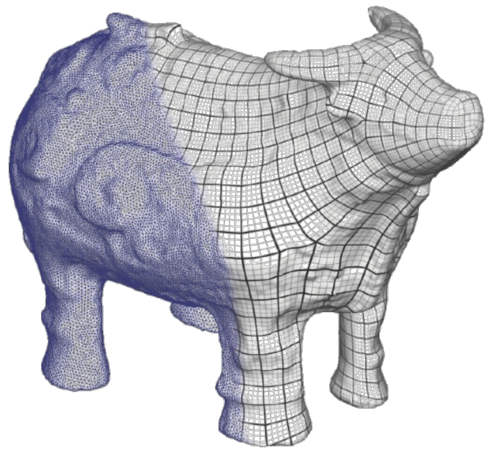
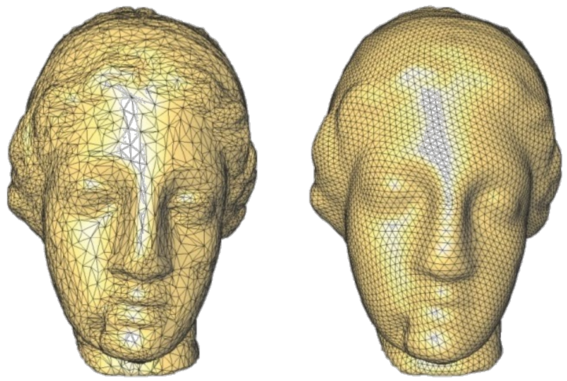
CS348a: Computer Graphics -- Geometric Modeling and Processing



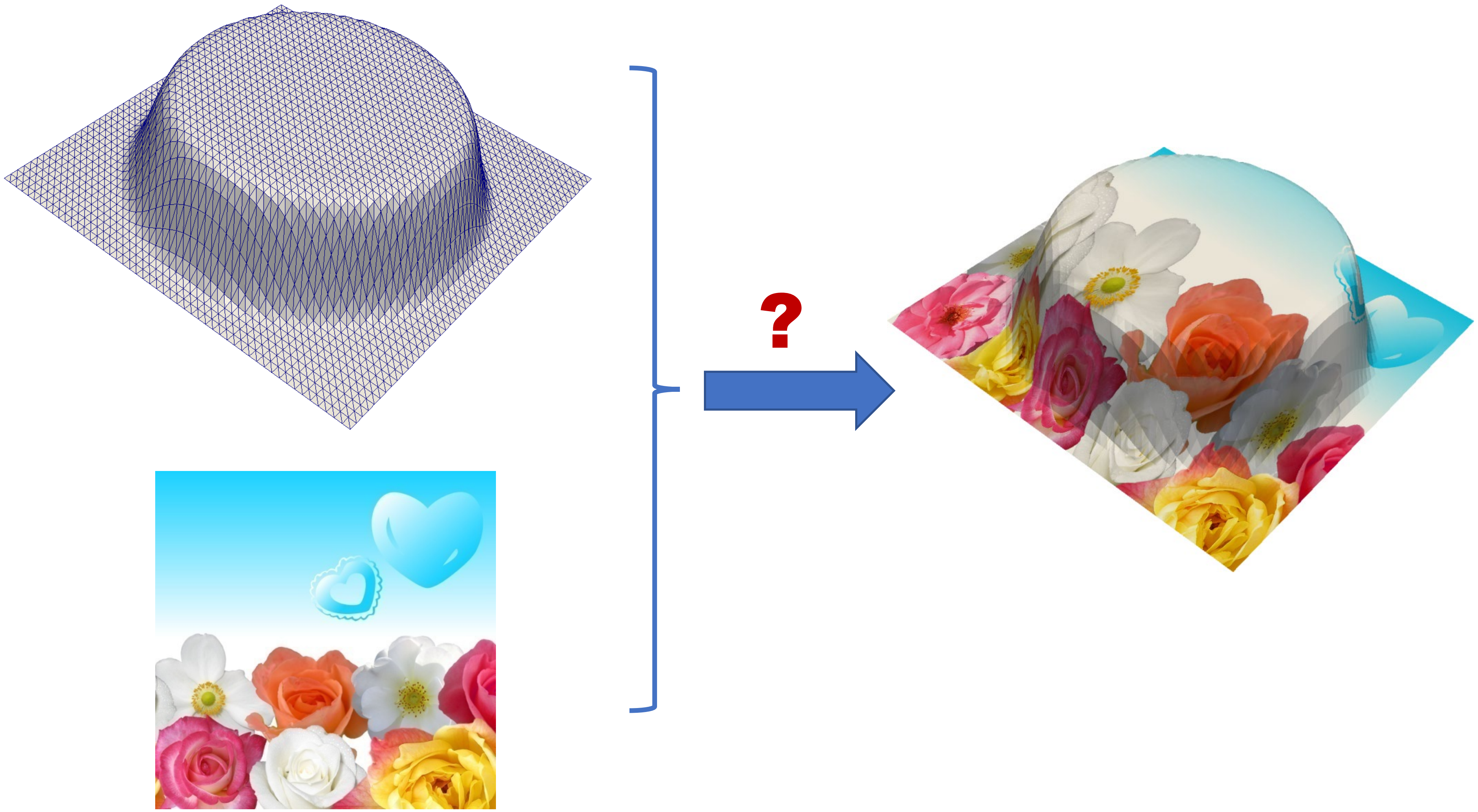


Leonidas Guibas
Computer Science Department
Stanford University

Leonidas Guibas Laboratory

Geometric Computing

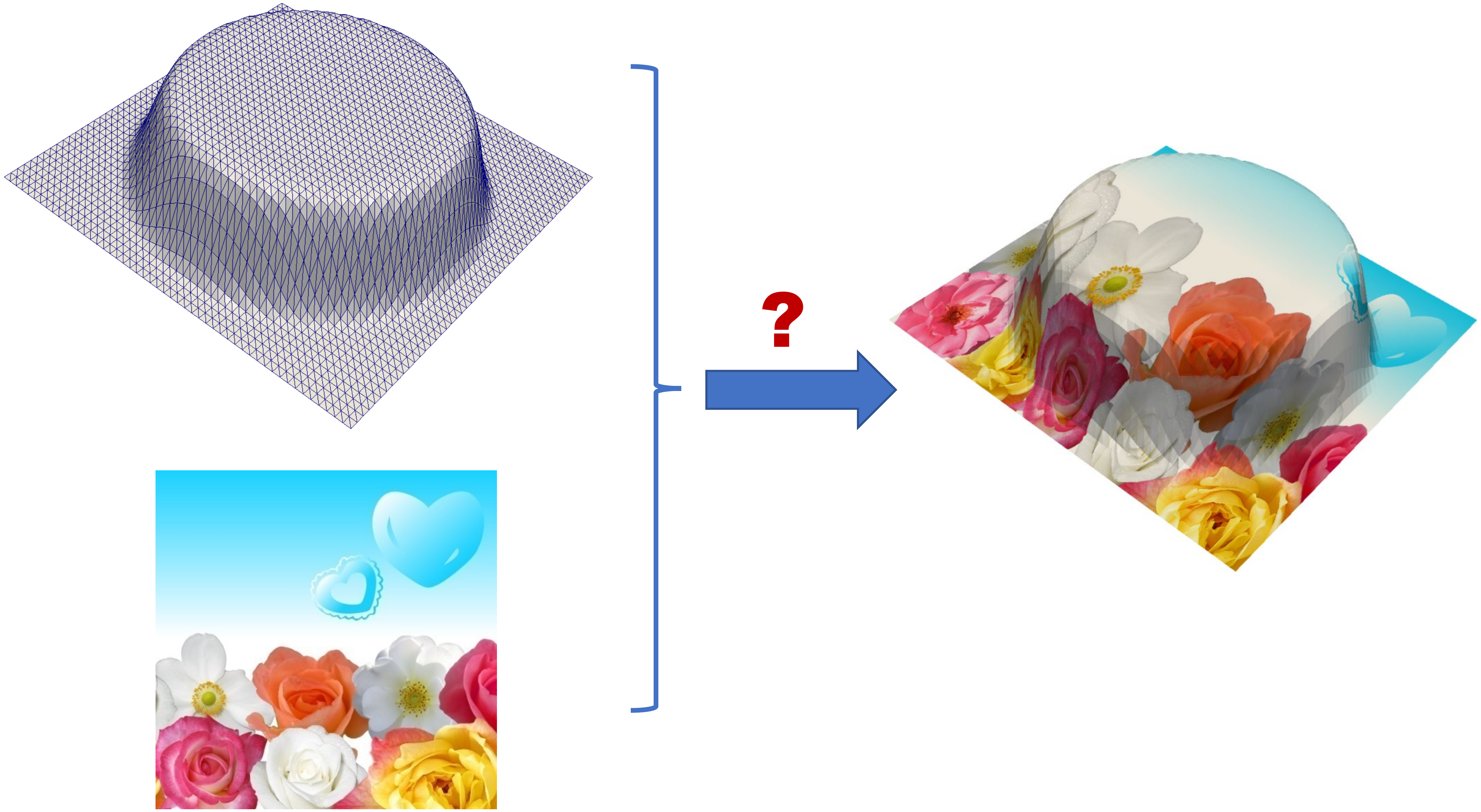
Parametrization



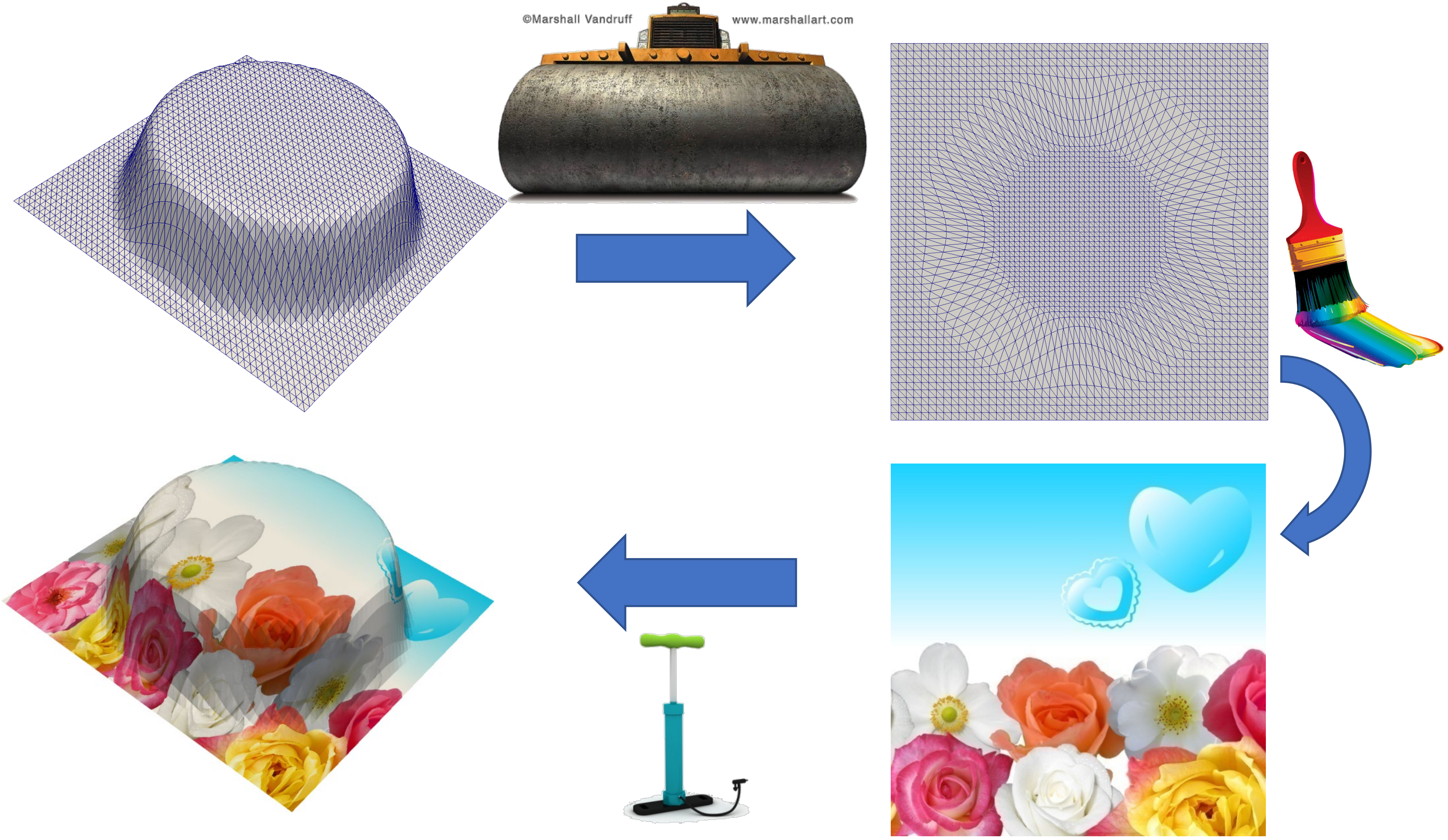
The Basic Problem

http://perso.telecom-paristech.fr/~tierny/stuff/teaching/tierny_surface_parameterization_web.pdf

The Basic Problem

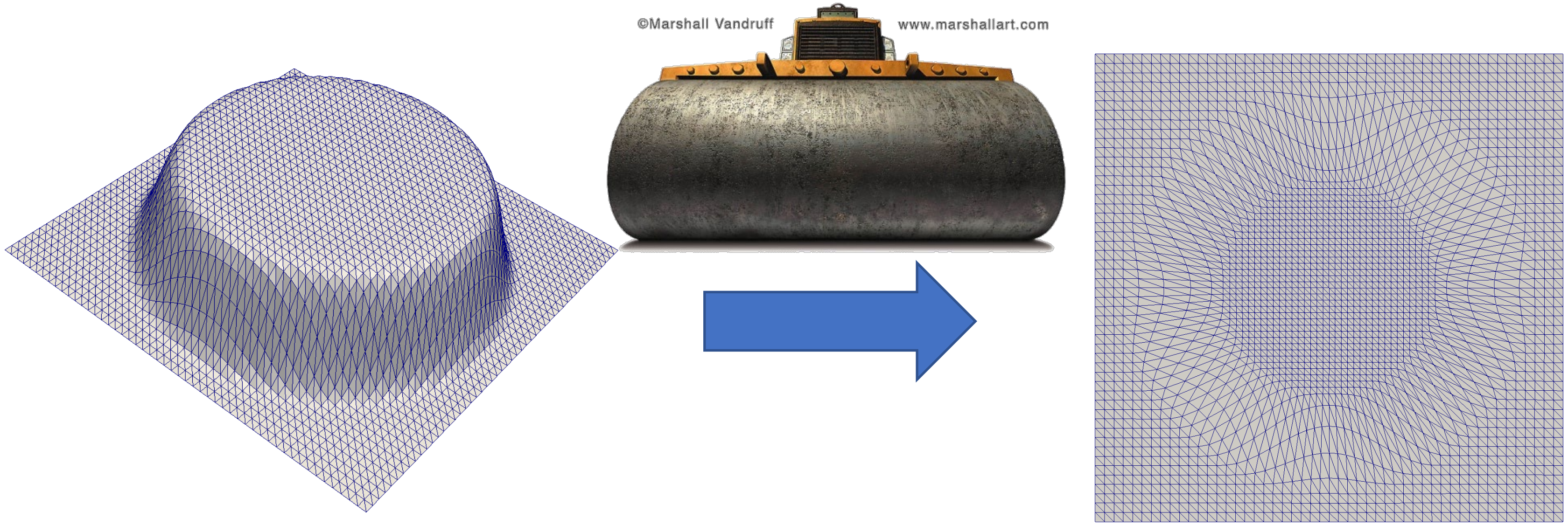


Solution



Flattening Surfaces

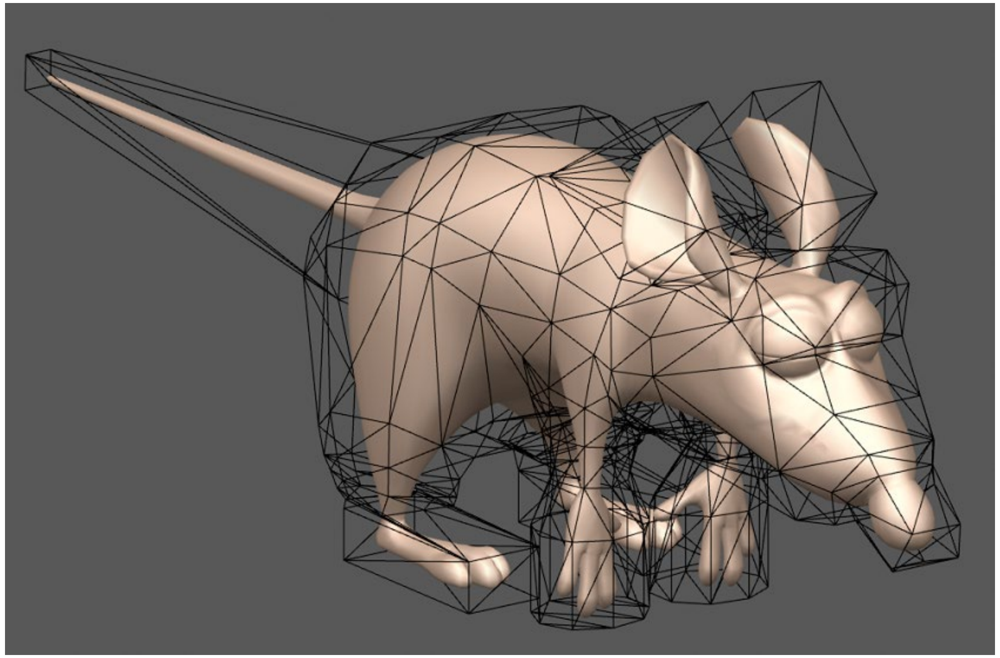
Parameterization is …

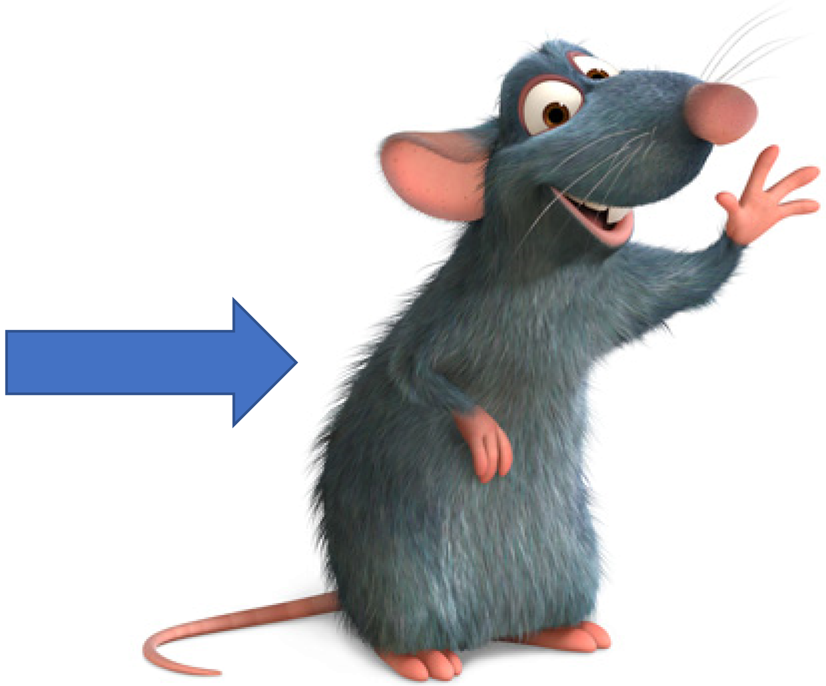


Why Parametrize?

R.I.P.
Really
Interested in
Parameterization

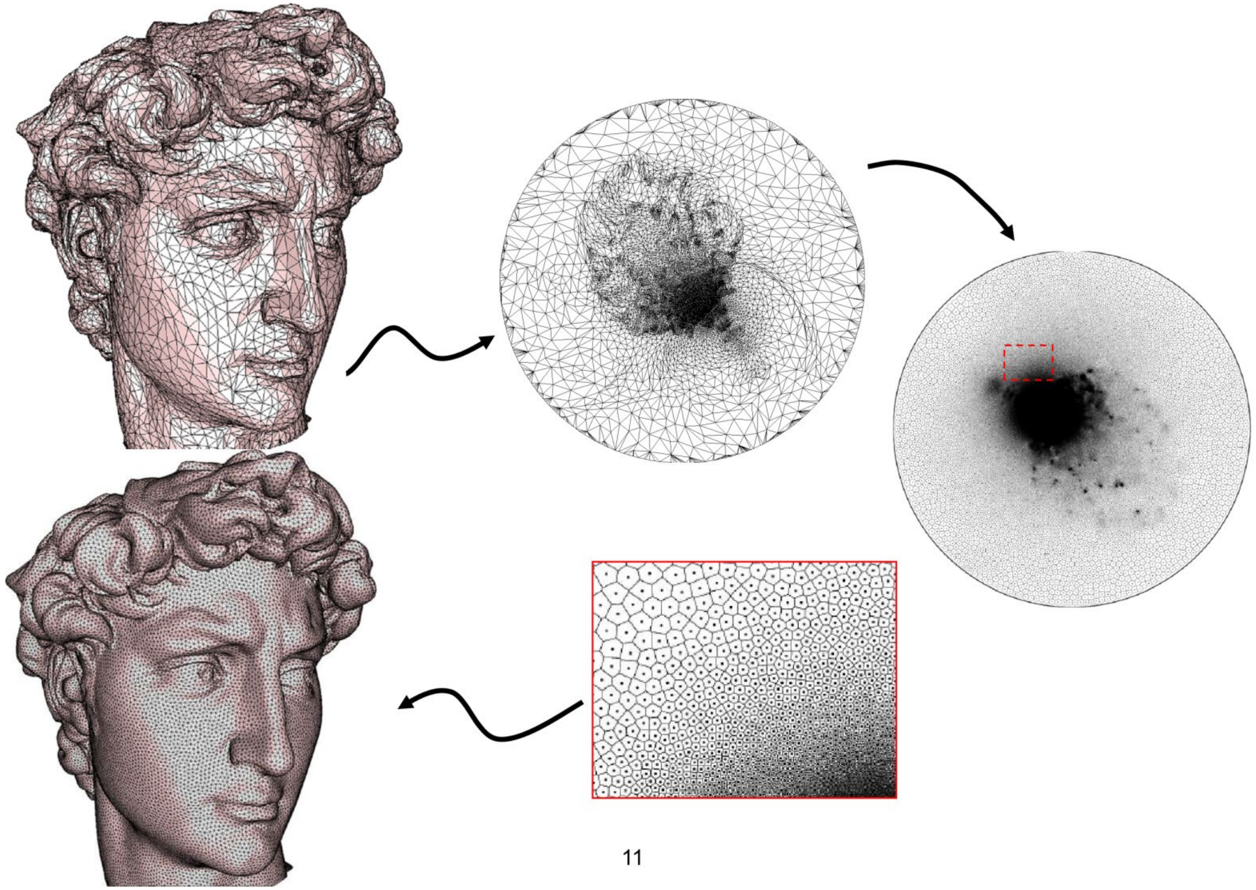
Why Parametrize?





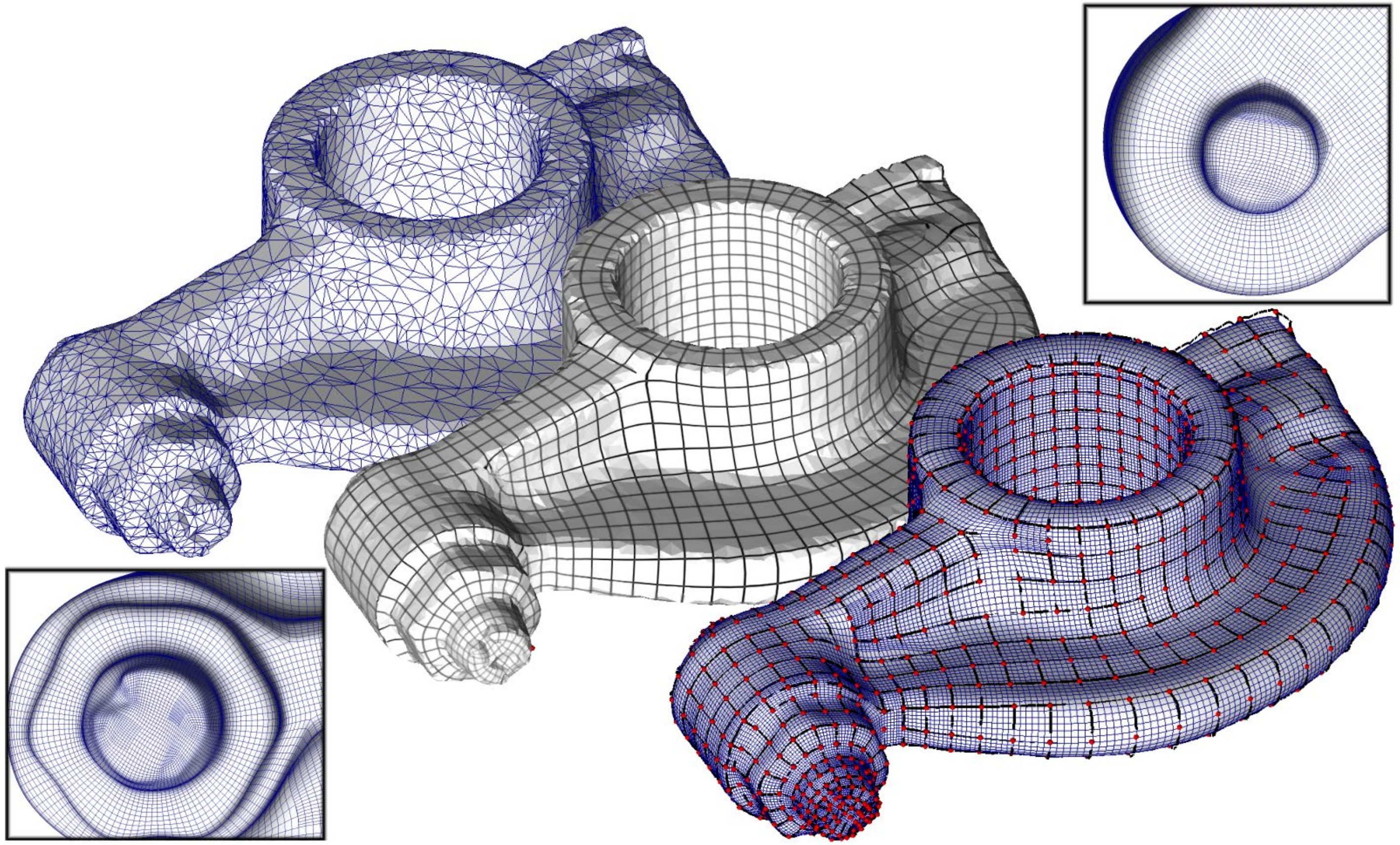
Texture Mapping

Why Parametrize?



Remeshing

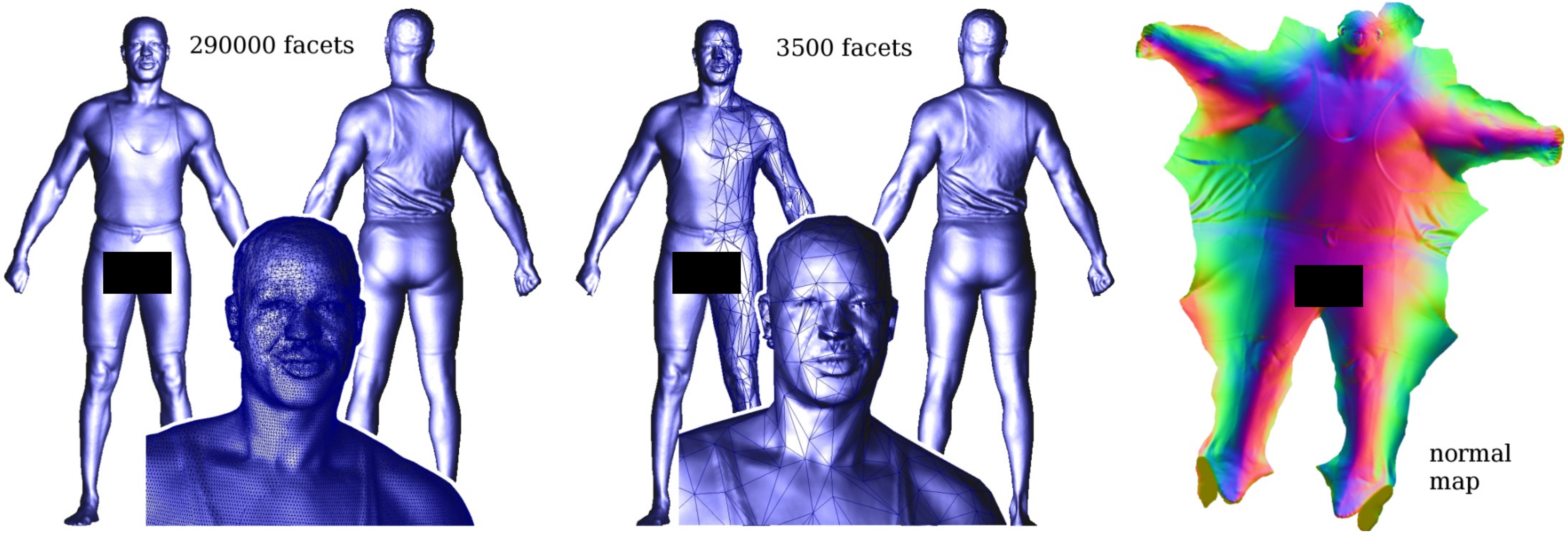
Why Parametrize?



Alternative Representations

http://www.inf.usi.ch/hormann/parameterization/CourseNotes2008.pdf

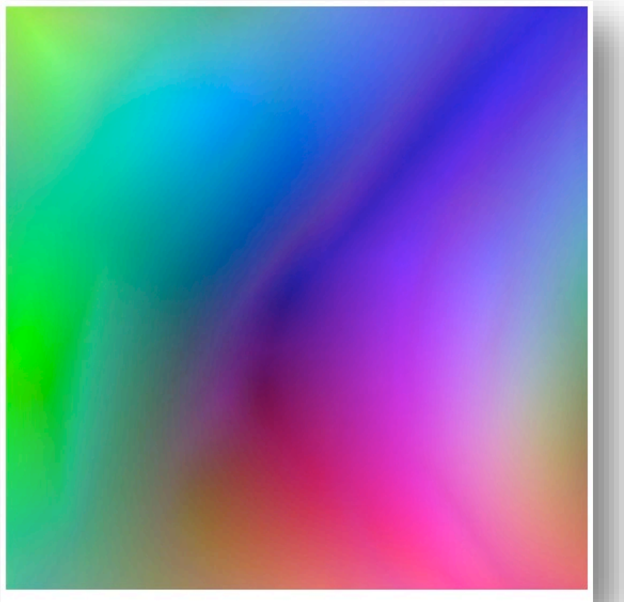
Why Parametrize?



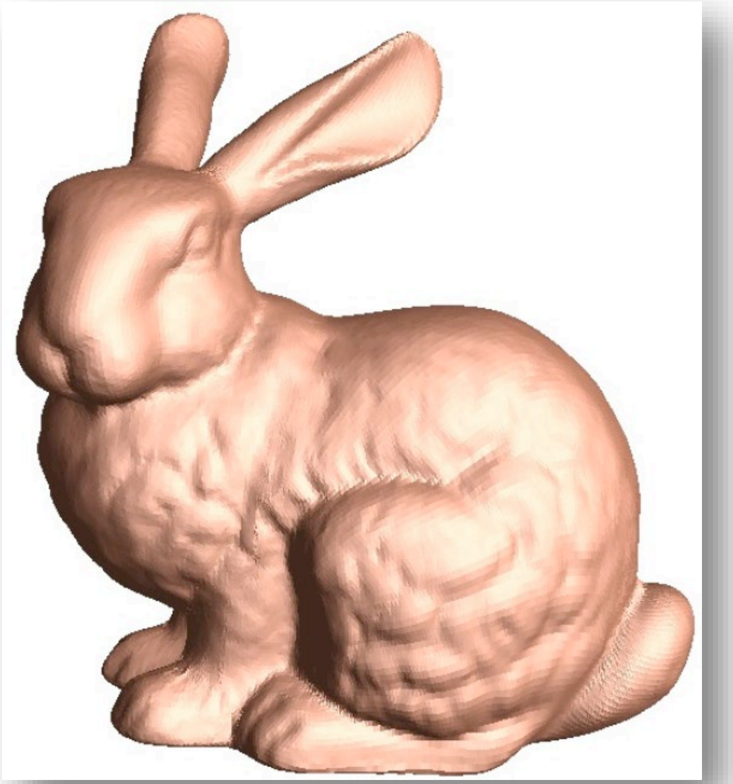
Simplification

http://www.inf.usi.ch/hormann/parameterization/CourseNotes2008.pdf

Why Parametrize?



=



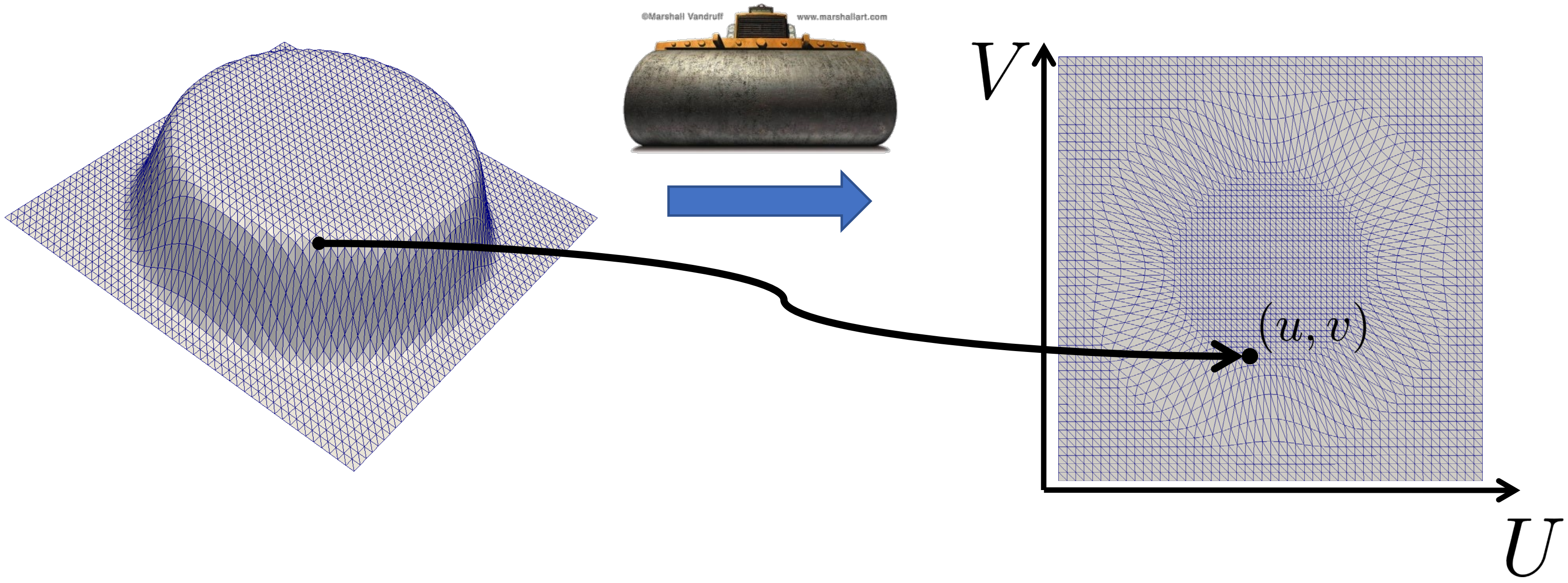
Compression

Gu, Gortler, Hoppe. Geometry Images. SIGGRAPH 2002

Outline

- Introduction
- Naïve approach and demonstration
- Distortion and desirable properties
- Fixed boundary: Harmonic parameterization
- Free-boundary: Eigenmap

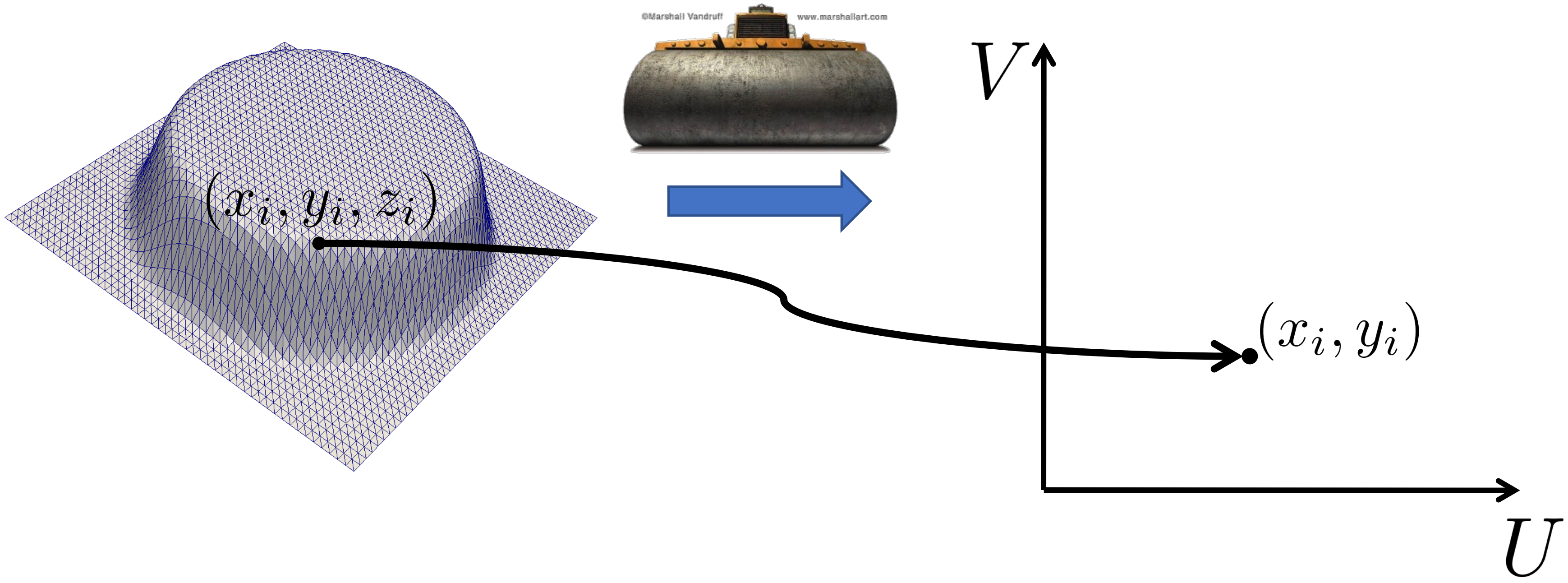
UV-Coordinates



For every mesh vertex determine its (u,v) coordinates

"Texture Coordinates"

Naïve Approach



Normalize so that UV-coordinates are in [0,1]

Look inside: VTK file format

(x_i, y_i, z_i)

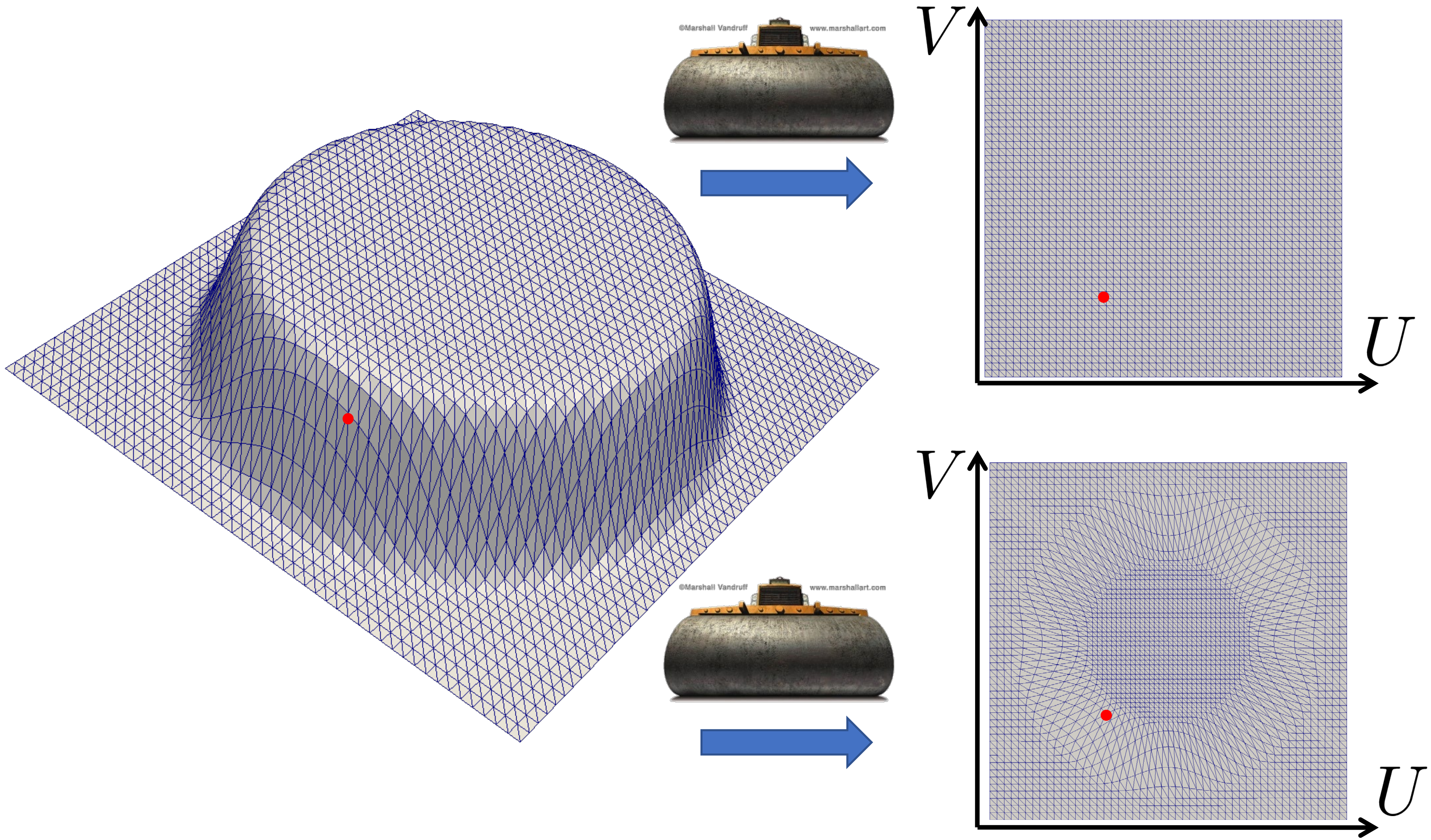
```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET POLYDATA
POINTS 10201 float
0      0 0.742357
0.02 0 0.794151
0.04 0 0.842234
0.06 0 0.885574
0.08 0 0.923144
0.1   0 0.953942
…
```

```
POLYGONS 20000 80000
3 0 2602 2601
3 2602 1 2603
3 2603 51 2601
3 2602 2603 2601
3 52 2605 2604
3 2605 51 2603
3 2603 1 2604
3 2605 2603 2604
…
```

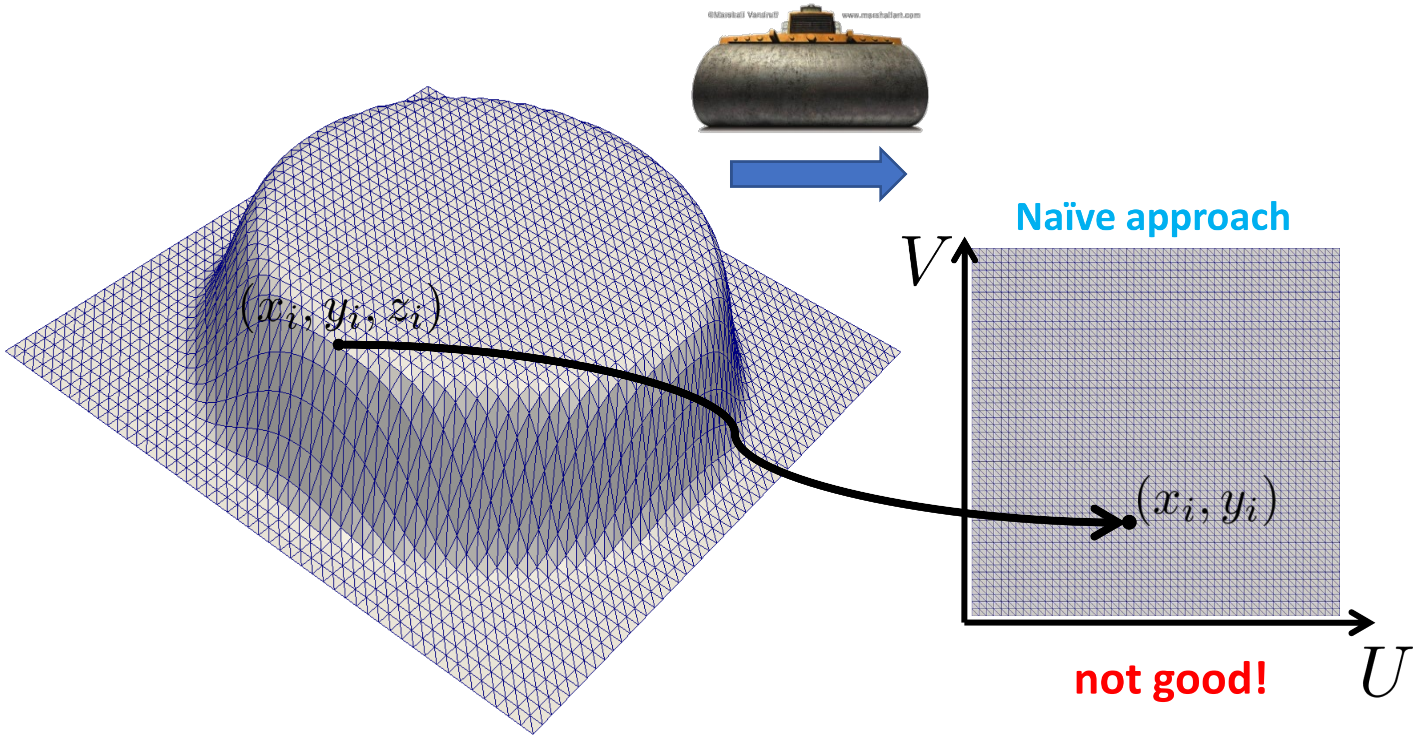
```
POINT_DATA 10201
TEXTURE_COORDINATES
Texture_Coordinates 2 float
0.000000 0.000000
0.020000 0.000000
0.040000 0.000000
0.060000 0.000000
0.080000 0.000000
0.100000 0.000000
…
```

(u_i, v_i)

Which is Better?



Main Issue: Distortion

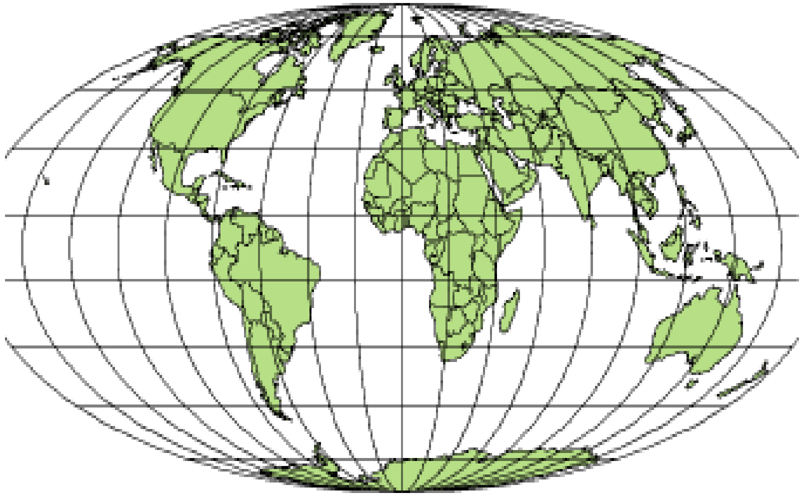


Triangle shapes (angles) and sizes (area) are not preserved!

Preserve shape & size = Triangle congruence (aka isometry)

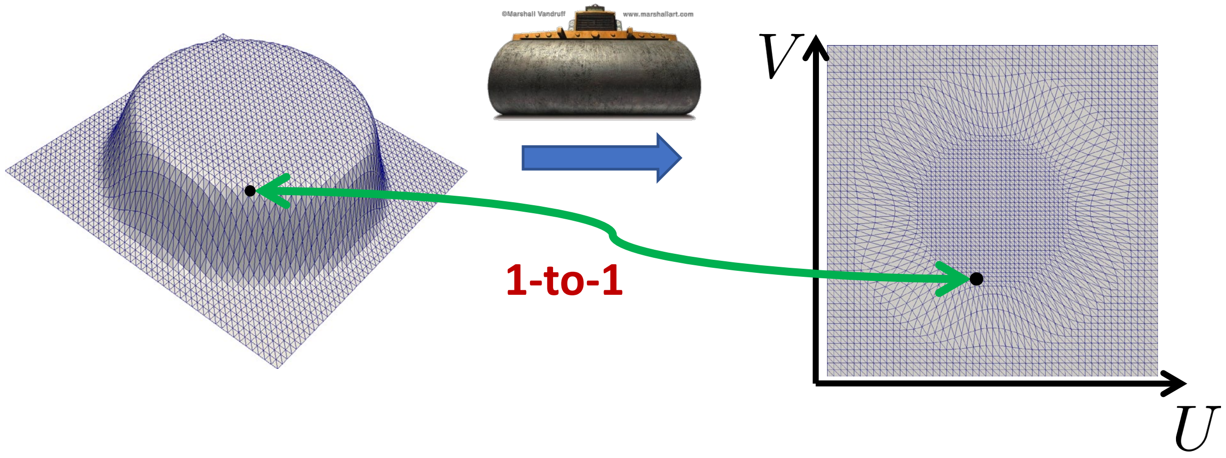
An Old Problem: Maps

Mercator-Projektion



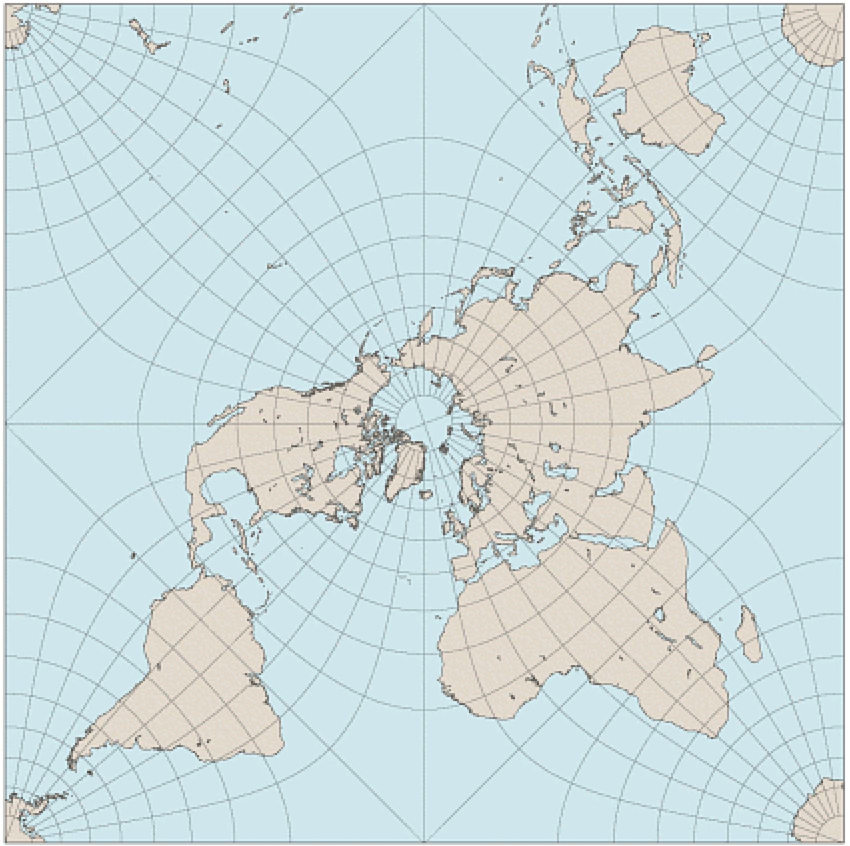
Mollweide-Projektion

Desirable Characteristics



Bijective: no fold overs

Desirable Characteristics



Conformal: Preserves angles

Desirable Characteristics

Equiareal: Preserves areas

http://en.wikipedia.org/wiki/Bonne_projection

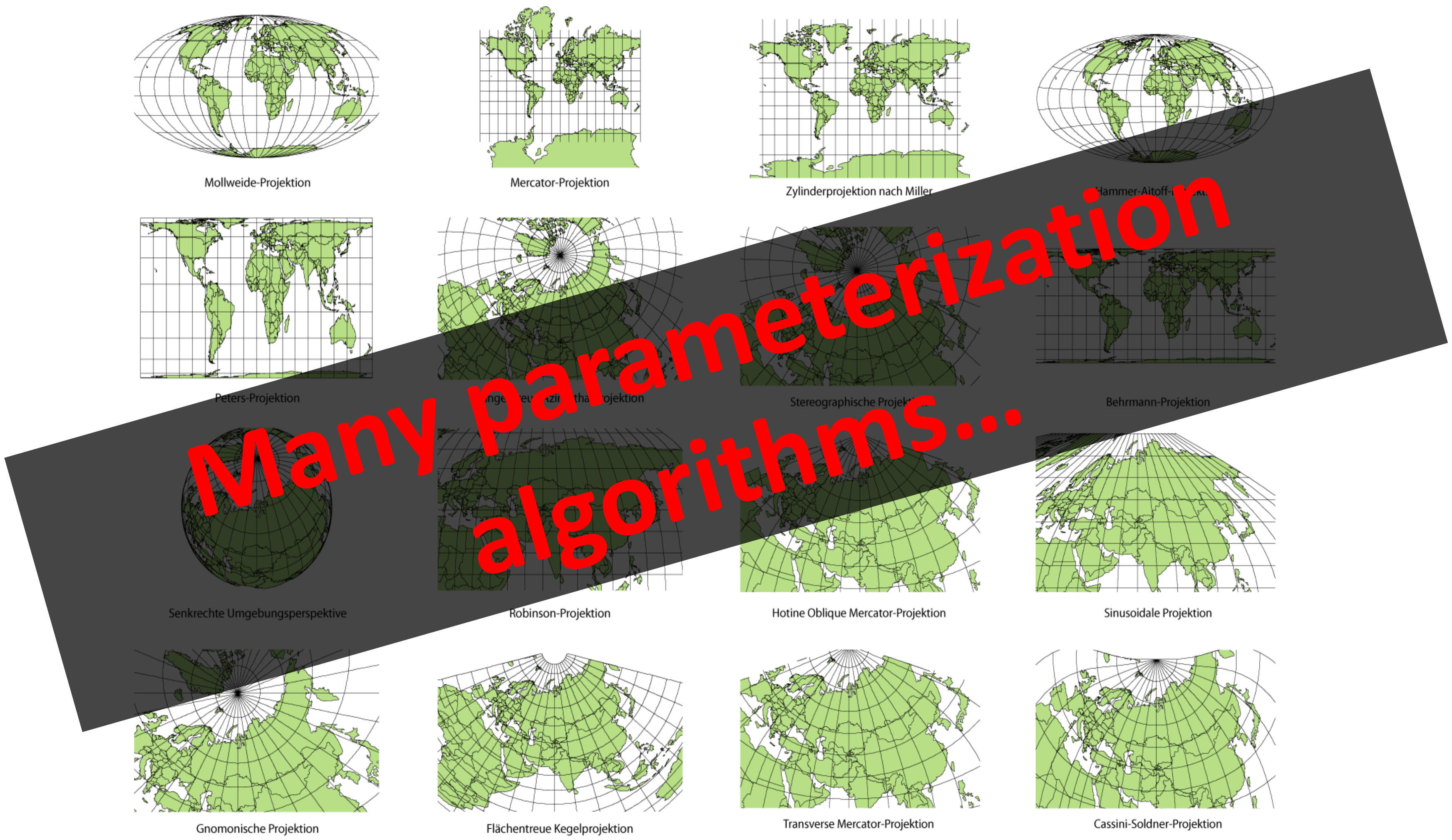
Desirable Characteristics

Isometric: conformal and equiareal

Sad Fact

Very few surfaces can be mapped **isometrically** to the plane.

Reason for so Many Types of Maps



Outline

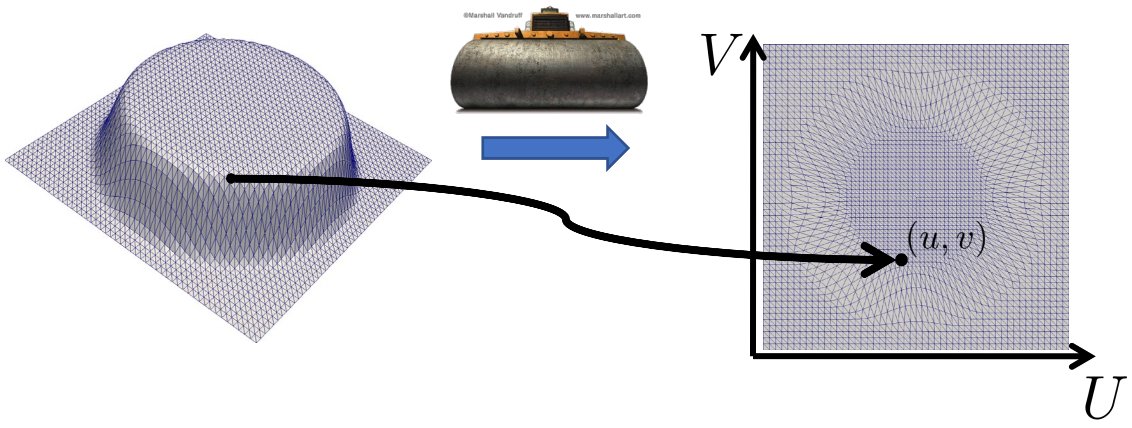
- Introduction
- Naïve approach and demonstration
- Distortion and desirable properties
- Fixed boundary: Harmonic parameterization
- Free-boundary: Eigenmap

Tutte's Theorem

- If the (u,v) coordinates at the boundary lie on a convex polygon, and if coordinates of the internal vertices are convex combination of their neighbors, then these (u,v) coordinates give a valid (bijective) parameterization.

- Convex combination = center of mass
- Can have different masses at different vertices
- Masses should be positive

Simple Realization



Goal: Assign (*u*,*v*) coordinate to each mesh vertex.

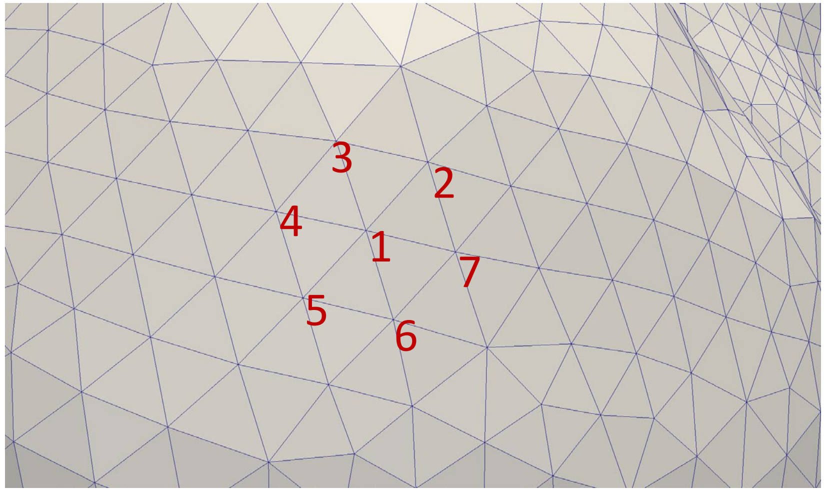
1. Fix (*u*,*v*) coordinates of boundary.
2. Want interior vertices to be at the center of mass of neighbors:

$$u_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} u_j \qquad v_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j$$

Algorithm

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

$$u_i \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} u_j \qquad v_i \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j$$

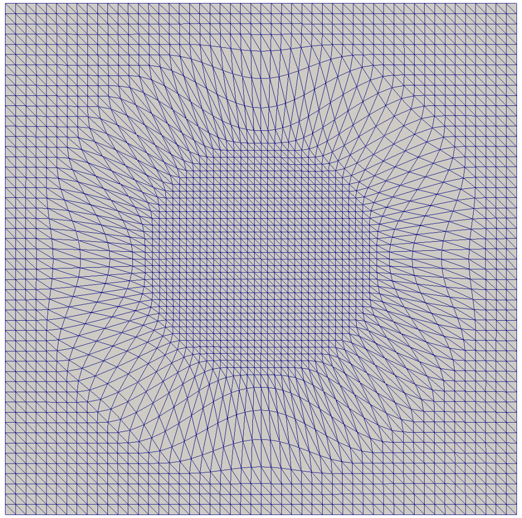


$$u_1 \leftarrow \frac{u_2 + u_3 + u_4 + u_5 + u_6 + u_7}{6}$$

$$v_1 \leftarrow \frac{v_2 + v_3 + v_4 + v_5 + v_6 + v_7}{6}$$

What do you think?

What do you think?

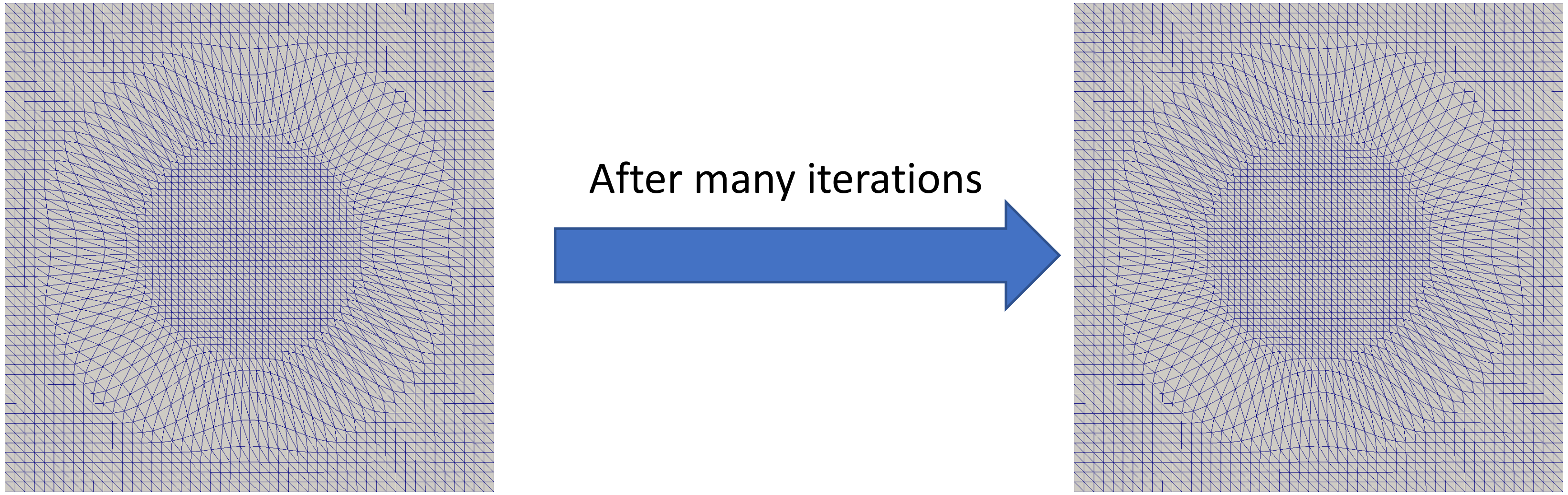


Some random planar mesh

After many iterations →

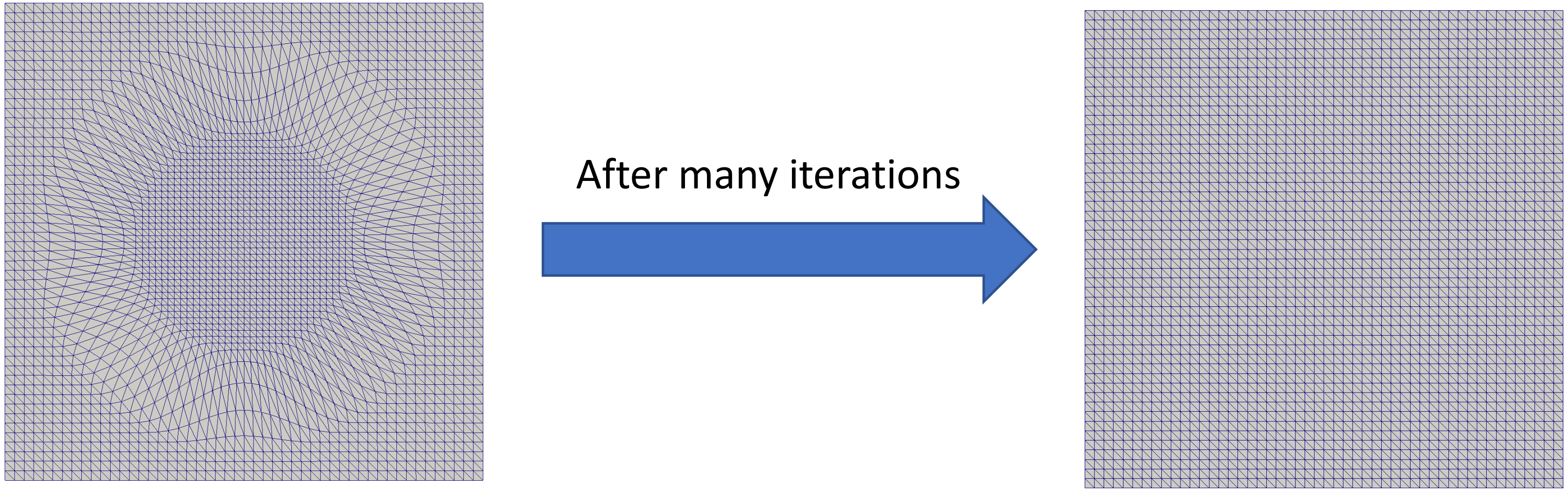
?

Expectation



It is already planar: best parameterization = itself

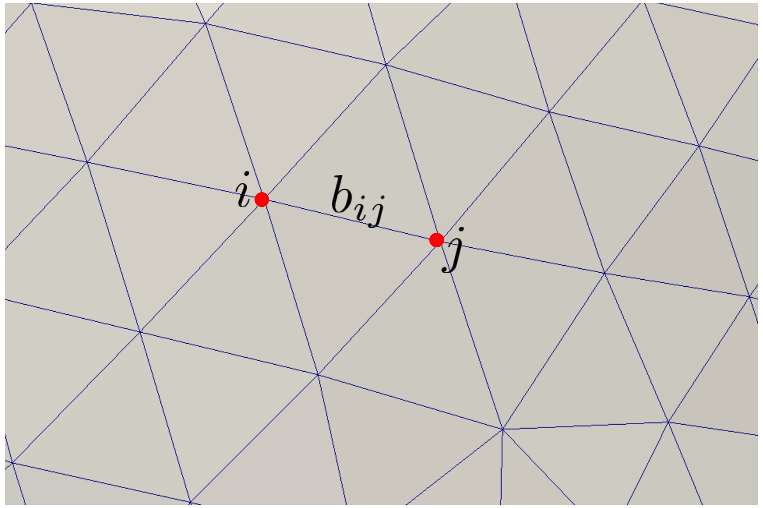
Reality… Why? How to Avoid?



Converges to a somewhat uniform grid!

Triangle shapes and sizes are not preserved!

Algorithm with Weights



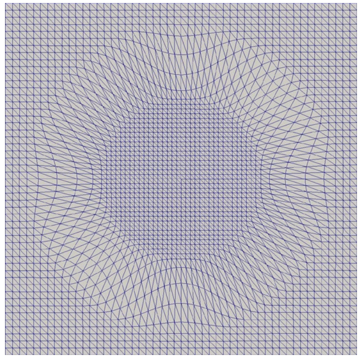
Introduce weights to capture geometric information:

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

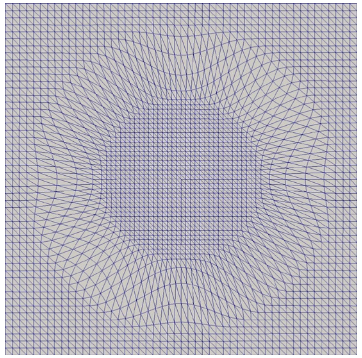
$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Weight Properties

GOAL:



After many iterations



For planar mesh with vertex coordinates (x_i, y_i):

$$\sum_{j \in N(i)} b_{ij} x_j = x_i$$

$$\sum_{j \in N(i)} b_{ij} y_j = y_i$$

With naïve initialization, iterations converge immediately!

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Weight Properties

$$\sum_{j \in N(i)} b_{ij} = 1$$

If weights w_{ij}, don't satisfy, then normalize:

$$b_{ij} = \frac{w_{ij}}{\sum_{j \in N(i)} w_{ij}}$$

For planar mesh with vertex coordinates (x_i, y_i):

$$\sum_{j \in N(i)} b_{ij} x_j = x_i$$

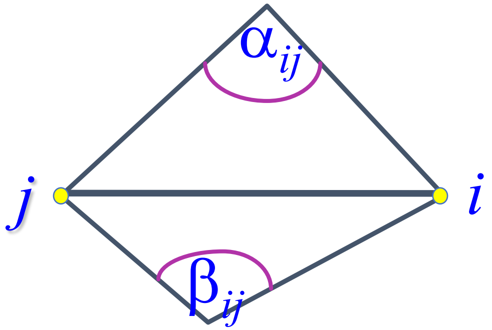
$$\sum_{j \in N(i)} b_{ij} y_j = y_i$$

For Tutte's theorem to hold, $b_{ij} > 0$

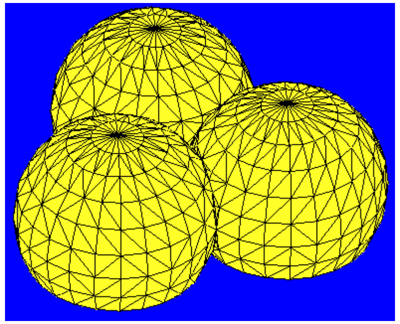
Barycentric Coordinates

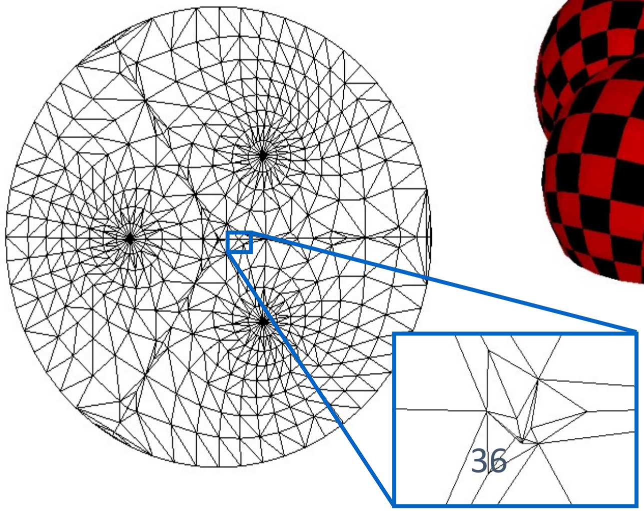
Harmonic Weights

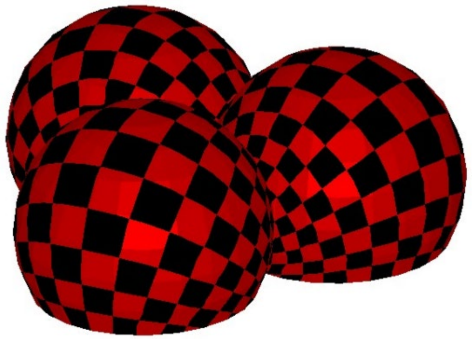
$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}$$



- Weights can be negative – not always bijective
- Weights depend only on angles - close to conformal
- 2D reproducible

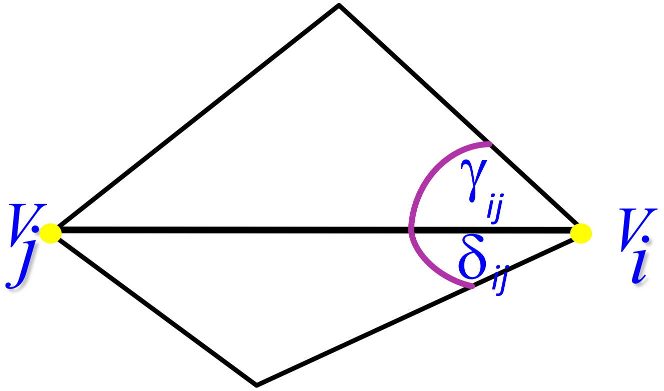




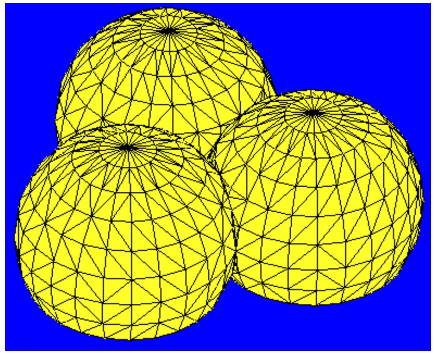


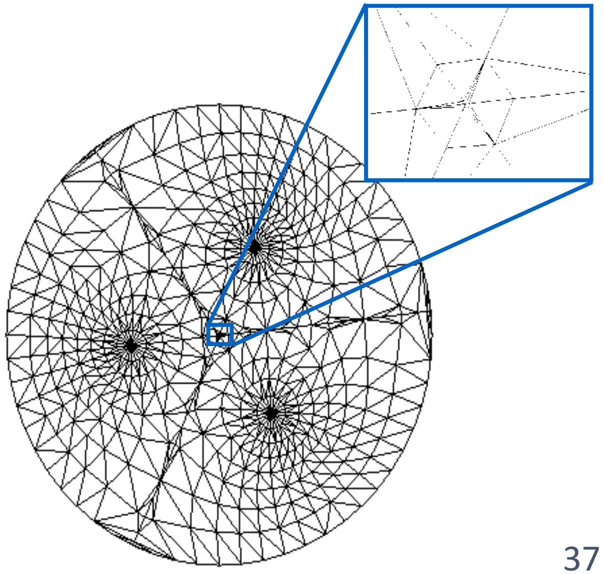
Mean-Value Weights

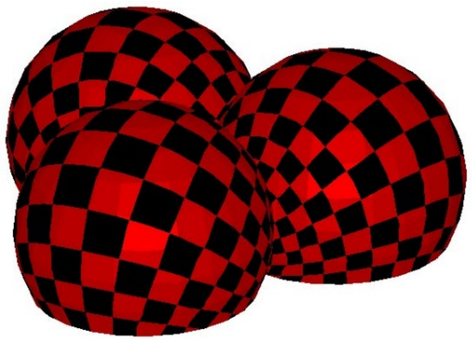
$$w_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{2\,||V_i - V_j||}$$



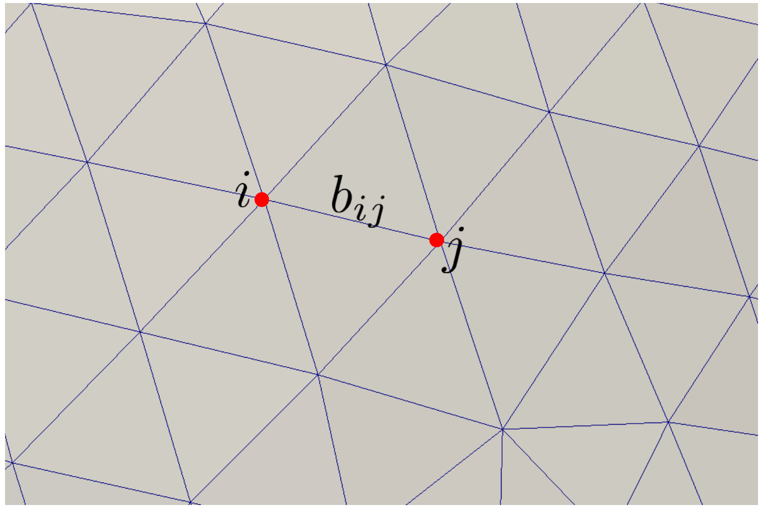
- Result visually similar to harmonic
- No negative weights – always bijective
- 2D reproducible







Recap: Algorithm with Weights



0. Pick some kind of barycentric coordinates as weights to capture geometric information.

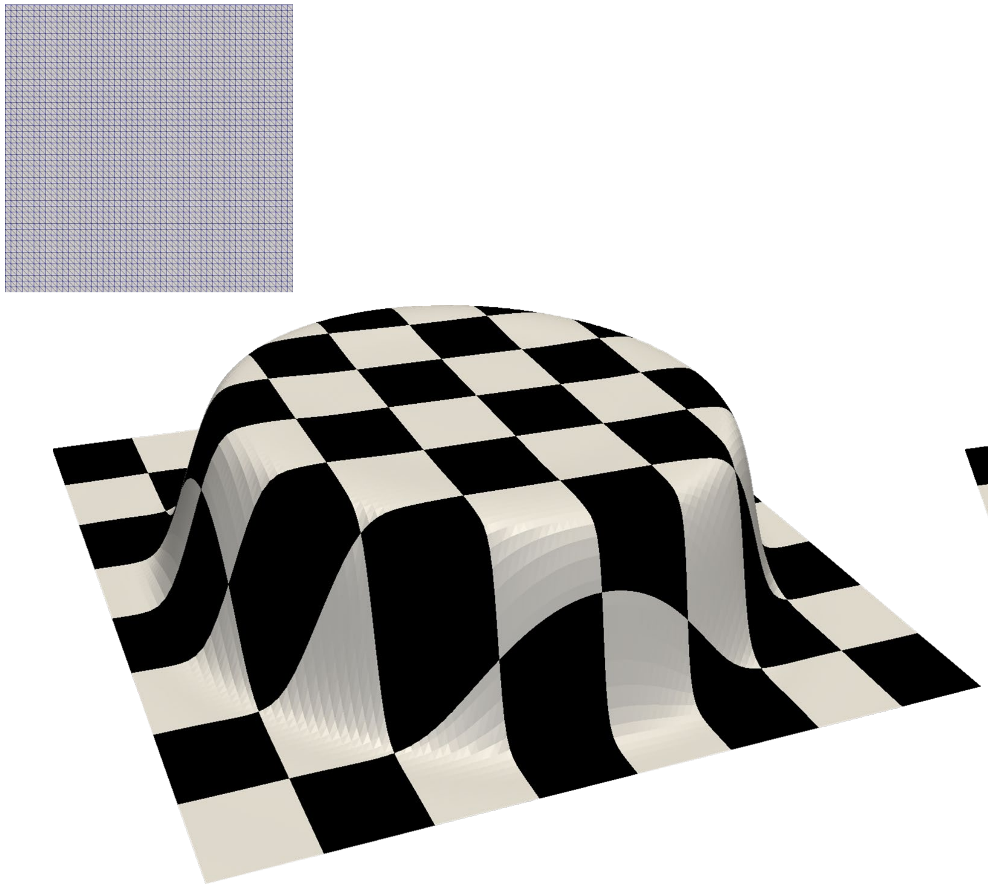
1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Implementation Note & Results

- Iterative algorithm = **Gauss-Seidel** for Ax = b
- Can solve Ax = b at once, “without” iterating!

Results



Naive

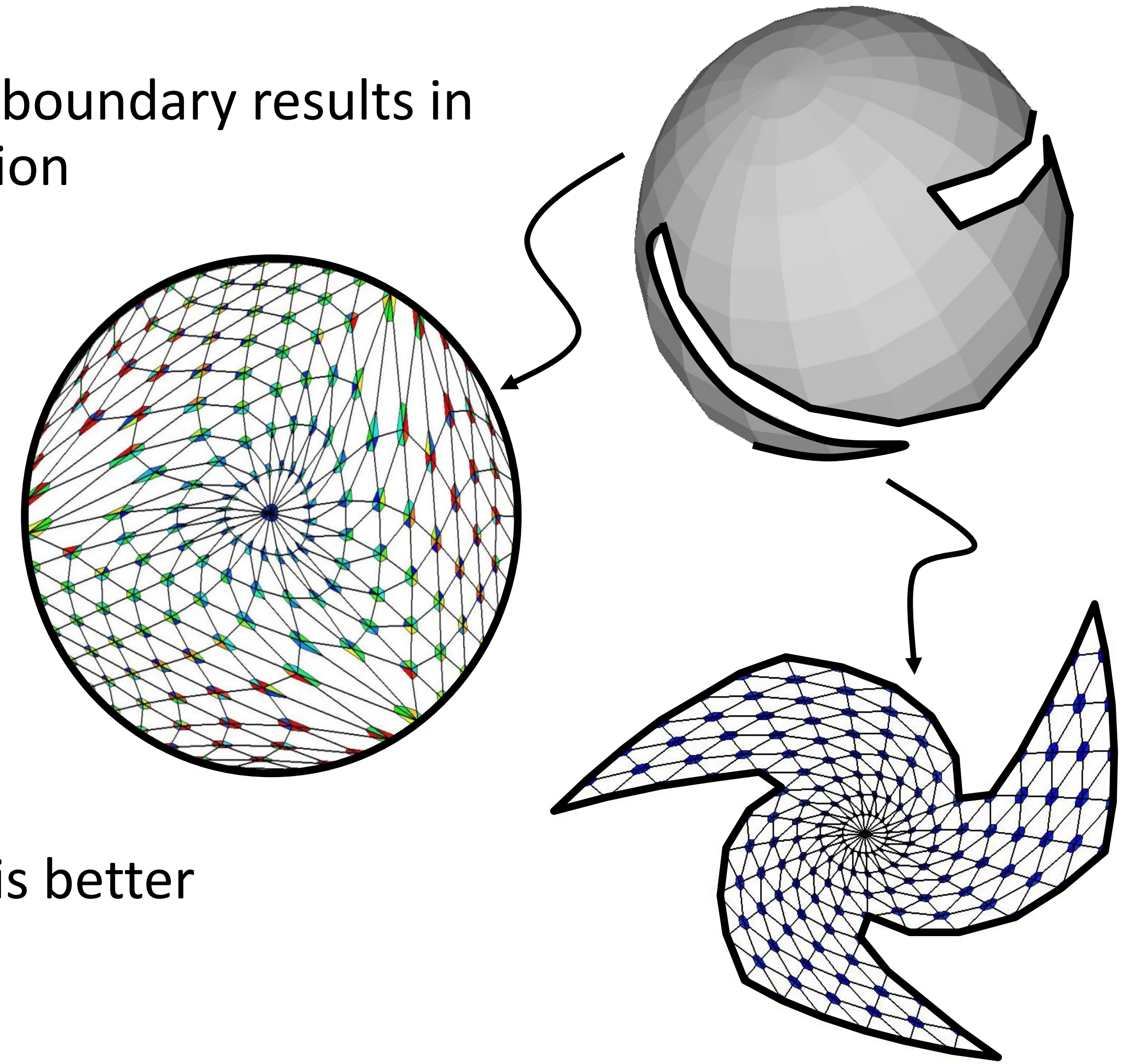
Harmonic

Outline

- Introduction
- Naïve approach and demonstration
- Distortion and desirable properties
- Fixed boundary: Harmonic parameterization
- Free-boundary: Eigenmap

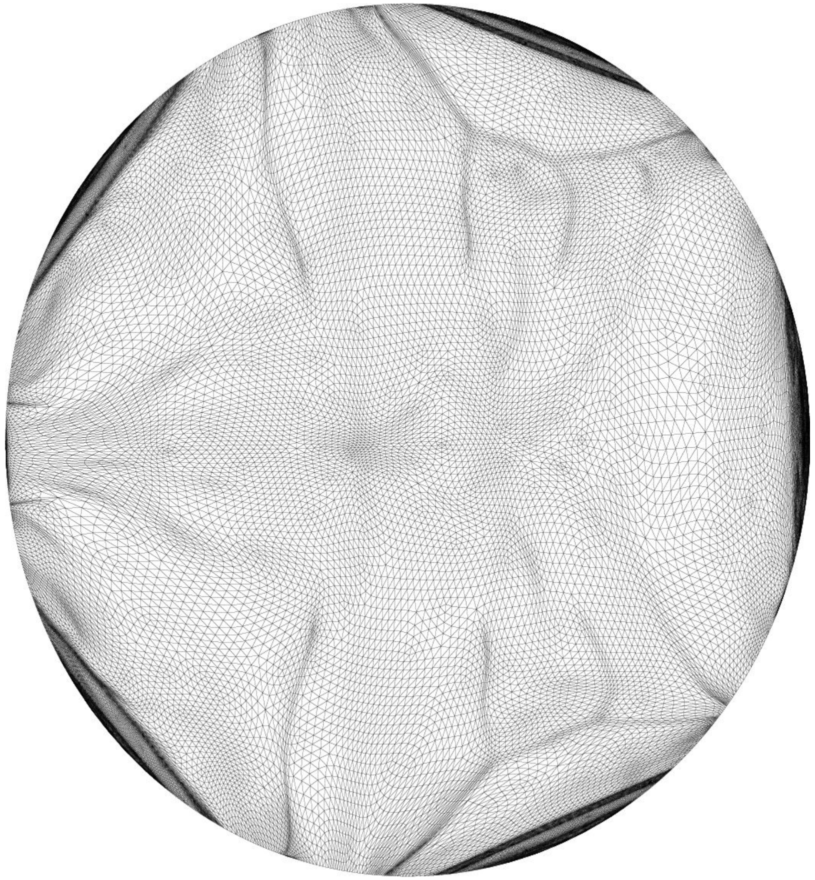
Non-Convex Boundary

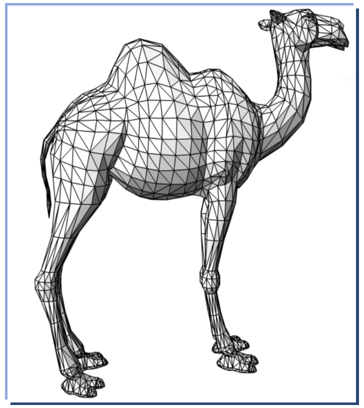
- Requiring convex boundary results in significant distortion



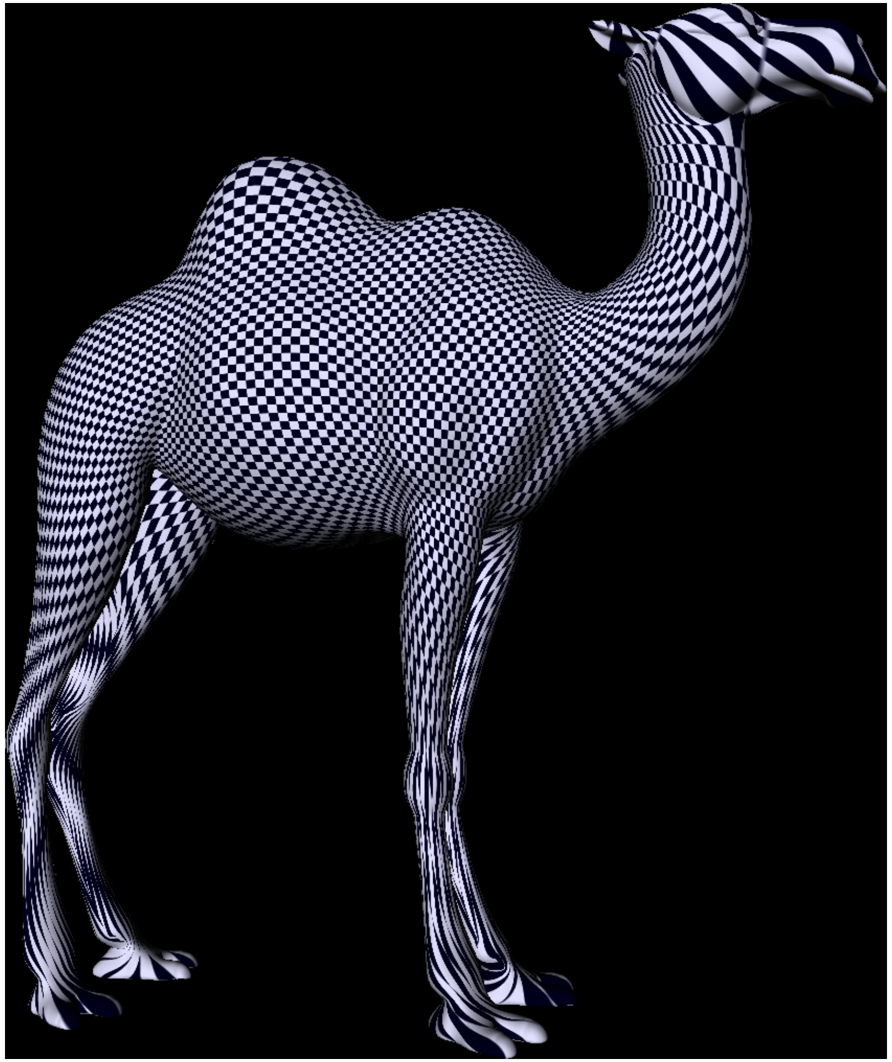
- “Free” boundary is better

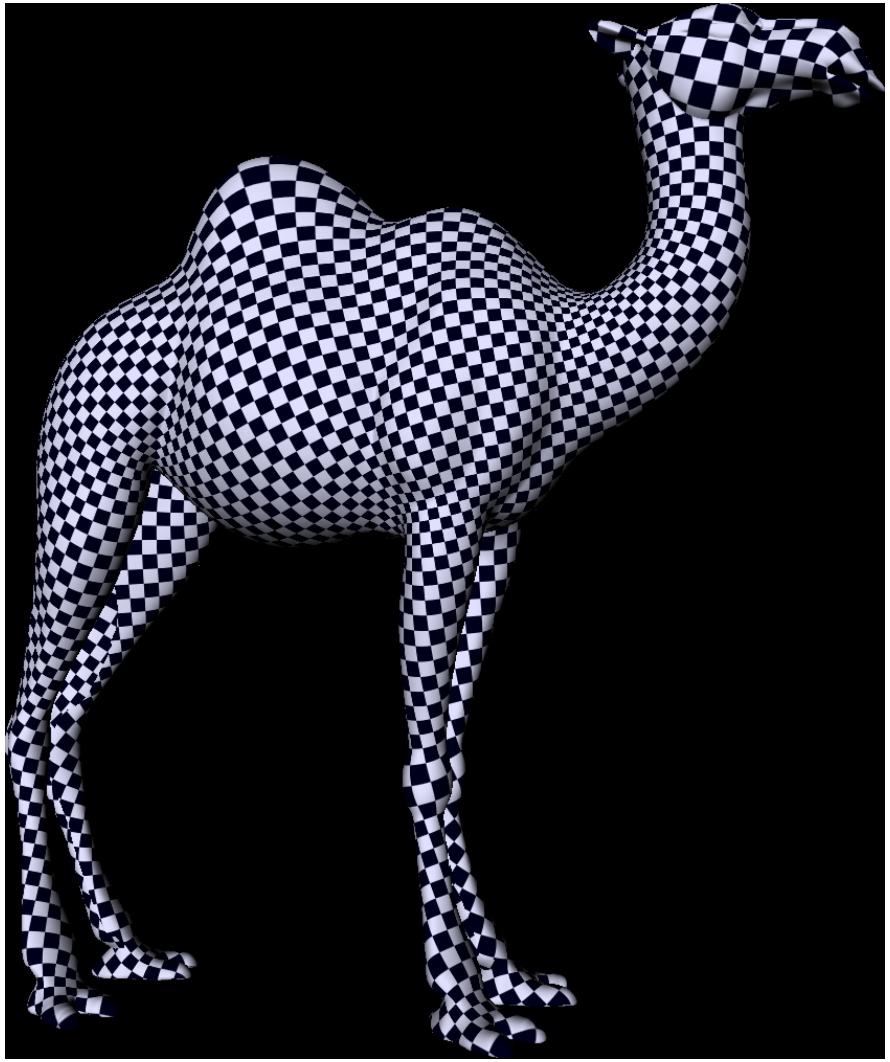
Fixed vs Free Boundary





Fixed vs Free Boundary





Old Algorithm

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Old Algorithm

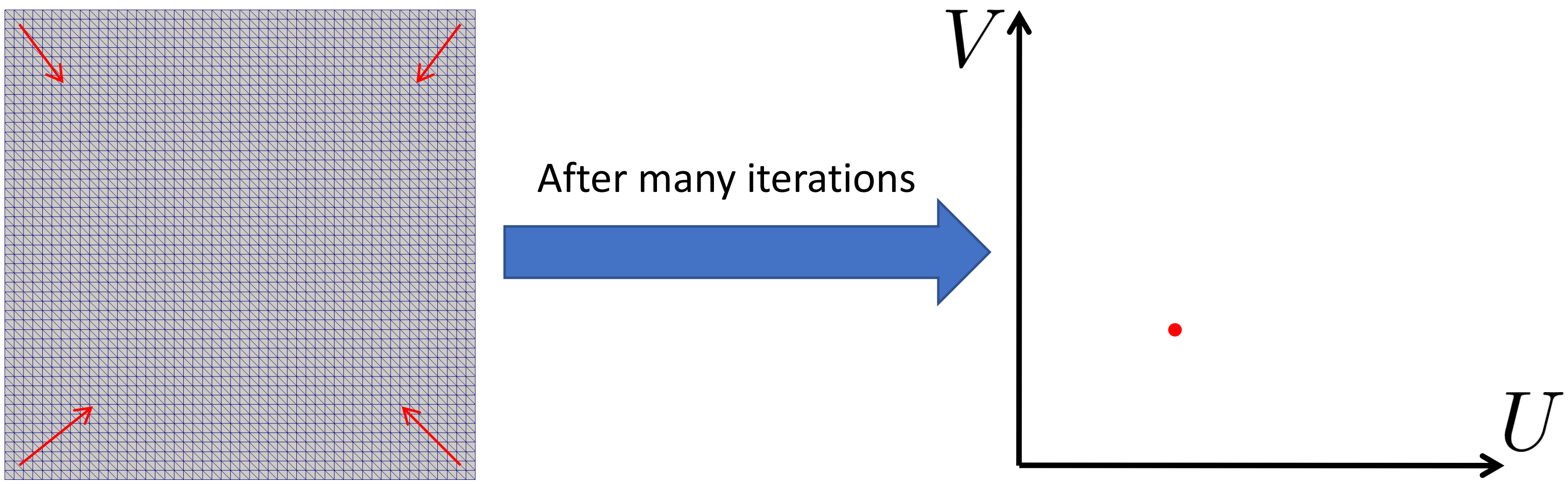
1. Initialize (u,v) of all points (e.g. using naïve).
2. While not converged: for each vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Why this would be problematic? How to fix this?

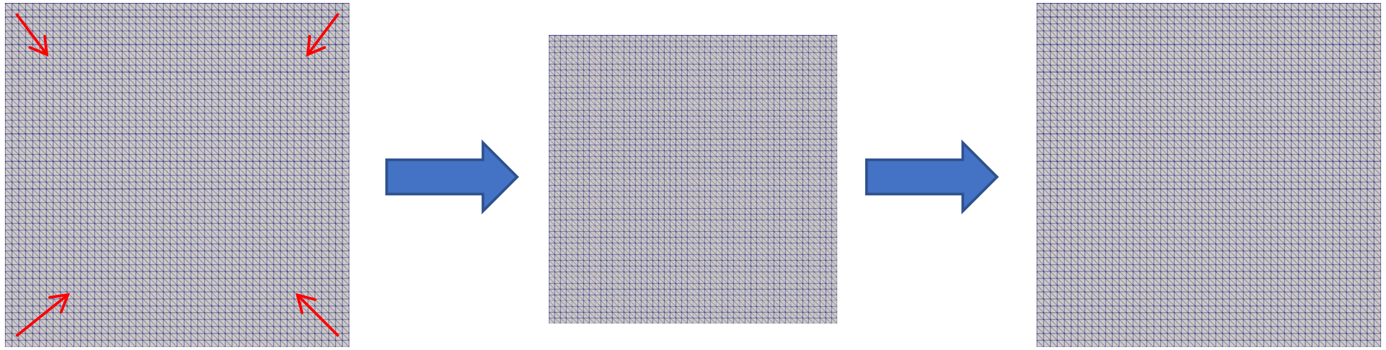
Problem

- Boundary vertices are pulled towards interior
- Shrinkage happens
- Collapse to a single point!



How to Fix this?

- Un-shrink at every iteration:
  - Move the center of mass at origin of UV-plane
  - Rescale in U direction to make std. deviation = 1
  - Rescale in V direction to make std. deviation = 1
  - Make sure covariance between U and V = 0
    - Subtract an appropriate multiple of U from V.



Fixed Algorithm

Initialize (u,v) for all vertices (e.g. using naïve)

While not converged:

    For several times:

        For each vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \quad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

        End For

    End For

    Un-shrink

End While

Fixed Algorithm = Eigenmap

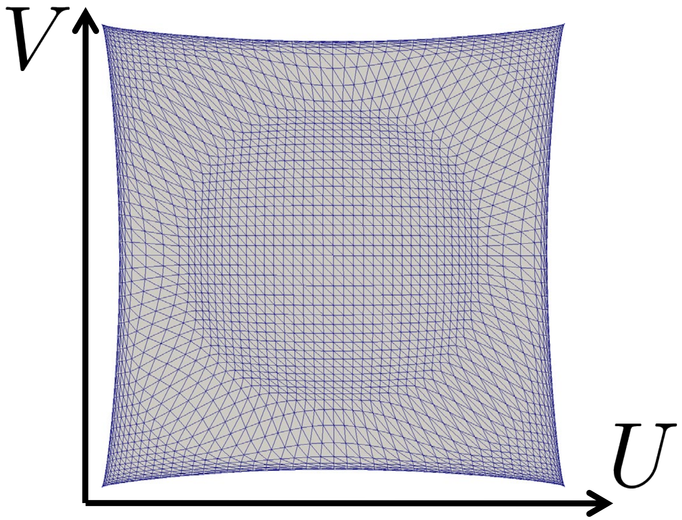
- Equivalent to inverse power iteration for solving an eigenvalue/eigenvector problem of type

$$Ax = \lambda x$$

- Pick the smallest two non-constant eigenvectors. Call these
- Set (u,v) coordinates as: $x^{(1)}, x^{(2)}$

$$u_i = x_i^{(1)} \qquad v_i = x_i^{(2)}$$

Eigenmap result



Connection Between Methods

- Fixed boundary, solve Ax=b
- Free boundary, solve Ax =0…
  - Problem: then solution x = 0!
  - Solution: solve instead eigenvalue problem

$$Ax = \lambda x$$

and pick eigenvectors corresponding to the smallest non-zero eigenvalues.

Summary

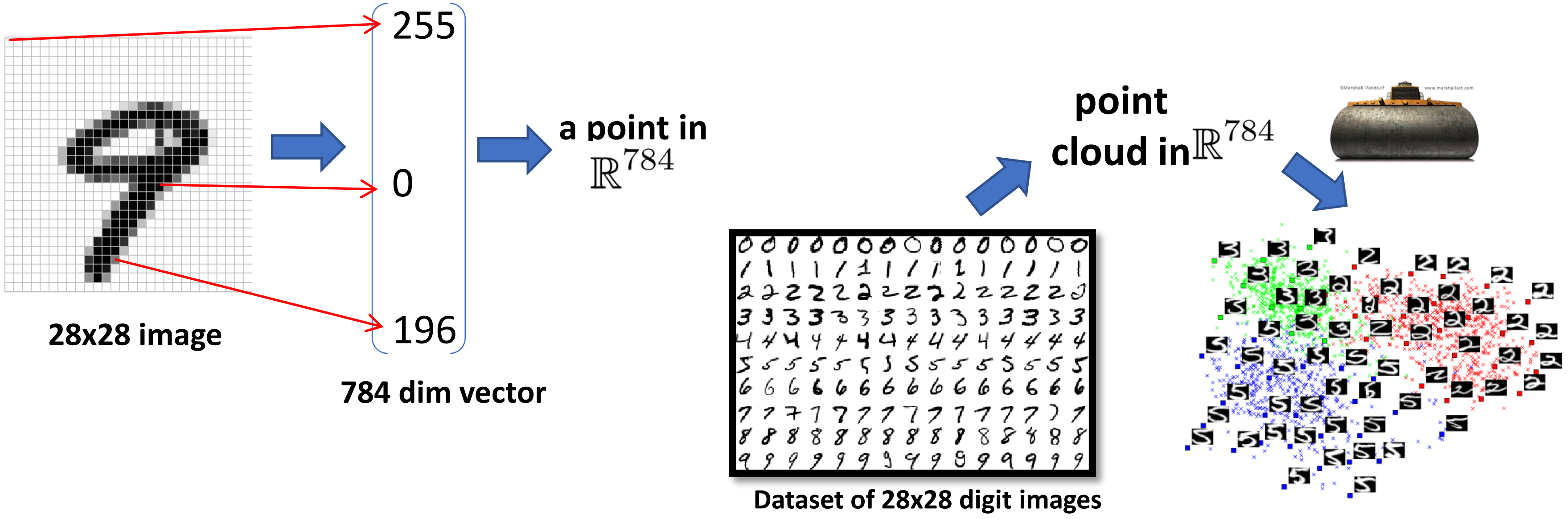
- Mesh parameterization = flattening
- Inspecting and classifying distortion:
  - Conformal/Equiareal/Isometric

- Methods
  - Fixed bndry: Harmonic parameterization
  - Free bndry: Eigenmaps

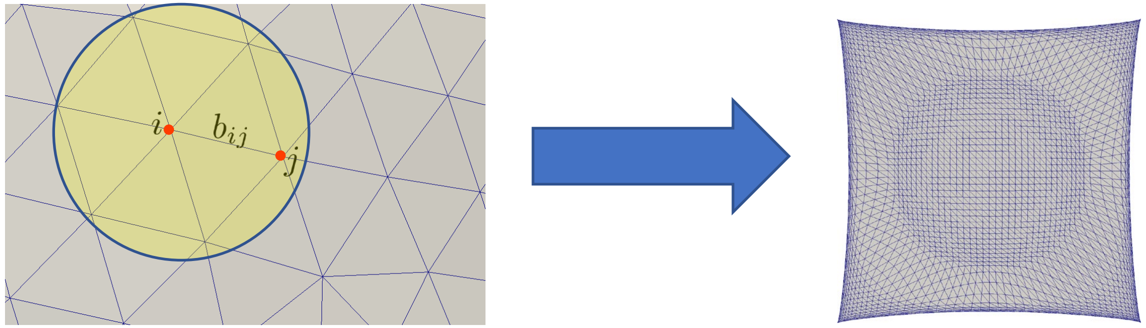
- These are easy to implement!

Connections to Other Areas – CS233

- Here: 3D reduced to 2D – "dimensionality reduction"
- Look up: "non-linear dimensionality reduction"
- Ways of organizing/visualizing high dim data



Food for Thought



Local info integrated into **global** embedding

Huge impact in geometry processing,
machine learning, and sensor networks.

End