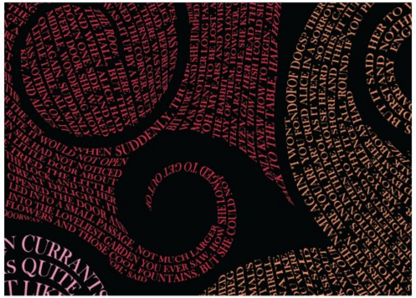
Digital Micrography

Ron Maharik
Mikhail Bessmeltsev
University of British Columbia

Alla Sheffer
University of British Columbia
INRIA Rhône-Alpes

Ariel Shamir
The Interdisciplinary Center

Nathan Carr
Adobe Systems Incorporated

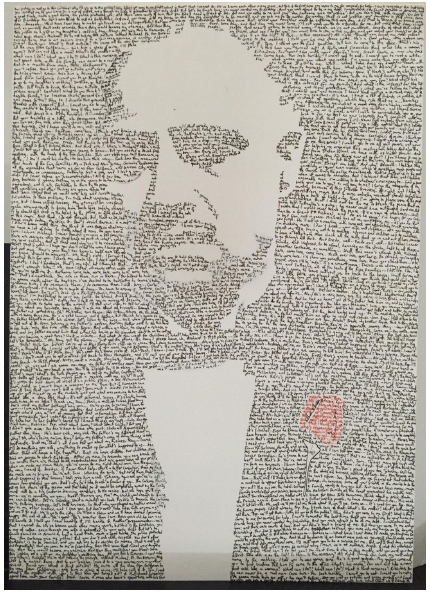


“Micrography”?

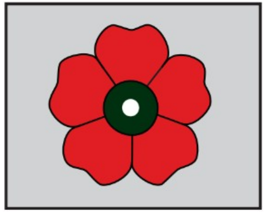
Calligrams

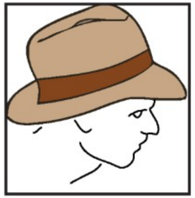
Calligraphic Packing by Jie Xu Craig S. Kaplan 2007

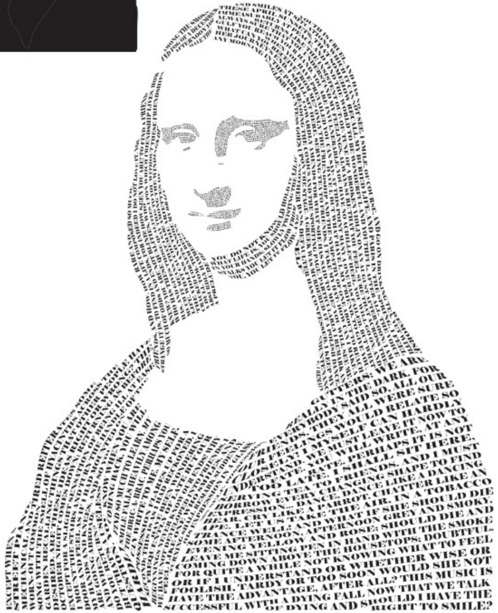
Micrograms

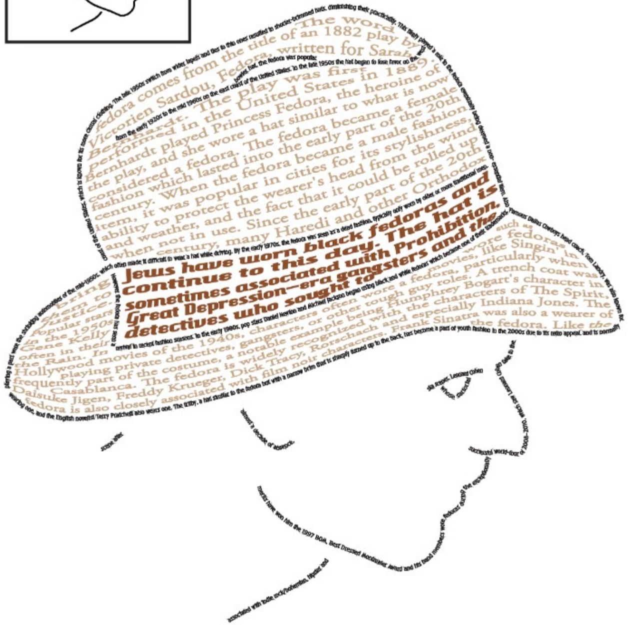


Vector images + Text → Microgram





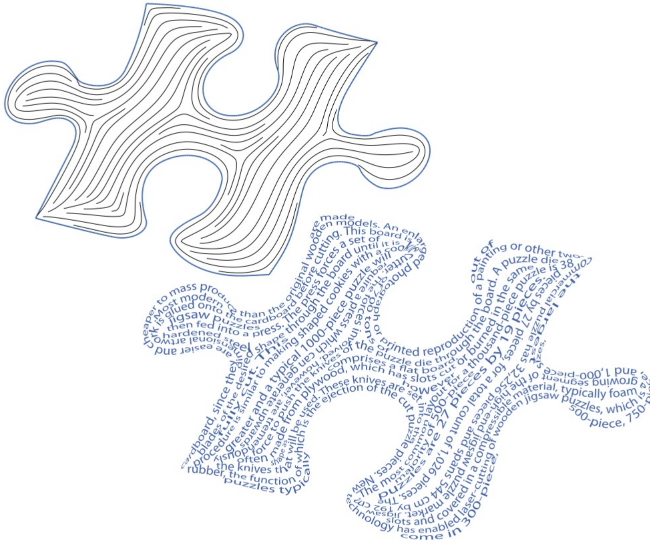




Readable Text

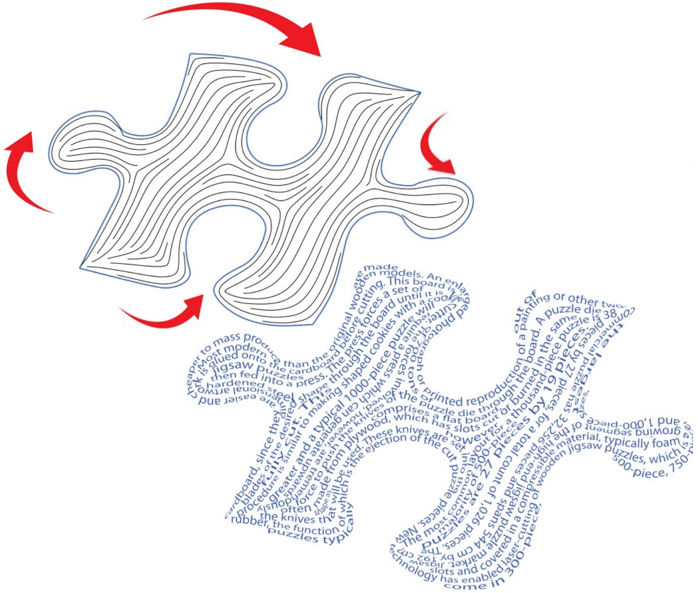
First attempt

Forcing alignment to boundaries



First attempt

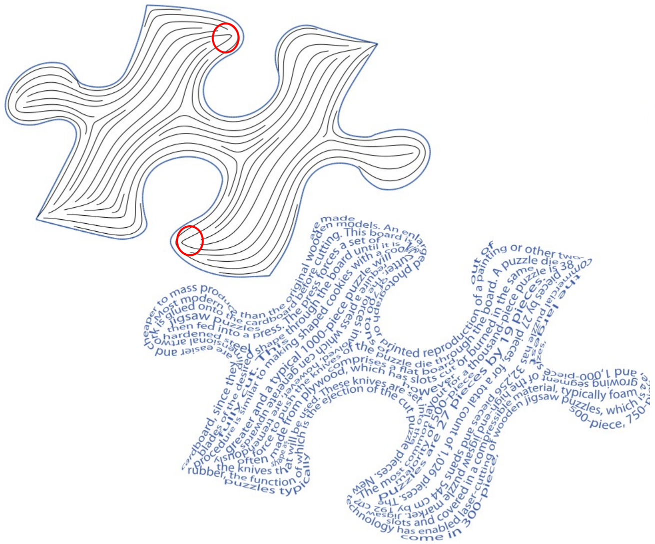
Forcing alignment to boundaries



No orientation

First attempt

Forcing alignment to boundaries

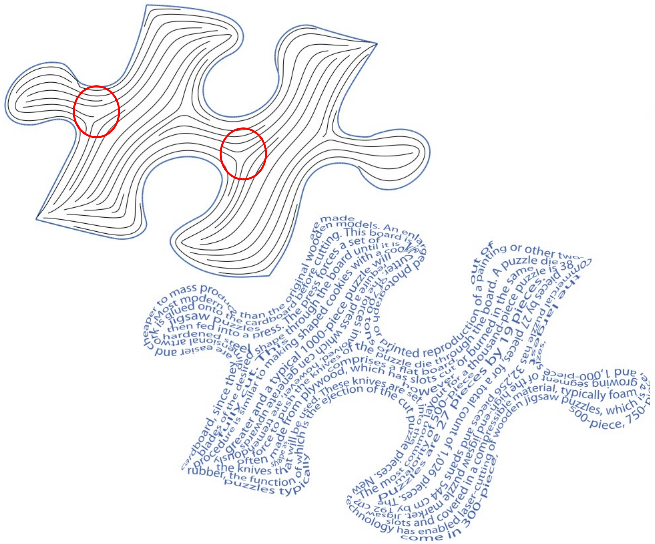


No orientation

High Curvature

First attempt

Forcing alignment to boundaries

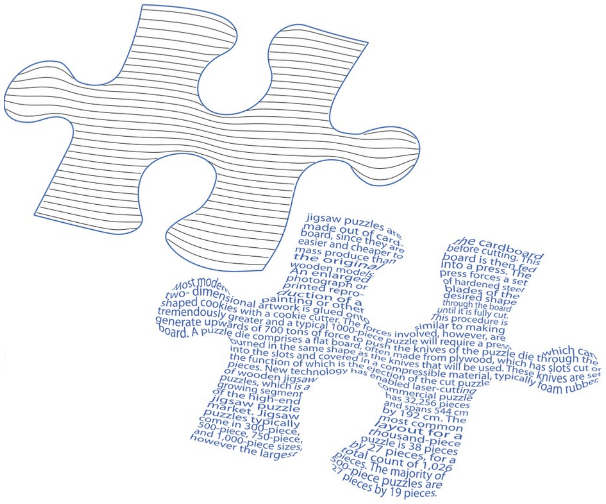


No orientation

High Curvature

Singularities

With smart alignment constraints

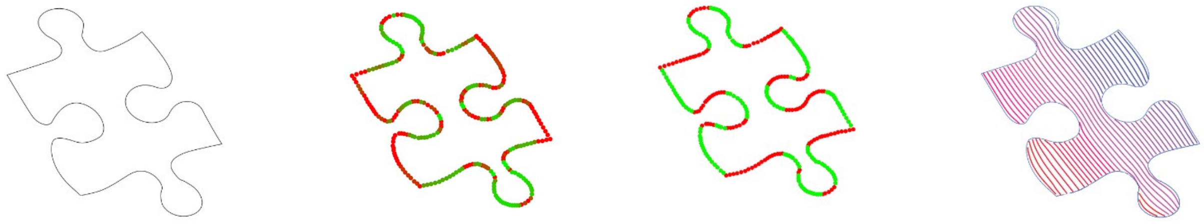


Orientation

Low Curvature

No Singularities

Algorithm pipeline

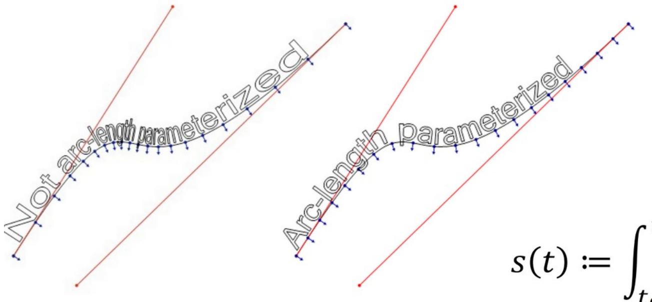


A B C D

Algorithm pipeline

PARAMETERIZATION BY ARC LENGTH

http://www.planetclegg.com/projects/WarpingTextToSplines.html



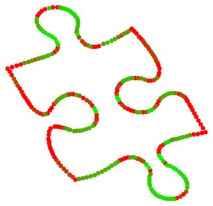
$$s(t) := \int_{t_0}^{t} \|\gamma'(t)\| dt$$
$$t(s) := \text{inverse of } s(t)$$
$$\gamma(s) := \gamma(t(s))$$

Constant-speed parameterization

D

Boundary condition

Orthogonal vs Aligned



B

Each colored vertice has an label weight going from 0 (red) to 1 (green)

Edge weights

$$w_{ij} = F_a(a_{ij}) \cdot F_d(d_{ij})$$

$$F_a(a_{ij}) = \tanh\left(\frac{4}{\pi}\left(\left|\frac{\pi}{2} - a_{ij}\right| - \frac{\pi}{4}\right)\right)$$

$$F_d(d_{ij}) = e^{-\tilde{d}_{ij}^2/\sigma_d^2}$$

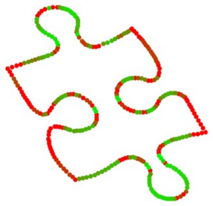
Where : a_{ij} = angle between normal of vertices *i,j*

d_{ij} = distance between vertices *i,j*

$\sigma_d = 10$

Boundary condition

Orthogonal vs Aligned



B

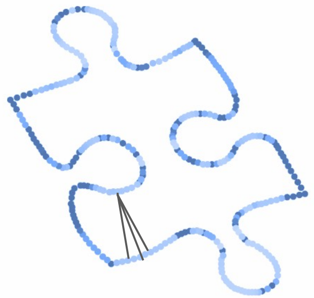
Each colored vertice has an label weight going from 0 (red) to 1 (green)

Vertice attraction weights

$$\alpha_i = F_a^+(a_i) \cdot F_d^+(d_i)$$

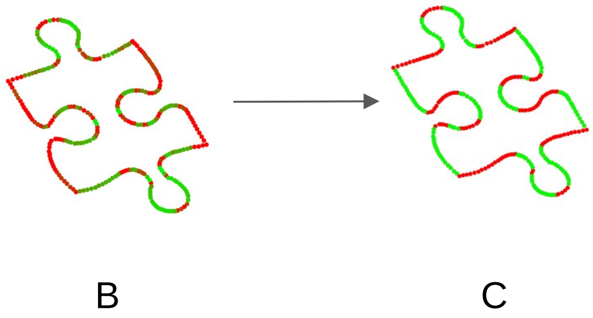
$$F_a^+ = max(0, F(a_i))$$

$\sigma_d = 20$



Boundary condition

Orthogonal vs Aligned



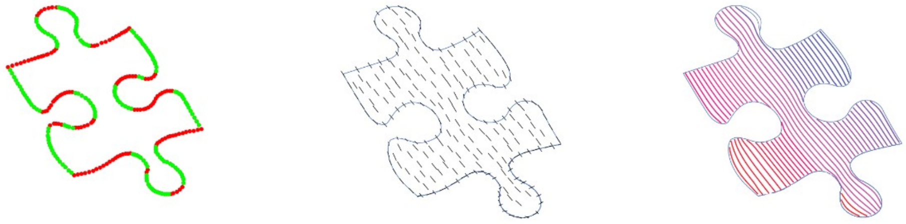
Each colored vertice has an label
weight going from 0 (red) to 1 (green)

Vertice attraction weights

$$\min \sum_{ij} \underbrace{w_{ij}}_{\text{can be negative}} (l_i - l_j)^2 + \overbrace{\omega}^{=1} \sum_i \alpha_i (1 - l_i)$$

$$\text{subj. to } 0 \leq l_i \leq 1$$

Vector field



C D

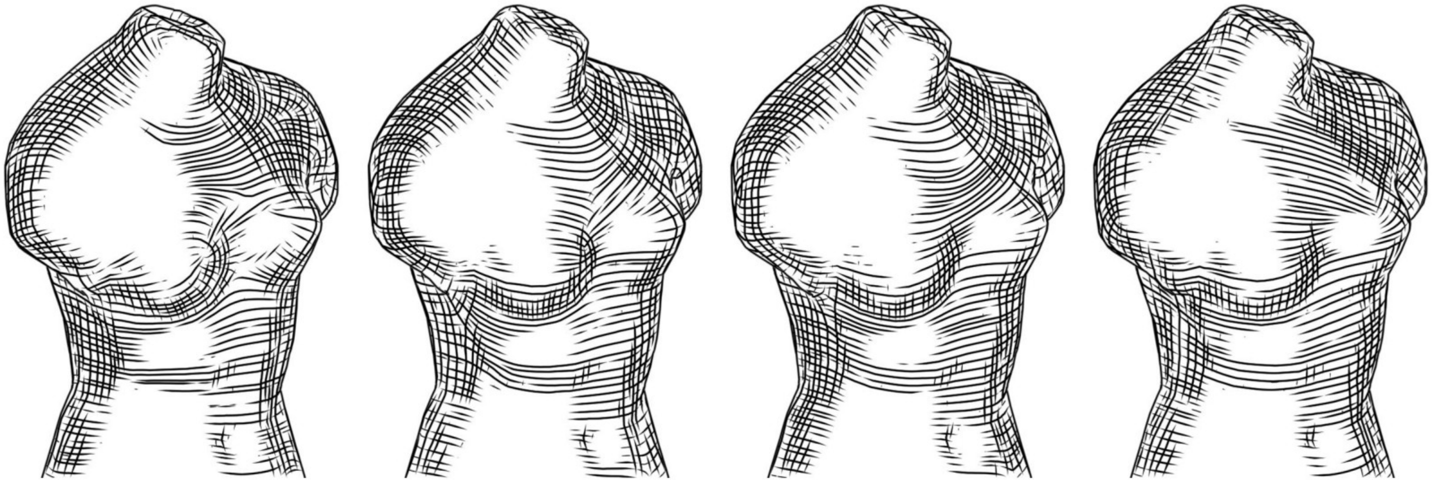
1. Triangulate shape
2. Compute vector field
3. Trace text lines

Vector field

Rotational Symmetry Field Design on Surfaces

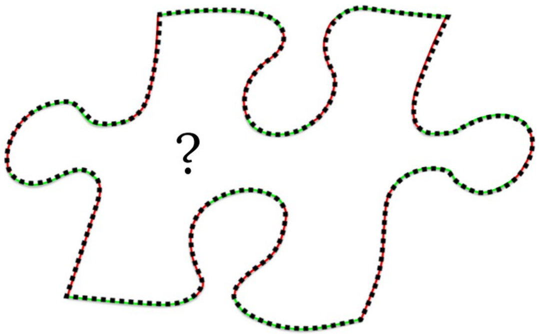
Jonathan Palacios*
Oregon State University

Eugene Zhang*
Oregon State University



(a) (b) (c) (d)

Inside?



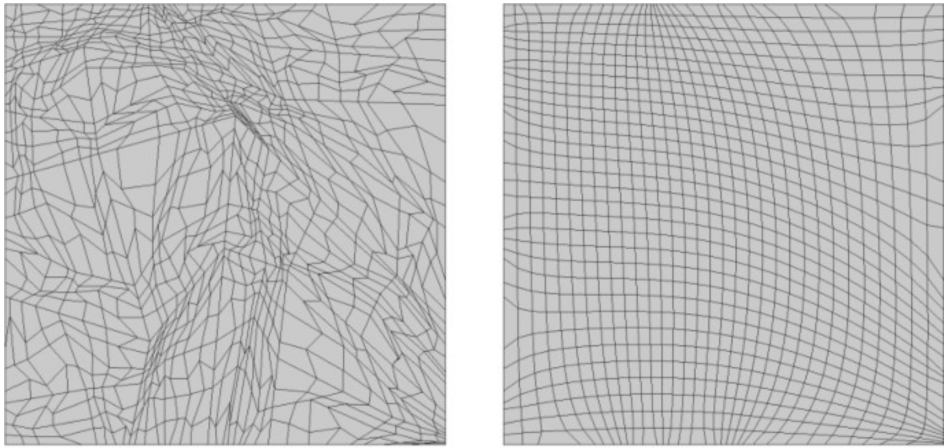
Inside?

- Smoothest interpolation of boundary values
- Laplace equation with Dirichlet boundary conditions
- Discretization?
- Representation?

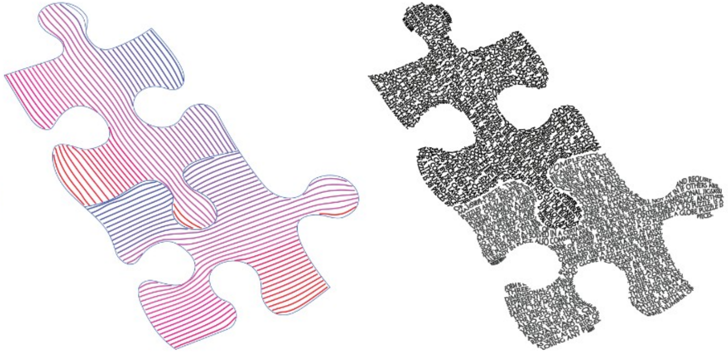
$$\Delta u = 0$$
$$u\Big|_{\partial\Omega} = v$$

Applications_of_Vector_Fields.pdf

Laplacian smoothing!

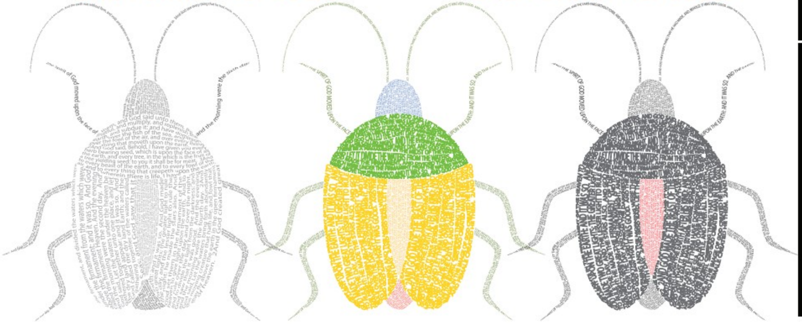


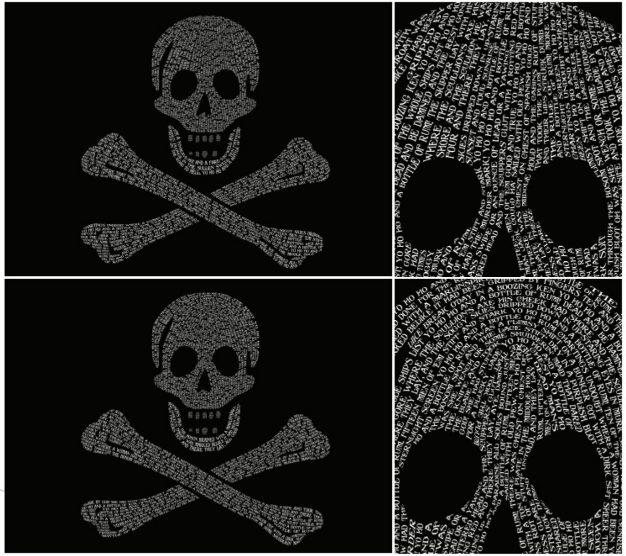
(a) before smoothing **(b)** after smoothing



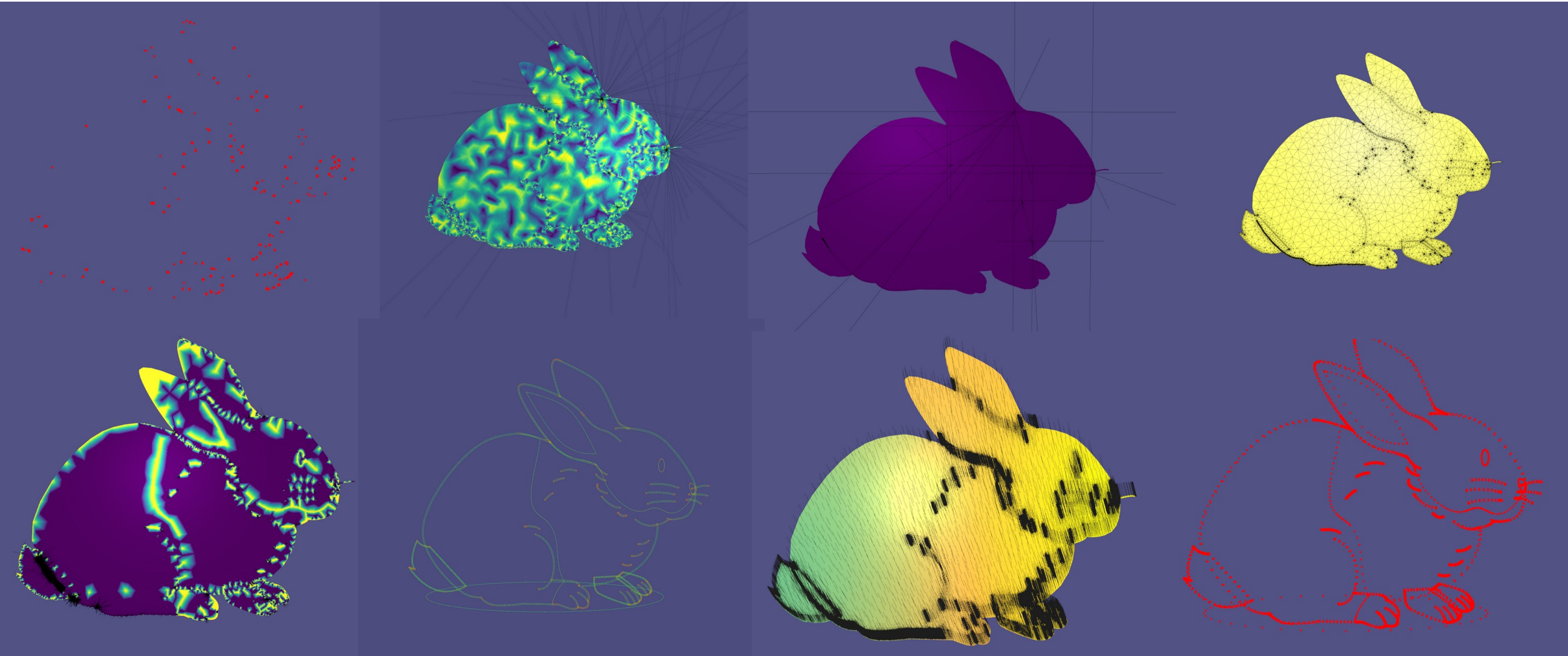

CALIFORNIA REPUBLIC


CALIFORNIA REPUBLIC



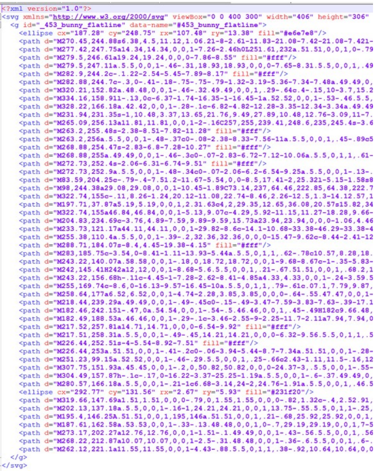


Implementation

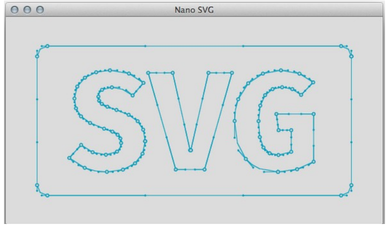


Rendering points on the screen

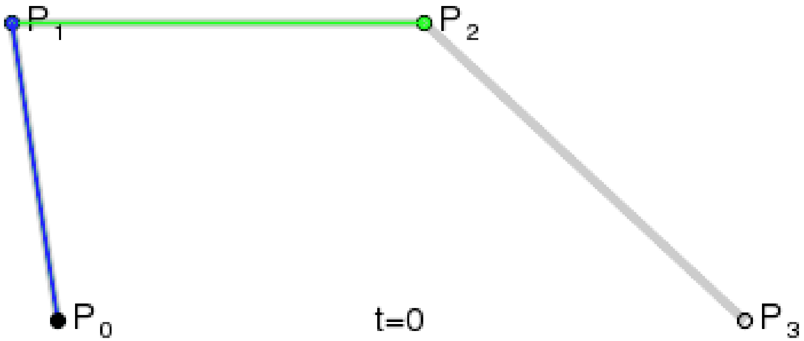
SVG files



Tool : NanoSVG



Cubic Bezier Curves

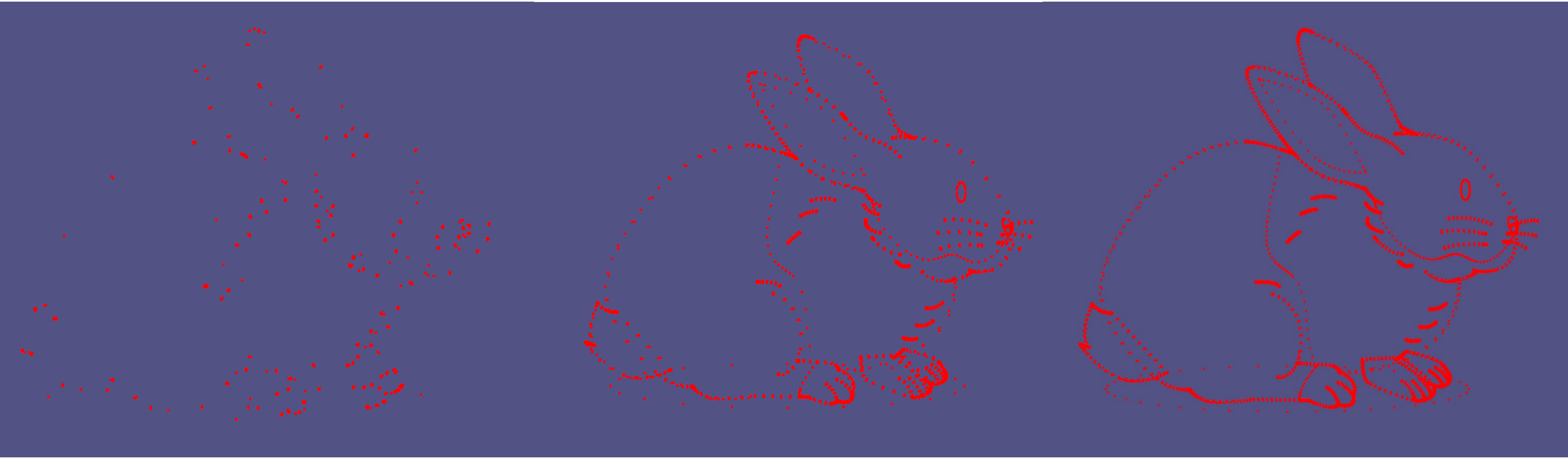


$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3,\ 0 \le t \le 1.$$

```
struct NSVGimage* image;
image = nsvgParseFromFile("../../../bunnyhd.svg", "px", 96);
std::vector<Eigen::MatrixXd> curves;
std::vector<Eigen::MatrixXi> edges;
for (auto shape = image->shapes; shape != NULL; shape = shape->next) {
  for (auto path = shape->paths; path != NULL; path = path->next) {
    for (auto i = 0; i < path->npts-1; i += 3)
    {
      float* p = &path->pts[i*2];
      bezierToVertex(5,p,curves,edges,path->closed);
    }
  }
}
```

```
for(double i=0;i<=dotsPerCurve;i++)
{
    double t = i/dotsPerCurve;
    double x = pow((1.0-t),3) * p[0] + 3.0 * t *pow((1.0-t),2) * p[2] + 3.0*pow(t,2)*(1.0-t) * p[4] + pow(t,3) * p[6];
    double y = pow((1.0-t),3) * p[1] + 3.0 * t *pow((1.0-t),2) * p[3] + 3.0*pow(t,2)*(1.0-t) * p[5] + pow(t,3) * p[7];
```

Rendering points on the screen

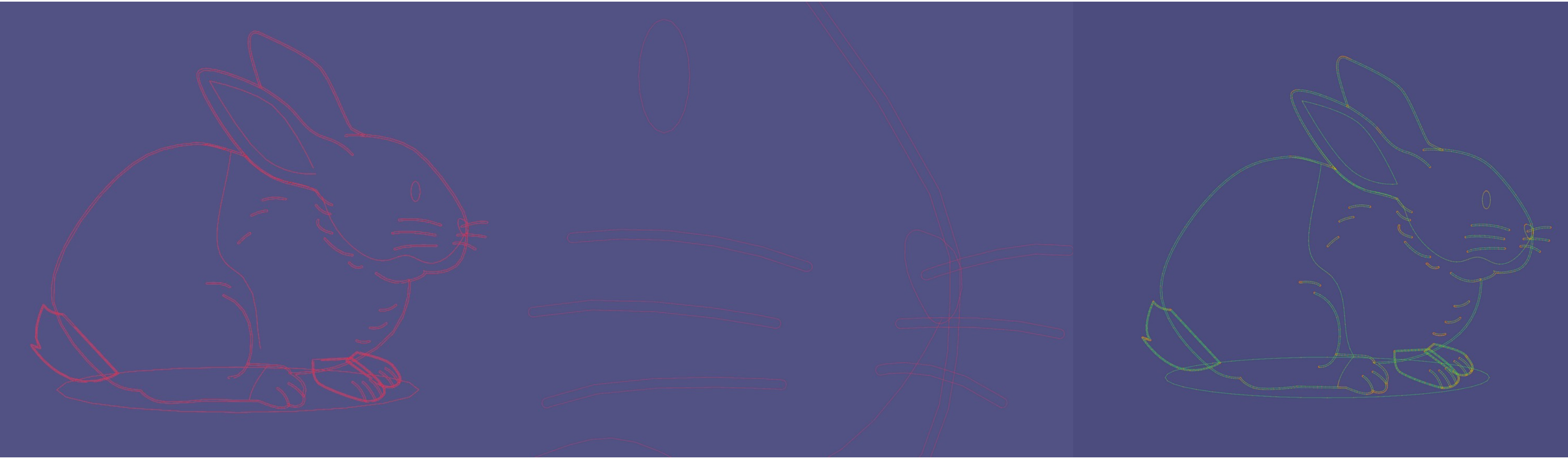


1 point per curve

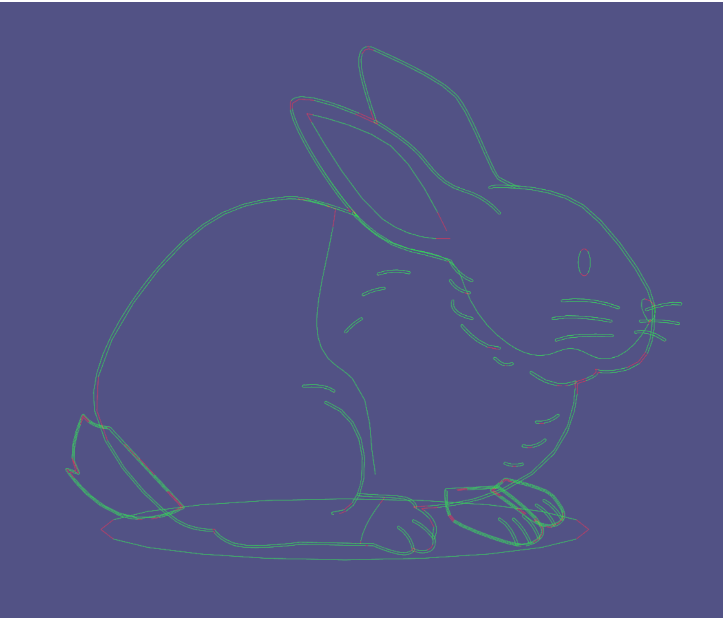
5 points per curve

10 points per curve

Edges



Calculating edge weights



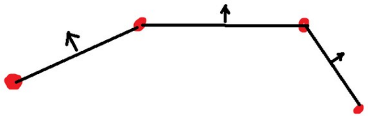
negative : red

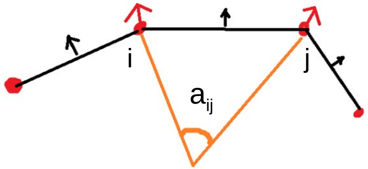
positive : green

$$w_{ij} = F_a(a_{ij}) \cdot F_d(d_{ij})$$

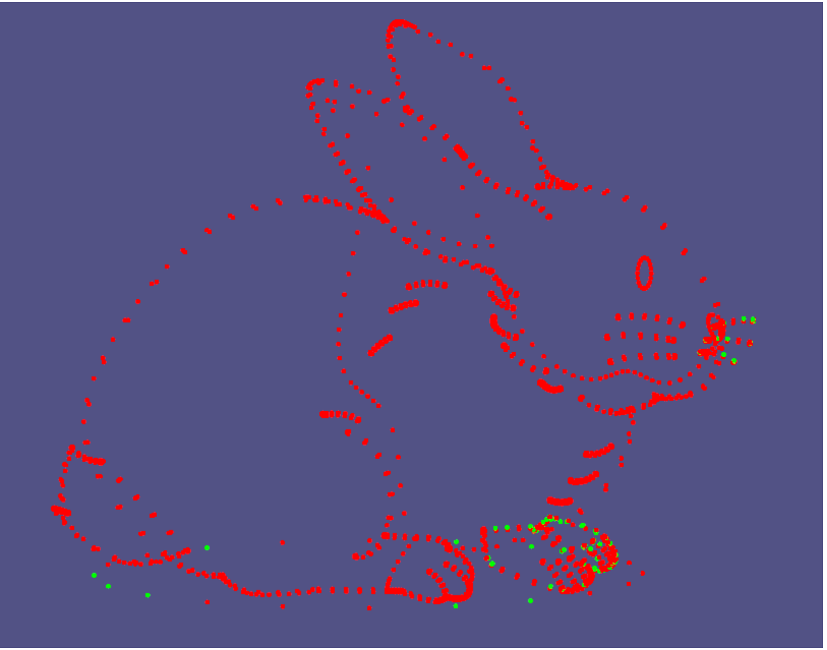
$$F_a(a_{ij}) = \tanh(\frac{4}{\pi}(|\frac{\pi}{2} - a_{ij}| - \frac{\pi}{4}))$$

$$F_d(d_{ij}) = e^{-\tilde{d}_{ij}^2/\sigma_d^2}$$





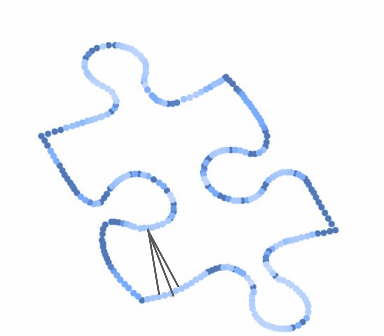
Calculating vertex attraction weights



$$\alpha_i = F_a^+(a_i) \cdot F_d^+(d_i)$$

$$F_a^+ = max(0, F(a_i))$$

$\sigma_d = 20$



igl::ray_box_intersect(V.row(i),Vnormal,box)

Find local minimum

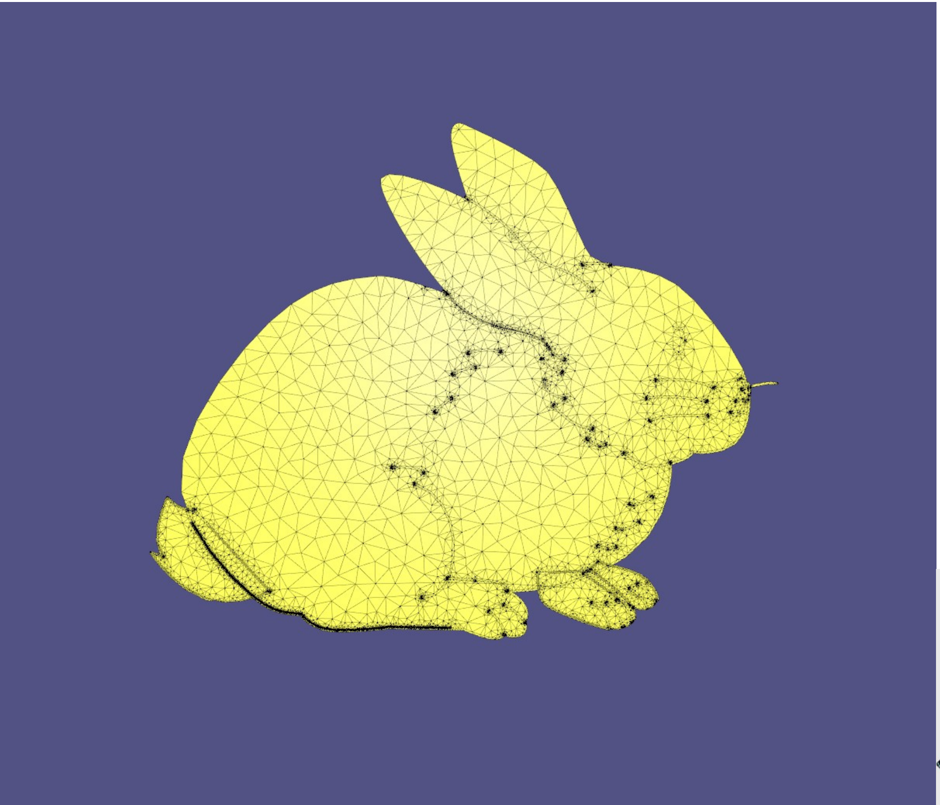
$$\min \sum_{ij} w_{ij}(l_i - l_j)^2 + \omega \sum_i \alpha_i(1 - l_i)$$

$$\text{subj. to } \; 0 \leq l_i \leq 1$$

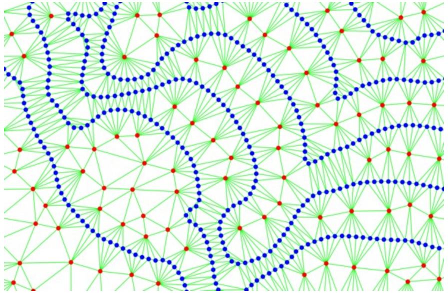
```
auto W = calculateEdgeWeights(V,E,VtoEnormals);
auto alpha = calculateVertexAttraction(V,VtoEnormals);
auto label = Eigen::VectorXd{V.rows()}.setConstant(0.5); //final labeling, starts at 0.5
double Wsum = 0.0;
//edge weight sum
for(int i=0;i<E.rows();i++)
{
  auto p1 = E.row(i).coeff(0,0);
  auto p2 = E.row(i).coeff(0,1);

  Wsum += W[i] * pow(label[p1]-label[p2],2.0);
}
double alphasum = 0.0;
//vertex weight sum
for(int i=0;i<V.rows();i++)
{
  alphasum += alpha[i] * (1-label[i]);
}

//minimise here
auto total = Wsum + alphasum;
```

:(

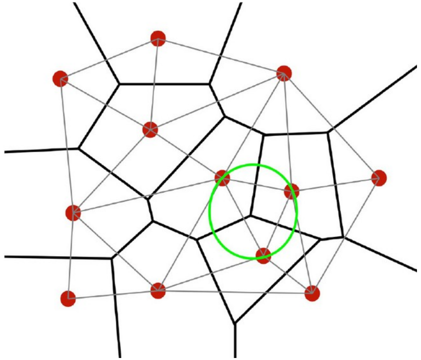
Triangulation

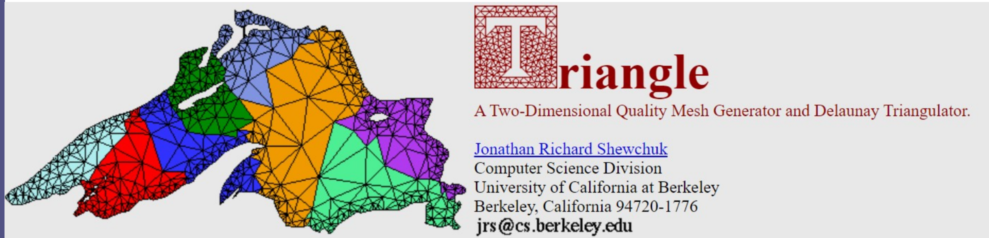


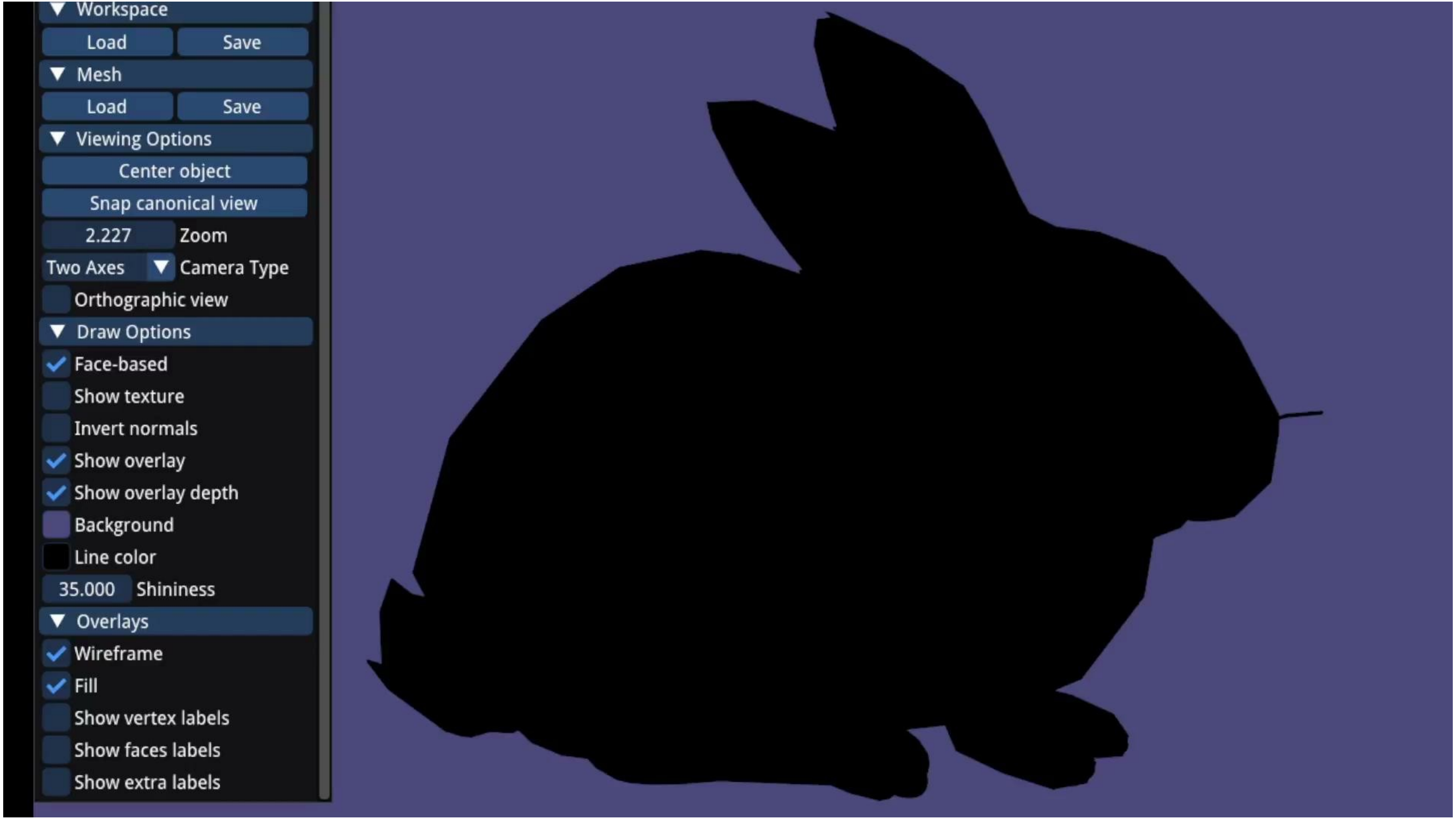
Generate Steiner points



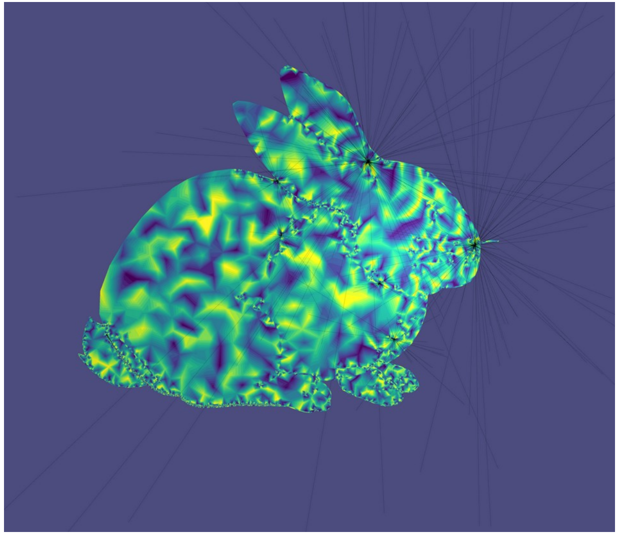
Perform Delaunay triangulation

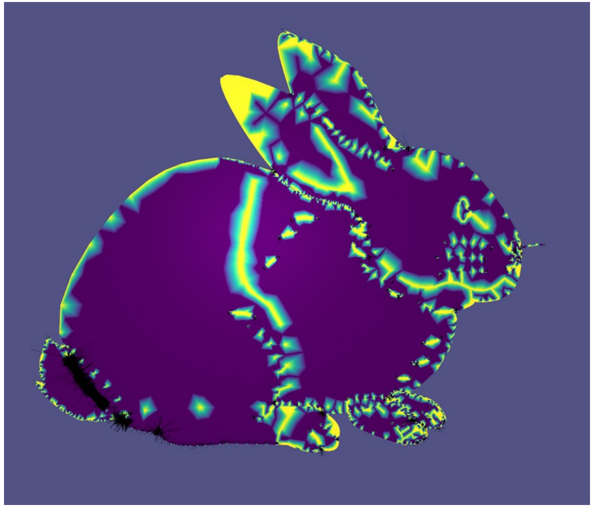


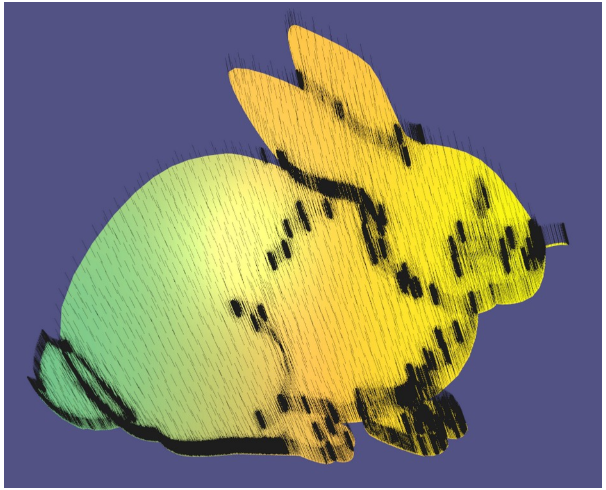



Workspace
Load
Save
Mesh
Load
Save
Viewing Options
Center object
Snap canonical view
2.227
Zoom
Two Axes
Camera Type
Orthographic view
Draw Options
Face-based
Show texture
Invert normals
Show overlay
Show overlay depth
Background
Line color
35.000
Shininess
Overlays
Wireframe
Fill
Show vertex labels
Show faces labels
Show extra labels

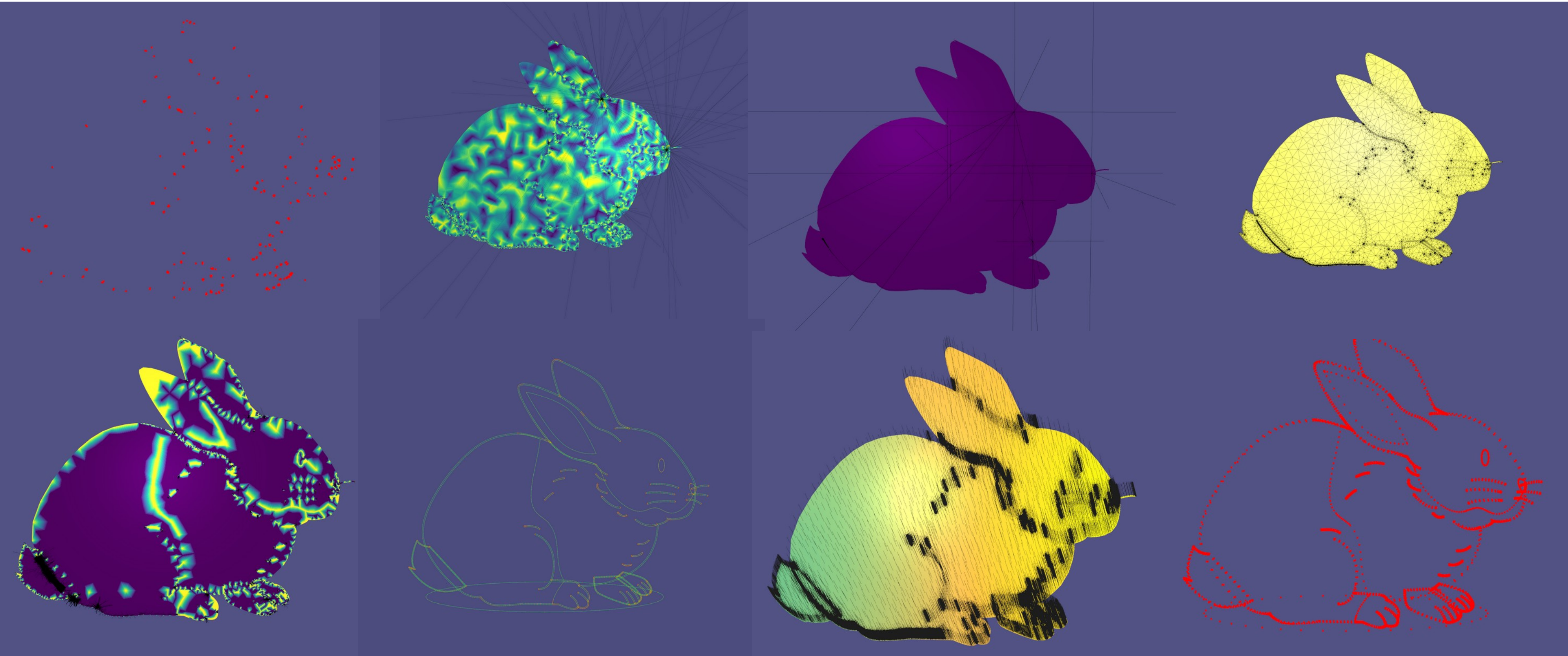
Exploring vector fields with Libigl

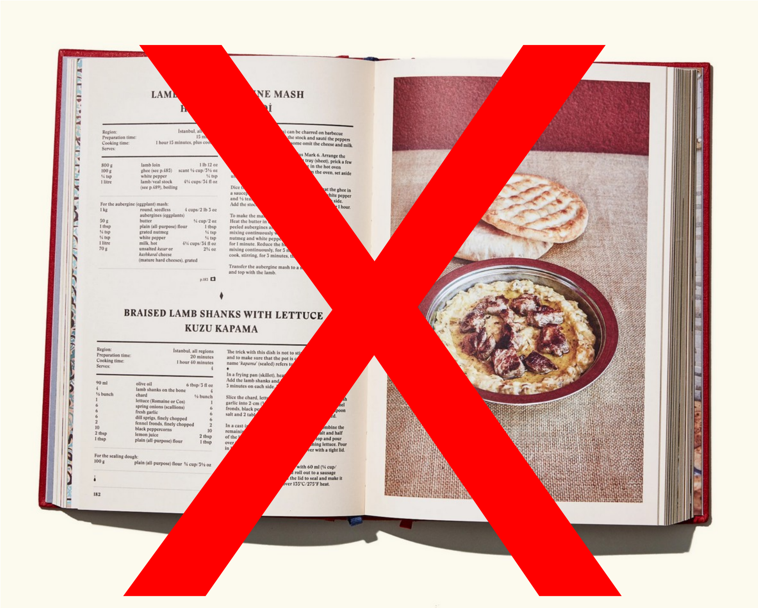






Conclusion?




BRAISED LAMB SHANKS WITH LETTUCE
KUZU KAPAMA
182