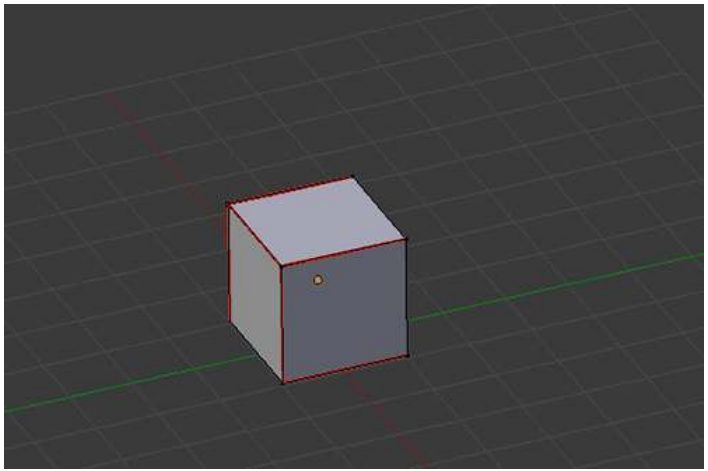
Introduction to Mesh Parameterization

Richard Liu



Source

Self Intro

5th (?) year PhD at UChicago 3DL lab

Self Intro

5th (?) year PhD at UChicago 3DL lab

3rd year presenting for SGI 🎉

Self Intro

5th (?) year PhD at UChicago 3DL lab 

3rd year presenting for SGI 🎉

AI for mesh stuff (generation, animation, texturing, deformation, etc…)

Self Intro

5th (?) year PhD at UChicago 3DL lab

3rd year presenting for SGI 🎉

AI for mesh stuff (generation, animation, texturing, deformation, etc…)

A PhD journey through hair

2022

2023

2024

2025

Yesterday

Session Content

Definition and mathematical foundations

Computing fixed-boundary parameterizations

How to measure parameterization distortion

Jupyter notebook tutorial

Exercises: 101_parameterization_fundamentals, 102_parameterization_exercises

Source Material

Hormann, K., Polthier, K., Sheffer, A. “Mesh Parameterization: Theory and Practice”. Siggraph Asia 2008 Course. https://www.cs.princeton.edu/courses/archive/fall12/cos526/papers/hormann08.pdf

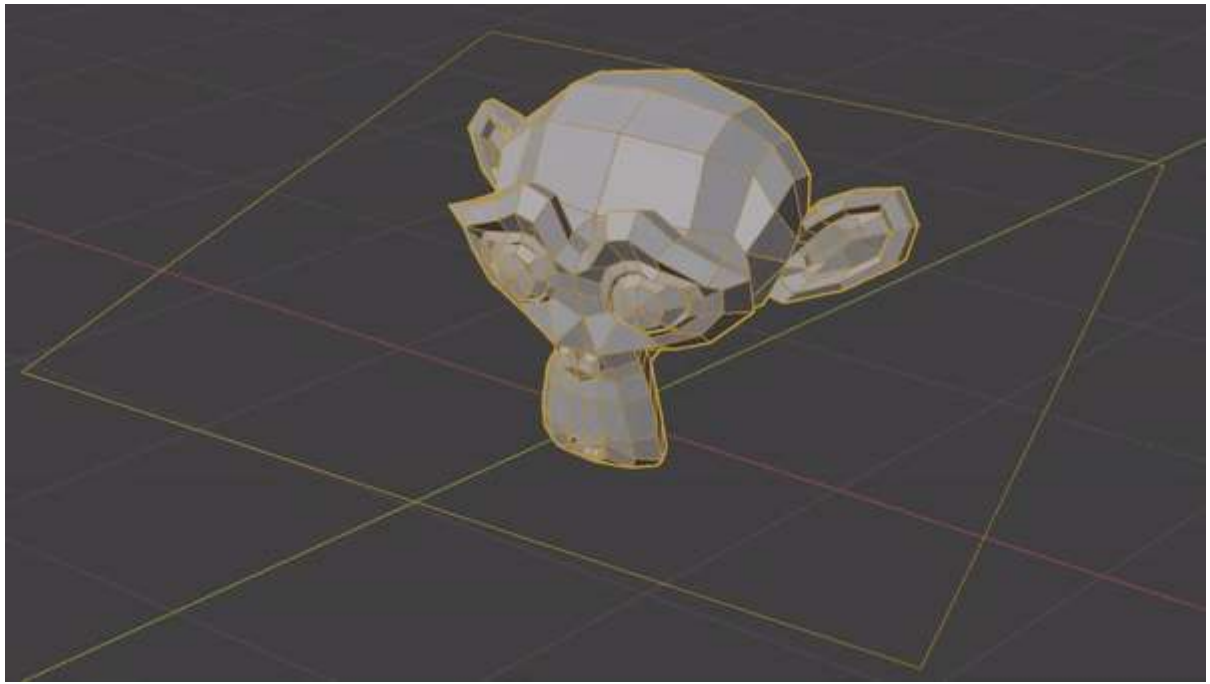
Jacobson, A. Geometry processing course parameterization assignment. https://github.com/alecjacobson/geometry-processing-parameterization

“Parameterization I”. Stanford CS-468. https://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/12_Parameterization1.pdf

Sheffer, A., Praun, E., Rose, K. “Mesh Parameterization Methods and Their Applications”. Trends in CG and Vision 2006. https://people.eecs.berkeley.edu/~jrs/meshpapers/ShefferPraunRose.pdf

Part 1: Foundations

What is mesh parameterization?

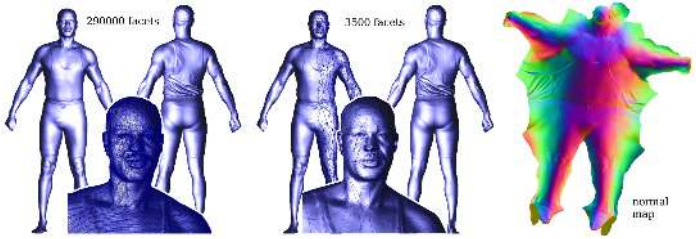


Process of mapping 3D surface mesh onto 2D plane

Source

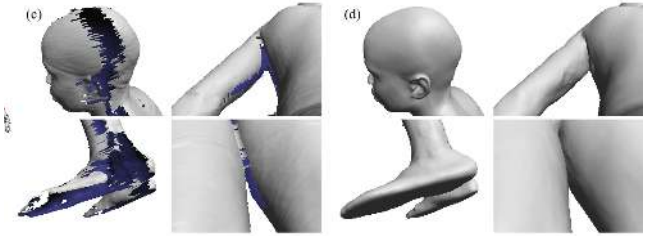
Lots of applications …

Normal Mapping/Simplification



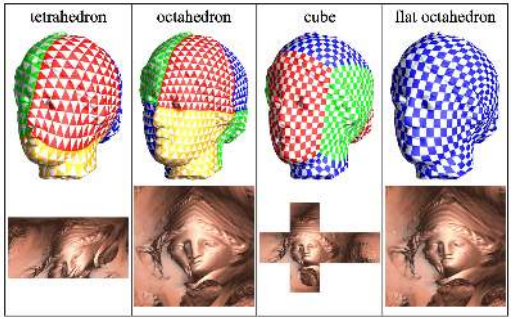
Source

Shape Completion



Source

Remeshing

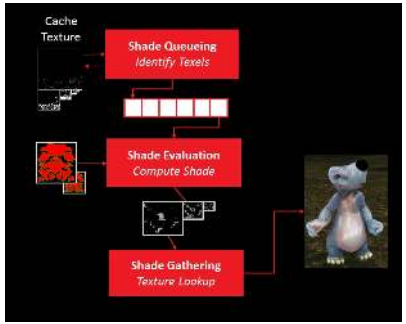


Source

Detail Transfer

Source

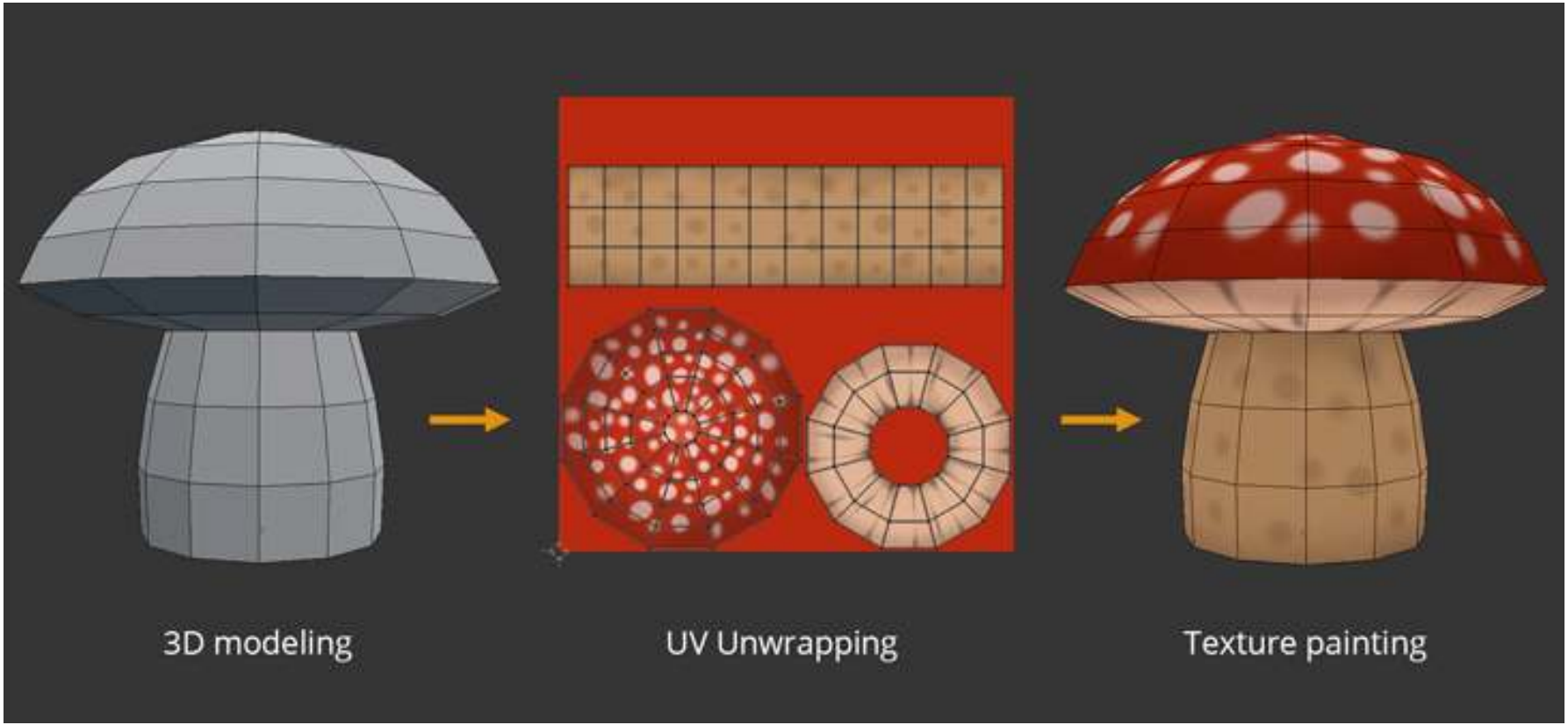
Texture Space Shading



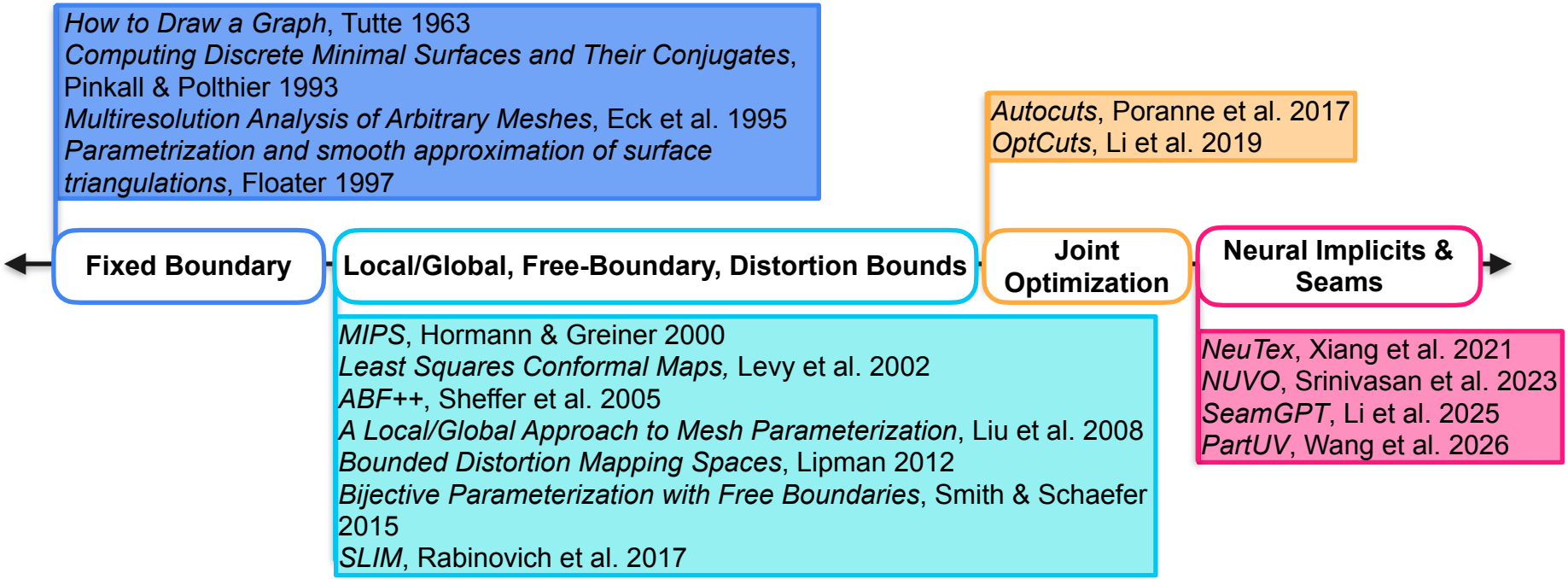
Source

Most Common Application: Texture Mapping

Source

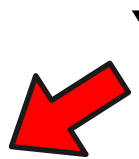


Timeline of Works*



* Not meant to be treated as a comprehensive survey. Important works/subproblems omitted due to presenter's ignorance and commitment to a clean, overlap-free narrative.

Timeline of Works*



You are here

How to Draw a Graph, Tutte 1963
Computing Discrete Minimal Surfaces and Their Conjugates, Pinkall & Polthier 1993
Multiresolution Analysis of Arbitrary Meshes, Eck et al. 1995
Parametrization and smooth approximation of surface triangulations, Floater 1997

Autocuts, Poranne et al. 2017
OptCuts, Li et al. 2019

Fixed Boundary — **Local/Global, Free-Boundary, Distortion Bounds** — **Joint Optimization** — **Neural Implicits & Seams**

MIPS, Hormann & Greiner 2000
Least Squares Conformal Maps, Levy et al. 2002
ABF++, Sheffer et al. 2005
A Local/Global Approach to Mesh Parameterization, Liu et al. 2008
Bounded Distortion Mapping Spaces, Lipman 2012
Bijective Parameterization with Free Boundaries, Smith & Schaefer 2015
SLIM, Rabinovich et al. 2017

NeuTex, Xiang et al. 2021
NUVO, Srinivasan et al. 2023
SeamGPT, Li et al. 2025
PartUV, Wang et al. 2026

* Not meant to be treated as a comprehensive survey. Important works/subproblems omitted due to presenter's ignorance and commitment to a clean, overlap-free narrative.

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

$$M = (\mathbf{V}, \mathbf{E}, \mathbf{F})$$

Mathematical Foundations

A **mesh** is a discretization of a smooth surface into a set of vertices, edges, and faces.

$$M = (\mathbf{V}, \mathbf{E}, \mathbf{F})$$

$$\mathbf{V} \subset \mathbb{R}^3$$

$$\mathbf{E} = \{(i, j) \mid \text{vertex i is connected to vertex j}\}$$

$$\mathbf{F} = \{(i, j, k) \mid \text{vertices i,j,k share a face}\}$$

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

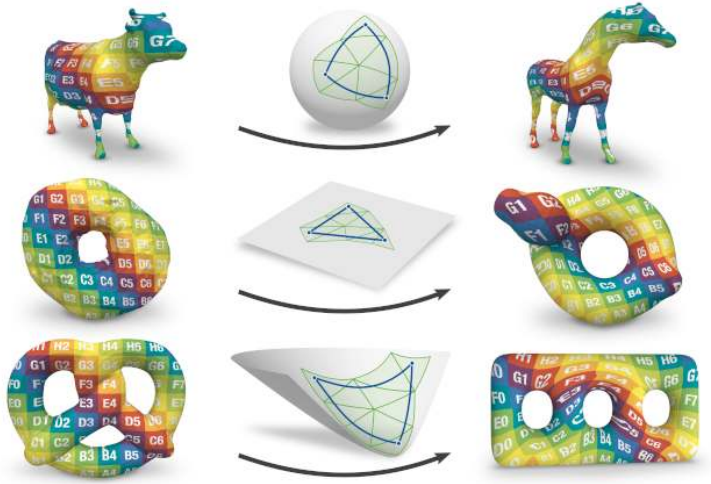
$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

Classic papers define F^{-1} as the parameterization but I find this direction to be more intuitive.

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$



Ω often in 2D but can really be anything!

Source

Mathematical Foundations

A mesh **parameterization** is defined as a function mapping from vertices to Ω

$$F : \mathbf{V} \subset \mathbb{R}^3 \rightarrow \Omega \subset \mathbb{R}^2$$

In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$

Mathematical Foundations

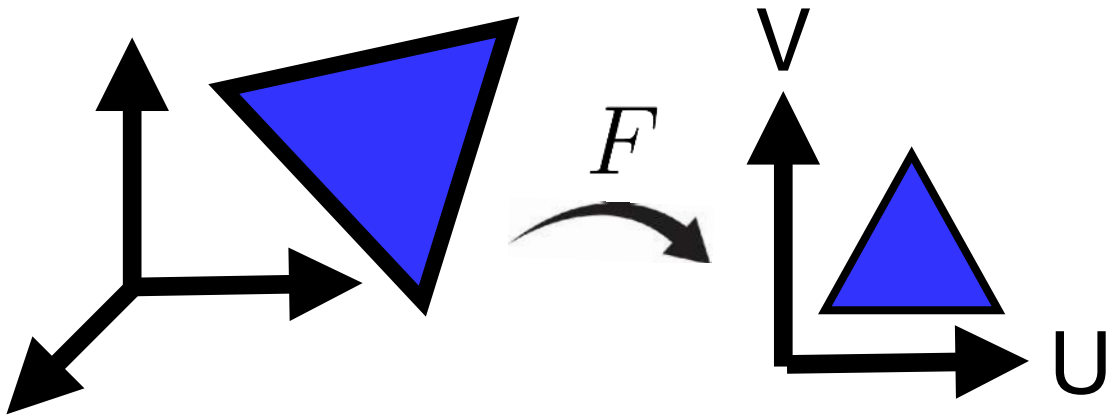
In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$

Mathematical Foundations

In the discrete case, we assign a 2D coordinate to every vertex

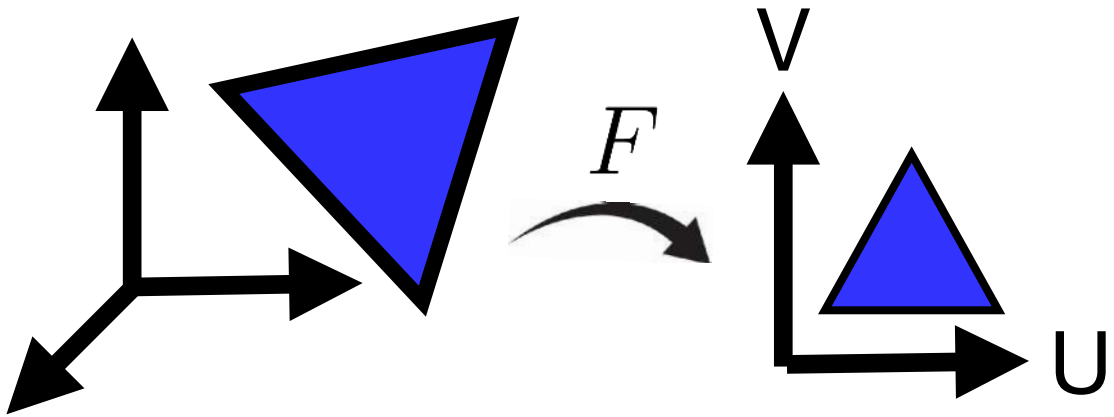
$$F(\mathbf{x}) = (u_i, v_i)$$



Mathematical Foundations

In the discrete case, we assign a 2D coordinate to every vertex

$$F(\mathbf{x}) = (u_i, v_i)$$



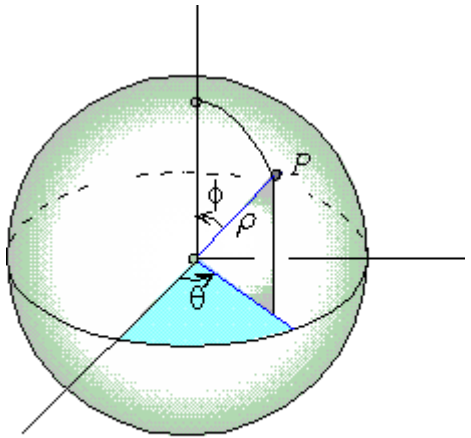
2D coordinates commonly referred by “U” and “V” axes, hence the colloquialism **UV coordinates** or **UVs**

Common Parameterization: Spherical Coordinates

For a sphere with radius R, we have the parameterization

$$U(x, y, z) = \left(\arctan\left(\frac{y}{x}\right), \arccos\left(\frac{z}{R}\right)\right) = (\theta, \phi)$$

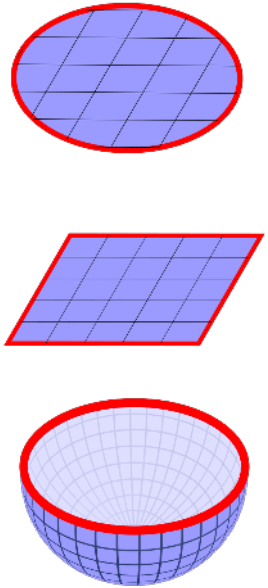
$$0 \leq \theta < 2\pi$$
$$0 \leq \phi < \pi$$



Source

Disk Topology Assumption

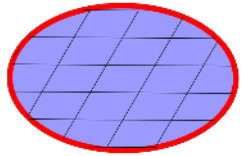
To compute the parameterization, we need to assume our surface is **disk topology.**

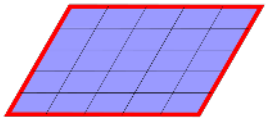


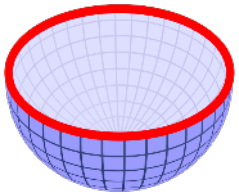
Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology.**

Disk topology: surface with 1 boundary loop.







Source

Disk Topology Assumption

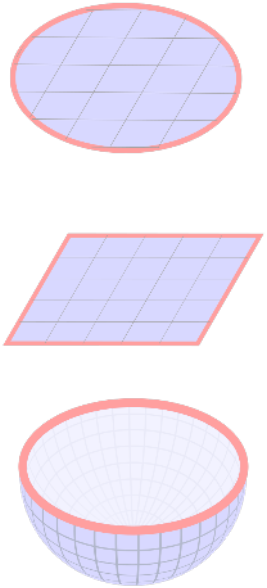
To compute the parameterization, we need to assume our surface is **disk**

Disk topology

Topology 101

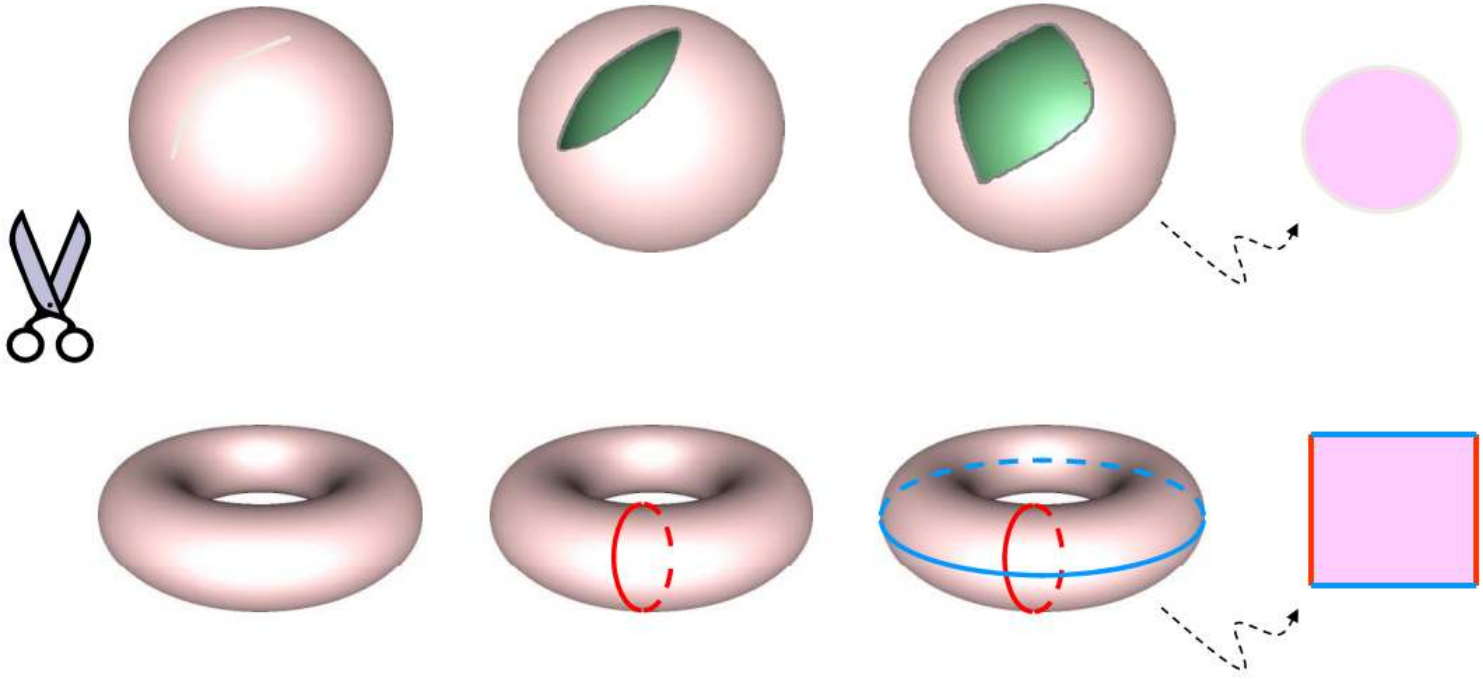
bijective map (homeomorphism) to the 2D plane.

We can always cut the mesh such that it becomes disk topology!



Source

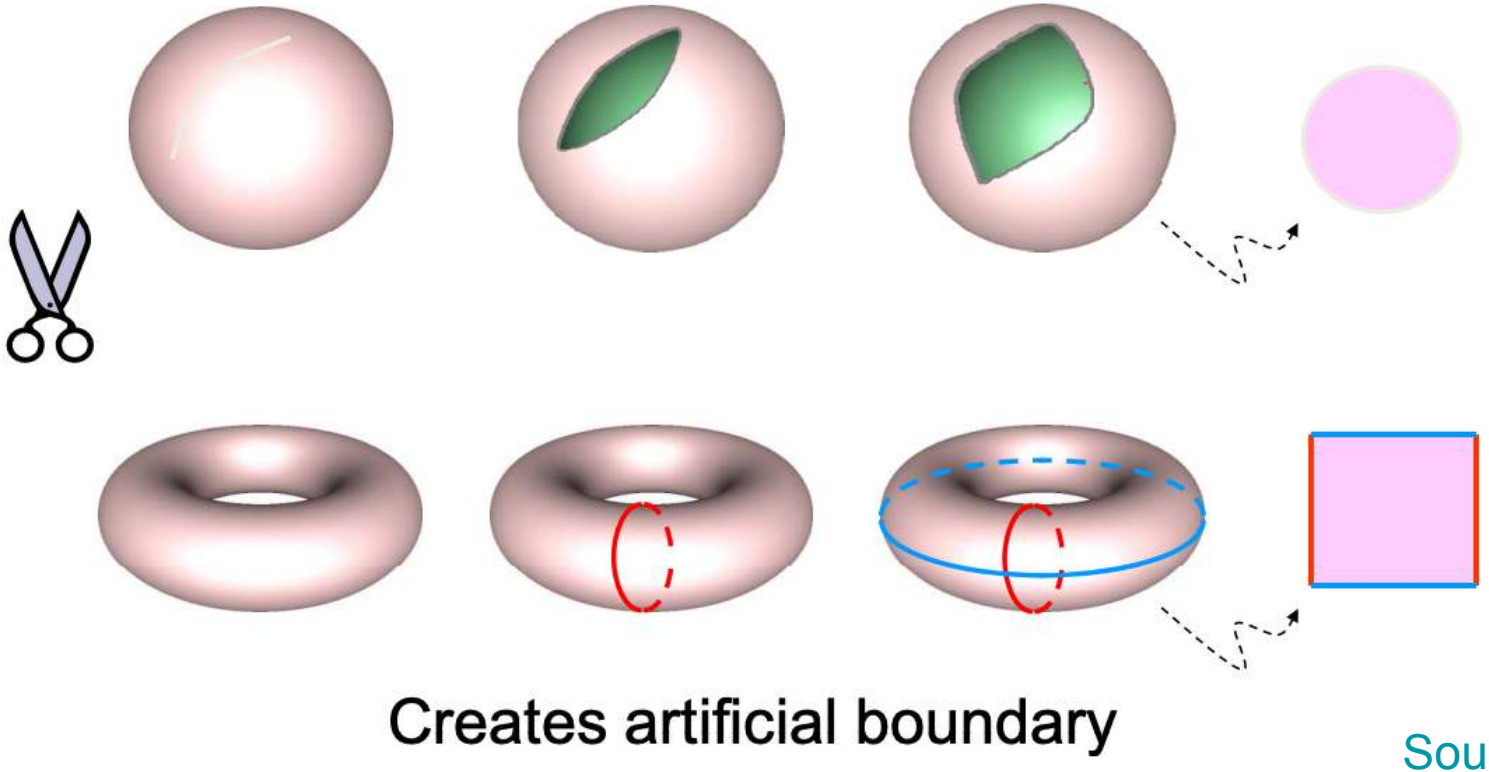
Disk Topology Assumption



Source

Disk Topology Assumption

The problem with seams is they create **discontinuities**.



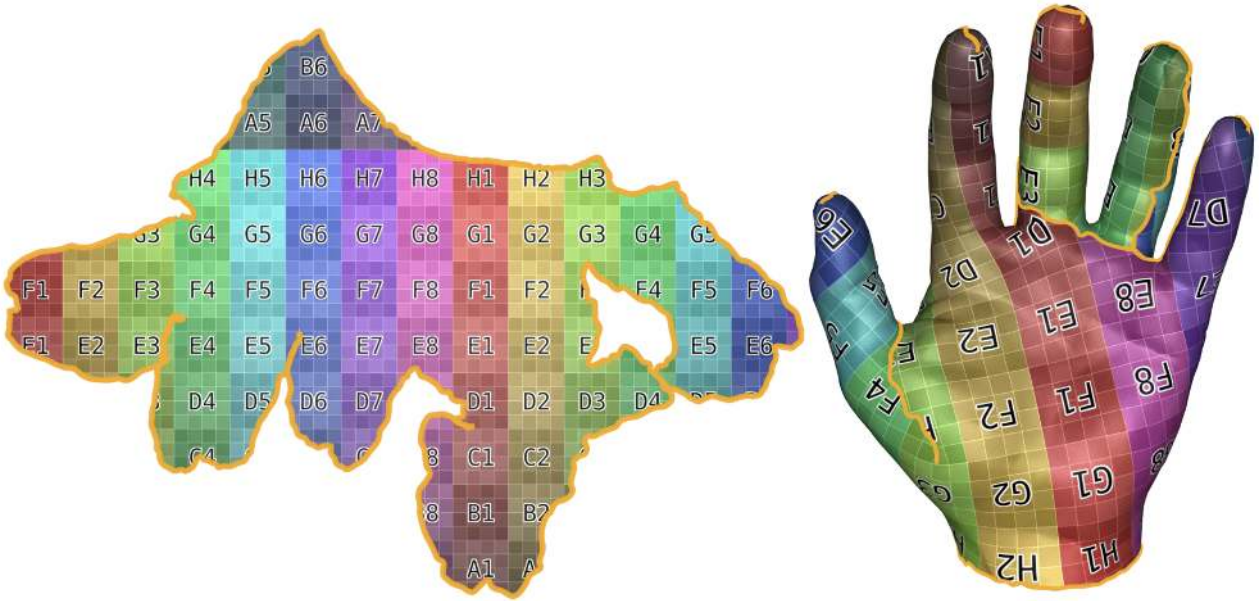
Source

Disk Topology Assumption

The problem with seams is they create **discontinuities**.

Disk Topology Assumption

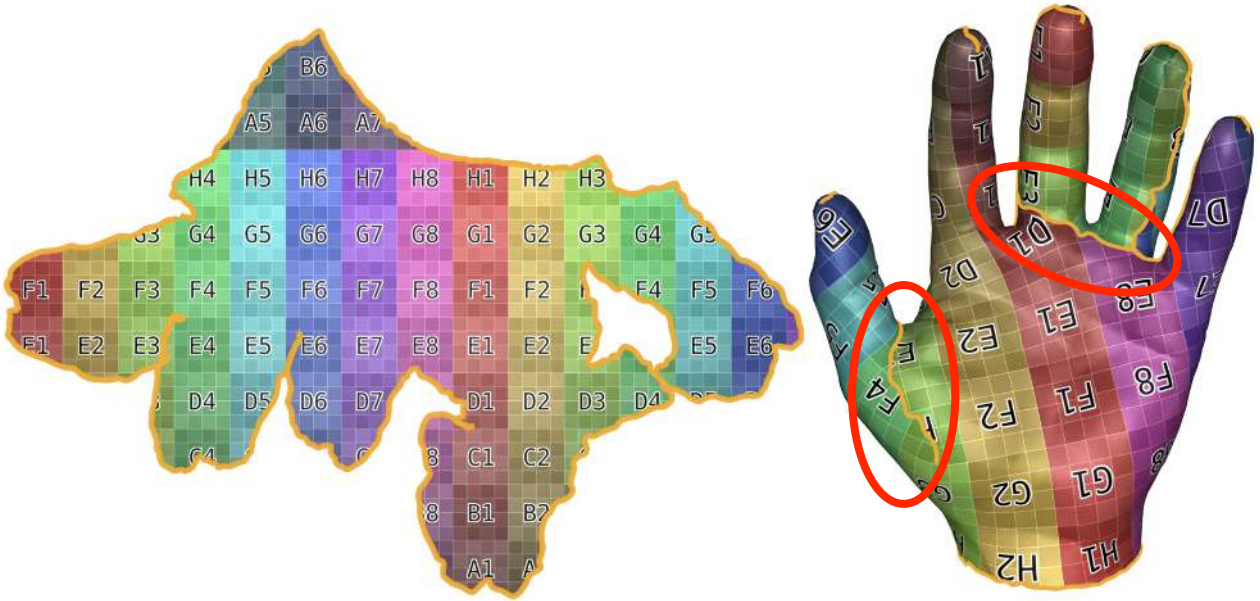
The problem with seams is they create **discontinuities**.



Source

Disk Topology Assumption

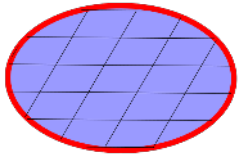
The problem with seams is they create **discontinuities**.



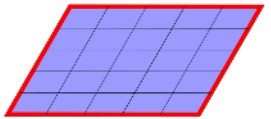
Source

Disk Topology Assumption

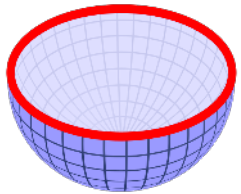
To compute the parameterization, we need to assume our surface is **disk topology.**



Disk topology: surface with 1 boundary loop.



Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



Source

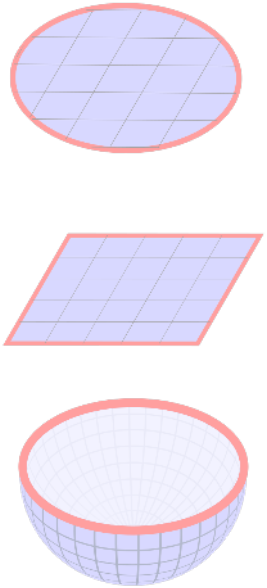
Disk Topology Assumption

To compute the parameterization, we need to assume our surface is **disk topology.**

Disk topology: surface with 1 boundary loop

Not continuous **across** disk boundary

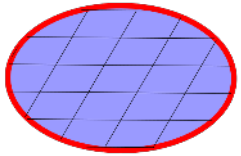
Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



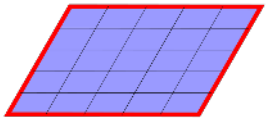
Source

Disk Topology Assumption

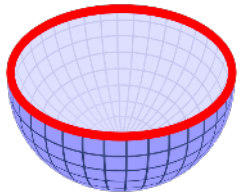
To compute the parameterization, we need to assume our surface is **disk topology.**



Disk topology: surface with 1 boundary loop.



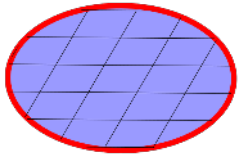
Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



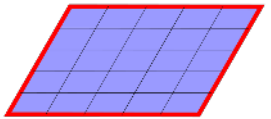
Source

Disk Topology Assumption

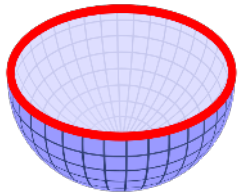
To compute the parameterization, we need to assume our surface is **disk topology.**



Disk topology: surface with 1 boundary loop.

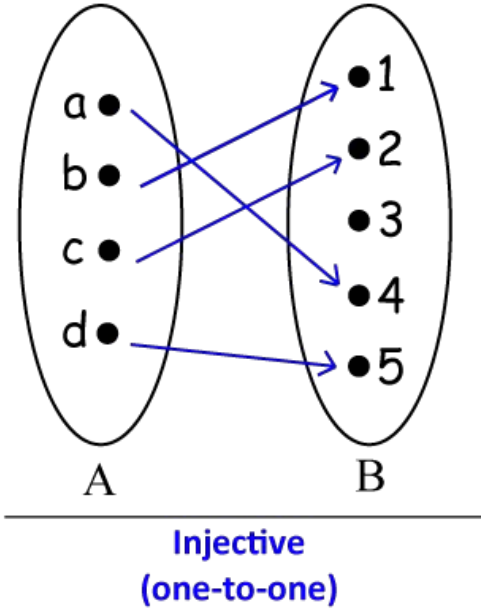


Topology 101. Disk topology allows for **continuous** and **bijective** map (homeomorphism) to the 2D plane.



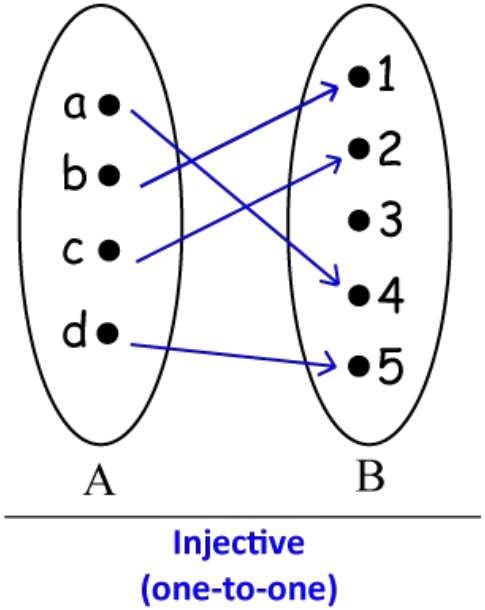
Source

Bijectivity



Source

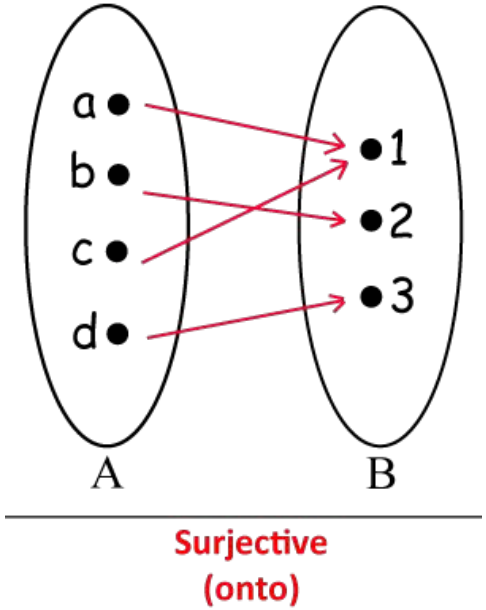
Bijectivity



Source

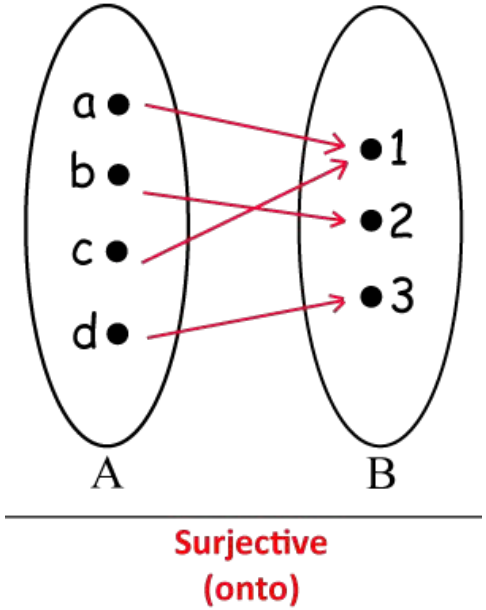
Every element of A maps to a **unique** element of B

Bijectivity



Source

Bijectivity

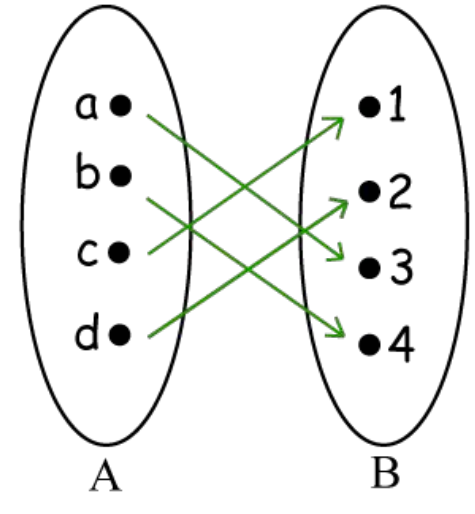


Surjective
(onto)

Source

Every element of B has **at least one** corresponding element of A

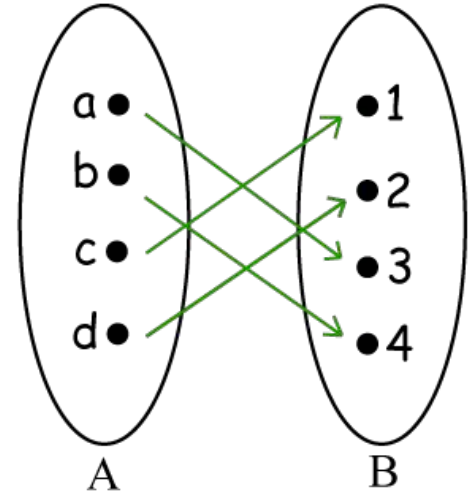
Bijectivity



Bijective
(one-to-one and onto)

Source

Bijectivity



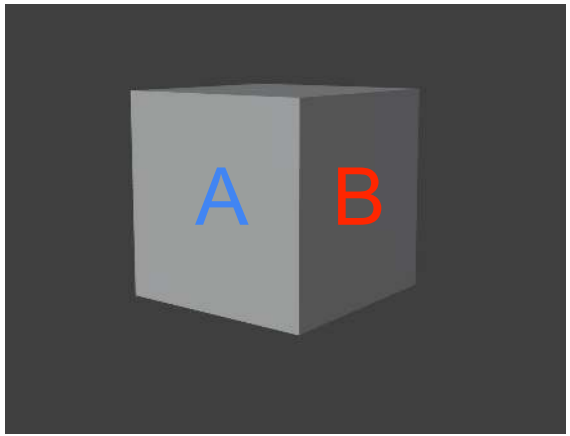
Bijective
(one-to-one and onto)

Source

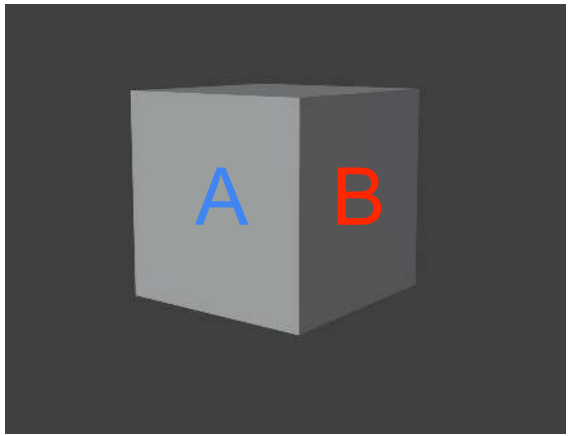
Both injective and surjective.

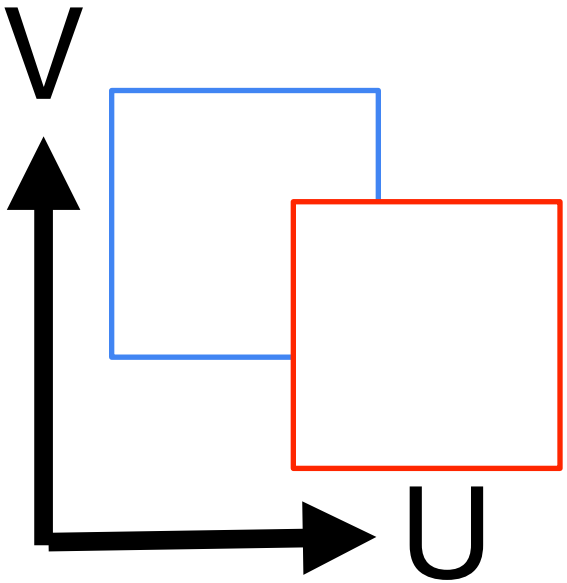
Exactly matched pairs between A and B.

Why Bijectivity (Injectivity) Matters

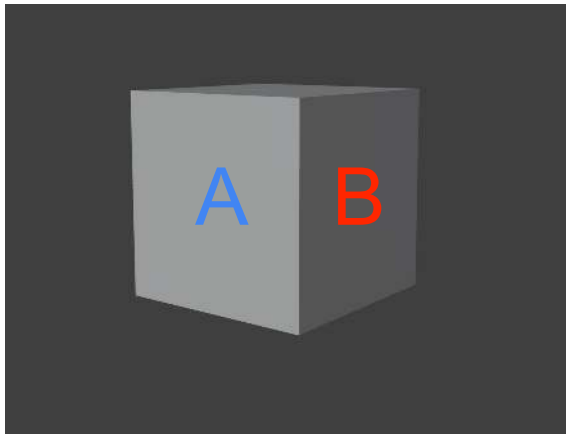


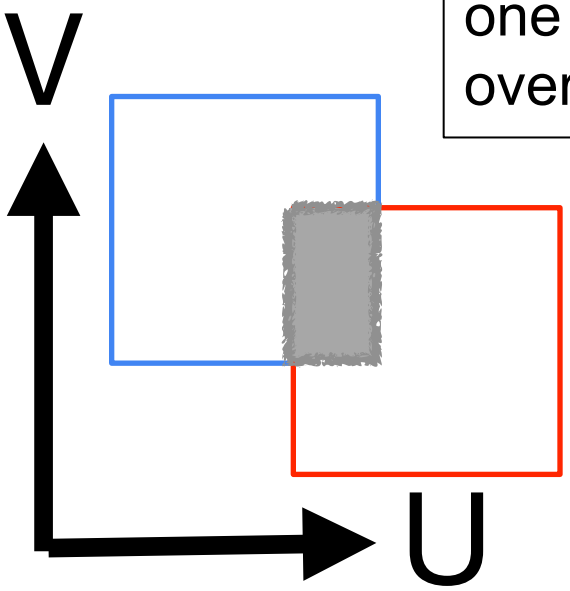
Why Bijectivity (Injectivity) Matters





Why Bijectivity (Injectivity) Matters



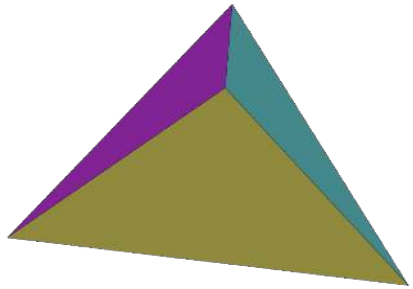


Can only assign one color to the overlap.

Part 2: Fixed-Boundary Parameterization

Fixed-Boundary Parameterization

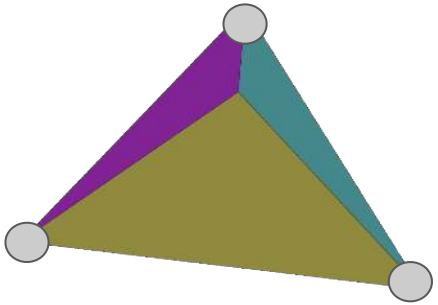
Treat mesh as a **mass-spring system**.



Fixed-Boundary Parameterization

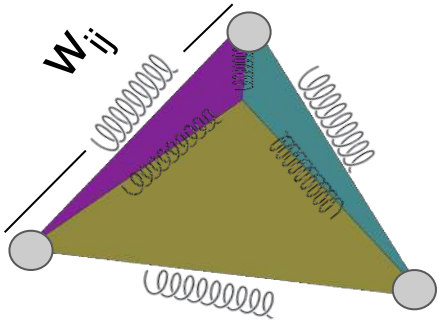
Treat mesh as a **mass-spring system**.

Mesh vertices: nodes with unit mass



Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.

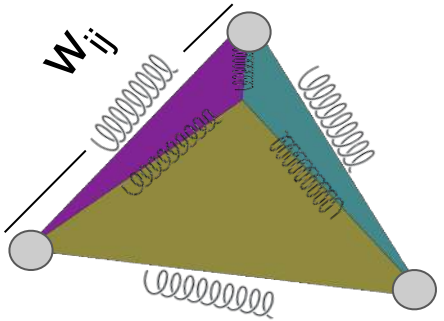


Mesh vertices: nodes with unit mass

Mesh edges: springs with stiffness constant w_{ij} with zero rest length

Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.



Mesh vertices: nodes with unit mass

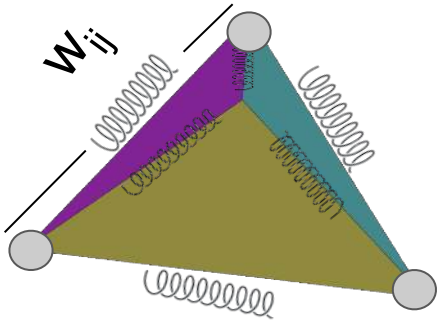
Mesh edges: springs with stiffness constant w_{ij} with zero rest length

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

Fixed-Boundary Parameterization

Treat mesh as a **mass-spring system**.



Mesh vertices: nodes with unit mass

Mesh edges: springs with stiffness constant w_{ij} with zero rest length

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

There's a trivial solution to this system. Anyone care to take a guess?

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

There's a trivial solution to this system. Anyone care to take a guess?

Set F = constant everywhere

Fixed-Boundary Parameterization

Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

There's a trivial solution to this system. Anyone care to take a guess?

Set F = constant everywhere

We need boundary conditions.

Fixed-Boundary Parameterization

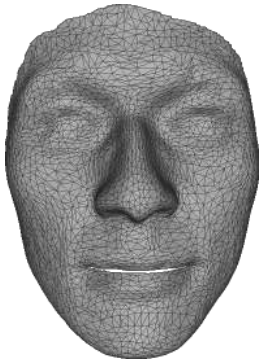
Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

There's a trivial solution to this system. Anyone care to take a guess?

Set F = constant everywhere



We need boundary conditions.

Pinning the mesh boundary is a natural choice!

Fixed-Boundary Parameterization

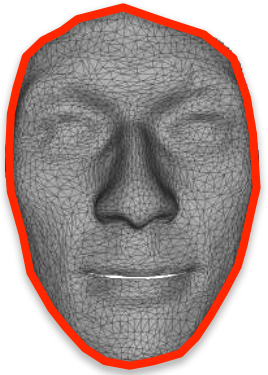
Optimization problem: minimize potential energy of system

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

"Make every adjacent vertex as close to each other as possible"

There's a trivial solution to this system. Anyone care to take a guess?

Set F = constant everywhere



We need boundary conditions.

Pinning the mesh boundary is a natural choice!

Fixed-Boundary Parameterization

Two key design choices to make

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

Fixed-Boundary Parameterization

Two key design choices to make

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

1. Where to place mesh boundary

Fixed-Boundary Parameterization

Two key design choices to make

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

1. Where to place mesh boundary
2. How to define w_{ij}

Fixed-Boundary Parameterization

Two key design choices to make

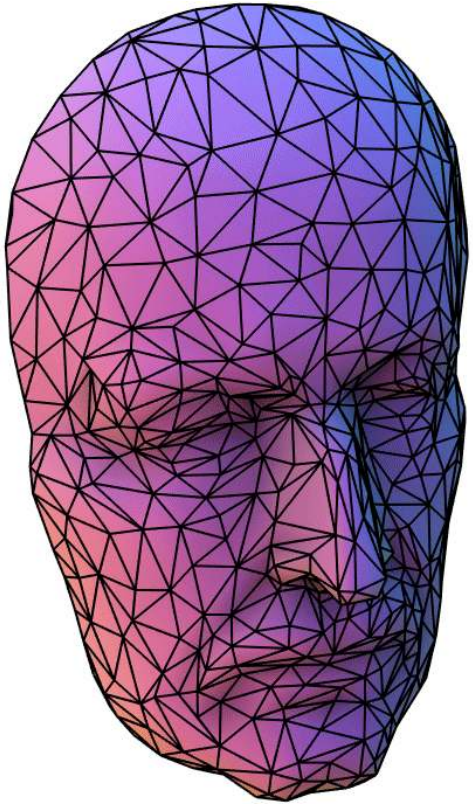
$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F(\mathbf{x_i}) - F(\mathbf{x_j})||^2$$

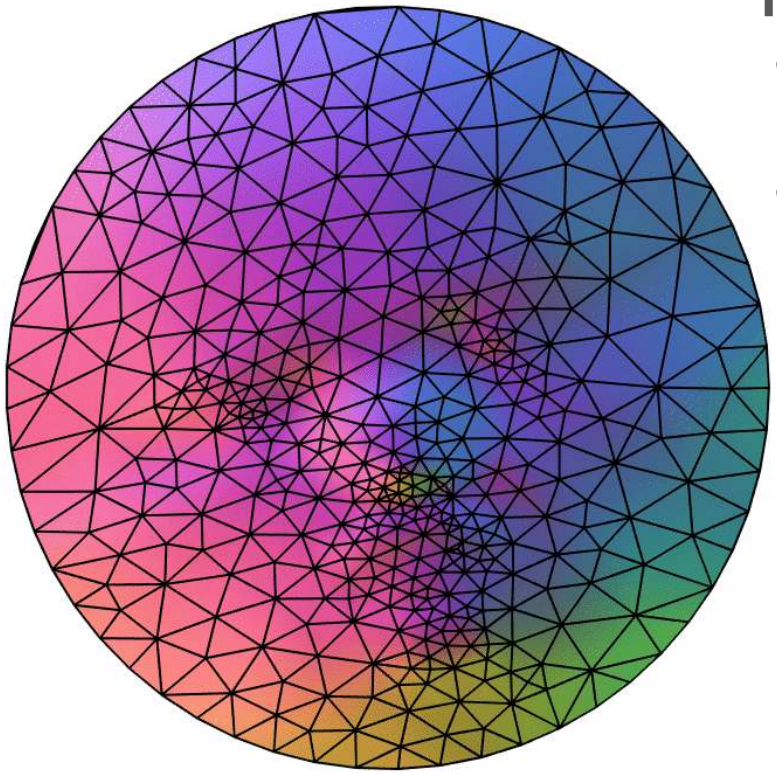
1. Where to place mesh boundary
2. How to define w_{ij}

[**Tutte 1963/Floater 1997**] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.

[**Tutte 1963/Floater 1997**] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.

[**Tutte 1963/Floater 1997**] If the boundary is **convex** and **weights are positive**, then the solution is guaranteed to be injective.





Tutte embedding

- Boundary is mapped to circle
- All weights are 1

Source

Fixed-Boundary Parameterization

How to solve for F?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

Fixed-Boundary Parameterization

How to solve for F?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

$$\frac{\partial E}{\partial F_i} = \sum_{j \in N_i} 2 * w_{ij}(F_i - F_j) = 0$$

N_i indexes all of the neighbors of vertex i.

Fixed-Boundary Parameterization

How to solve for F?

$$E = \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

$$\frac{\partial E}{\partial F_i} = \sum_{j \in N_i} 2 * w_{ij}(F_i - F_j) = 0$$

N_i indexes all of the neighbors of vertex i.

Set boundary vertices and solve for remaining F_i

Fixed-Boundary Parameterization

$$\min_{F} \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

Fixed-Boundary Parameterization

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

In matrix form, this looks like

$$\min_F \frac{1}{2} \mathrm{tr}(F^T L F)$$

Fixed-Boundary Parameterization

$$\min_F \sum_{i,j \in \mathbf{E}} w_{ij} ||F_i - F_j||^2$$

In matrix form, this looks like

$$\min_F \frac{1}{2} \mathrm{tr}(F^T L F)$$

L is the mesh Laplacian! w_{ij} determines what "type" of discrete Laplacian you're using. (recall 013_laplacian)

Fixed-Boundary Parameterization

$$\min \sum_{i,j} w_{ij} ||F_i - F_j||^2$$

In matr

$$\min_F \frac{1}{2} \mathrm{tr}(F^T \boxed{L} F)$$

L is sparse, so the system is sparse.
We will explore the effects of different choice of weights in the exercises.

L is the mesh Laplacian! w_{ij} determines what "type" of discrete Laplacian you're using. (recall 013_laplacian)

Part 3: Computing Parameterization Distortion

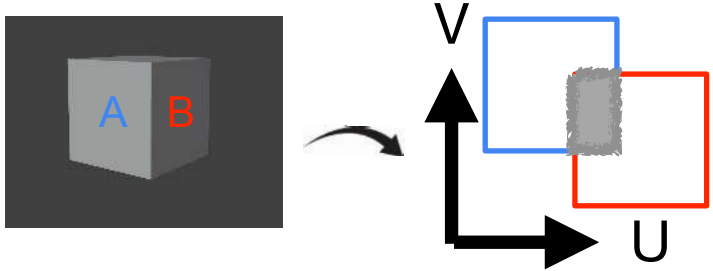
Desiderata Review

Desirable properties of parameterizations

Desiderata Review

Desirable properties of parameterizations

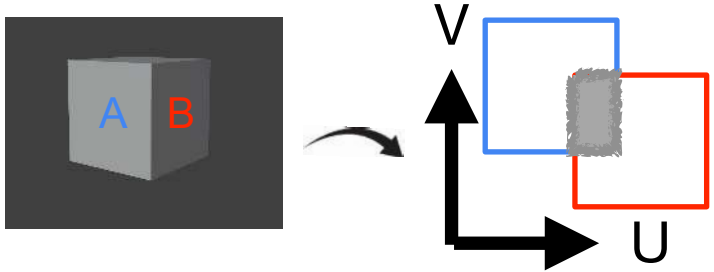
- Injectivity



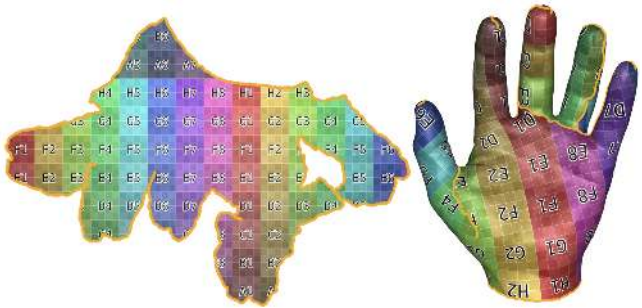
Desiderata Review

Desirable properties of parameterizations

- Injectivity



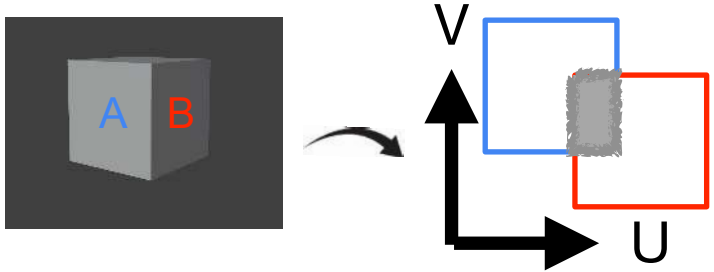
- Seam (discontinuity) minimization



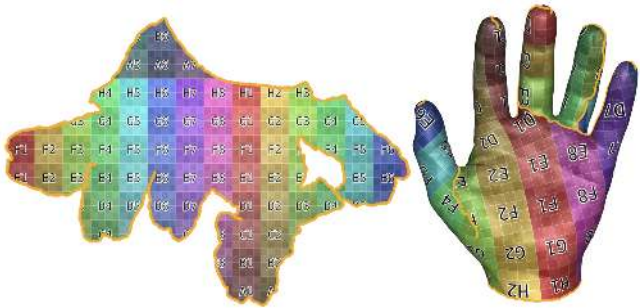
Desiderata Review

Desirable properties of parameterizations

- Injectivity

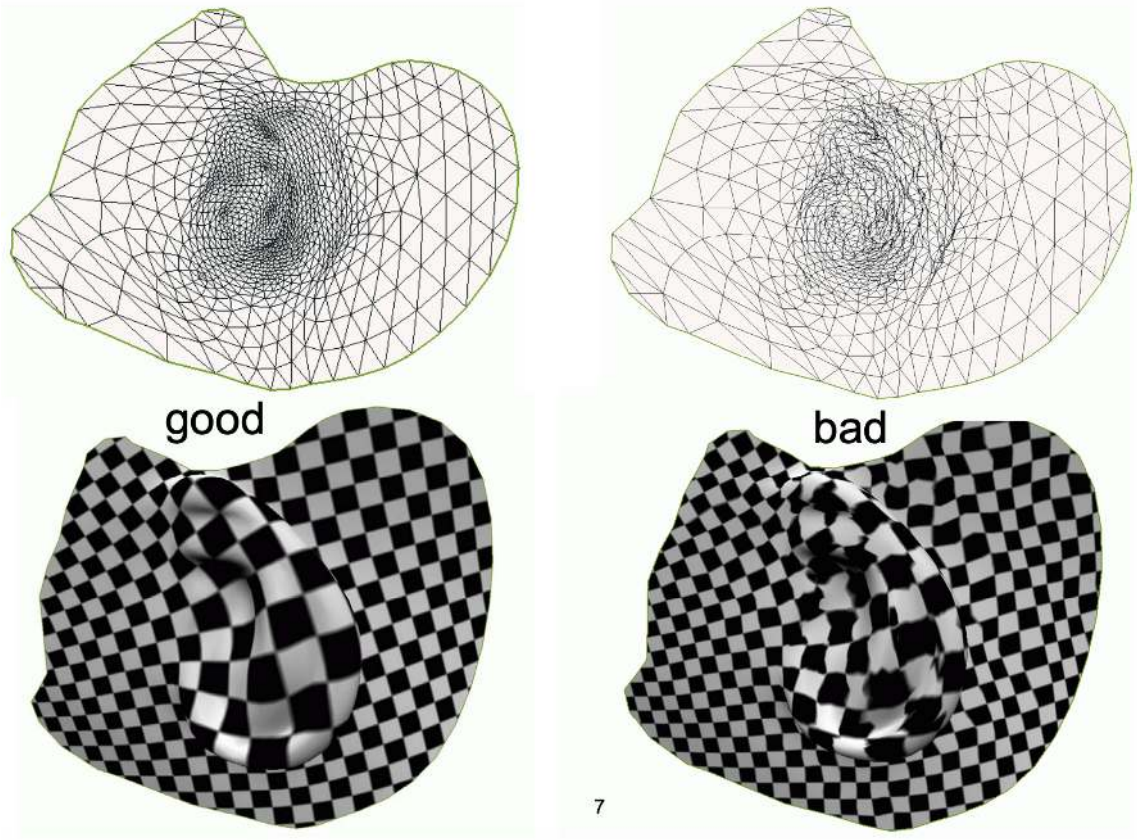


- Seam (discontinuity) minimization



- **Low distortion**

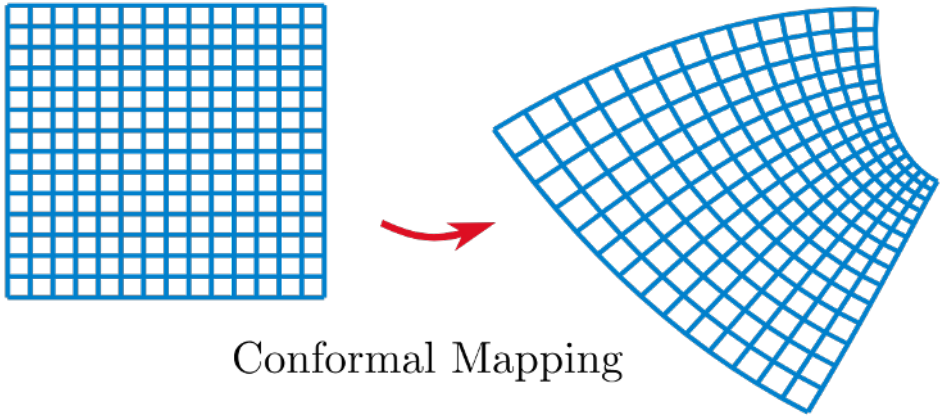
Types of Distortion



Source

Types of Distortion

Conformal: angle distortion



Source

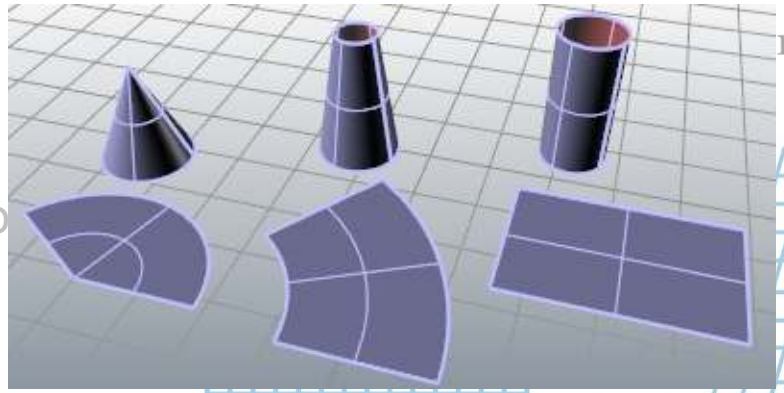
Types of Distortion

Conformal: angle distortion

Isometric maps: both angle and area (length) preserving

ng

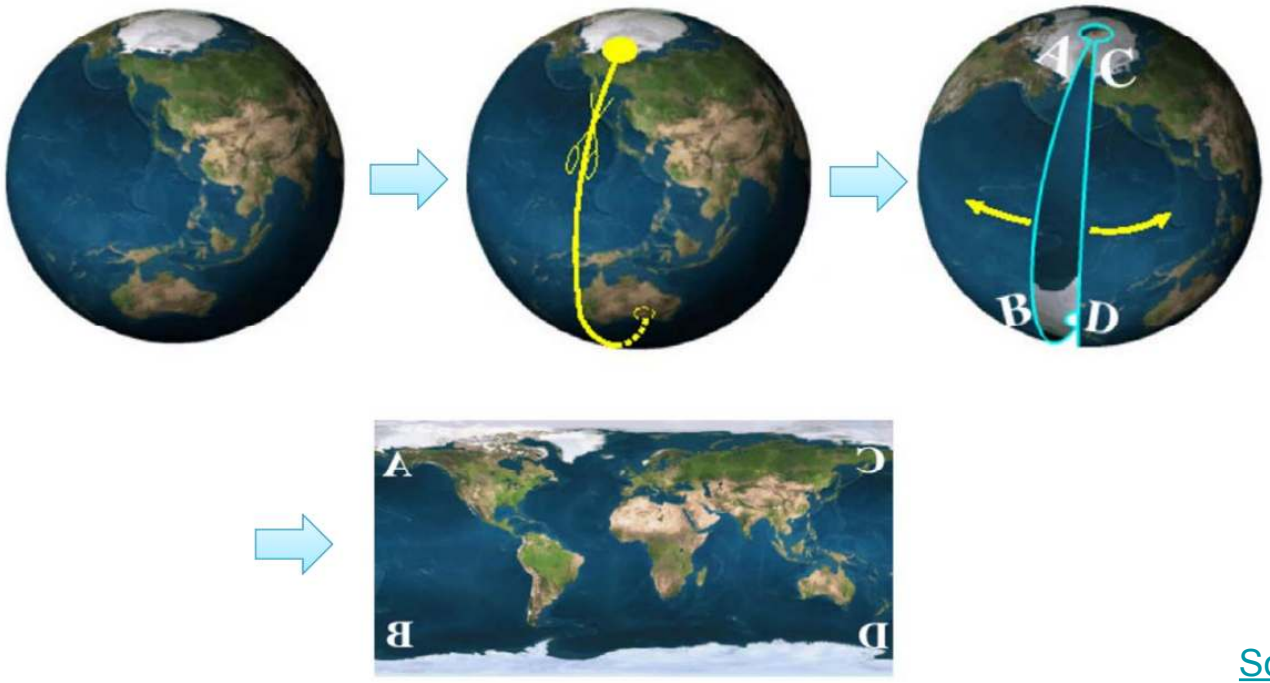
Authalic: area distortio



Authalic Mapping

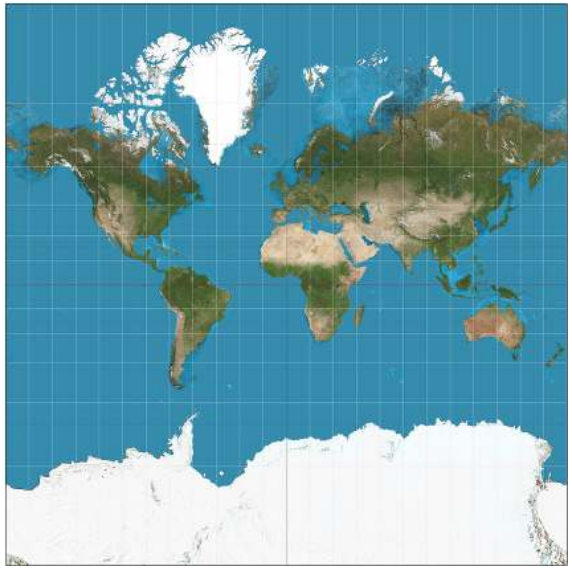
Source

Distortion: Motivations in Cartography



Source

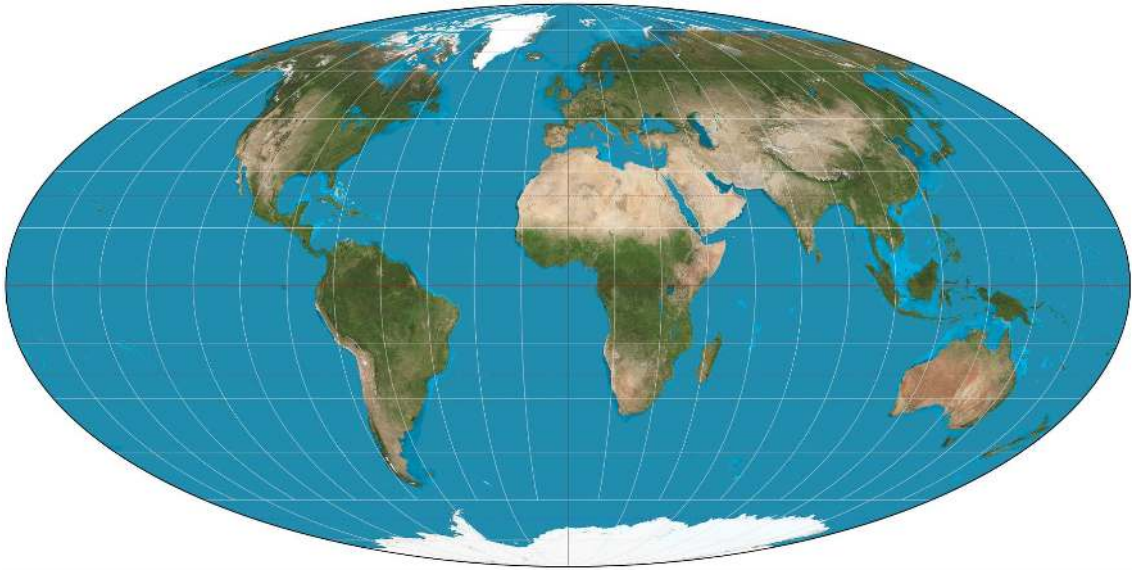
Distortion: Motivations in Cartography



Mercator projection
(conformal)

Source

Distortion: Motivations in Cartography



Mollweide projection
(equiareal/authalic)

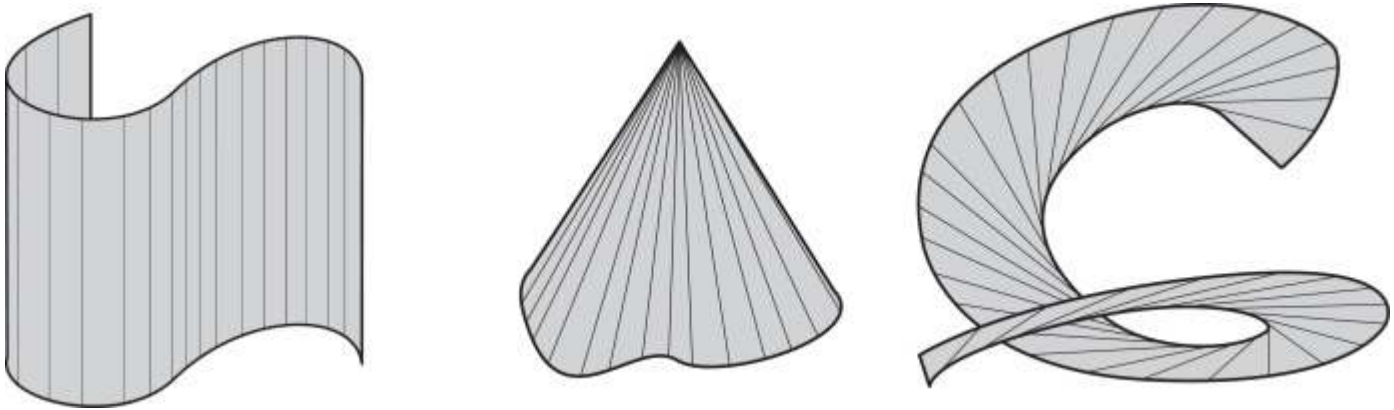
Distortion: Motivations in Cartography

Why are there no isometric (distortionless) maps of the world?

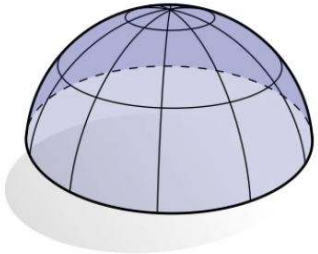
Distortion: Motivations in Cartography

Why are there no isometric (distortionless) maps of the world?

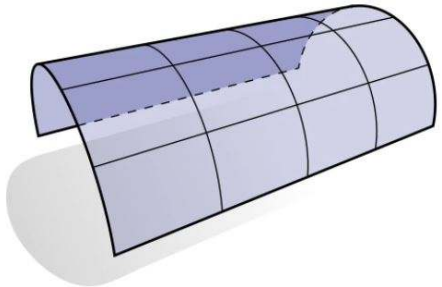
Theorema Egregium (Gauss 1827) Only developable (zero Gaussian curvature) surfaces can be isometrically flattened to the plane.



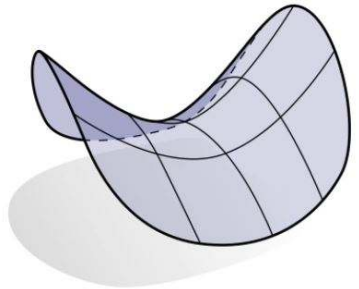
Gaussian Curvature



$K > 0$



$K = 0$



$K < 0$

Source

Sphere has positive gaussian curvature everywhere

Measuring Distortion

Ok if most parameterizations create distortion how do we measure it?

Measuring Distortion

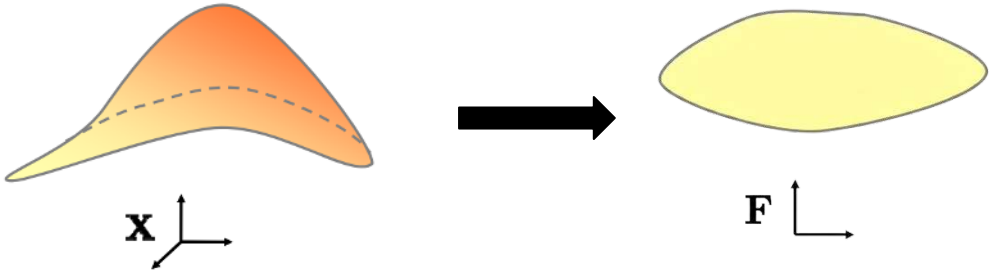
Ok if most parameterizations create distortion how do we measure it?

The Jacobian tells you everything about distortion.

Measuring Distortion

Ok if most parameterizations create distortion how do we measure it?

The Jacobian tells you everything about distortion.



Source

$$F(x,y,z) = \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \end{pmatrix} \qquad J_F = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \end{pmatrix}$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

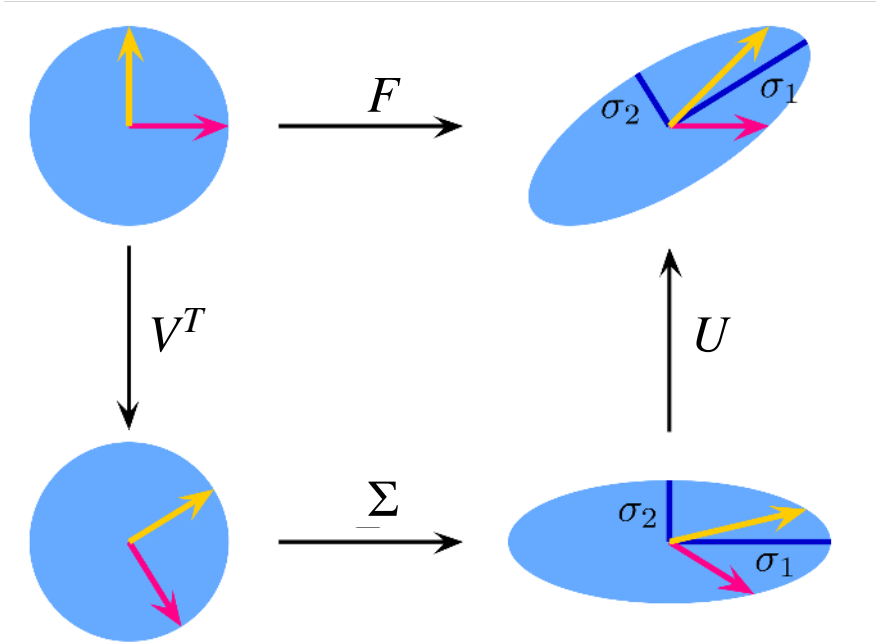
$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

Measuring Distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

Measuring Distortion

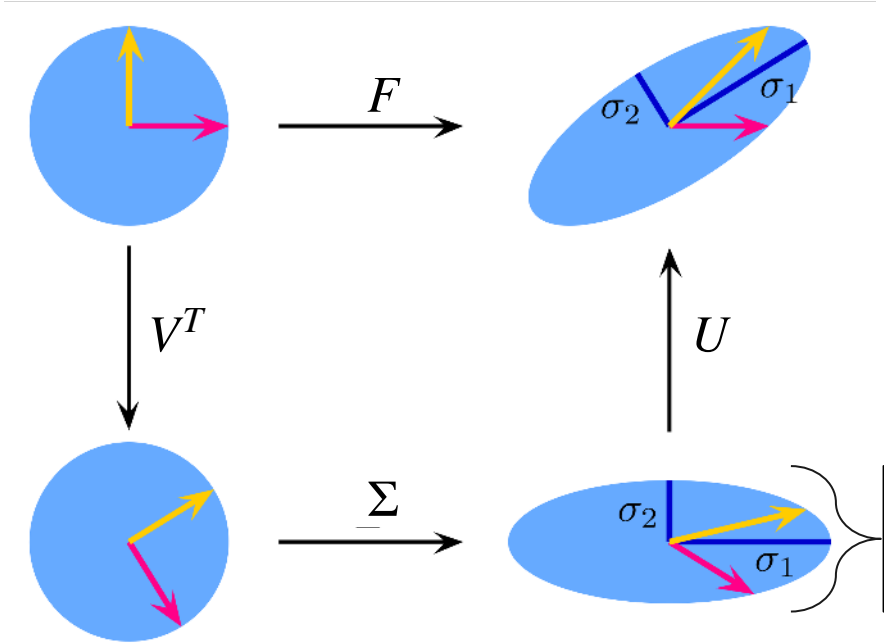
$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$



Source

Measuring Distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$



Source

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

$$f \text{ is } \textit{equiareal} \text{ or } \textit{area-preserving} \iff \sigma_1\sigma_2 = 1$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

$$f \text{ is } \textit{equiareal} \text{ or } \textit{area-preserving} \iff \sigma_1\sigma_2 = 1$$

$$f \text{ is } \textit{conformal} \text{ or } \textit{angle-preserving} \iff \sigma_1 = \sigma_2$$

Measuring Distortion

Specifically, the **singular values** of the Jacobian tell you everything about distortion

$$J_F = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} V^T$$

$$f \text{ is } \textit{equiareal} \text{ or } \textit{area-preserving} \iff \sigma_1\sigma_2 = 1$$

$$f \text{ is } \textit{conformal} \text{ or } \textit{angle-preserving} \iff \sigma_1 = \sigma_2$$

$$f \text{ is } \textit{isometric} \text{ or } \textit{length-preserving} \iff \sigma_1 = \sigma_2 = 1$$

Measuring Distortion

Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F|x|V|}$ (recall 012_gradient)

Measuring Distortion

Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F|x|V|}$ (recall 012_gradient)

$$\mathbf{G}F = J_F \in \mathbb{R}^{3|F|\times 2}$$

Measuring Distortion

Practically, you can find the Jacobian of the UV map function F using the mesh gradient operator $\mathbf{G} \in \mathbb{R}^{3|F| x |V|}$ (recall 012_gradient)

$$\mathbf{G}F = J_F \in \mathbb{R}^{3|F| \times 2}$$

Jacobian is a 2x3 matrix per triangle

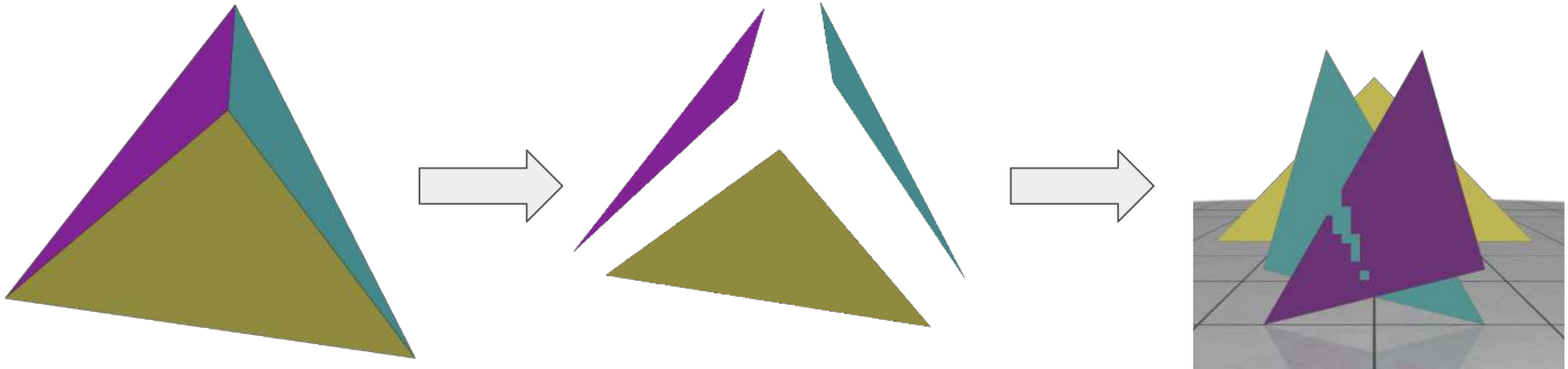
Afternoon Teaser

If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.

Afternoon Teaser

If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.

Maximizes discontinuities, so typically a bad solution.

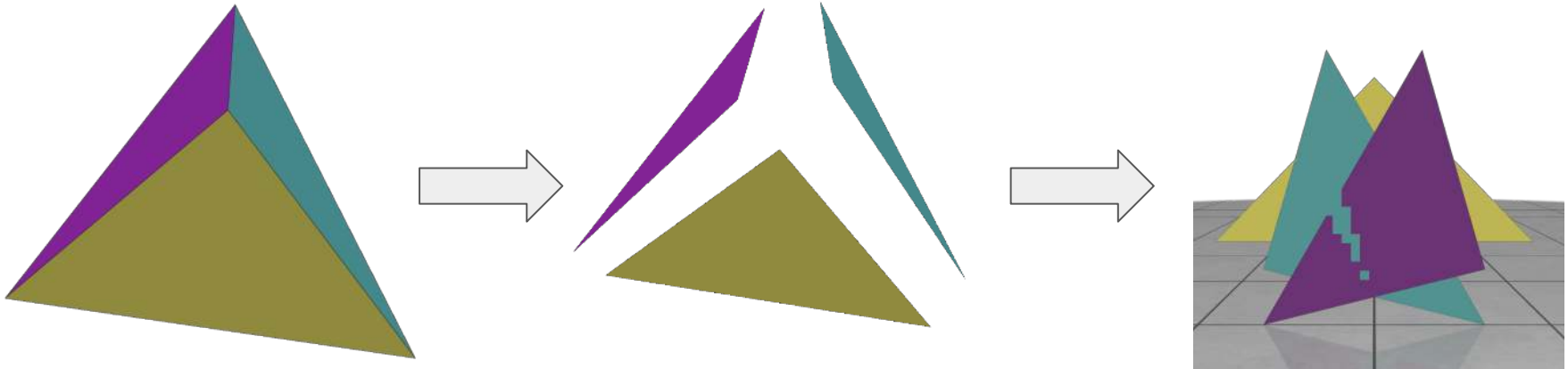


Afternoon Teaser

If we split the mesh up into individual triangles (triangle soup) we can trivially obtain a distortionless parameterization.

Maximizes discontinuities, so typically a bad solution.

Dale will show how we can bypass this discontinuity problem with neural networks (stay tuned!)



Exercises!

Make sure you have the latest version of the repo

```
(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull
Already up to date.
```

Exercises!

Make sure you have the latest version of the repo

```
(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull
Already up to date.
```

101_parameterization_fundamentals.ipynb: step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

Exercises!

Make sure you have the latest version of the repo

```
(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull
Already up to date.
```

101_parameterization_fundamentals.ipynb: step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

102_parameterization_exercises.ipynb: will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

Exercises!

Make sure you have the latest version of the repo

```
(sgi) guanzhi@MacBookPro sgi-introduction-course % git pull
Already up to date.
```

101_parameterization_fundamentals.ipynb: step-by-step walkthrough on how to compute, visualize, and analyze distortion for parameterizations of some simple meshes.

102_parameterization_exercises.ipynb: will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

Solution code for reference is in 102_solution.ipynb

Exercises!

Make sure you have the latest version of the repo

```
[(sgi) guanzh
 Already up t
```

Lots of stuff but most of it is copy-paste work. The goal is exploratory: skip around as you please and try to build some intuition. See if you can parameterize some of your own meshes!

101_paramete[illegible]n on how to compute, visu[illegible]ome simple meshes.

102_parameterization_exercises.ipynb: will ask you to apply these same computations on some more complex meshes. And try your hand at some free-boundary methods!

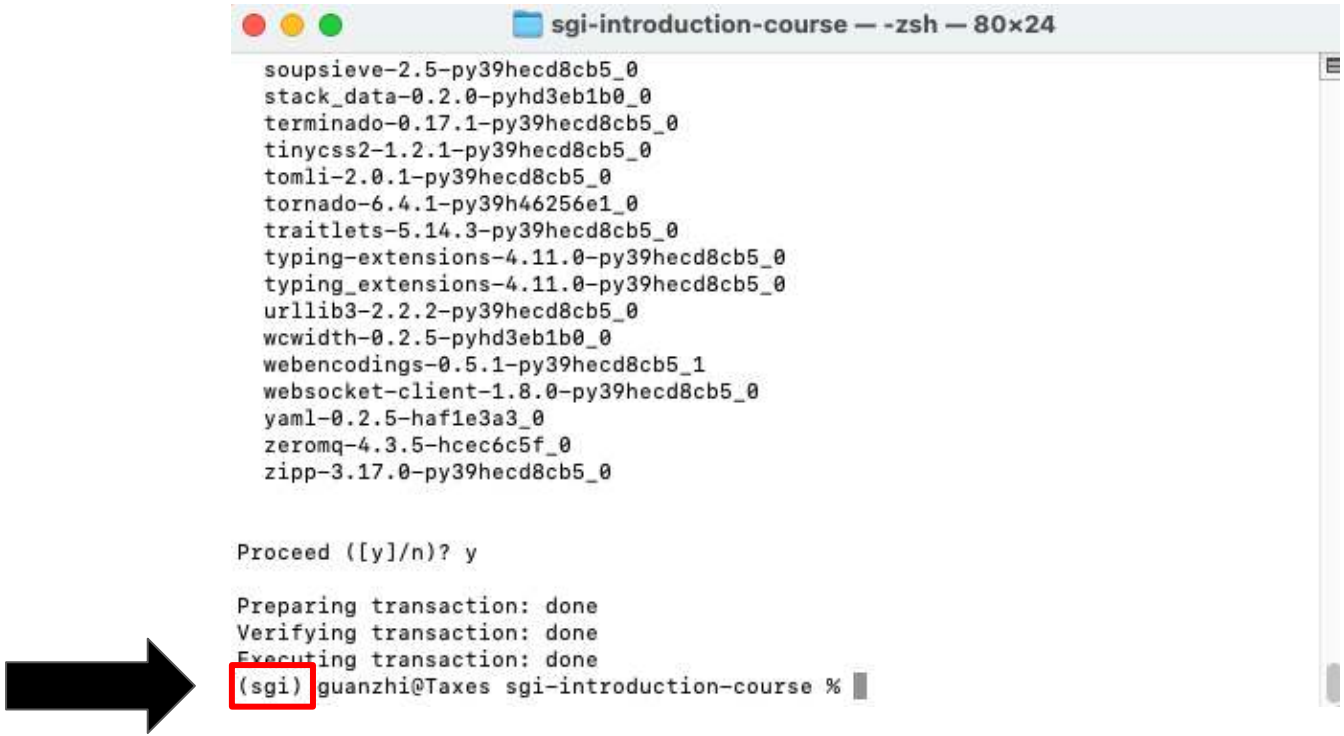
Solution code for reference is in 102_solution.ipynb

Environment Setup

Feel free to skip / check out if you are already comfortable with notebook coding.

Environment Setup

1) Make sure your conda/python virtual environment for SGI is activated

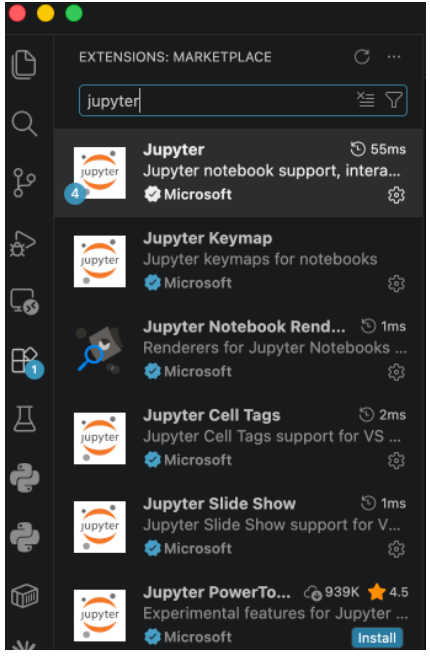


Environment Setup

2) pip install jupyterlab

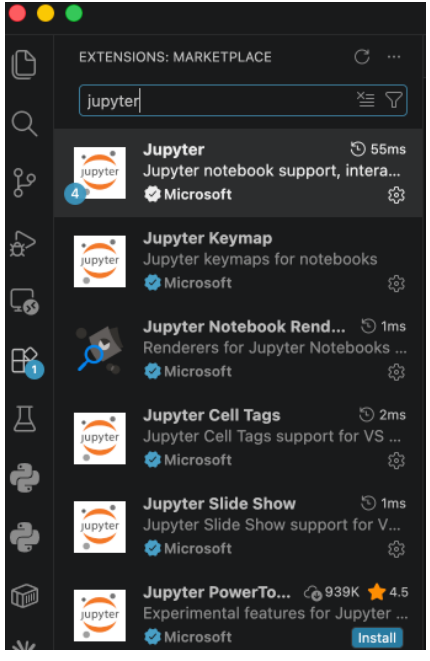
Environment Setup

2.5) If you are on VSCode / some other IDE which has native support for notebook, then can install the Jupyter extension and open the file directly in IDE.

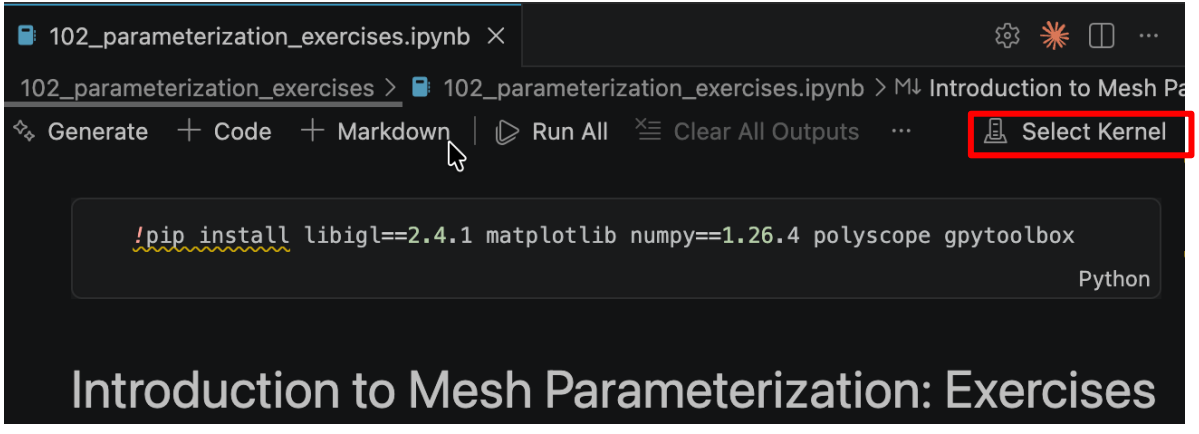


Environment Setup

2.5) If you are on VSCode / some other IDE which has native support for notebook, then can install the Jupyter extension and open the file directly in IDE.

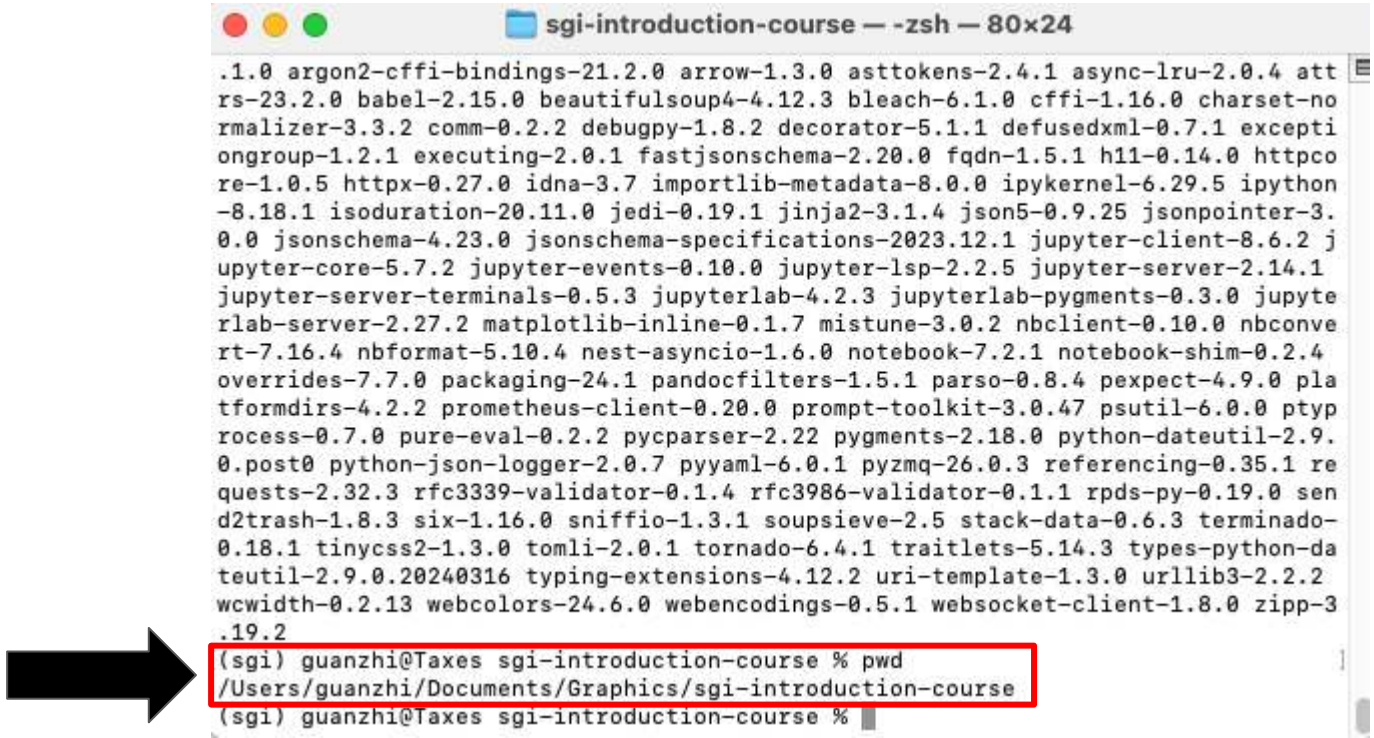


Select the sgi environment when prompted. May request to install ipykernel if not already installed.



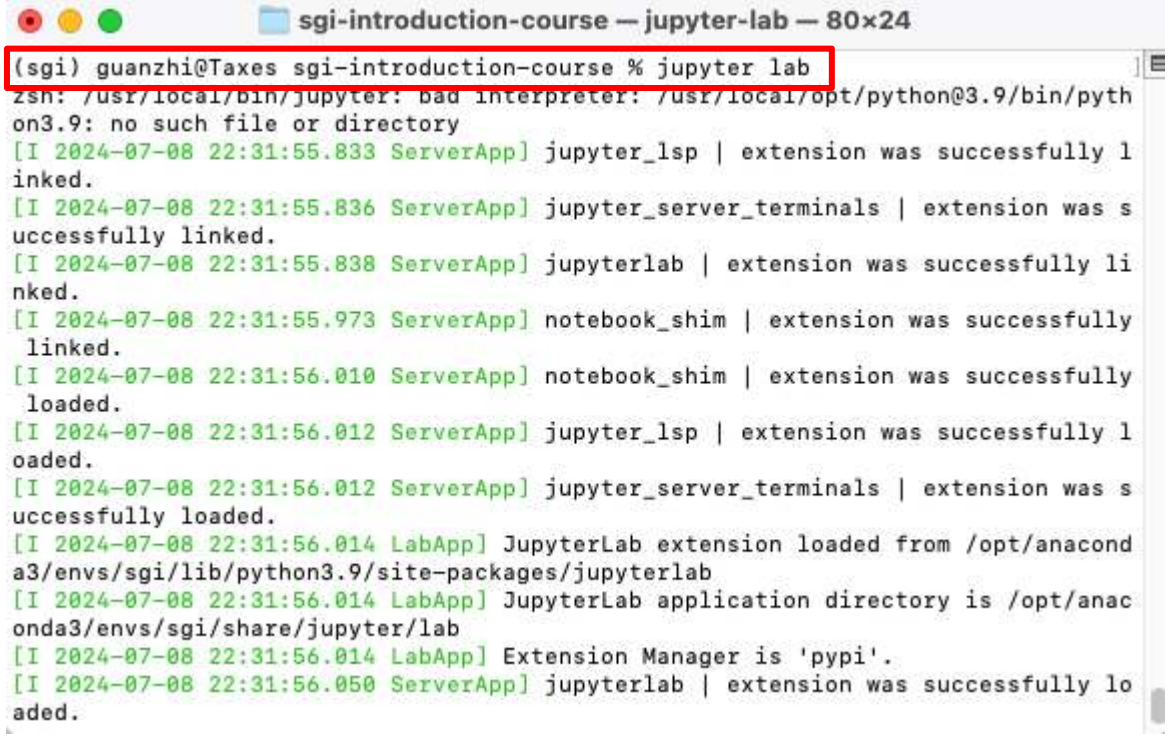
Environment Setup

3) Make sure you are in the sgi-introduction-course directory



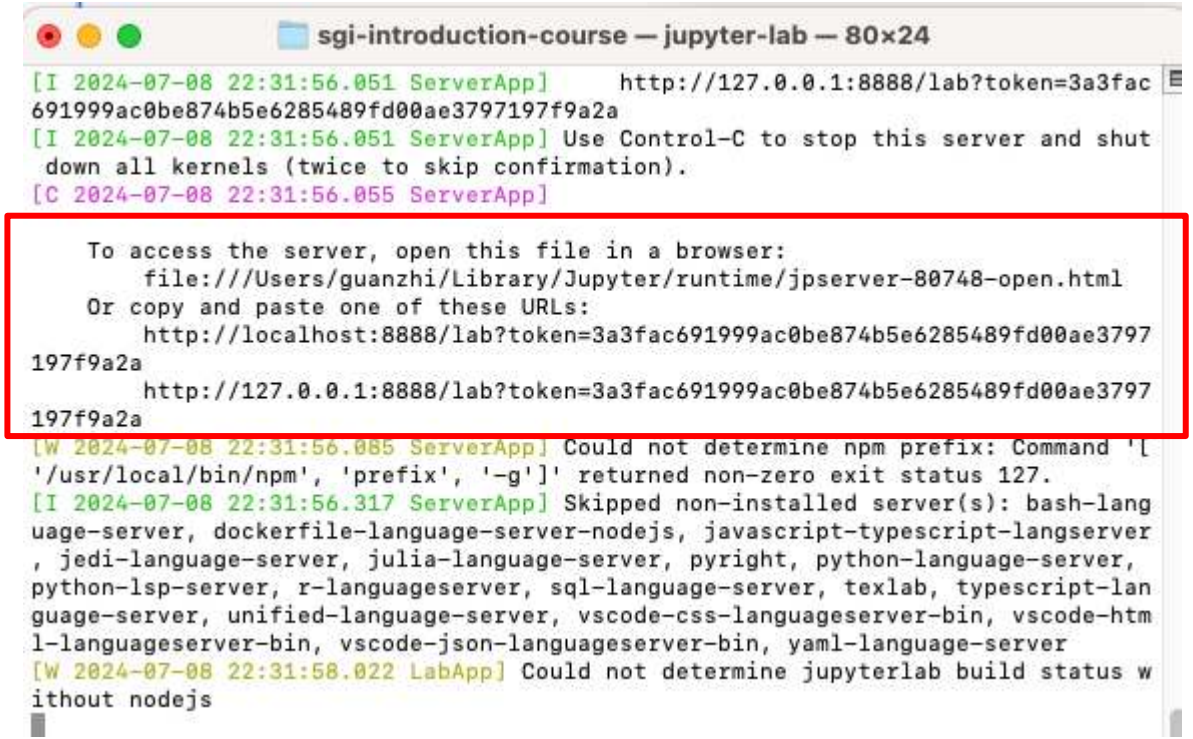
Environment Setup

4) Call `jupyter lab` to launch the local server



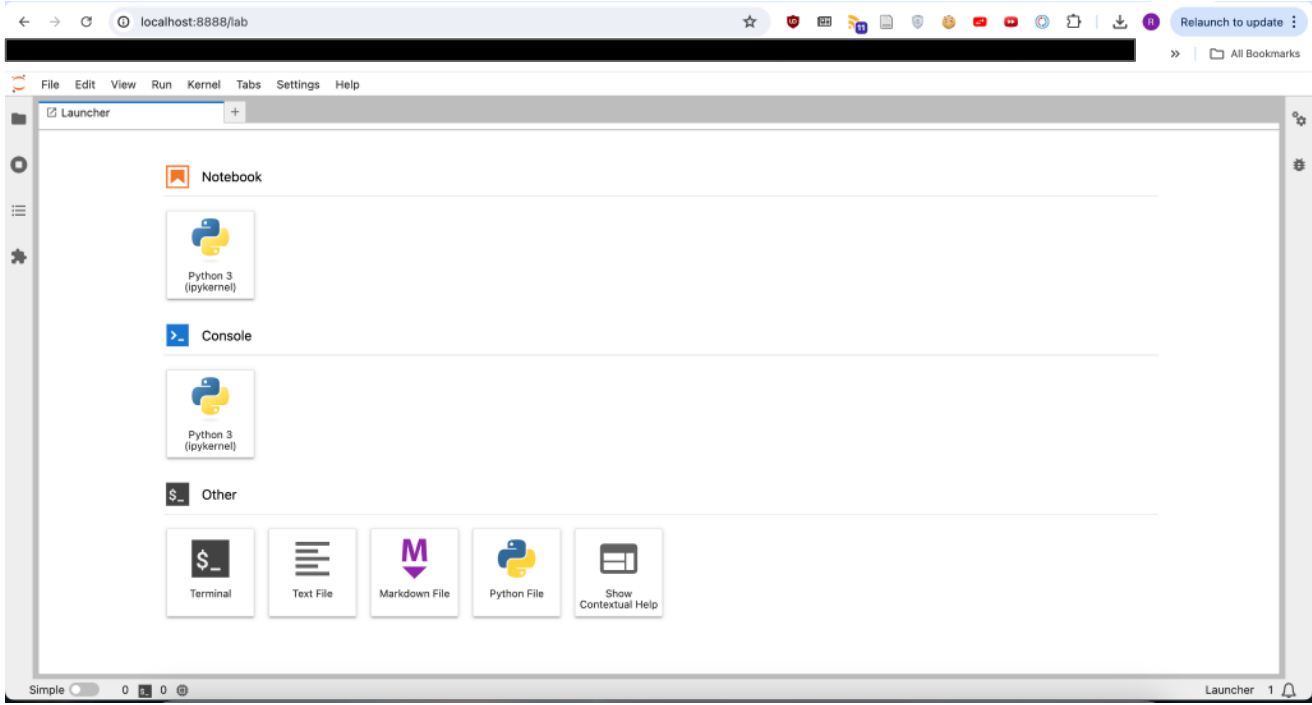
Environment Setup

4) If the server doesn't start in your browser automatically, then use one of the links provided



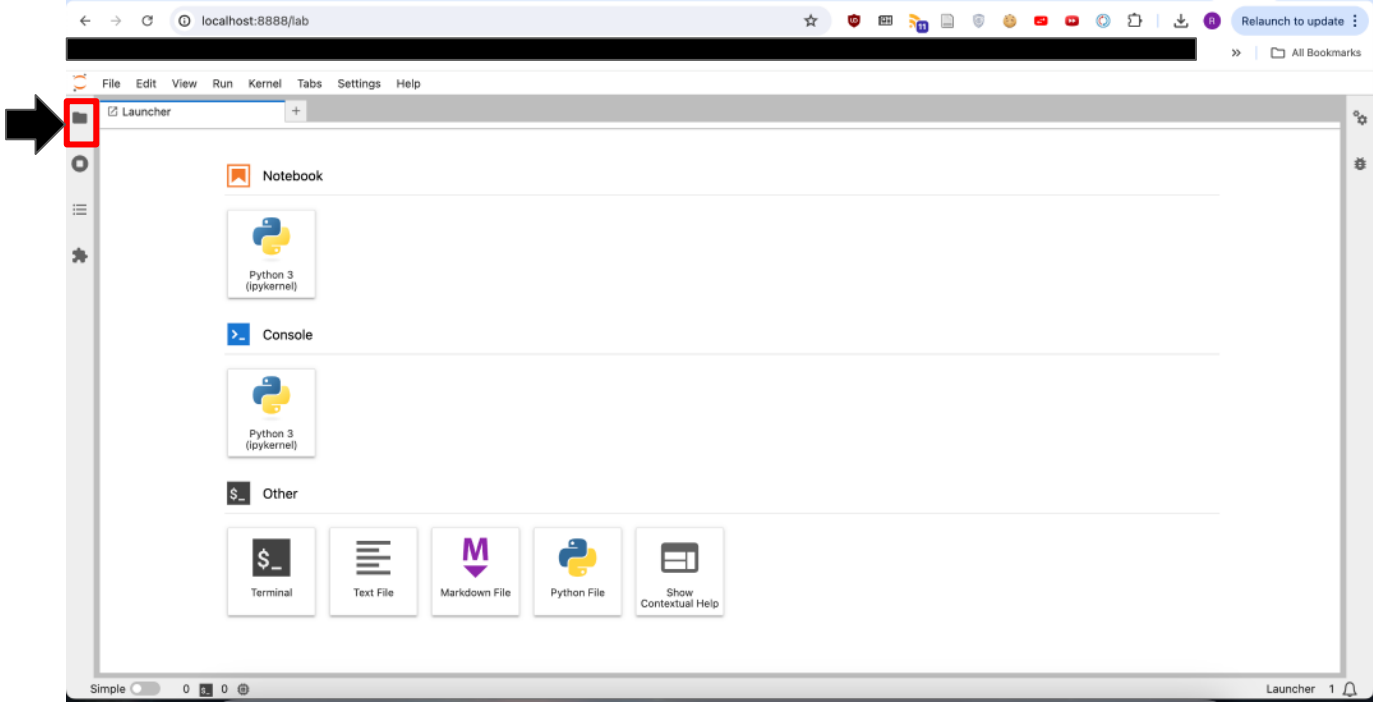
Environment Setup

5) You should see a browser tab that looks like this.



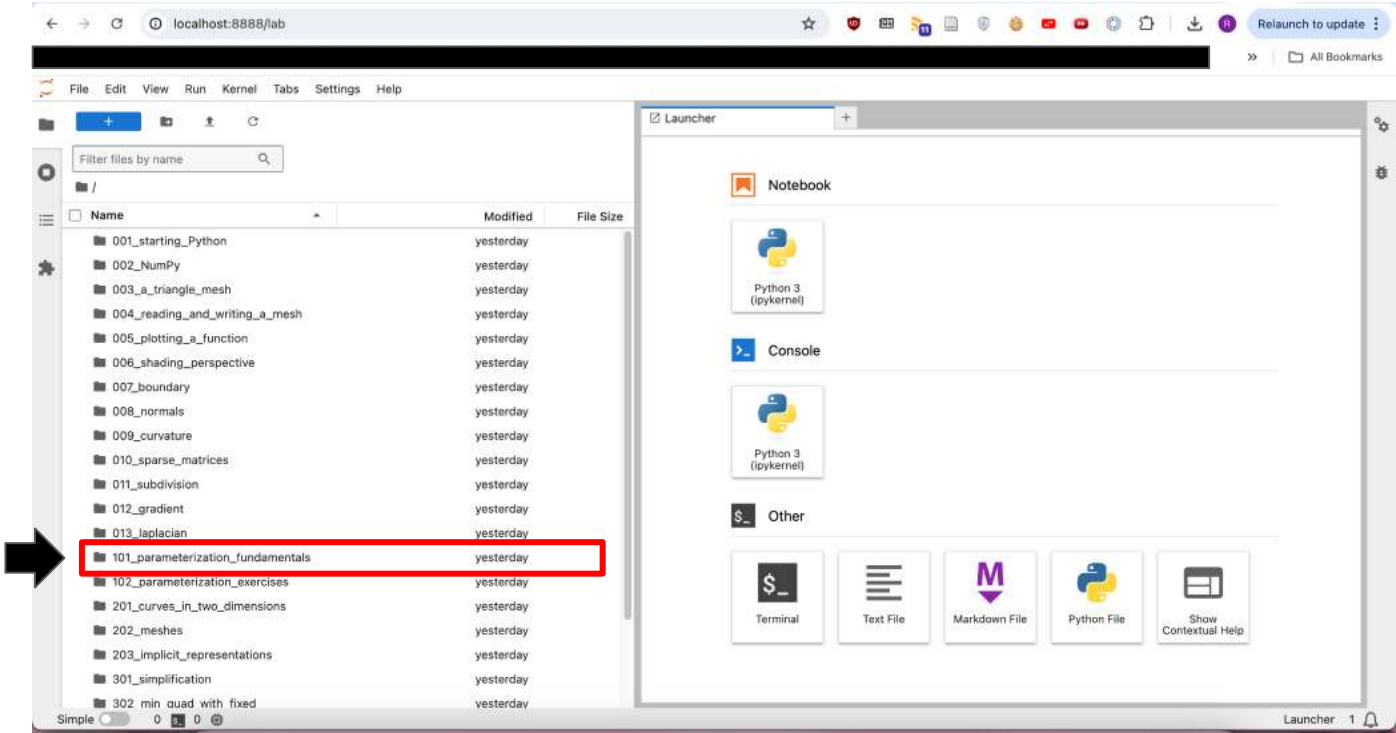
Environment Setup

6) You can access the folders in sgi-introduction-course by clicking the upper left folder icon



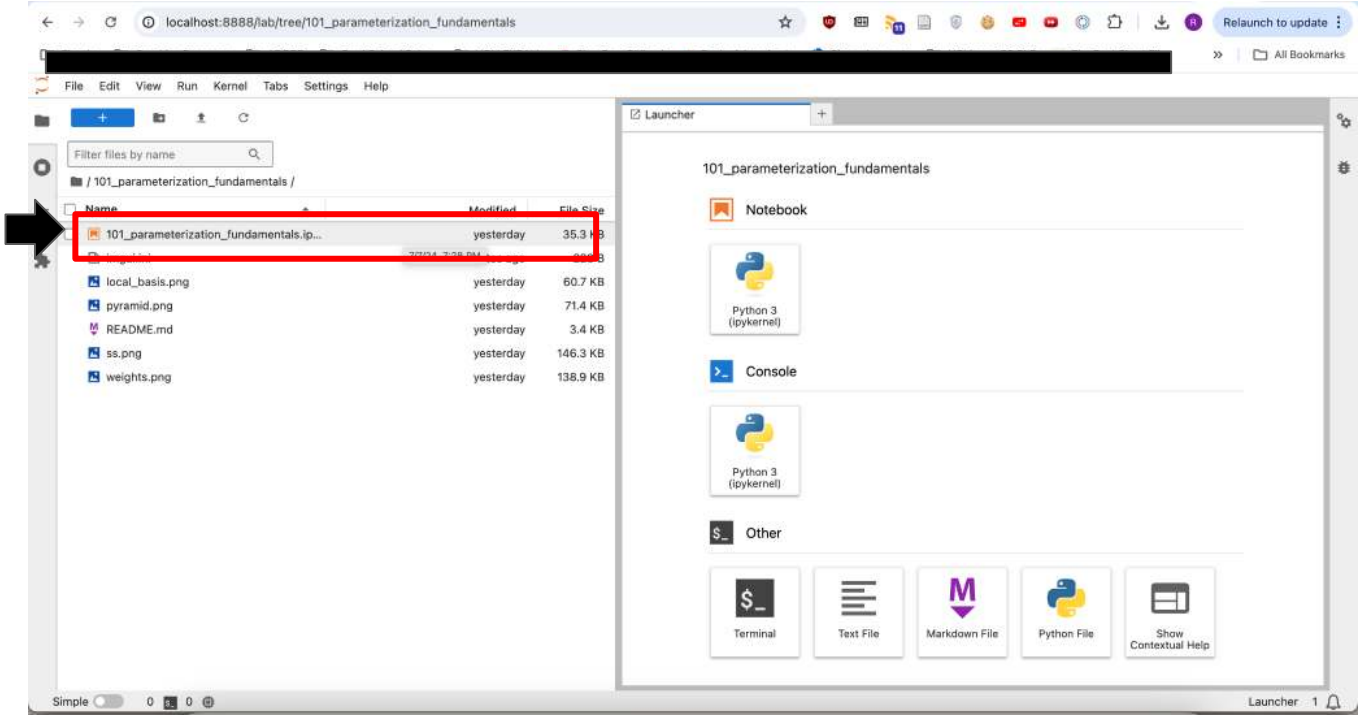
Environment Setup

7) Navigate to 101_parameterization_fundamentals



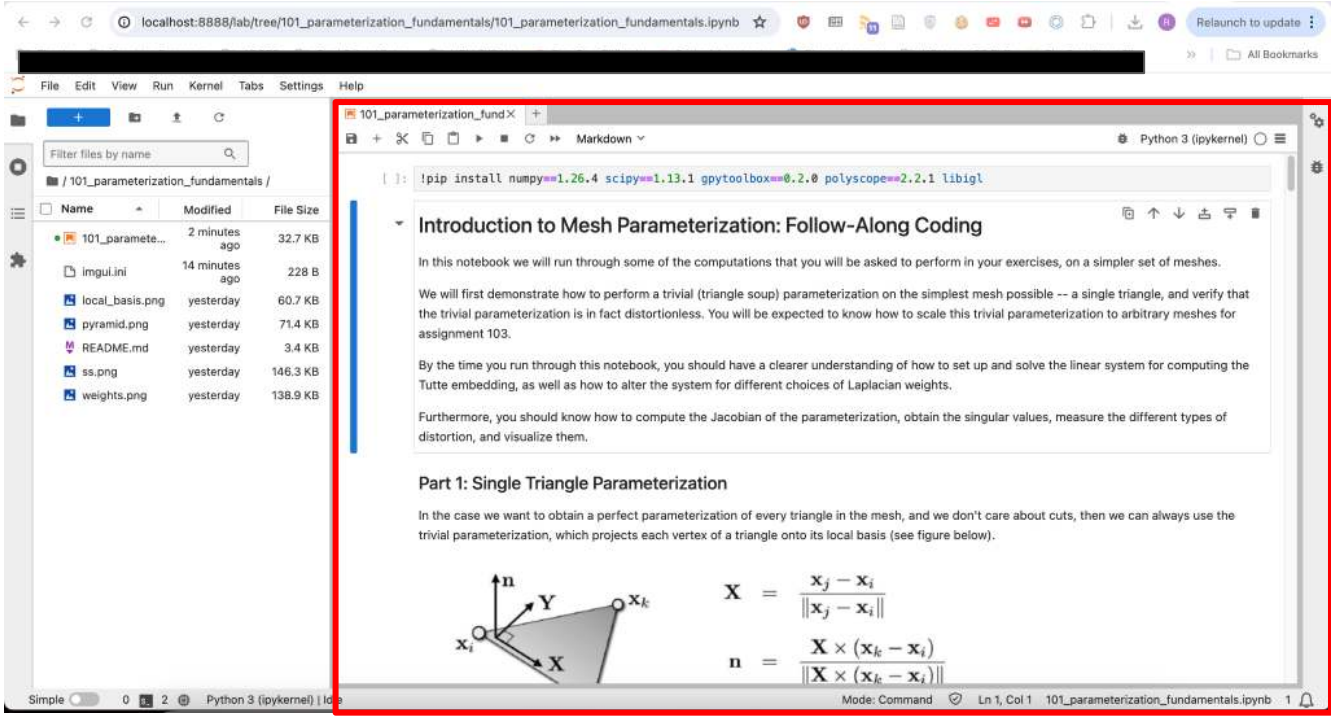
Environment Setup

8) Open 101_parameterization_fundamentals.ipynb



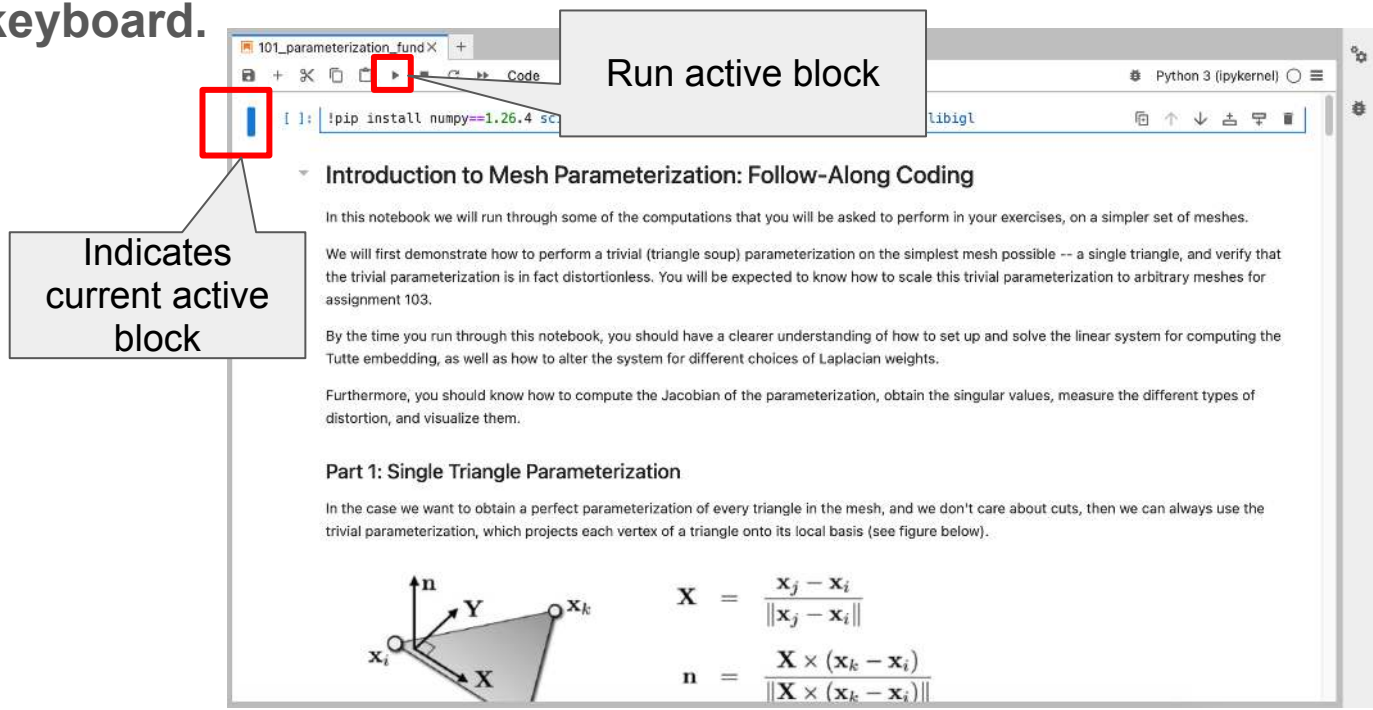
Environment Setup

9) You should now see the notebook open in the right panel.

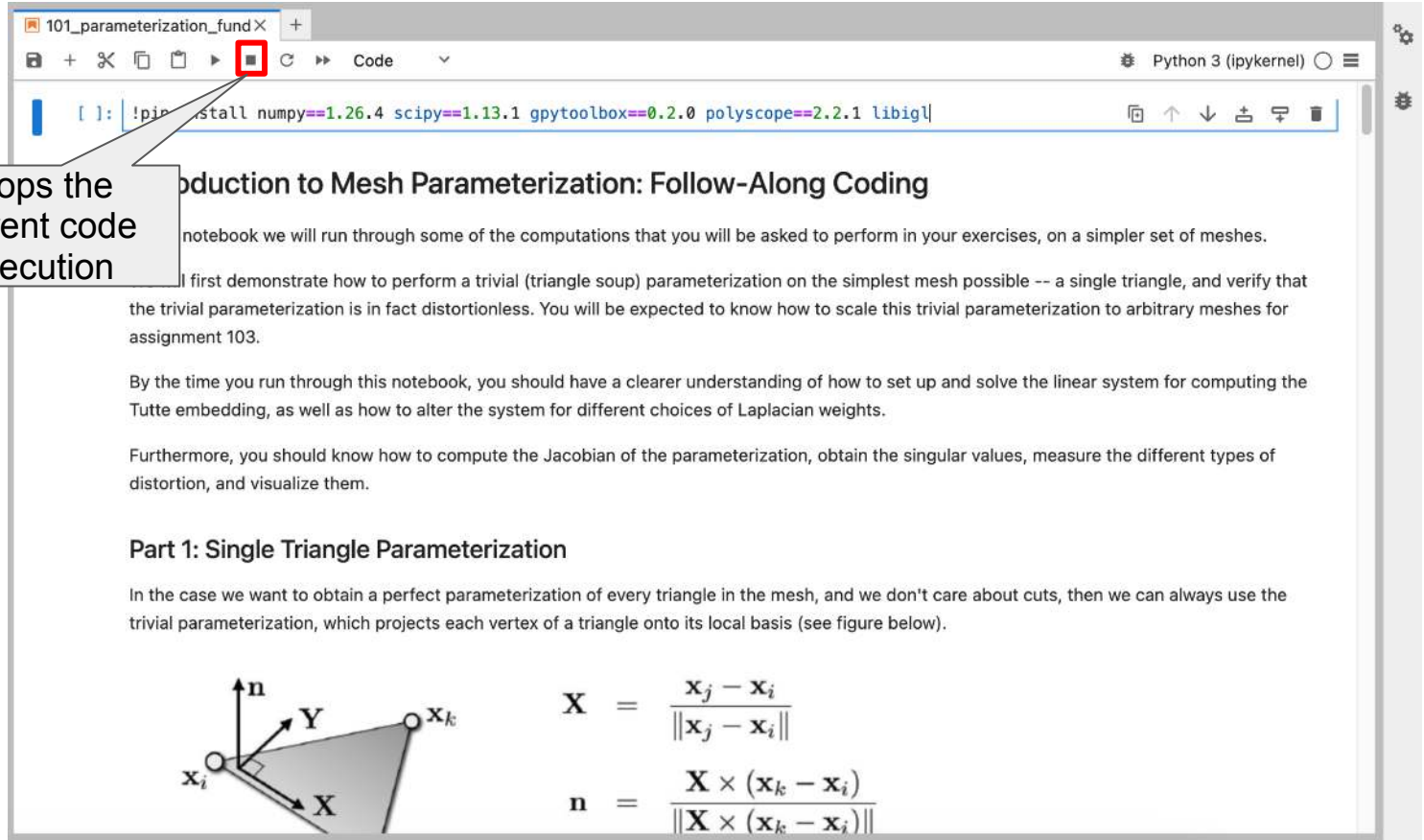


Environment Setup

10) Groups of code are separate into blocks. The blue bar indicates the active block. You can run the active block by pressing the ► button or pressing "shift+enter" on your keyboard.

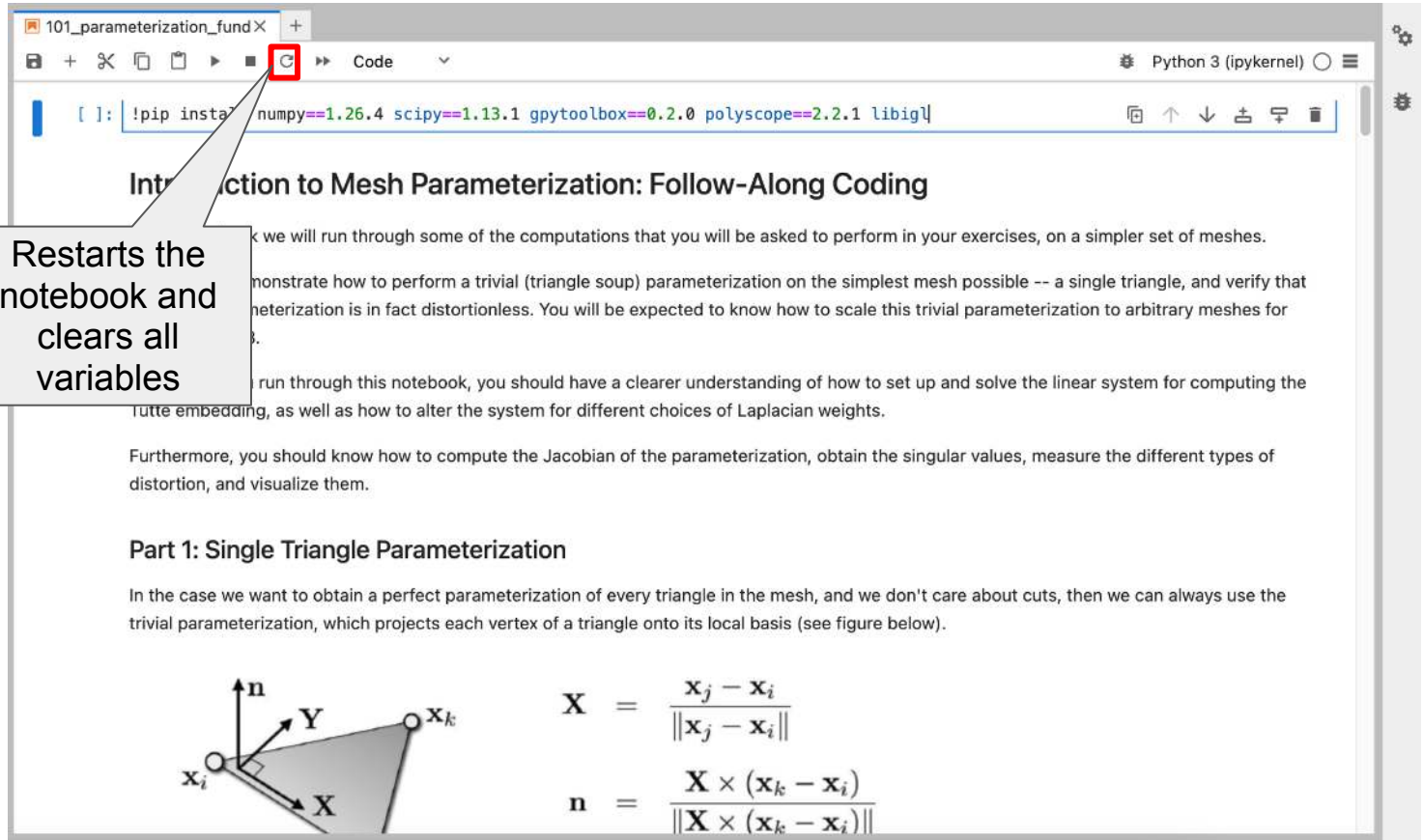


Environment Setup



Stops the current code execution

Environment Setup



Restarts the notebook and clears all variables

Environment Setup

Add block above

101_parameterization_fund

Code

Python 3 (ipykernel)

```
[ ]: !pip install numpy==1.26.4 scipy==1.13.1 gpytoolbox==0.2.0 polyscope==2.2.1 libigl
```

Introduction to Mesh Parameterization: Follow-Along Coding

In this notebook we will run through some of the computations that you will be asked to perform in your exercises, on a simpler set of meshes.

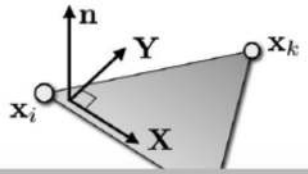
We will first demonstrate how to perform a trivial (triangle soup) parameterization on the simplest mesh possible -- a single triangle, and verify that the trivial parameterization is in fact distortionless. You will be expected to know how to scale this trivial parameterization to arbitrary meshes for assignment 103.

By the time you run through this notebook, you should have a clearer understanding of how to set up and solve the linear system for computing the Tutte embedding, as well as how to alter the system for different choices of Laplacian weights.

Furthermore, you should know how to compute the Jacobian of the parameterization, obtain the singular values, measure the different types of distortion, and visualize them.

Part 1: Single Triangle Parameterization

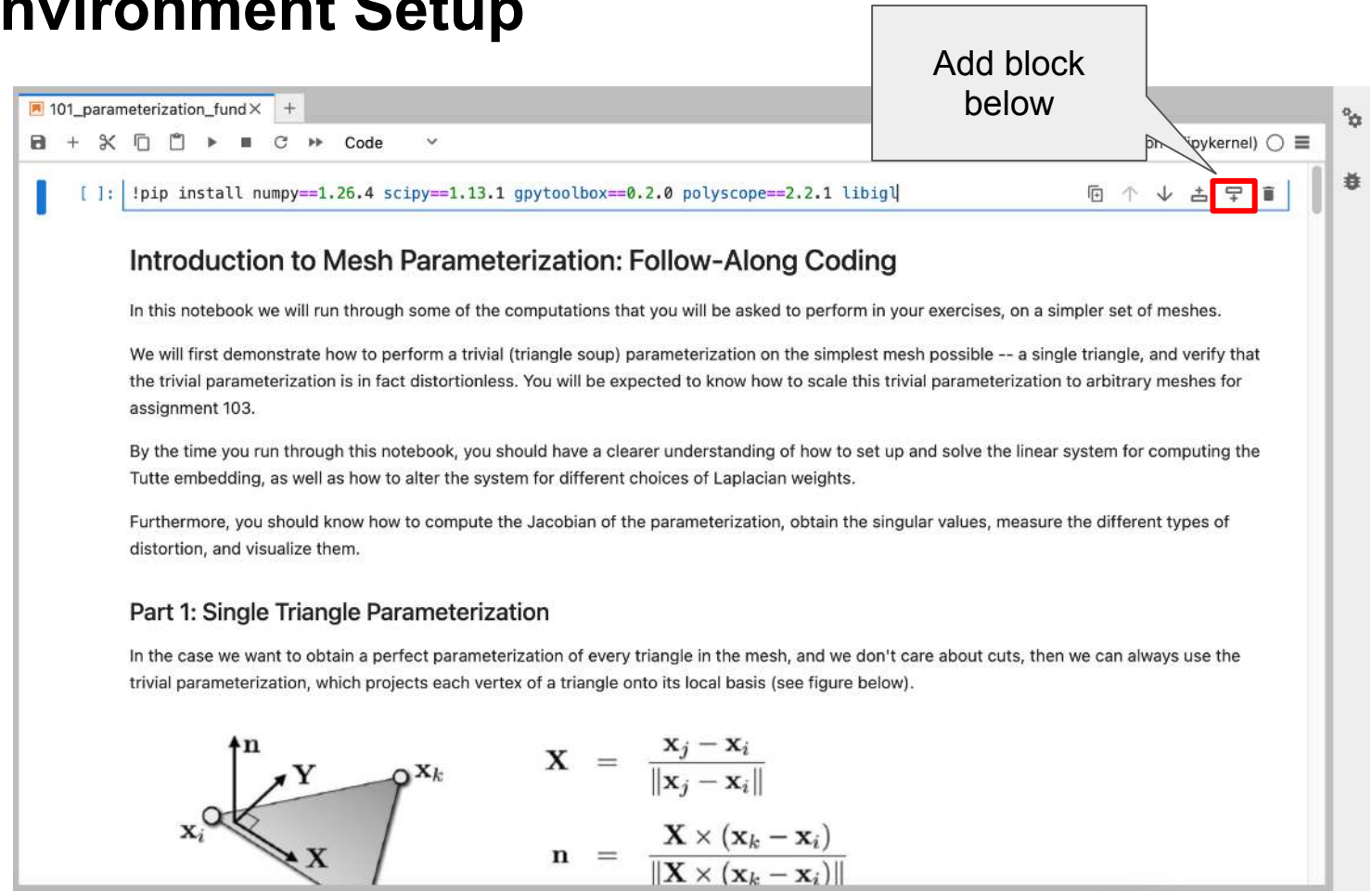
In the case we want to obtain a perfect parameterization of every triangle in the mesh, and we don't care about cuts, then we can always use the trivial parameterization, which projects each vertex of a triangle onto its local basis (see figure below).



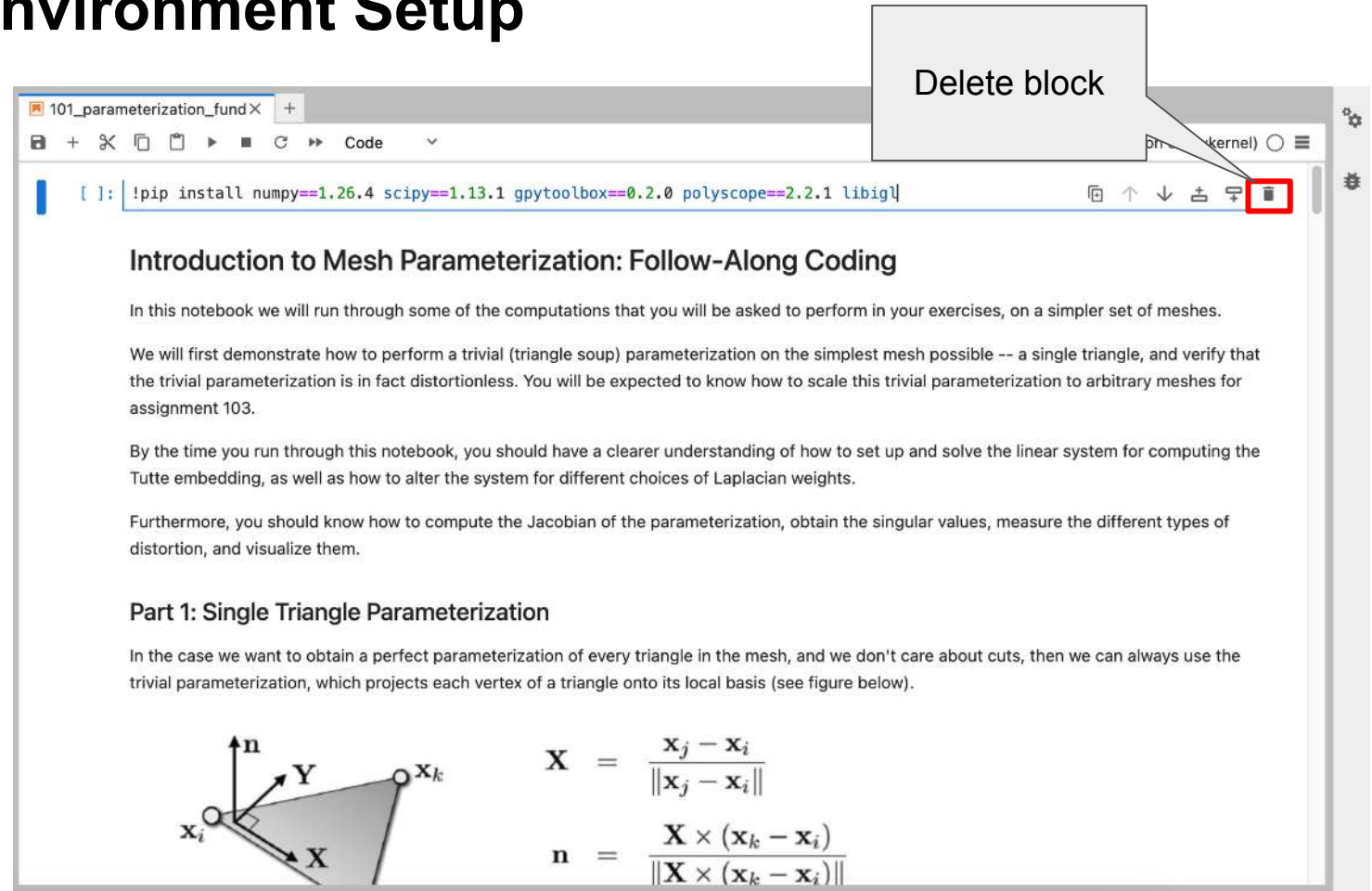
$$\mathbf{X} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$

$$\mathbf{n} = \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

Environment Setup



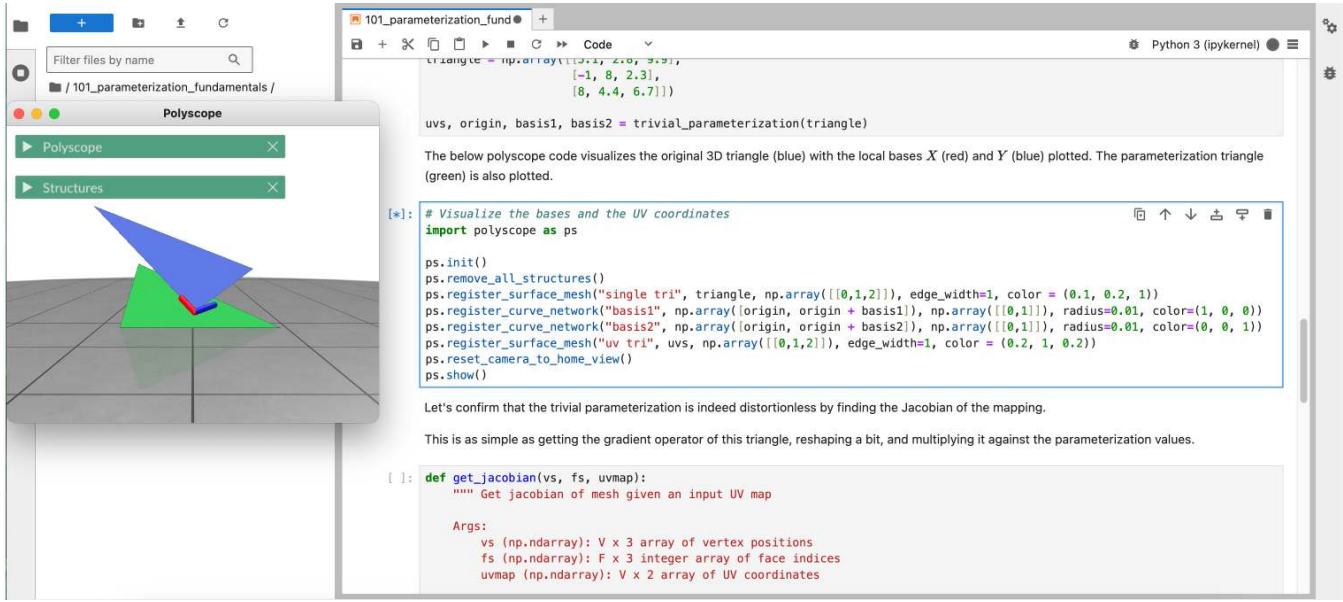
Environment Setup



Environment Setup

101 is designed for you to be able to run the code all the way without any additional coding.

Polyscope will work when run from a cell!



Thank you!