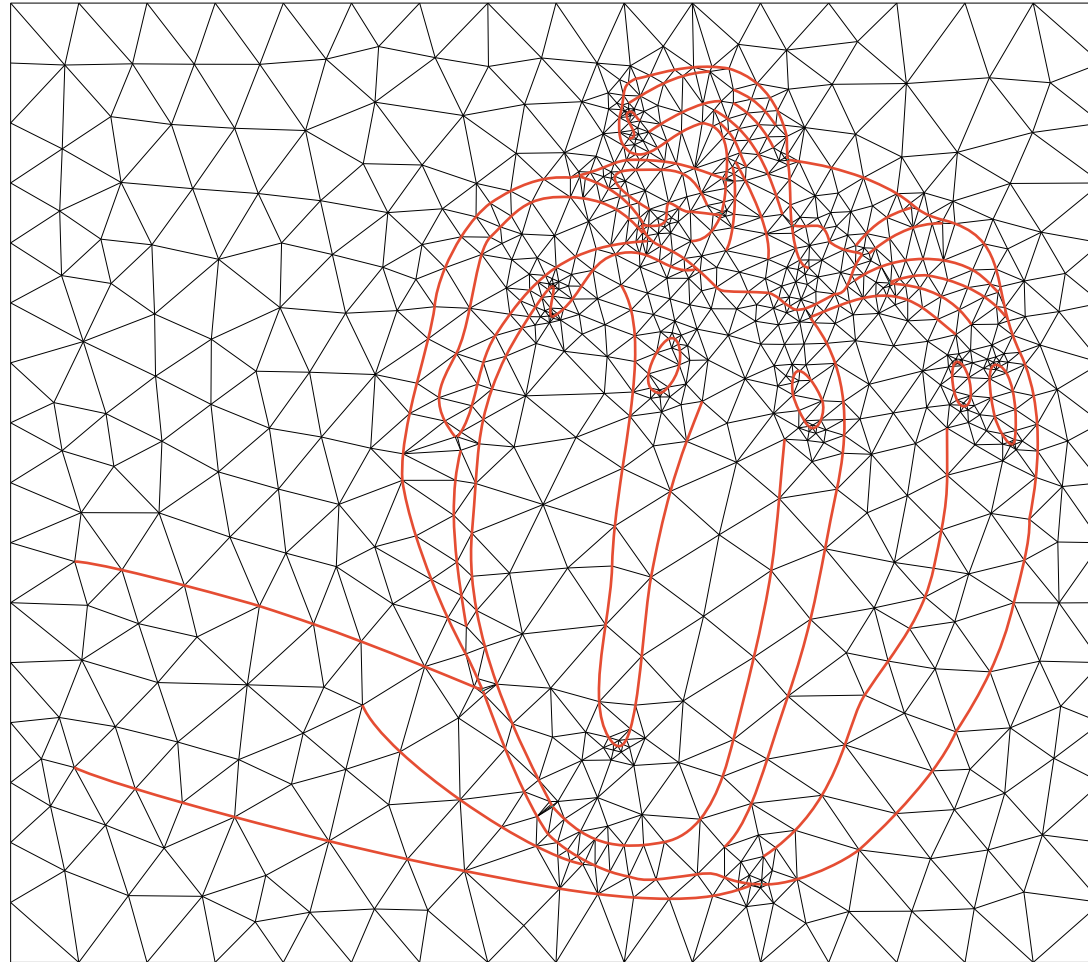
Parameters – Targeted Edge Length

Edge_length = diag_bbox/5



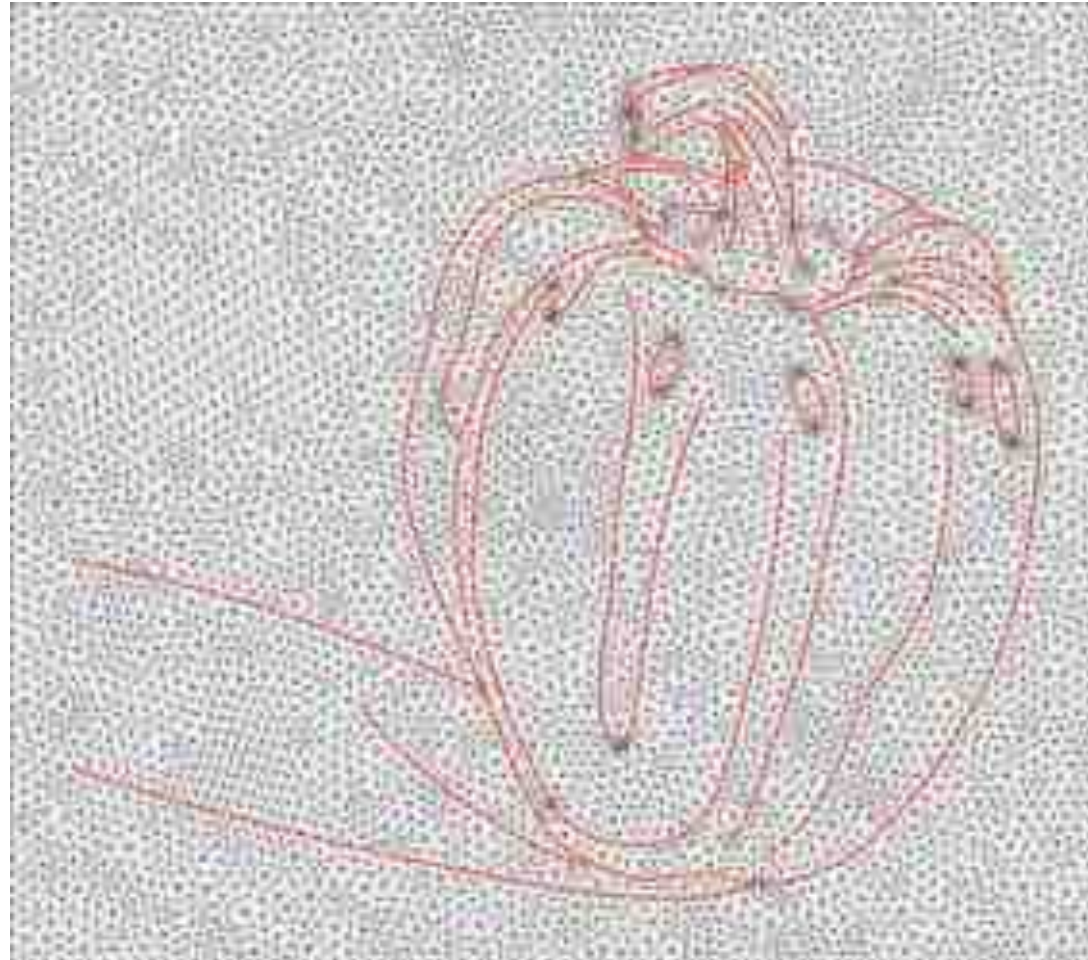
Parameters – Targeted Edge Length

Edge_length = diag_bbox/20

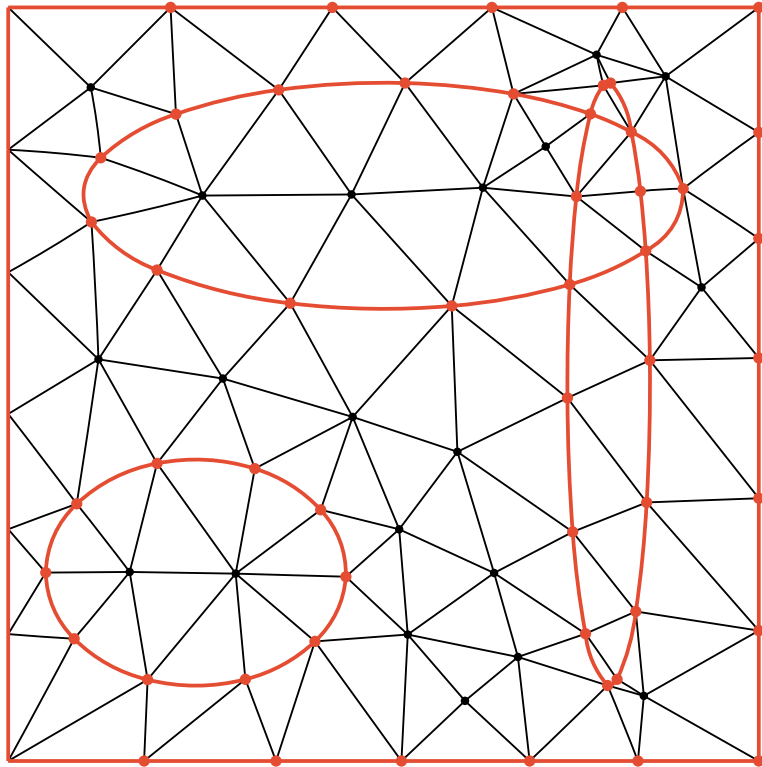


Parameters – Targeted Edge Length

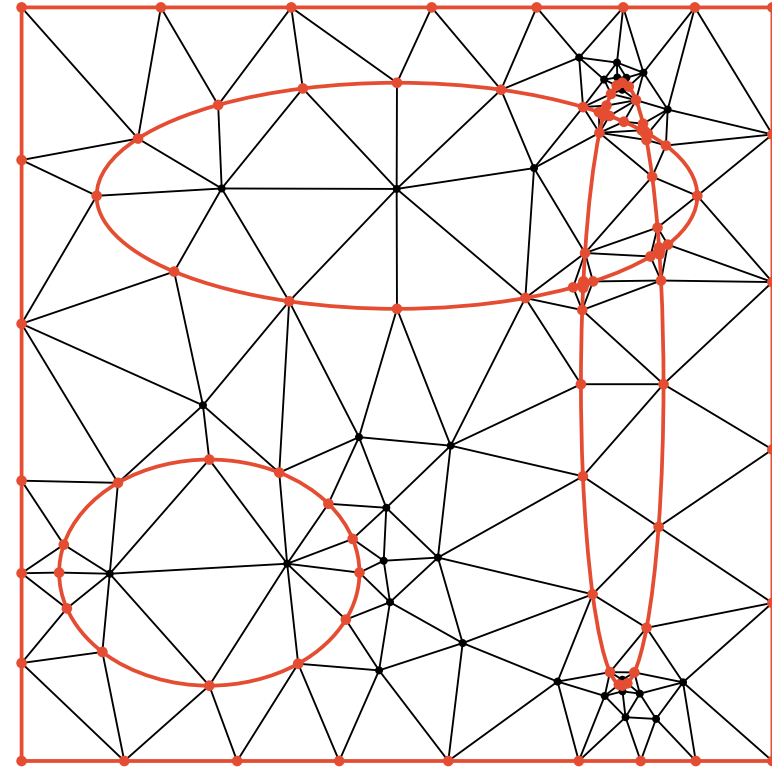
Edge_length = diag_bbox/*100*



MATLAB vs TriWild

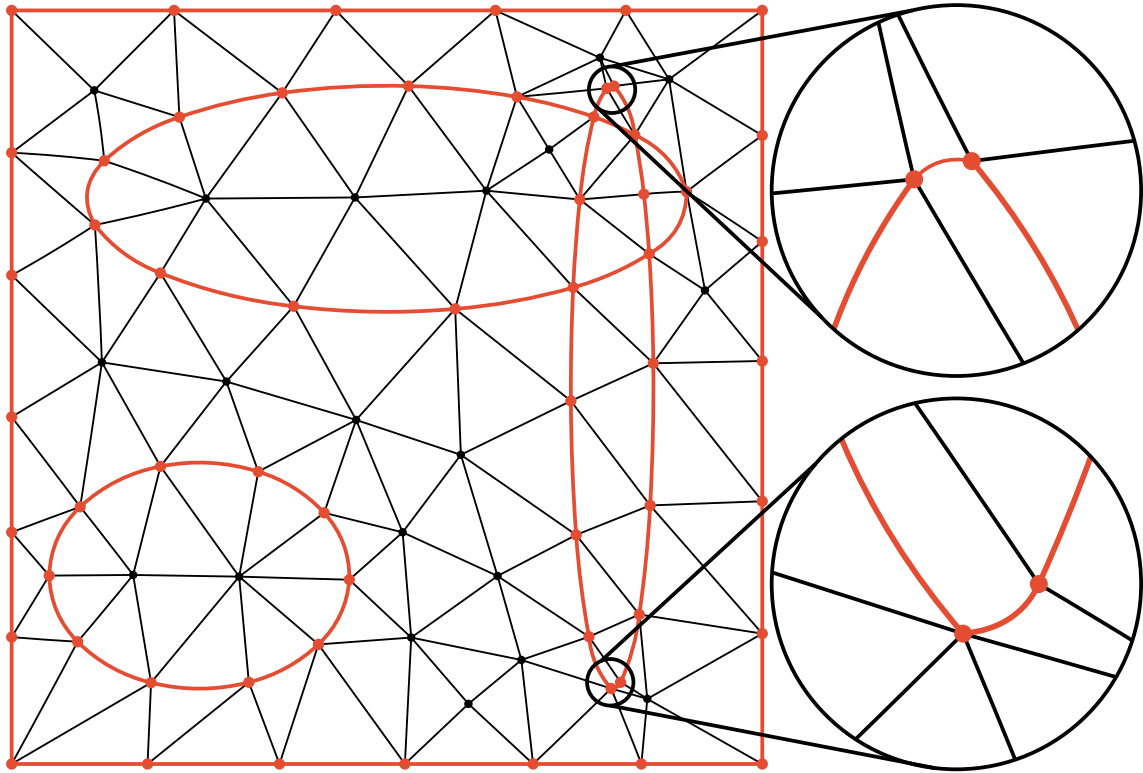


(Generated by MATLAB)

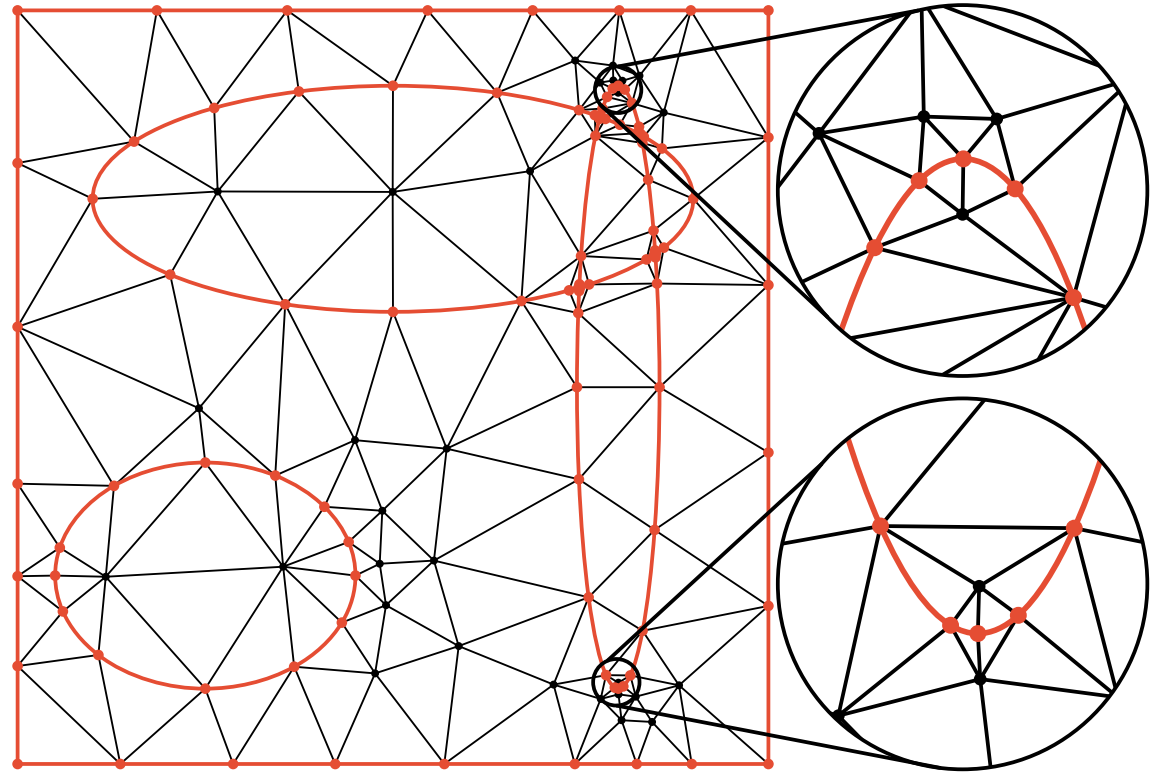


(Generated by TriWild)

MATLAB vs TriWild

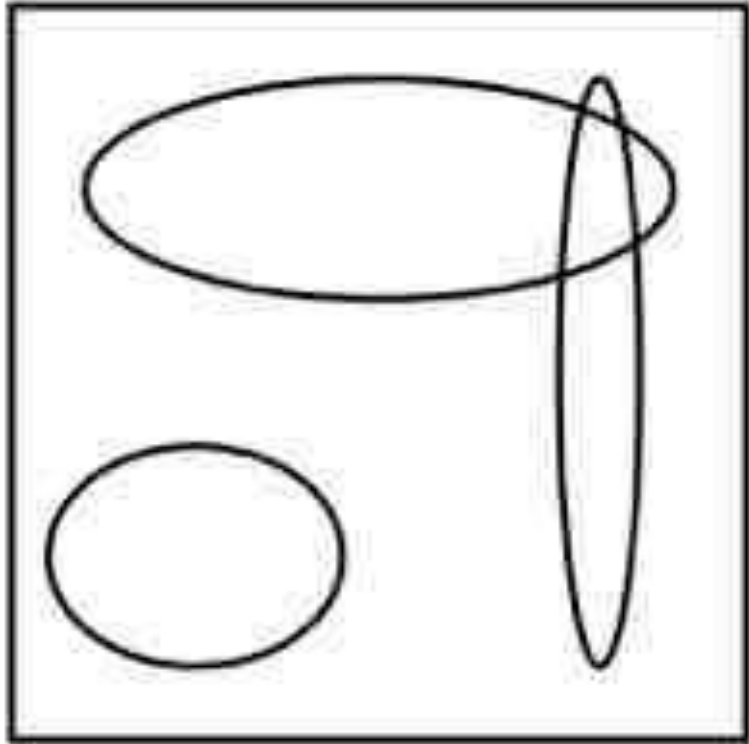


(Generated by MATLAB)

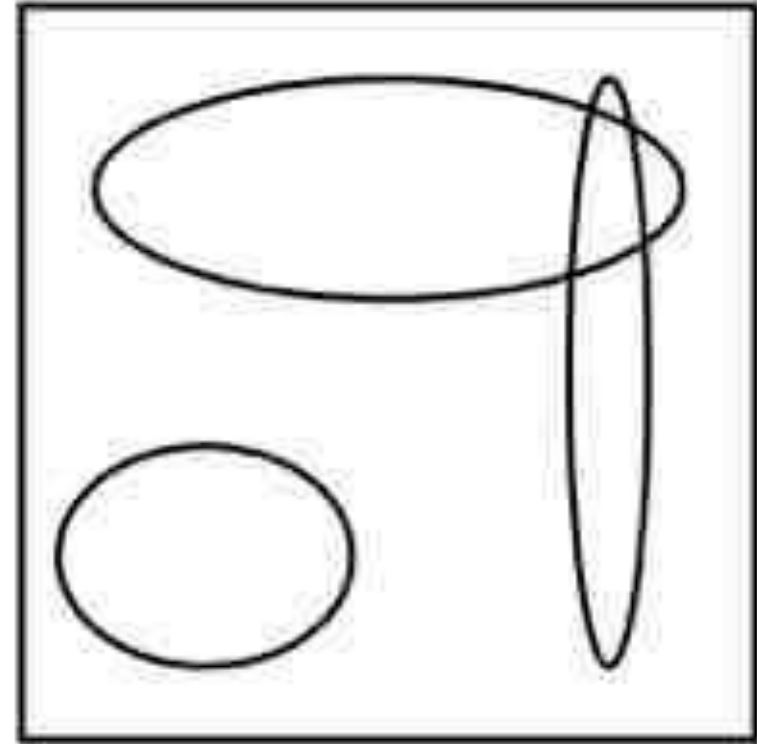


(Generated by TriWild)

MATLAB vs TriWild

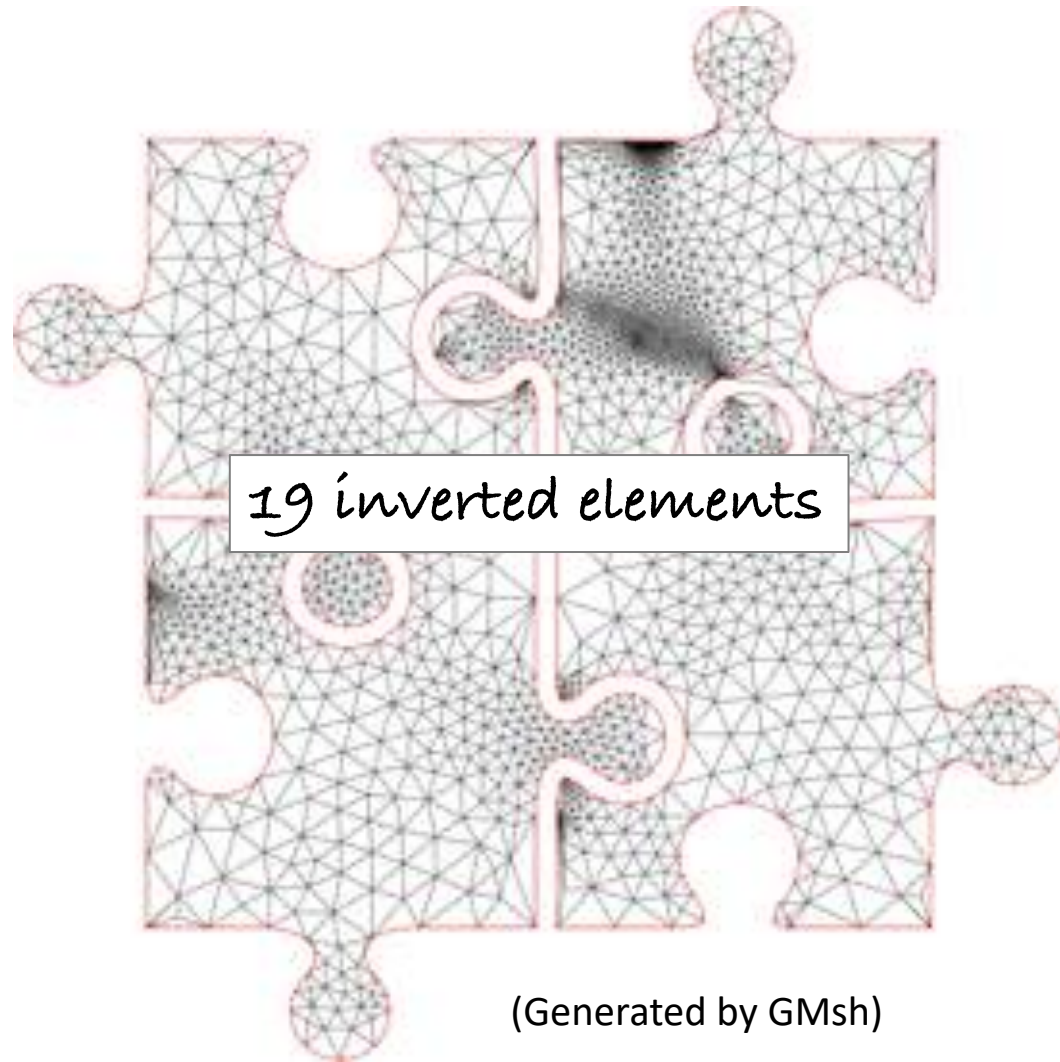


Original Input

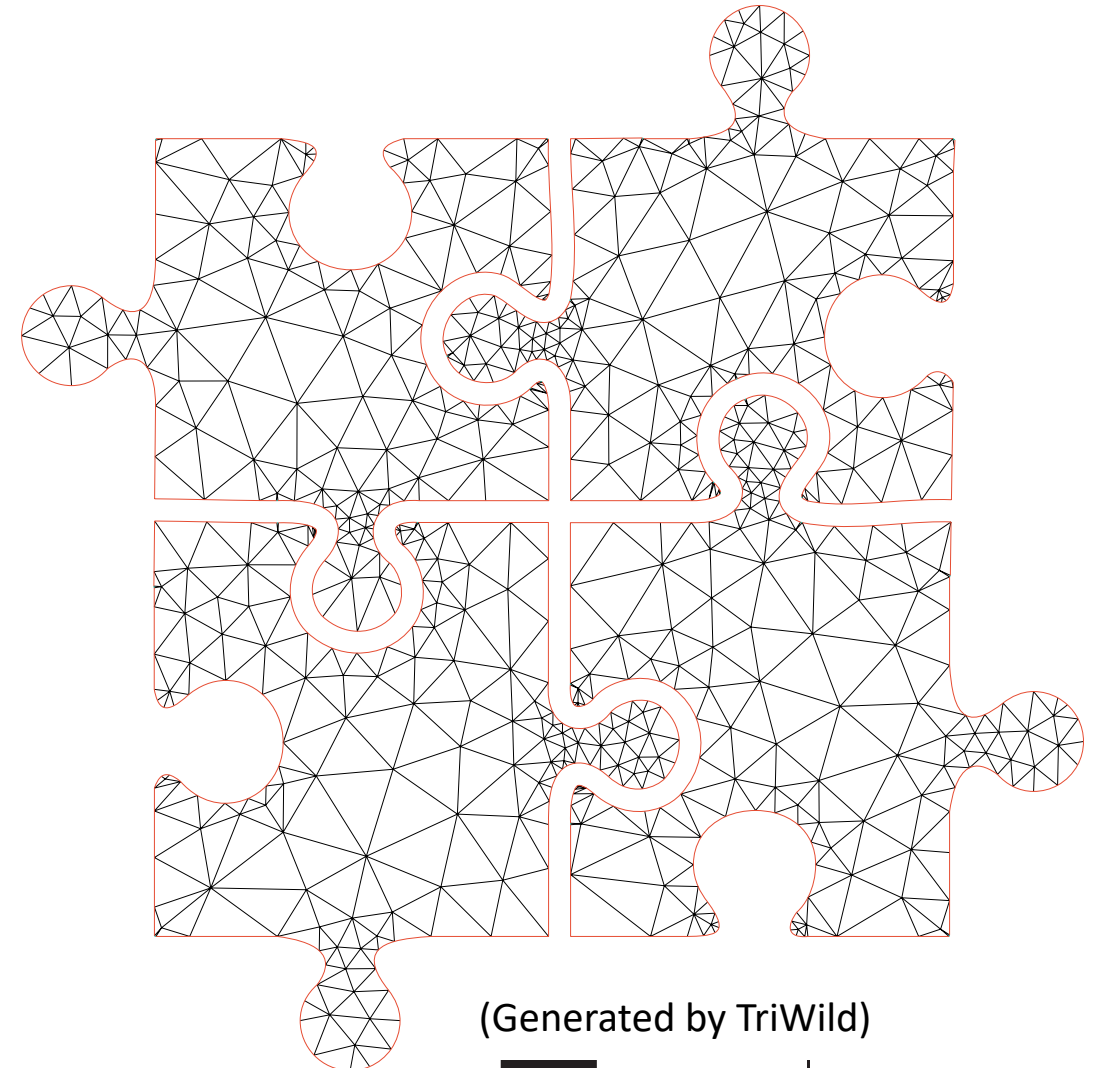


Failure Input

GMsh vs TriWild



(Generated by GMsh)



(Generated by TriWild)

Large-scale Test

- Dataset: **19,686** real-world SVG images from openclipart.org.
- Success rate: **19,685** output meshes
 - The only failure is due to large input size (1.5GB).

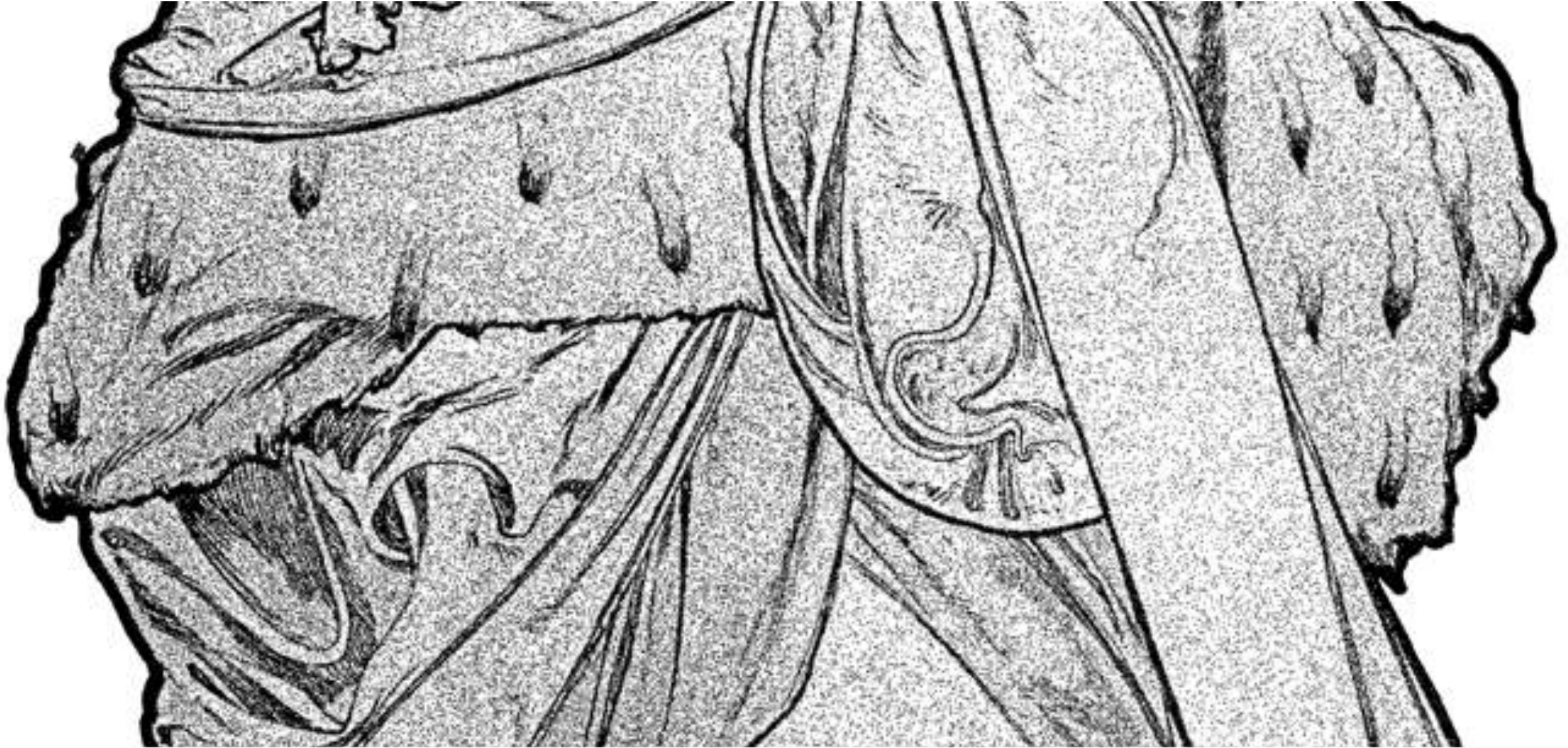
Success:

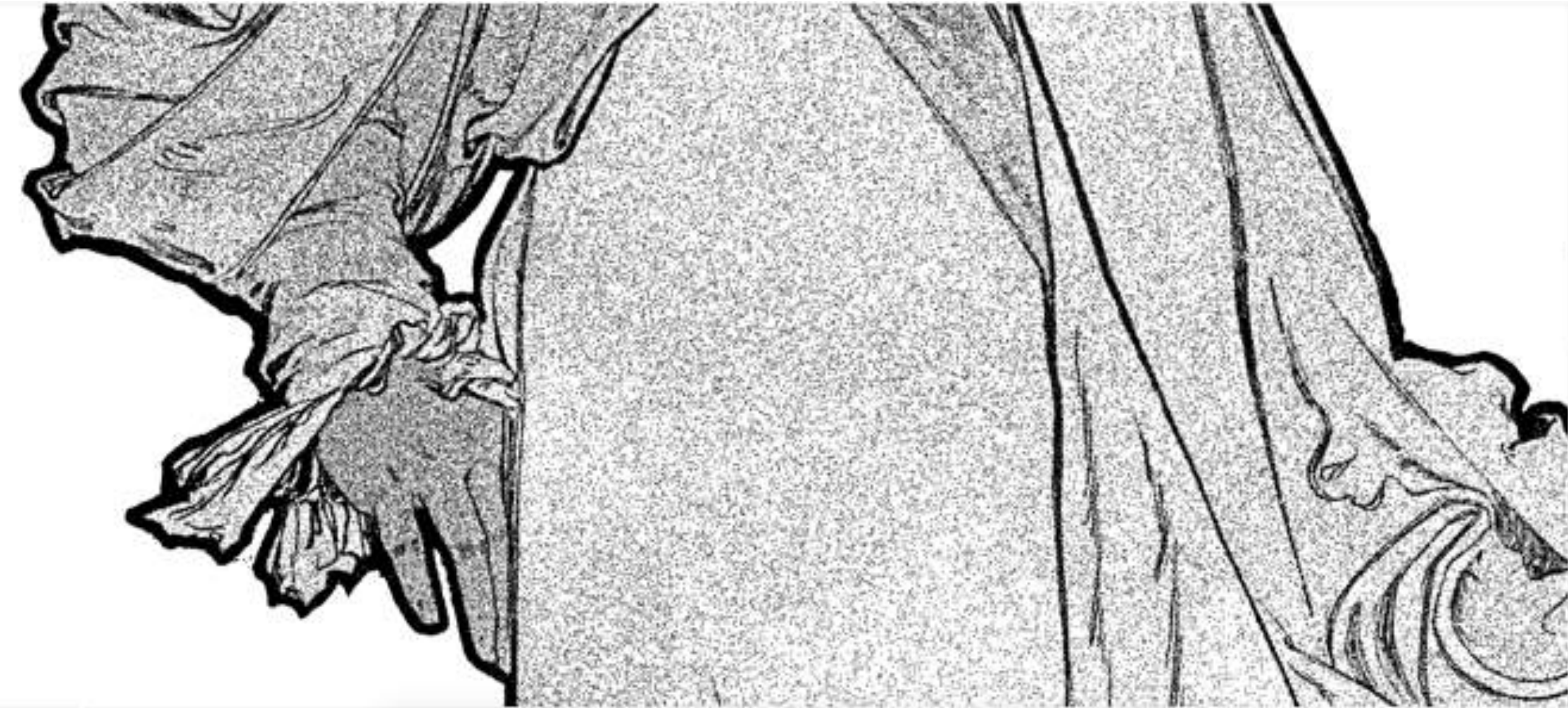
< 6 hours

< 64GB memory











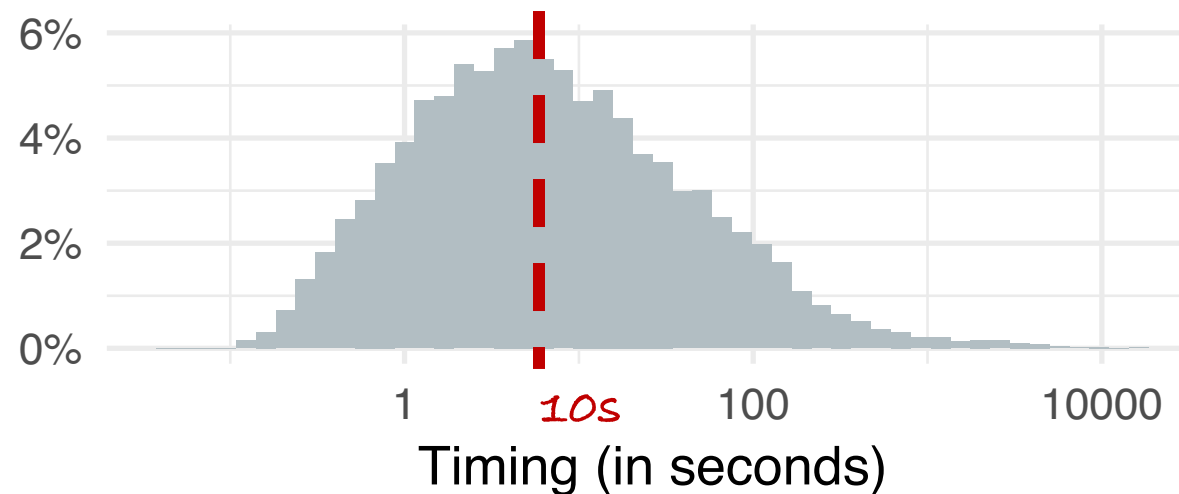






Large-scale Test

- Dataset: **19,686** real-world SVG images from openclipart.org.
- Success rate: **19,685** output meshes
 - The only failure is due to large input size (1.5GB).



Applicat



Diffusion Curves: A Vector Representation for Smooth-Shaded Images

Alexandrina Orzan
Inria - LJK (U. Grenoble -
CNRS)

Adrien Bousseau
Inria Sophia Antipolis

Holger Winnemöller
Adobe Systems

Pascal Barla
Inria - U. Bordeaux - ICGS -
CNRS

Joëlle Thollot
Inria - LJK (U. Grenoble -
CNRS)

David Salesin
Adobe Systems

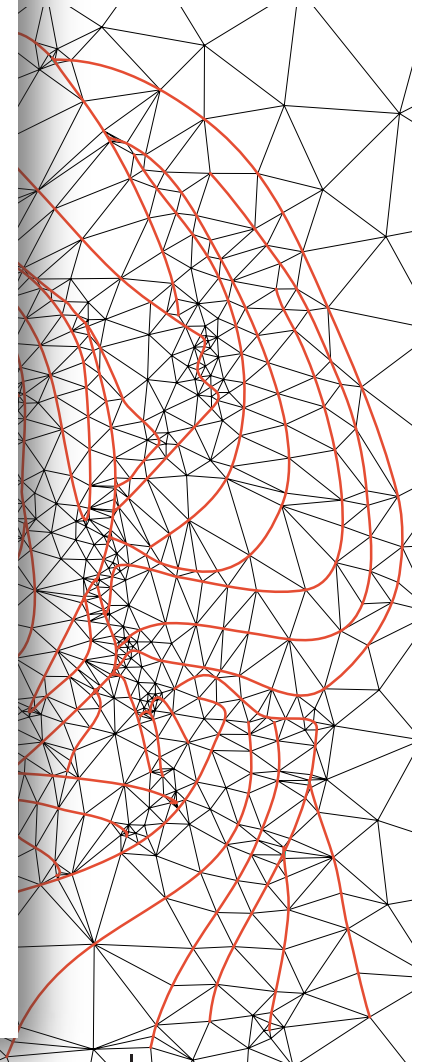
ABSTRACT

We describe a new vector-based primitive for creating smooth-shaded images, called the *diffusion curve*. A diffusion curve partitions the space through which it is drawn, defining different colors on either side. These colors may vary smoothly along the curve. In addition, the sharpness of the color transition from one side of the curve to the other can be controlled. Given a set of diffusion curves, the final image is constructed by solving a Poisson equation whose constraints are specified by the set of gradients across all diffusion curves. Like all vector-based primitives, diffusion curves conveniently support a variety of operations, including geometry-based editing, keyframe animation, and ready stylization. Moreover, their representation is compact and inherently resolution independent. We describe a GPU-based implementation for rendering images defined by a set of diffusion curves in real time. We then demonstrate an interactive drawing system for allowing artists to create artworks using diffusion curves, either by drawing the curves in a freehand style, or by tracing existing imagery. Furthermore, we describe a completely automatic conversion process for taking an image and turning it into a set of diffusion curves that closely approximate the original image content.

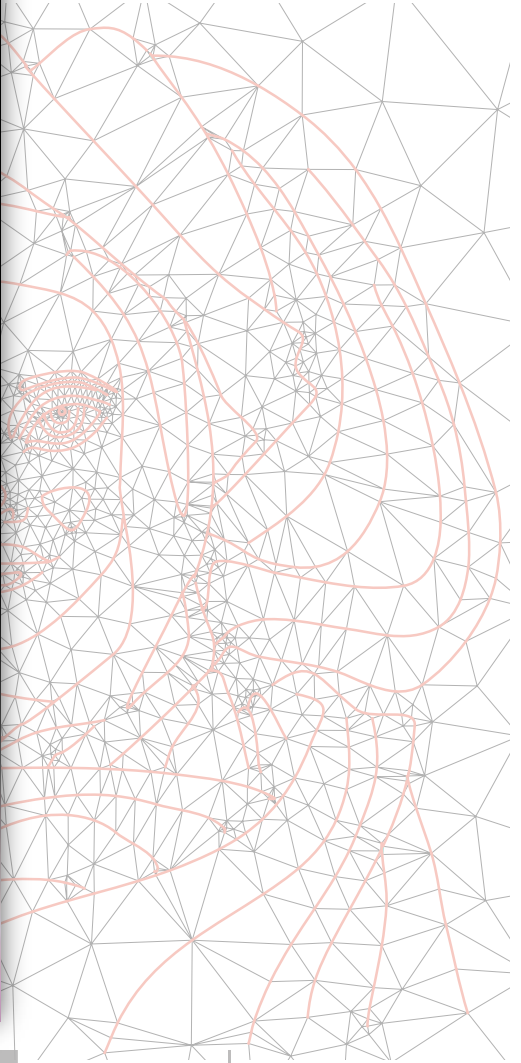


Figure 1: Diffusion curves (left), and the corresponding color image (right). Note the complex shading on the folds and blur on the face.

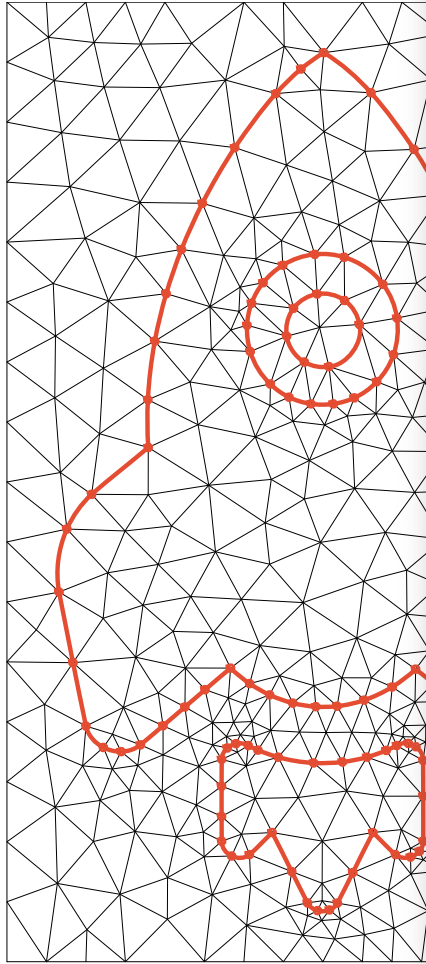
for graphics — primarily, a more direct mapping between the hardware devices used for acquisition and display of images, and their internal representation — vector graphics continues to provide certain benefits as well. Most notably, vector graphics offers a more compact representation, geometric editability, and resolution independence (allowing scaling of images while retaining sharp edges, see Figure 2). Vector-based images are also more easily animated through keyframe animation of their individual geometric primi-



Application



Application



Curved Mesh

EUROGRAPHICS Workshop on March-Based Interfaces and Modeling (2008)
C. Alvarez and M. P. Coll (Editors)

Repoussé: Automatic Inflation of 2D Artwork

Pradipt Joshi^{1,2} Nathan A. Carr²

¹U.C. Berkeley, ²Adobe Systems Inc.

Abstract

We describe a new system for the interactive enhancement of 2D art with 3D geometry. *Repoussé* creates a 3D shape by inflating the surface that interpolates the input curves. By using the mean curvature stored at boundary vertices as a degree of freedom, we are able to control the inflated surface intuitively and efficiently using a single linear system. Refined handles both smooth and sharp position constraints. Position constraint vertices can also have curvature constraints for controlling the inflation of the local surface. We show the applications of our system to font design, stroke design, photo enhancement and freeform 3D shape design.

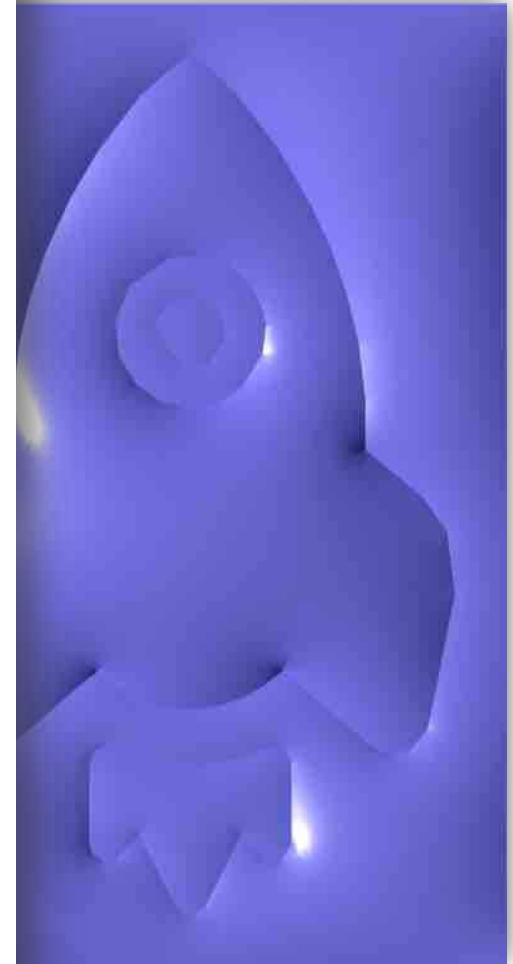
Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

March-based interfaces are becoming popular as a method for quick 3D shape modeling. With an ever-increasing set of modeling features, the powerful 3D sketching interface can construct shapes that range from industrial, technical objects (DIERKE) to smooth, organic shapes (SMITH), (KANG), (MISAMOTI). Usually, the application of these shape sketching interfaces is 3D shape design, but such interfaces can also be helpful for adding thickness to 2D images (PETER). The 3D curves drawn by the user are simply a means to get to the end result: the 3D shape. Instead of using the 2D curves to design 3D shapes, we use the resulting interpolating 3D shapes to enhance the existing 2D curves. That is, we apply the shape sketching interface to another, highly interesting application: 2D art creation and design.



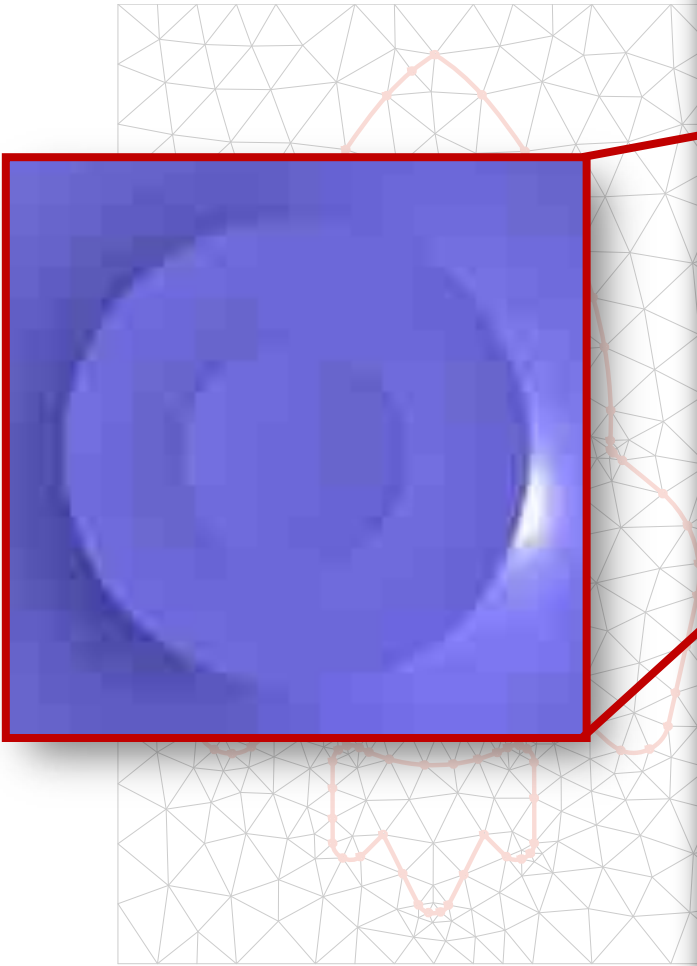
Figure 1: Boundary curve inflation using *Repoussé*.



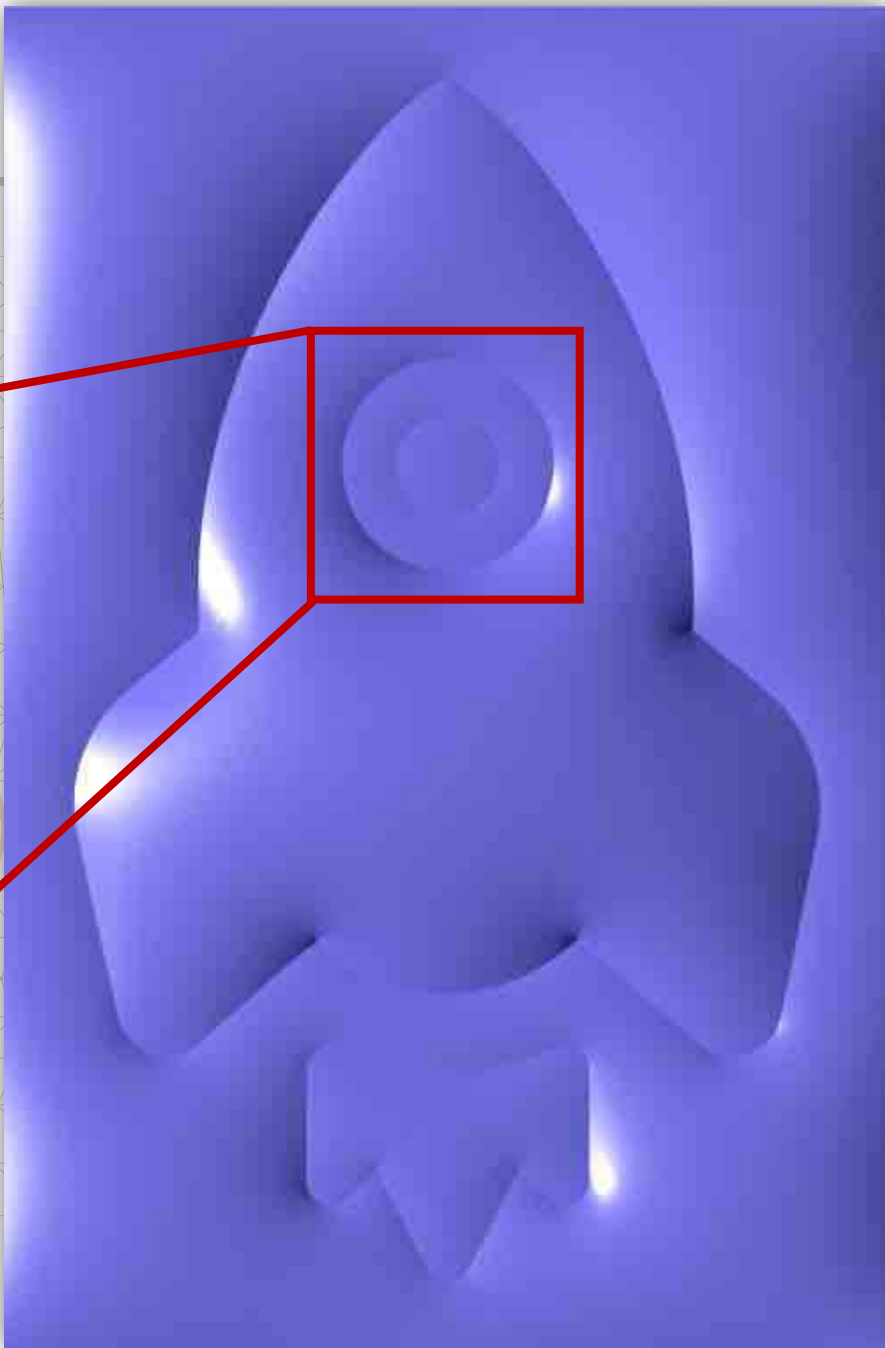
Using Linear Output



Application –

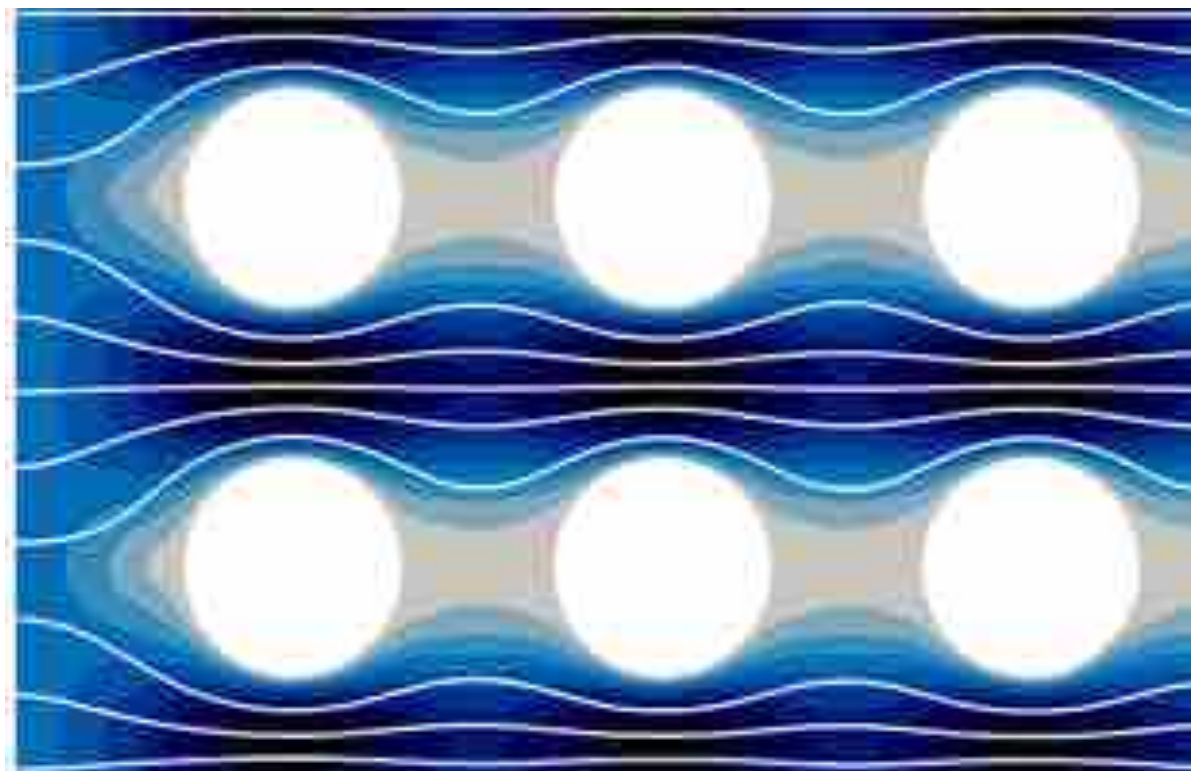
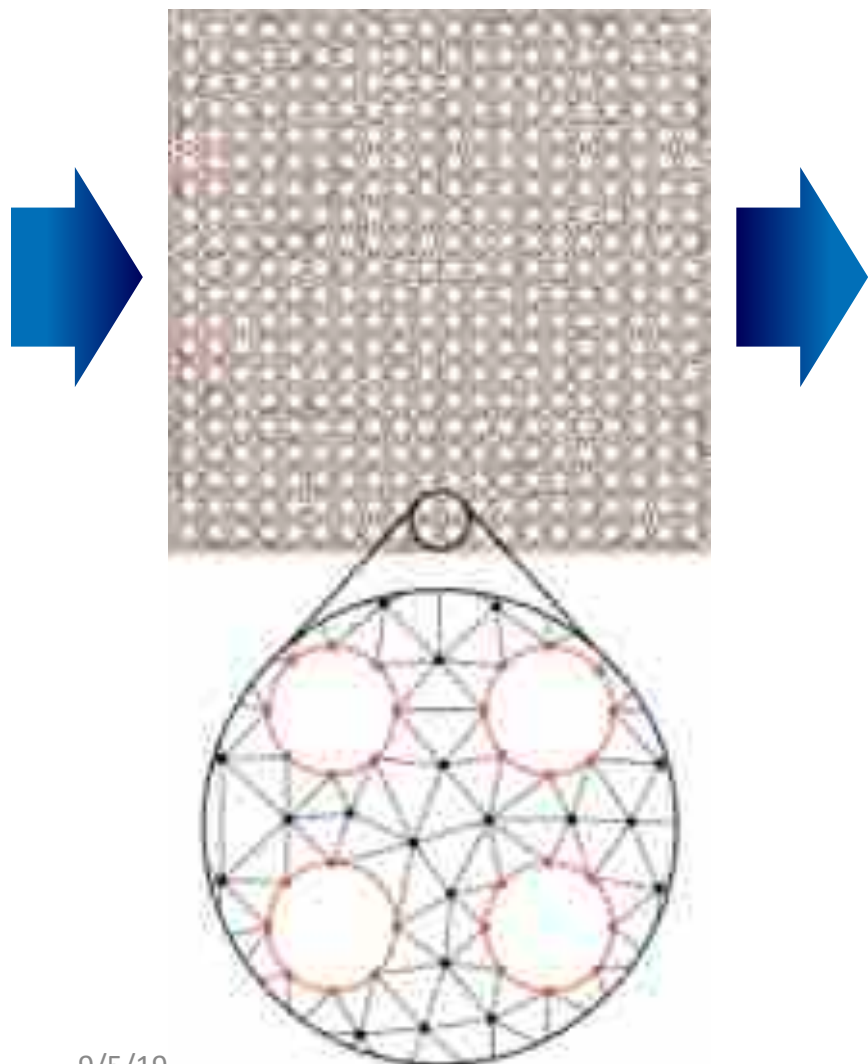


Curved Mesh



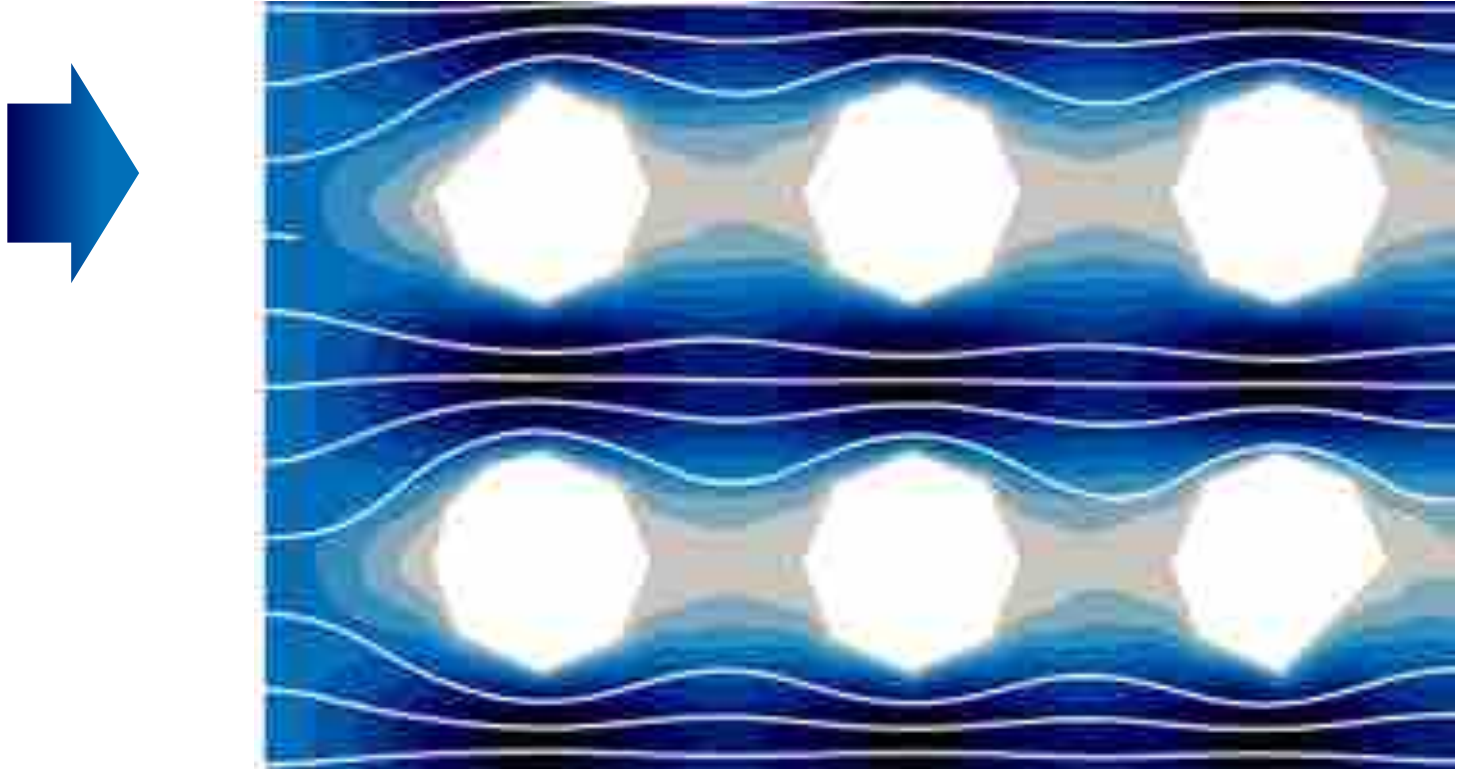
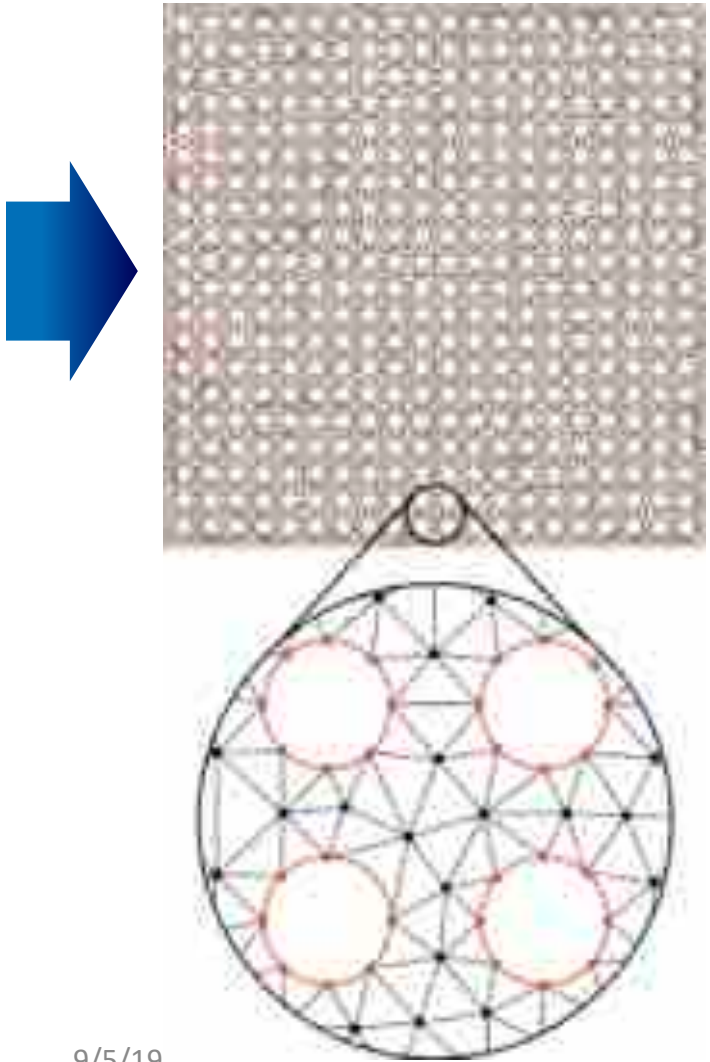
Using Linear Output

Application – Stokes



Using Curved Mesh

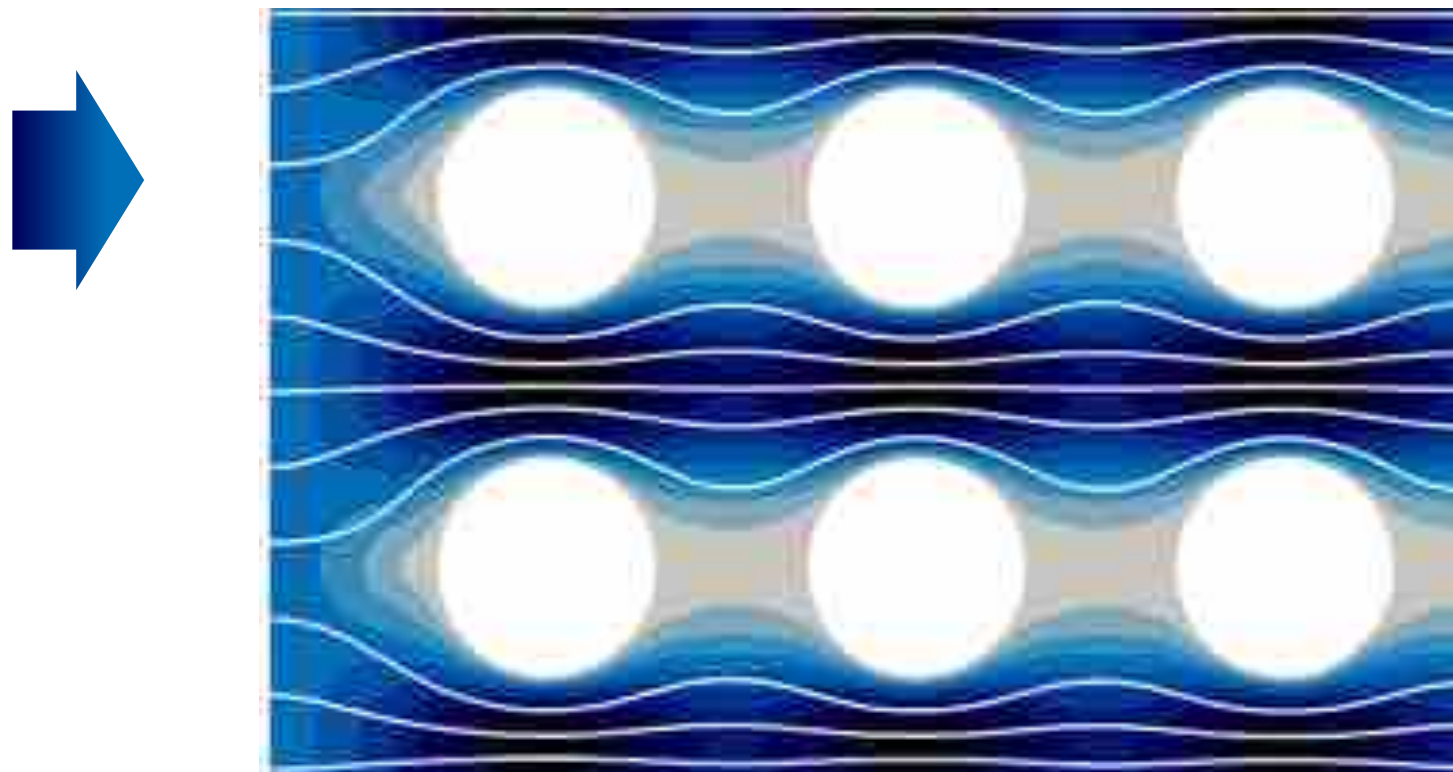
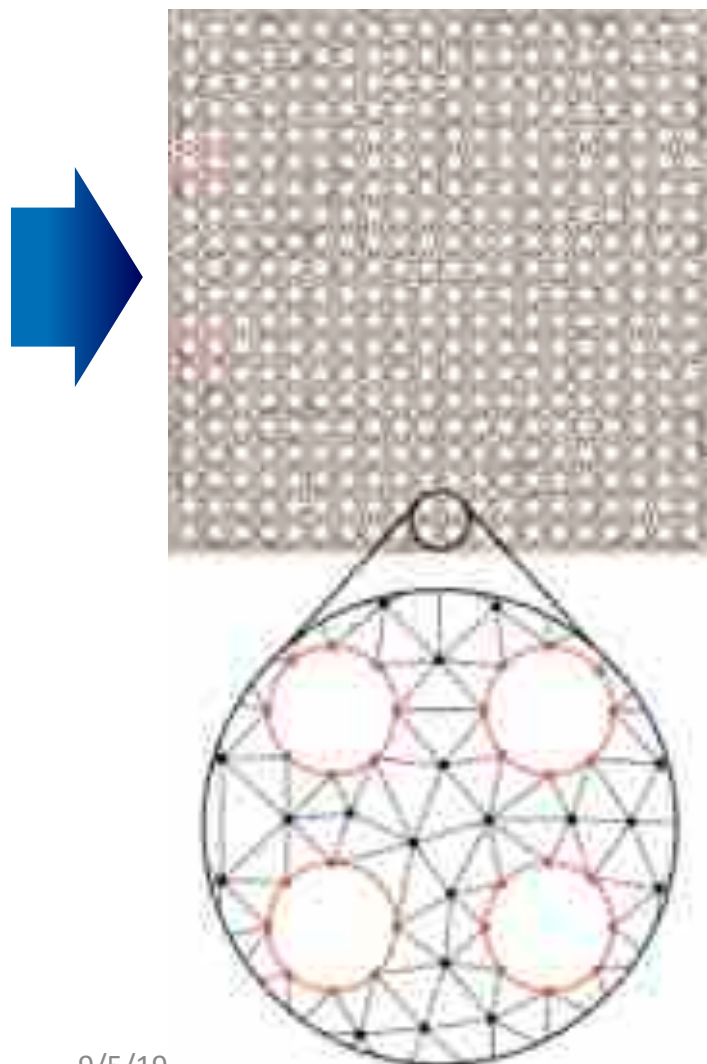
Application – Stokes



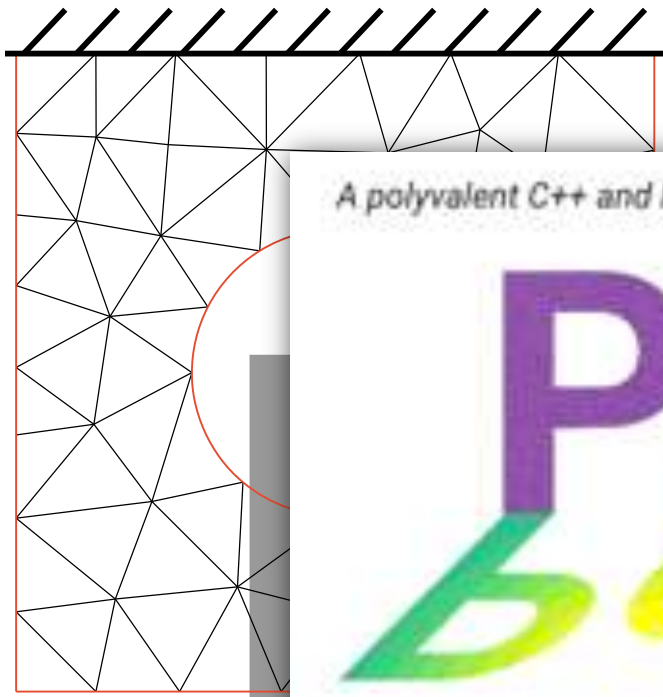
Using Linear Mesh



Application – Stokes



Application – Elasticity



Gravity

Curved Mesh

0.09s



Using Curved Mesh

115s



Using **Dense Linear** Mesh

A polyvalent C++ and Python FEM library.

PolyFEM
BONEFEM

Linear Pipeline

Dataset 1: Raw OpenClip20k

- Piece-wise linear approximations of 19,686 SVG images.

Dataset 2: Cleaned OpenClip20k

- Raw dataset with duplication and degeneracy removed.

Dataset 3: Snapped OpenClip20k

- Raw dataset with iteratively snap rounding using ϵ as pixel size.

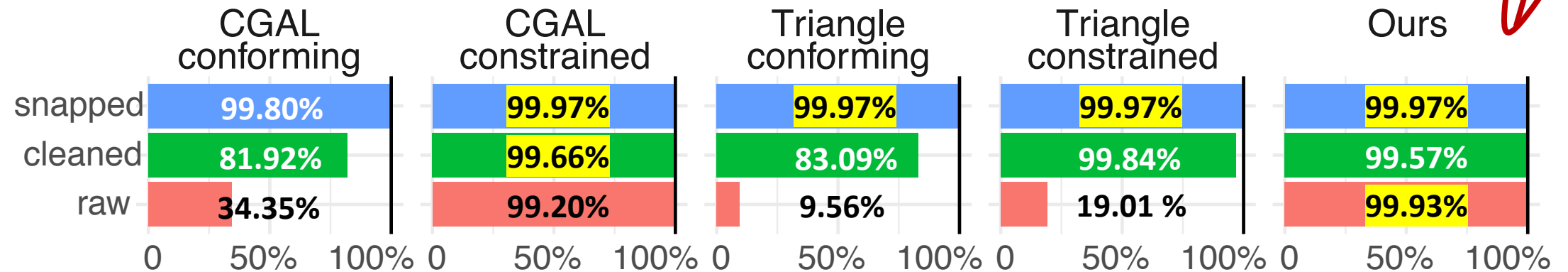


Linear Pipeline

Success:
<1 hour
<16GB memory

Compared with:

- 1) CGAL Conforming/Constrained Delaunay Triangulation
- 2) Triangle Conforming/Constrained Delaunay Triangulation



Linear Pipeline

Success:

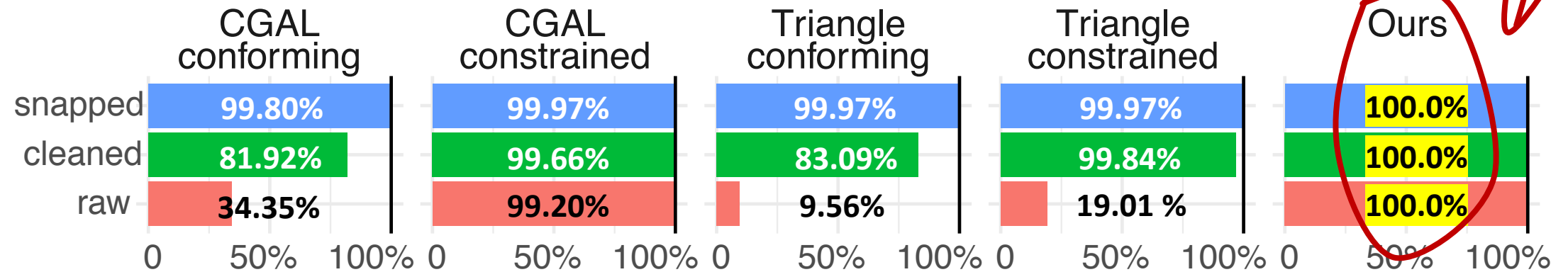
< 1 hour

~~< 10 GB~~ memory

64GB

Compared with:

- 1) CGAL Conforming/Constrained Delaunay Triangulation
- 2) Triangle Conforming/Constrained Delaunay Triangulation



Thank You!



<https://github.com/wildmeshing/TriWild>

ACKNOWLEDGMENTS

- This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise.
- This work was partially supported by the NSF CAREER award under Grant No. 1652515, the NSF grant IIS-1320635, the NSF grant DMS-1436591, the NSF grant 1835712, the SNSF grant P2TIP2_175859, NSERC Discovery Grants (RGPIN-2017-05235 & RGPAS-2017-507938), Canada Research Chair award, Connaught Fund, a gift from Adobe Research, and a gift from nTopology.



Thank You!



<https://github.com/wildmeshing/TriWild>

- Please scan the QR code for reference implementation in C++ and data.

- Python wrapper via Conda:

```
conda config --add channels conda-forge  
conda install wildmeshing
```

- Contact me if you have any questions!

yixin.hu@nyu.edu

