Intro. to Geometry Processing 600.756

http://www.cs.jhu.edu/~misha/Spring17a/

Assignment 1

Nice job!

- Wanted to have subdivision independent of smoothing
  (Should have made that clearer)

Questions:

- Can you sharpen by using a value less than zero?
- Can you get huge amounts of smoothing by using a value greater than one?
- How does the triangulation affect the results?

Assignment 1

- The command line parameters come (templated) as different types:
  - Base: defines a `set` member
    - `cmdLineReadable`: a flag that is on if the command line argument was specified
  - Inherited: typically has a member `value` or `values`
    - `cmdLineParameter< char* >`: a string
    - `cmdLineParameter< int >`: an (32-bit) integer value
    - `cmdLineParameter< float >`: a (32-bit) floating point value
    - …

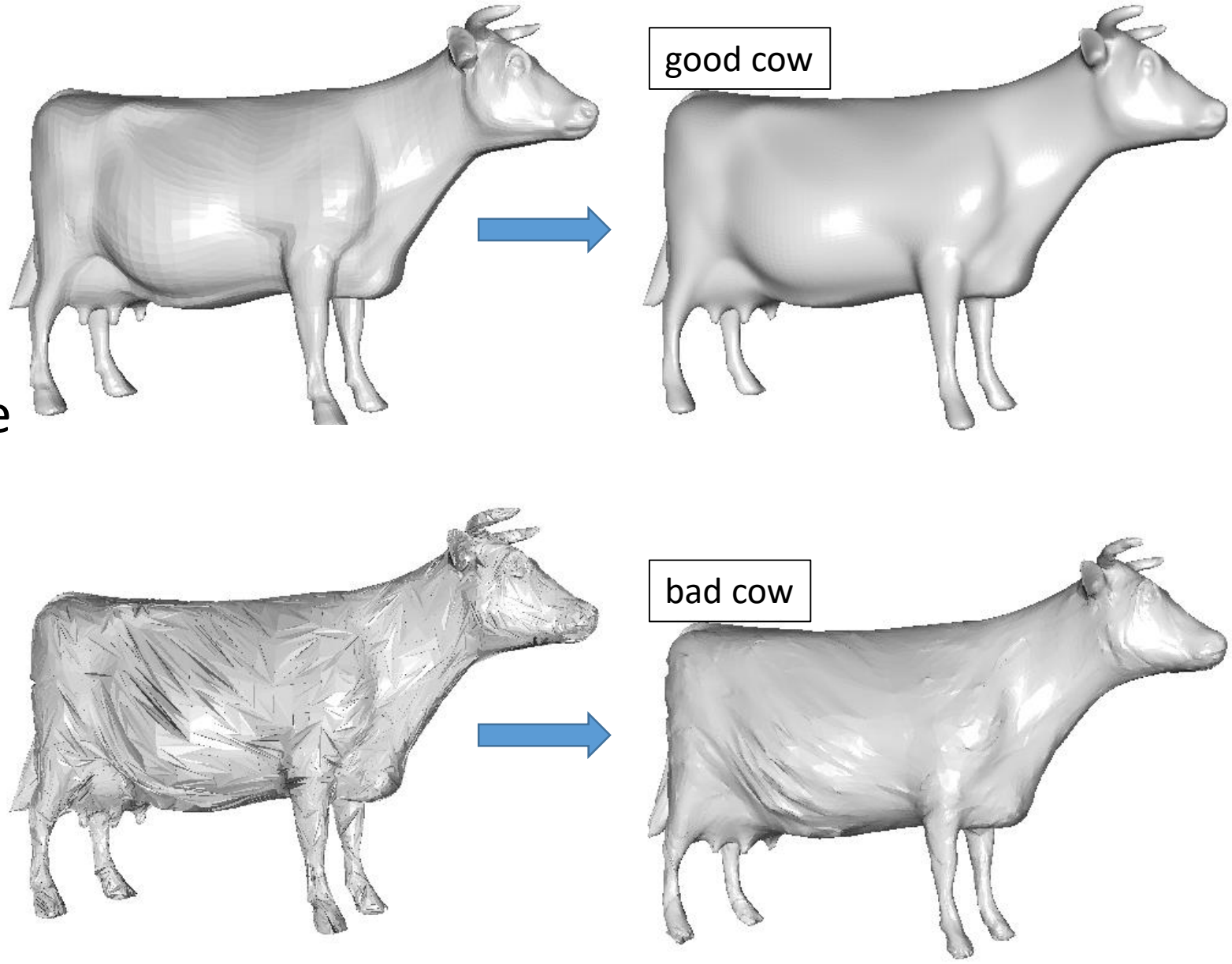
    Many of these can be constructed with both a name and a default value

Assignment 1

Results:

- Both (good/bad) models have vertices in the same locations
- The smoothing results look quite different

⇒ How this smoothing behaves depends on tessellation

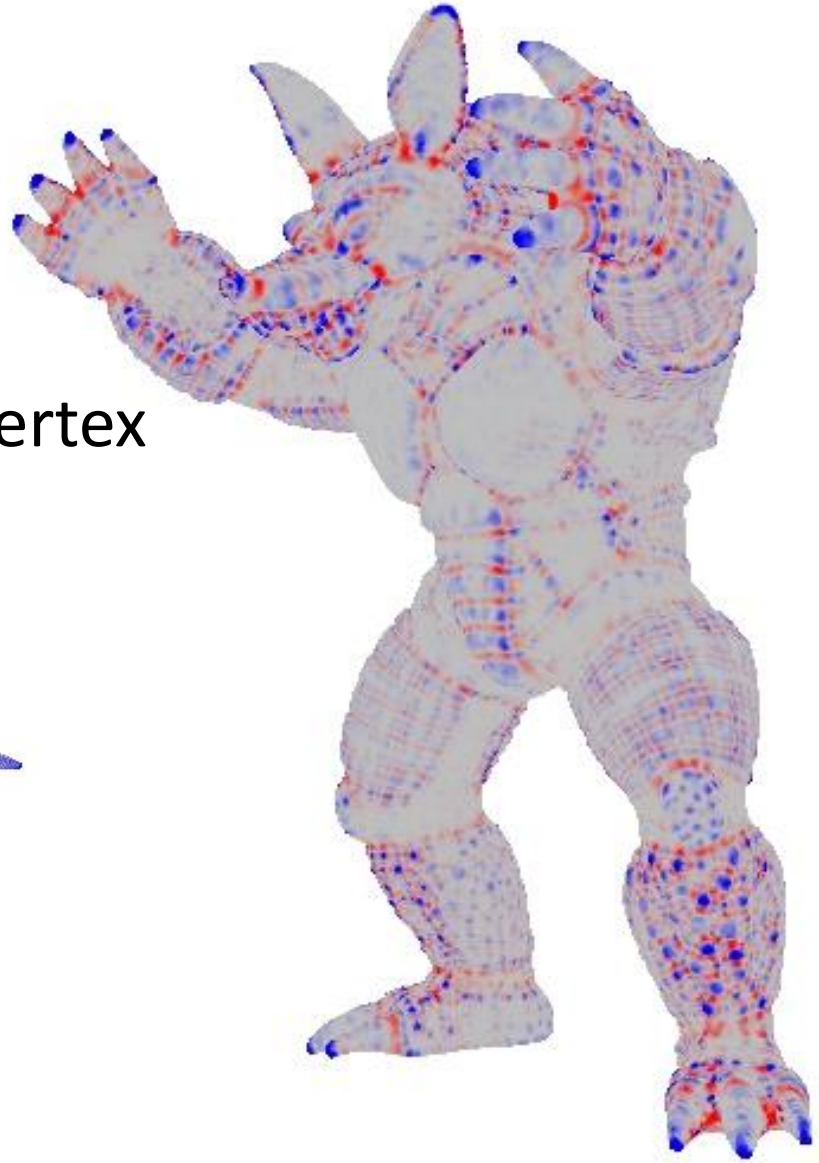


iters = 10, blending weight = 1

Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
- Compute the integrated Gaussian curvature

```
curvature --out o.ply --k 0 --in raptor.ply
```



```
curvature --out o.ply --k 1 --in armadillo.ply
```

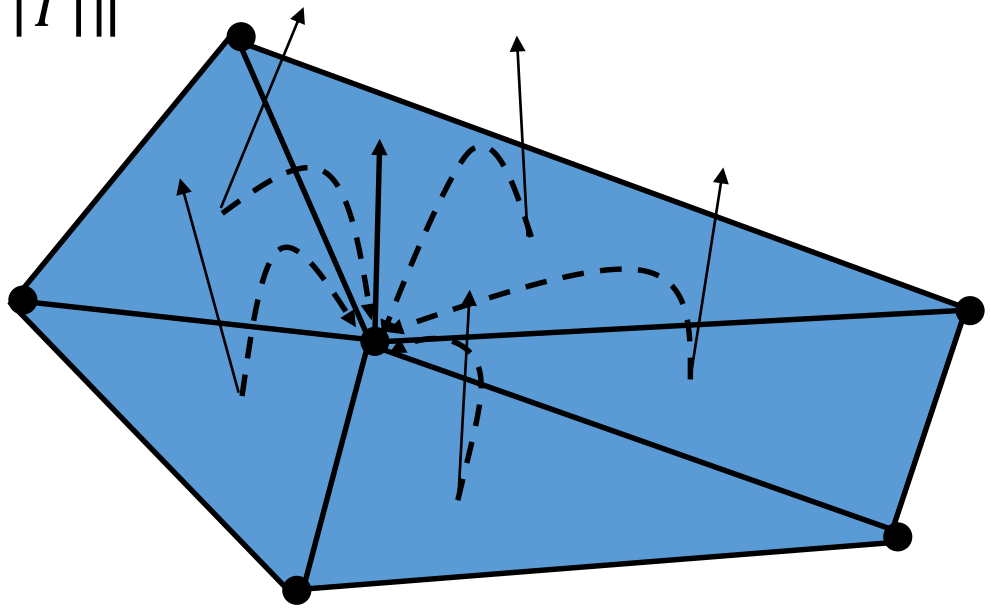
Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle

Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle
    - Compute the normal at each vertex:

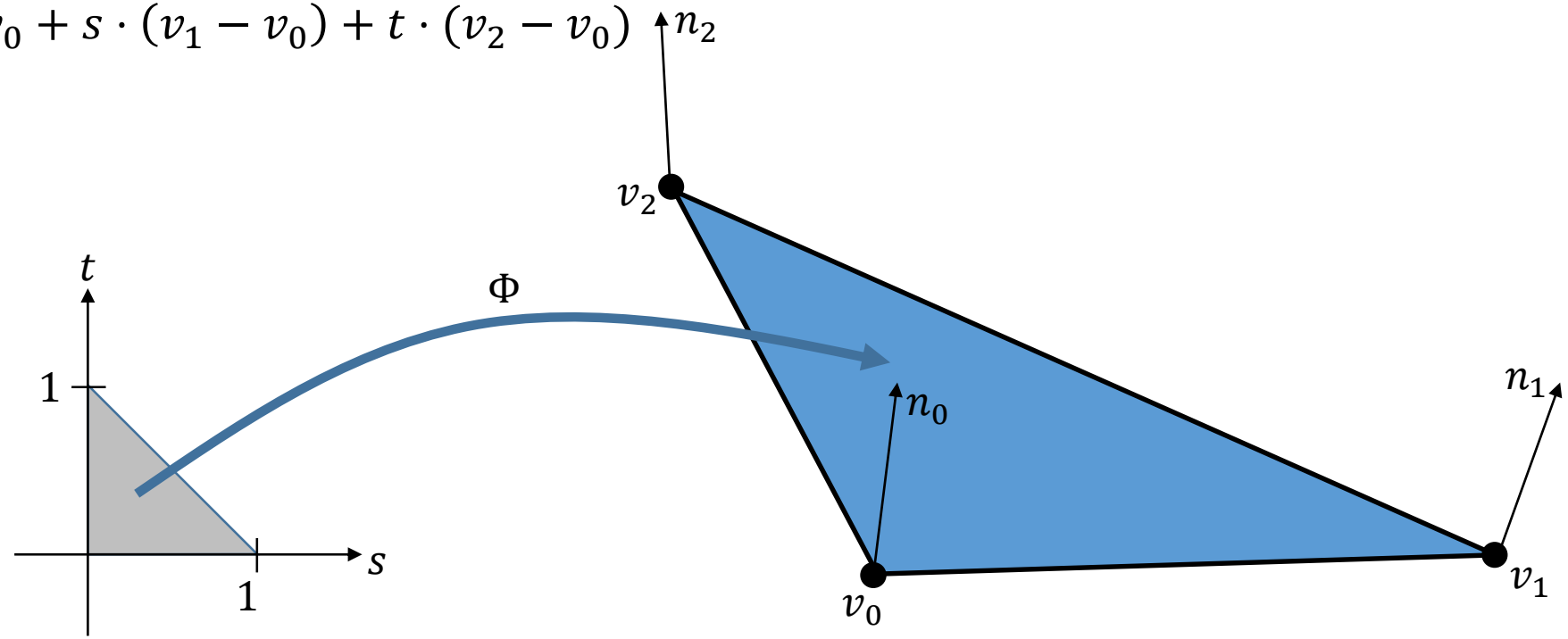
$$\vec{n}(v) = \frac{\sum_{T \ni v} \vec{n}(T) \cdot |T|}{\left\| \sum_{T \ni v} \vec{n}(T) \cdot |T| \right\|}$$



Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle
    - Compute the normal at each vertex
    - Define a parameterization from the unit right triangle to the triangle in 3D:

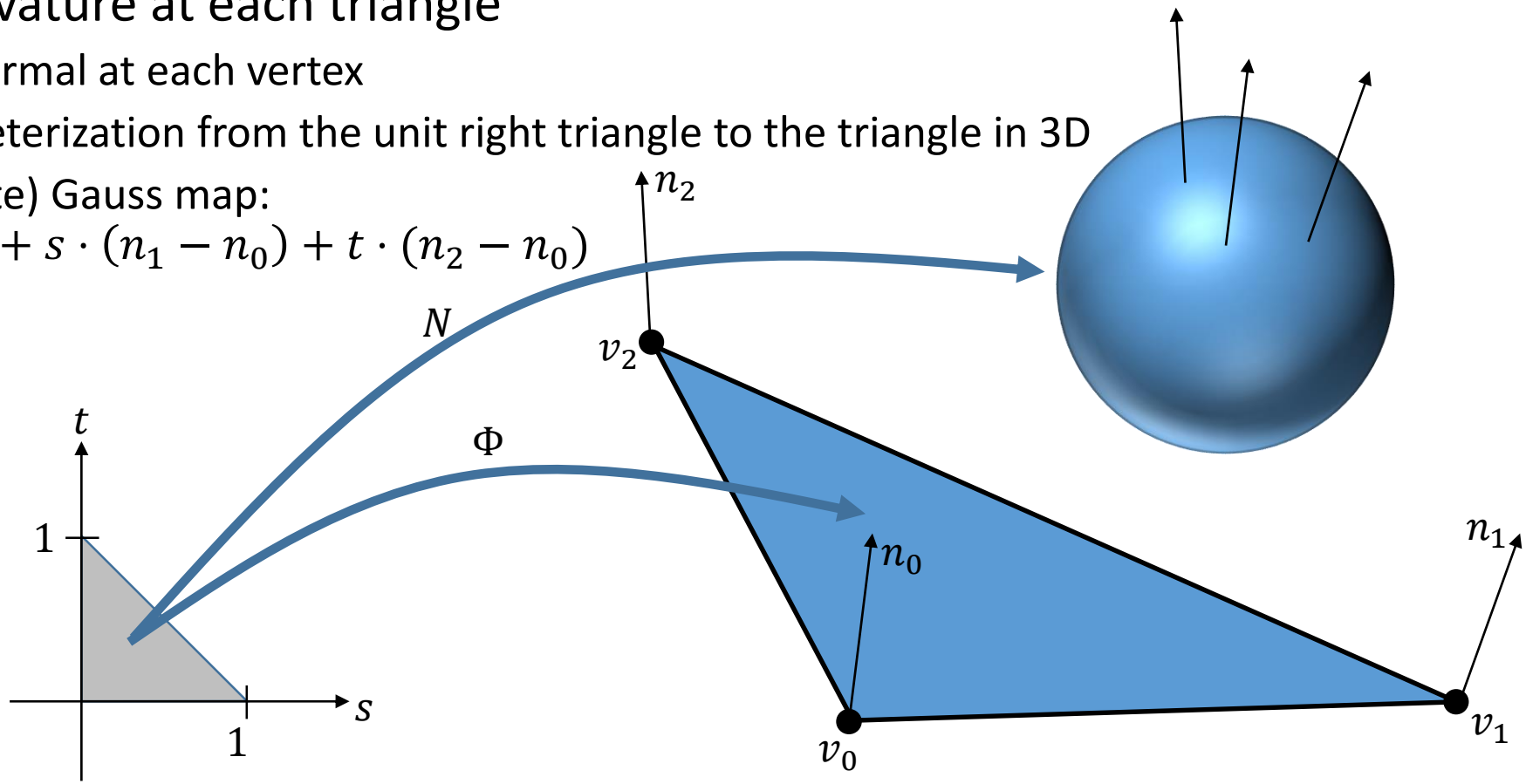
$$\Phi(s,t) = v_0 + s \cdot (v_1 - v_0) + t \cdot (v_2 - v_0)$$



Assignment 2 (curvature)

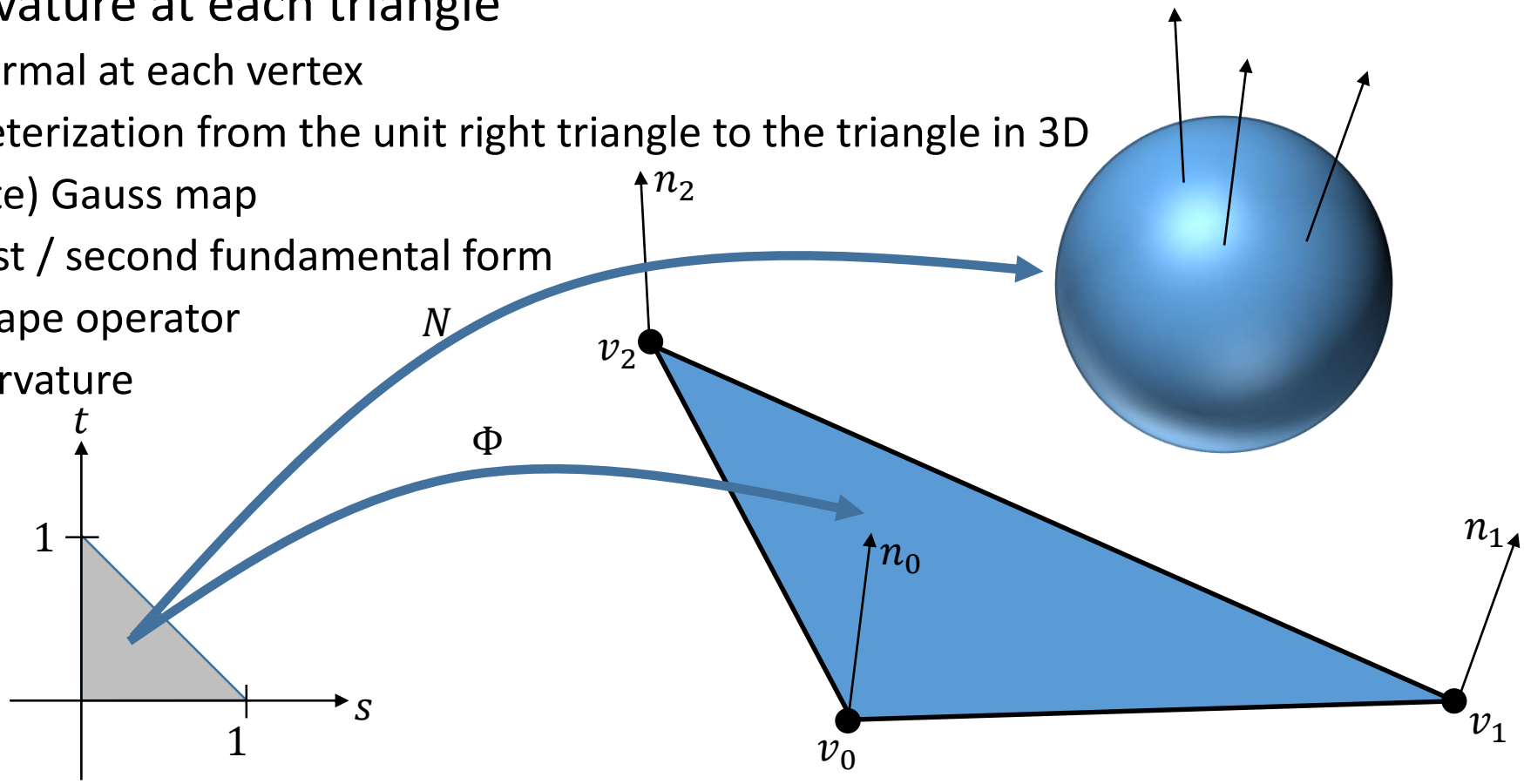
- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle
    - Compute the normal at each vertex
    - Define a parameterization from the unit right triangle to the triangle in 3D
    - Define a (discrete) Gauss map:

$$N(s,t) = n_0 + s \cdot (n_1 - n_0) + t \cdot (n_2 - n_0)$$



Assignment 2 (curvature)

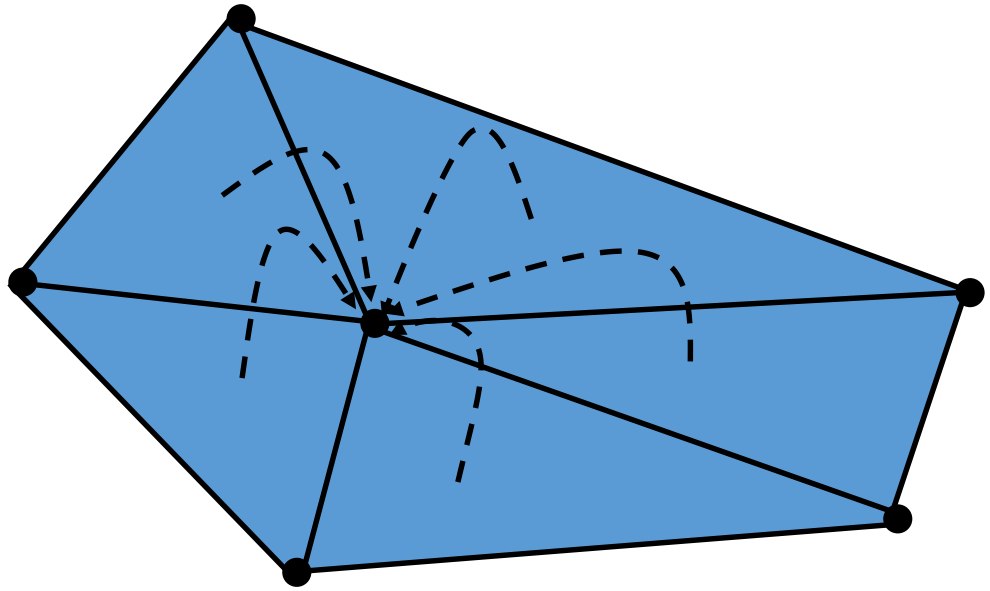
- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle
    - Compute the normal at each vertex
    - Define a parameterization from the unit right triangle to the triangle in 3D
    - Define a (discrete) Gauss map
    - Compute the first / second fundamental form
    - Compute the shape operator
    - Compute the curvature



Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
  - Compute the curvature at each triangle
  - Compute the curvature at each vertex
    - (Area-weighted) average the curvatures from adjacent triangles

$$c(v) = \frac{\sum_{T \ni v} c(T) \cdot |T|}{\sum_{T \ni v} |T|}$$



Assignment 2 (curvature)

- Compute mean / Gaussian curvature at each vertex
- Compute the integrated Gaussian curvature, I:
  - You can do this using the curvature estimated at the triangles

$$I = \sum_{T} c(T) \cdot |T|$$

  - What happens if you refine?
  - What happens if you remap $I \rightarrow 1 - \frac{I}{4\pi}$?

Assignment 2 (curvature)

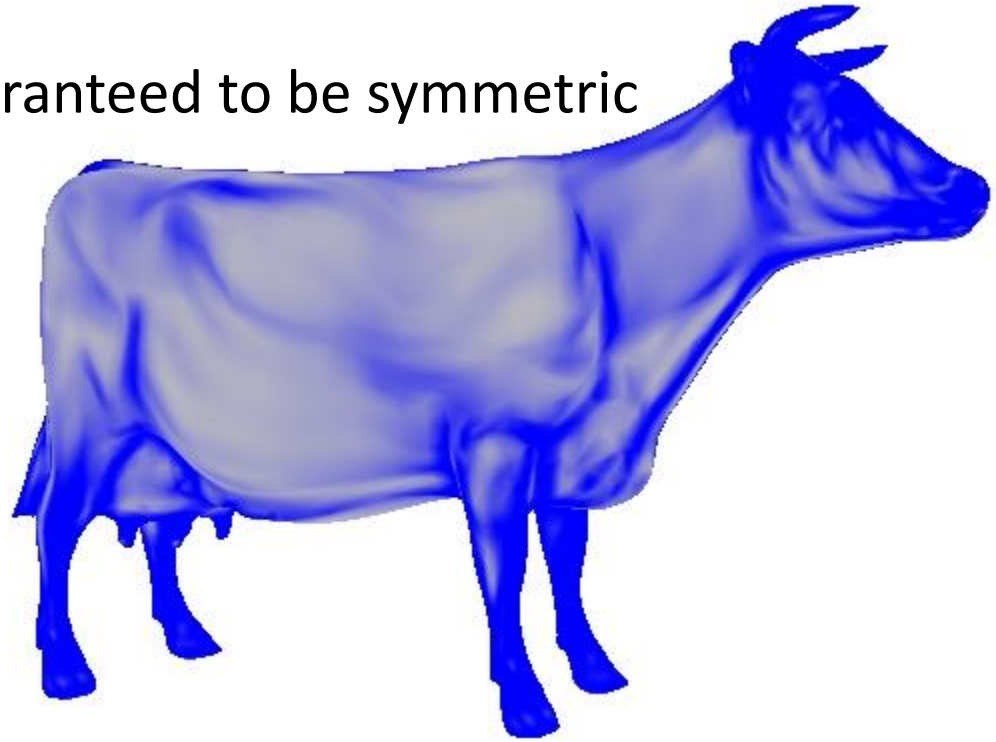
- Recommendation:
  - Use `SquareMatrix< float , 2 >` to represent 2×2 matrices.
    - This class supports standard matrix operations including multiplication, inversion, transposition, computing the trace and the determinant, etc.

Assignment 2 (curvature)

- Extra credit 1:
  - Compute the two principal curvature values and display the difference between the maximum and the minimum*

  [WARNING]:
  The discrete second fundamental form is not guaranteed to be symmetric so the eigenvalues may be imaginary!

  You can fix this by replacing the computed second fundamental form with the closest symmetric matrix



```
curvature --out o.ply --k 2 --in cow.ply
```

*You may need to switch to double-precision

Assignment 2 (curvature)

- Extra credit 2:
  - Compute the principal curvature directions and curvature values*

  [WARNING]:
  The discrete second fundamental form is not guaranteed to be symmetric so even if the eigenvalues are real, the eigenvectors may not be orthogonal!

  You can fix this by replacing the computed second fundamental form with the closest symmetric matrix

*This will (almost) require performing an eigen-decomposition

Assignment 2 (curvature)

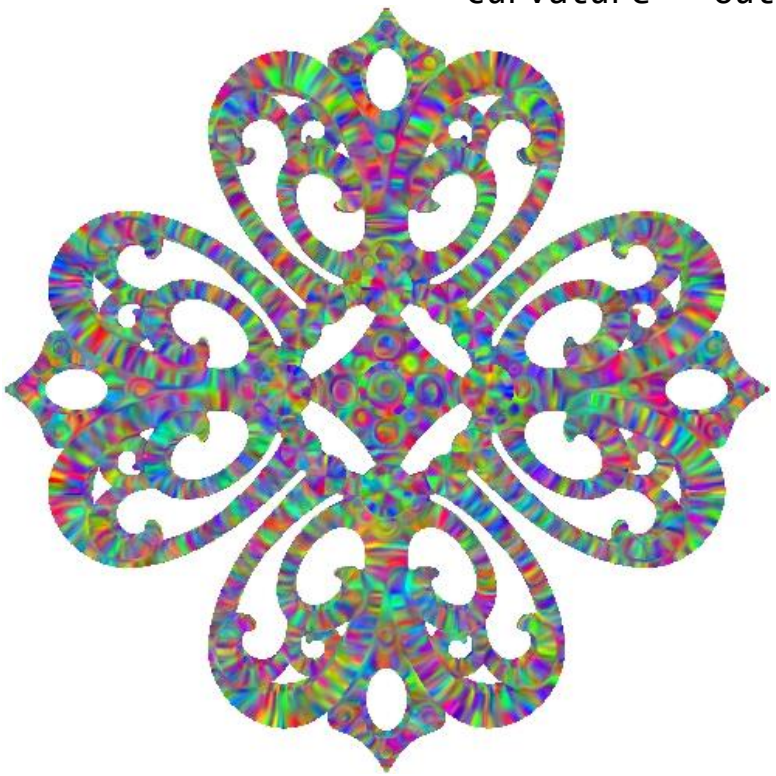
- Extra credit 2:
  - Compute the principal curvature directions and curvature values
    - To visualize this, write out your principal curvature directions and principal curvature values into the struct `CurvatureInfo`, with the first value/direction corresponding to the direction with largest (signed) curvature
    - This information will get written out with the `.ply` file
    - The `LIC` executable will take a `.ply` file with curvature information and generate a color per vertex, with color lines following the (largest) principal curvature direction

```
struct CurvatureInfo
{
	float k1 , k2; // k1 >= k2
	Point3D< float > dir1 , dir2;
	…
}
```

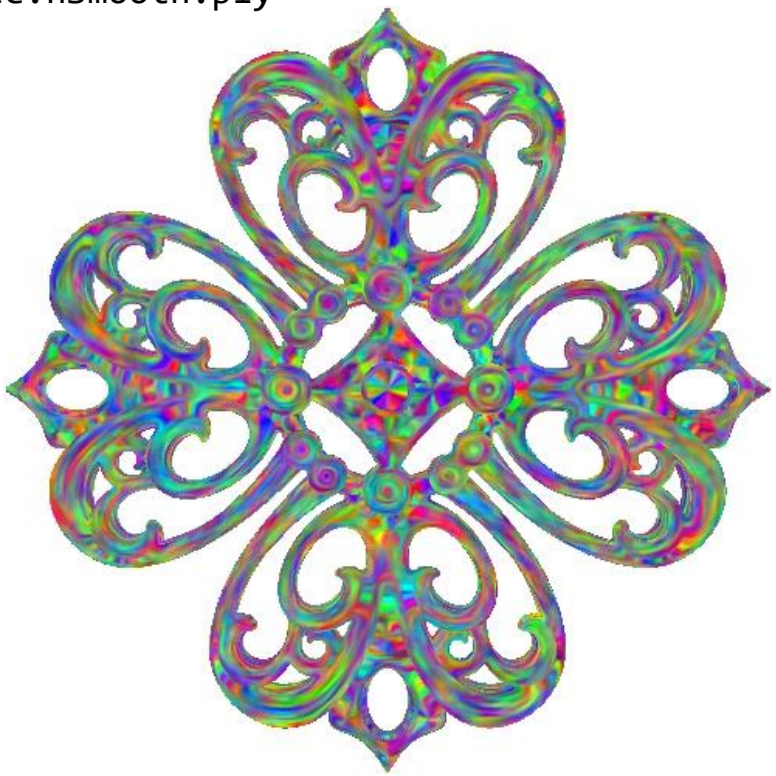
Assignment 2 (curvature)

- Extra credit 2:
  - Compute the principal curvature directions and curvature values

```
curvature --out o.ply --k 3 --in filigree.nSmooth.ply
```



```
LIC --in o.ply --out k.ply --stepSize 1e-7 --brighten 5
```

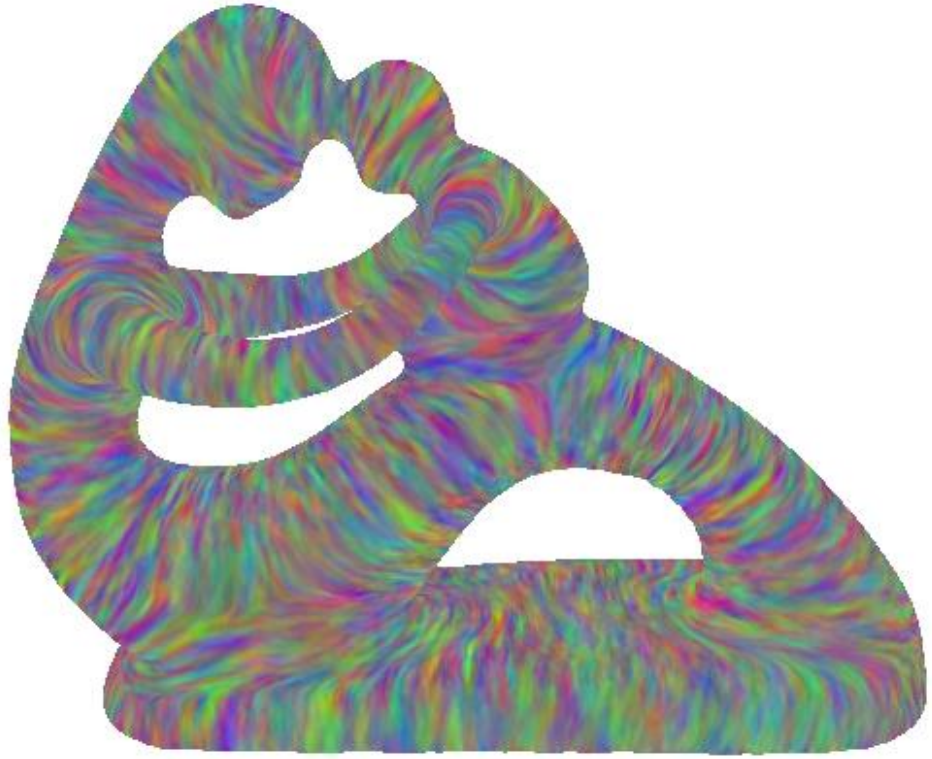


```
LIC --in o.ply --out k.ply --stepSize 1e-7 --brighten 5 --smallest
```

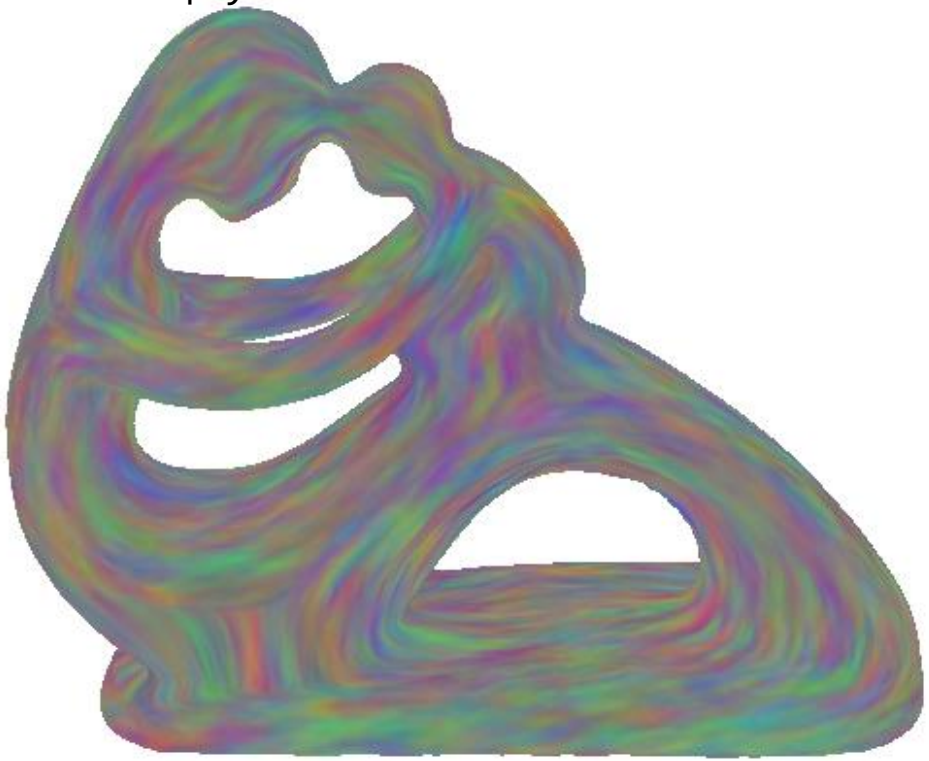
Assignment 2 (curvature)

- Extra credit 2:
  - Compute the principal curvature directions and curvature values

```
curvature --out o.ply --k 3 --in fertility.nSmooth.ply --refine 1
```



```
LIC --in o.ply --out k.ply --stepSize 1e-6 --brighten 5
```

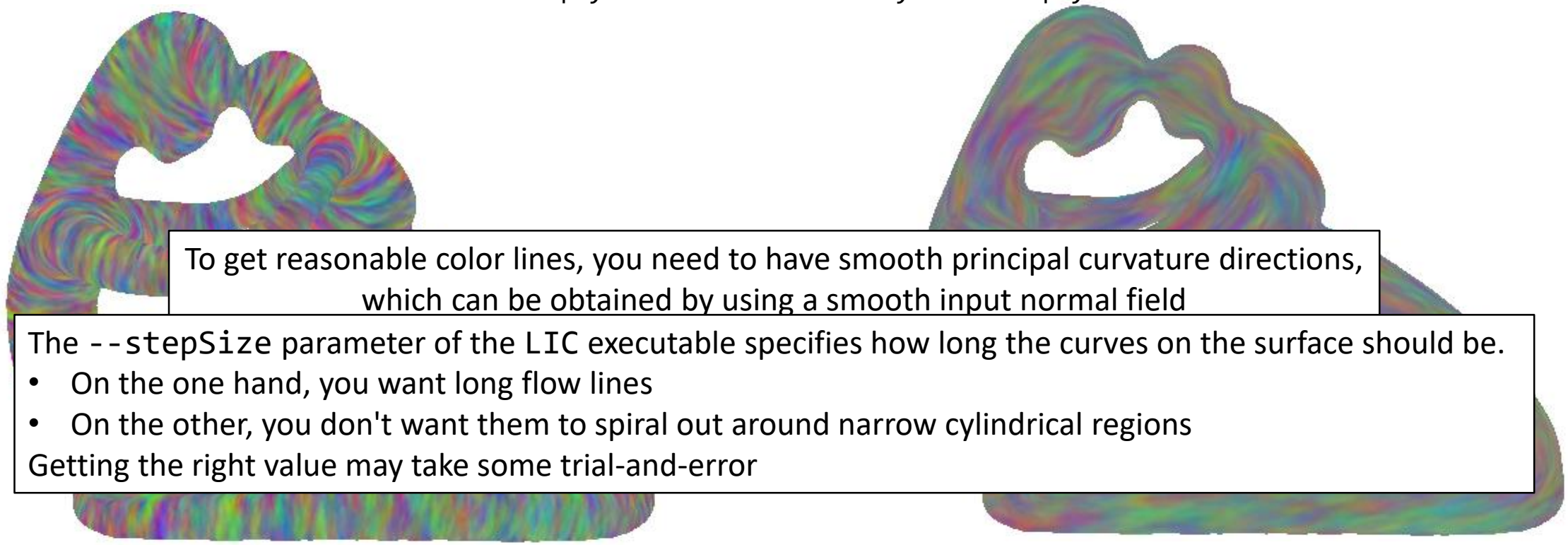


```
LIC --in o.ply --out k.ply --stepSize 3e-6 --brighten 5 --smallest
```

Assignment 2 (curvature)

- Extra credit 2:
  - Compute the principal curvature directions and curvature values

```
curvature --out o.ply --k 3 --in fertility.nSmooth.ply --refine 1
```



To get reasonable color lines, you need to have smooth principal curvature directions, which can be obtained by using a smooth input normal field

The `--stepSize` parameter of the `LIC` executable specifies how long the curves on the surface should be.

- On the one hand, you want long flow lines
- On the other, you don't want them to spiral out around narrow cylindrical regions

Getting the right value may take some trial-and-error

```
LIC --in o.ply --out k.ply --stepSize 1e-6 --brighten 5
```

```
LIC --in o.ply --out k.ply --stepSize 3e-6 --brighten 5 --smallest
```