

3D Differential Geometry

Paolo Cignoni, Massimiliano Corsini, Daniela Giorgi, Luigi Malomo

Visual Computing Lab, ISTI-CNR, Pisa

PhD in Artificial Intelligence

AY 2024 – 2025



Motivation

- Images have a quite straightforward interpretation as a regularly sampled signal over a 2D domain
- Many calculus concepts have straightforward interpretation and ways of being computed
- Derivatives and Edges extraction etc.
- For 3D surfaces things are much more complicated
 - No clear domain (parametrization issue)
 - Multiple different “signals”: the shape vs the appearance

(Discrete) Differential Geometry

How the surface is
locally changing

Compute approximations of the ***differential properties*** of underlying (discretized) surface from the 3D description

Basic concepts:

- Normal Vectors
- (Discrete) Curvature
- Local Averaging Region
- Gradients
- Laplace-Beltrami Operator

Normal

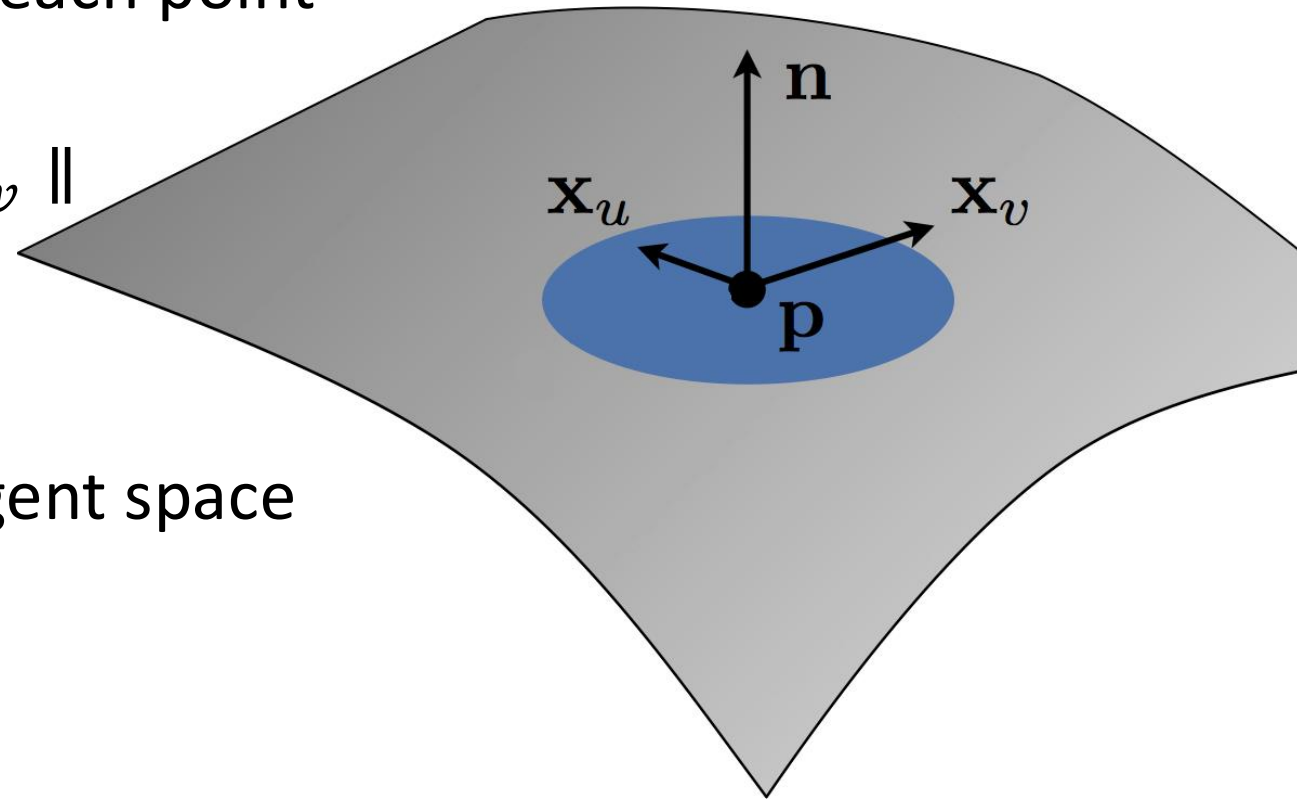
Let's consider a parametric surface 2-manifold
 $S \in \mathbb{R}^3$

$$S(u, v): \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Then we can define the normal for each point
of the surface as:

$$\mathbf{n} = (\mathbf{x}_u \times \mathbf{x}_v) / \|\mathbf{x}_u \times \mathbf{x}_v\|$$

Where $\mathbf{x}_u, \mathbf{x}_v$ are vectors on tangent space

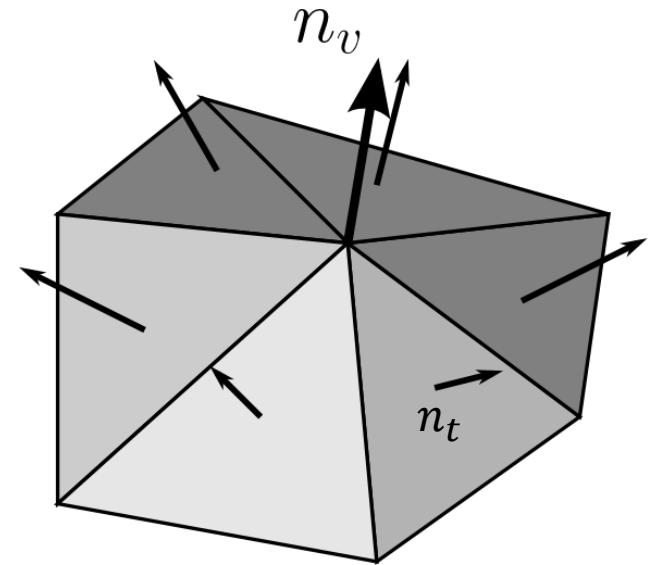


Normals on triangle meshes

Normals are well defined only inside the triangular faces n_t

Per-vertex normal can be computed by averaging normal of adjacent triangles

$$n_v = \sum_{t \in Adj(v)} n_t$$



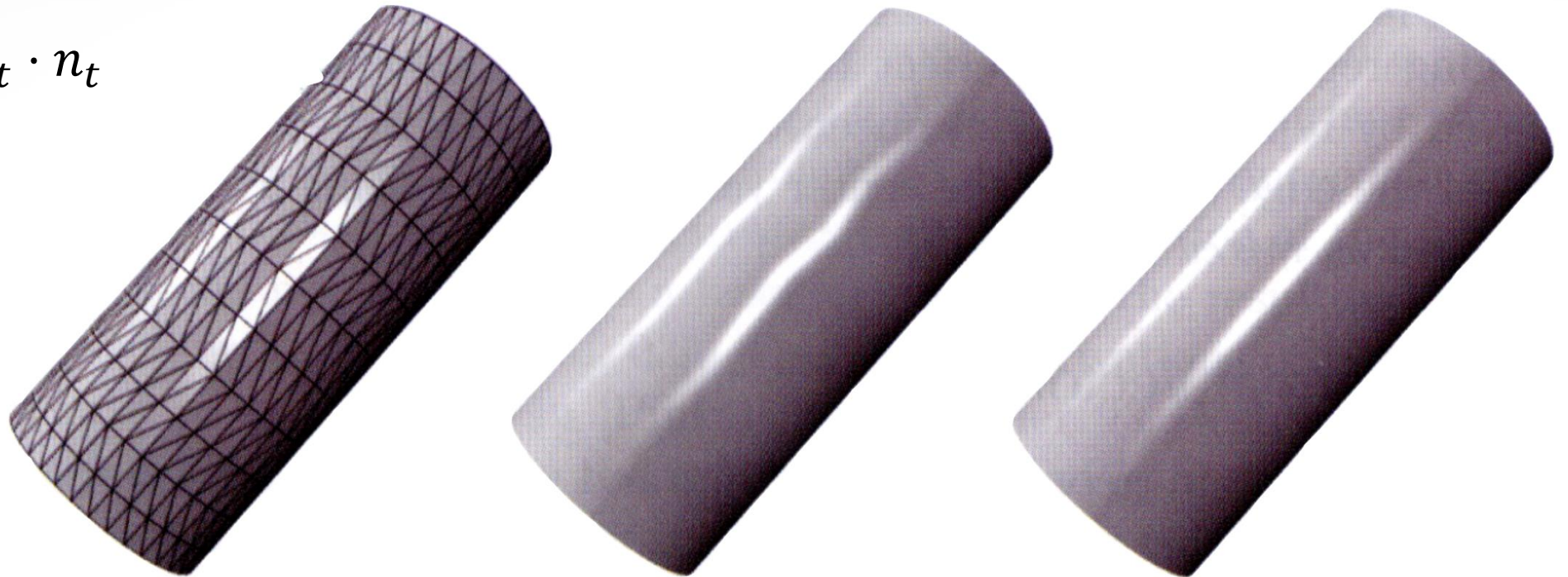
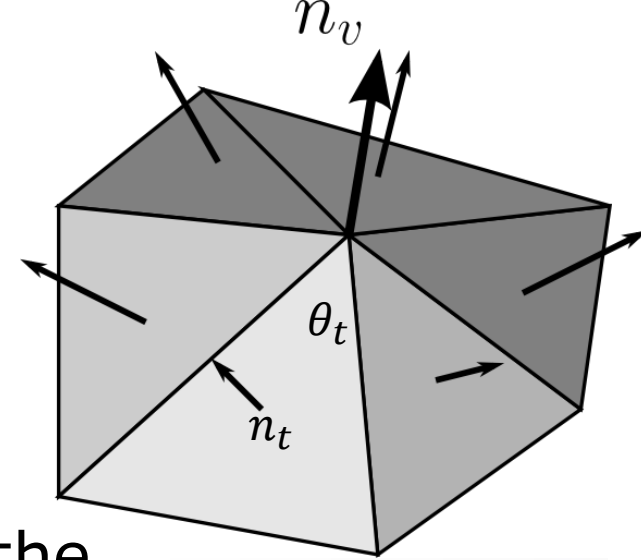
Normals on triangle meshes

Does simple averaging work? Almost, for a “good” triangulation

Uneven triangles may generate biased normals

Weighting the average by angle θ (or area) improves the results

$$n_v = \sum_{t \in Adj(v)} \theta_t \cdot n_t$$



Normals on triangle meshes

Does simple averaging work? Almost, for a “good” triangulation

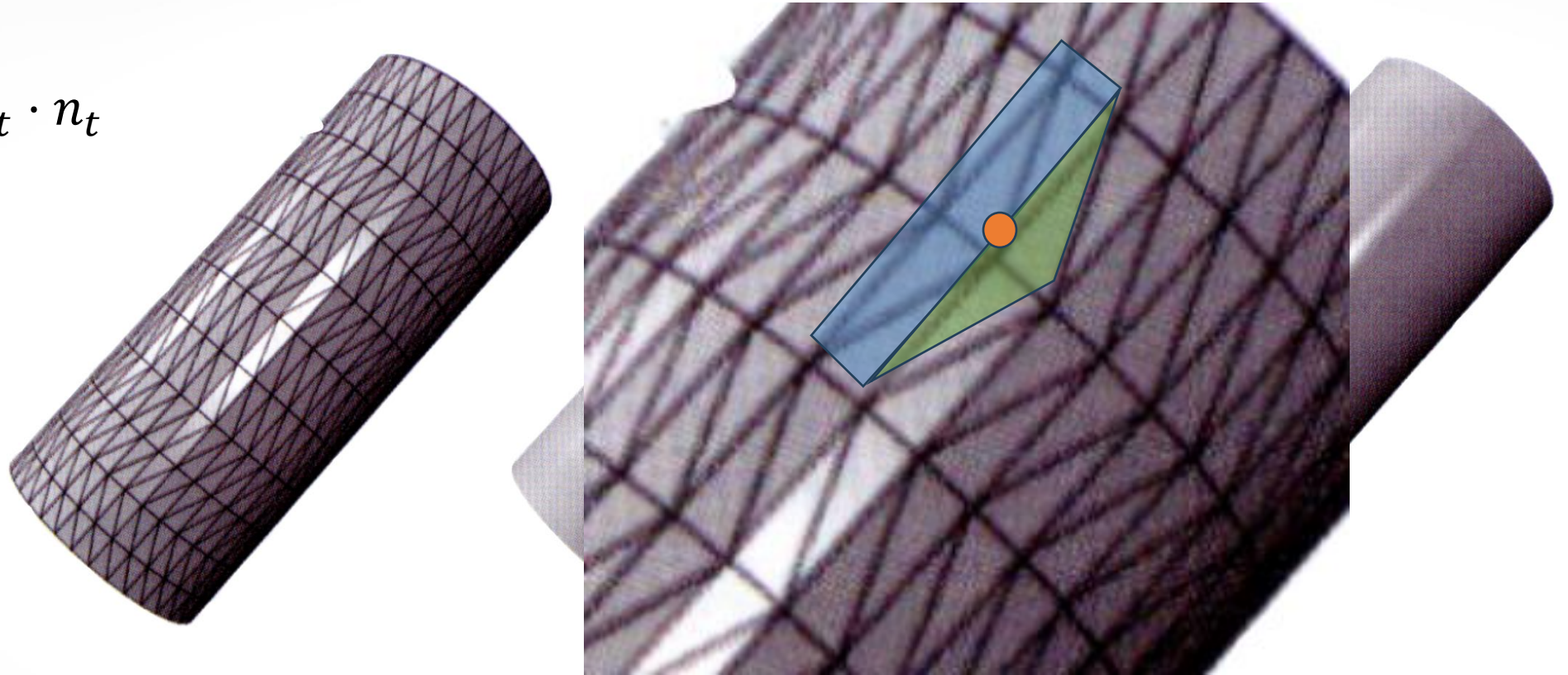
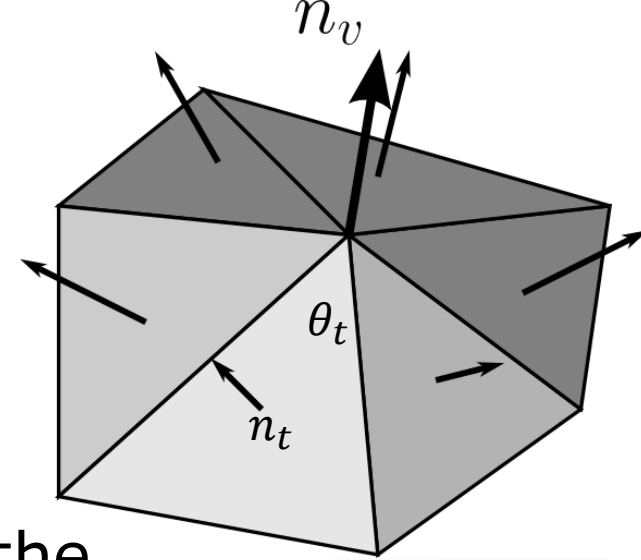
Uneven triangles may generate biased normals

Weighting the average by angle θ (or area) improves the results

$$n_v = \sum_{t \in Adj(v)} \theta_t \cdot n_t$$

Trick: if you get the normal as cross product of edges, its length is twice the area of the triangle

you can get the area weighting for free



Curvature

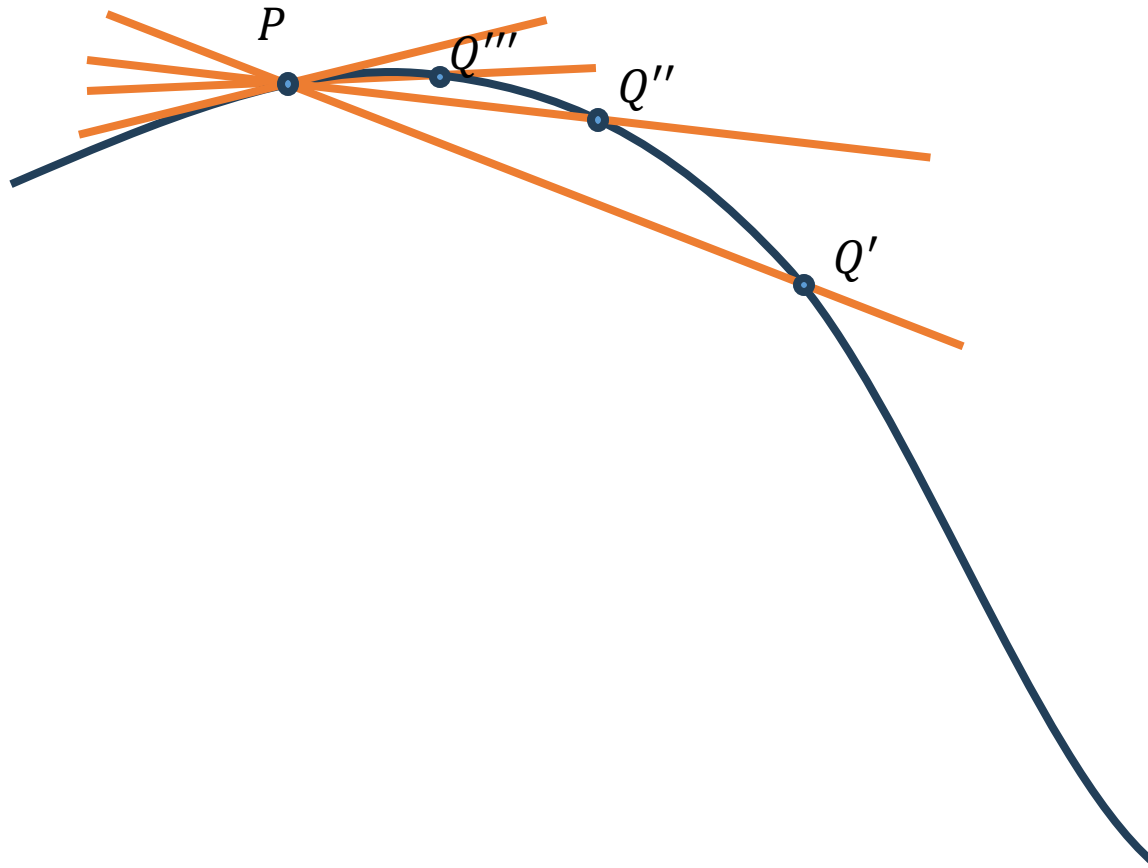
Capture the intuitive concept of how much a surface is bending in a point

Extension of the 2D concept

Not a single number but something more complex

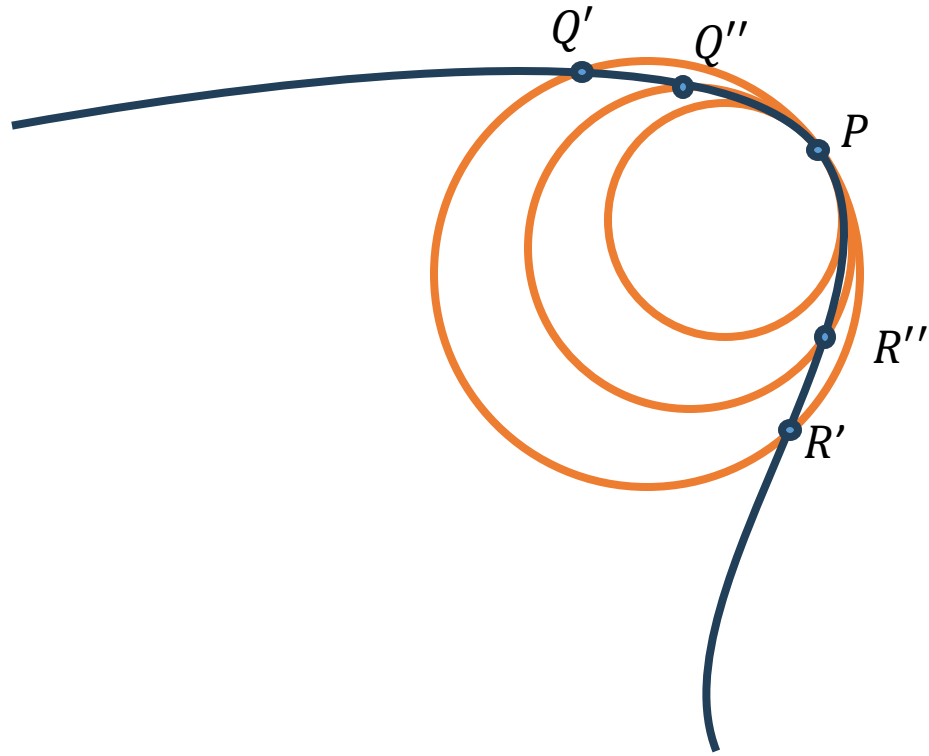
Tangent

- The tangent is limiting secant as the two points come together



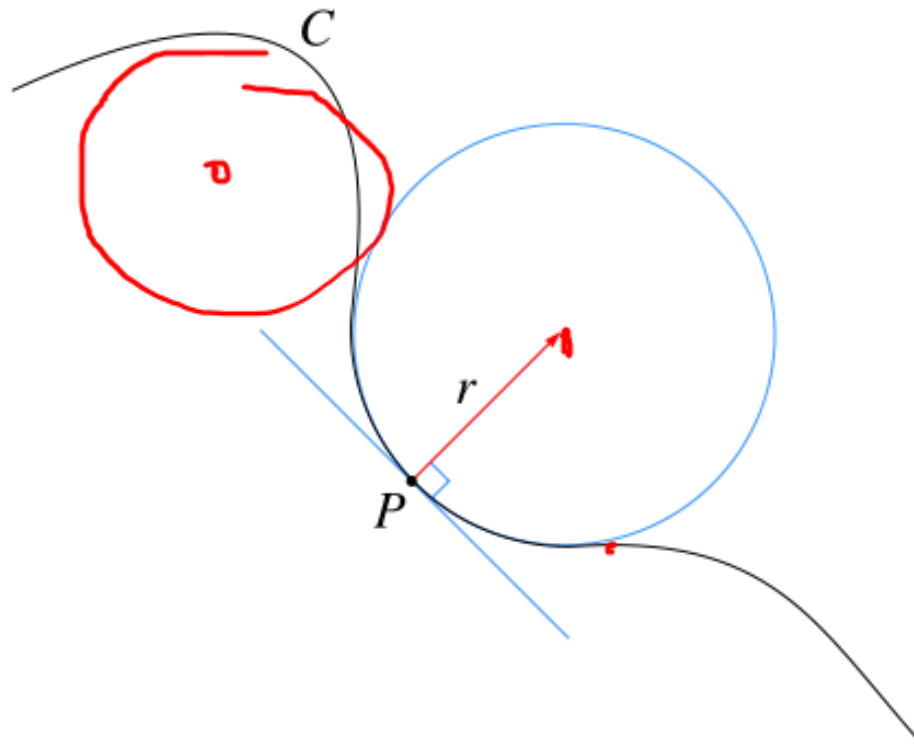
Circle of curvature

- Consider the circle passing through 3 points P, R, Q , **the circle of curvature** (*osculating circle*) it is the limiting circle when the three points come together



Curvature in the plane

The curvature of a curve c at a point P is defined to be the reciprocal of the radius of osculating circle at point P .

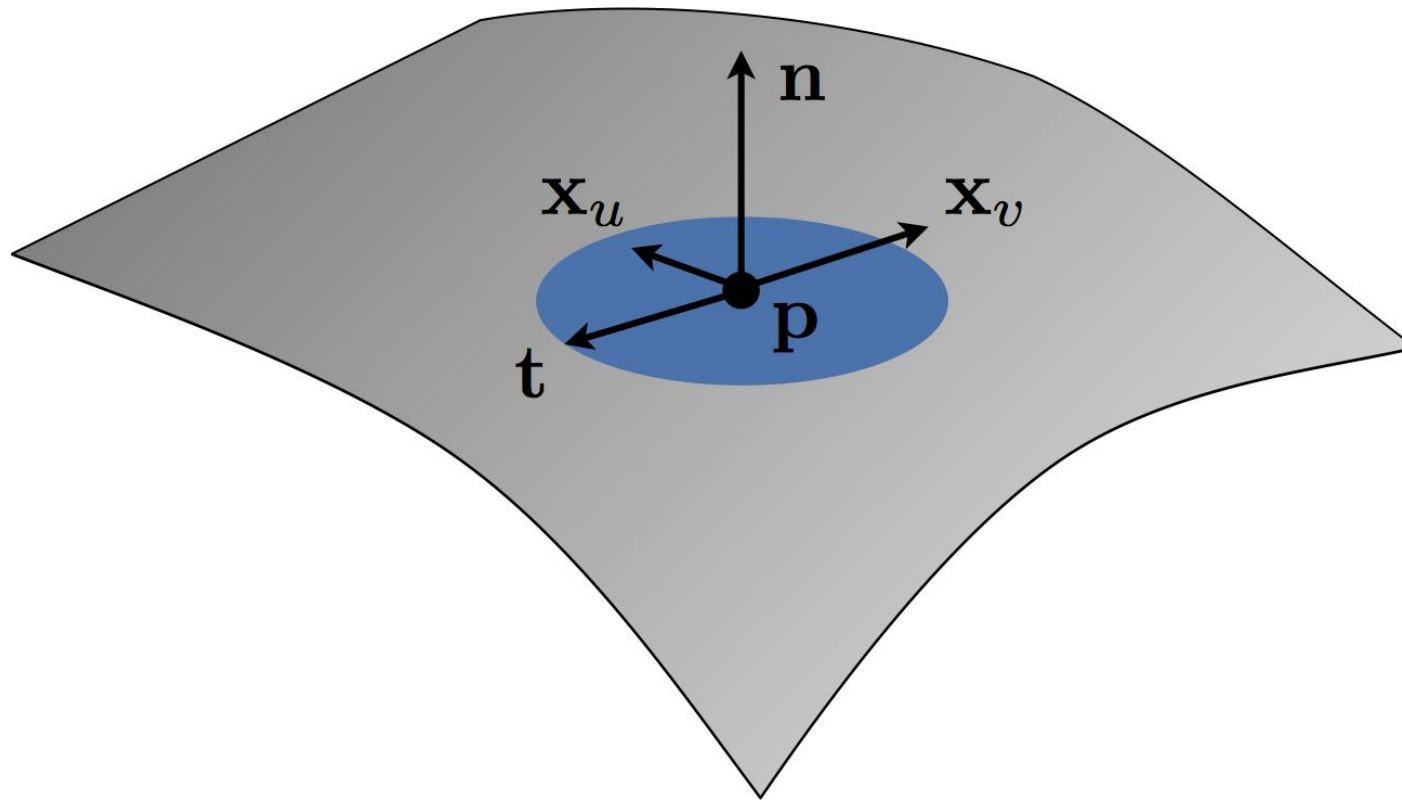


The osculating circle of a curve C at a given point P is the circle that has the same **tangent** as C at point P as well as the same **curvature**.

Just as the tangent line is the line best approximating a curve at a point P , the osculating circle is the best circle that approximates the curve at P .

Curvature in space

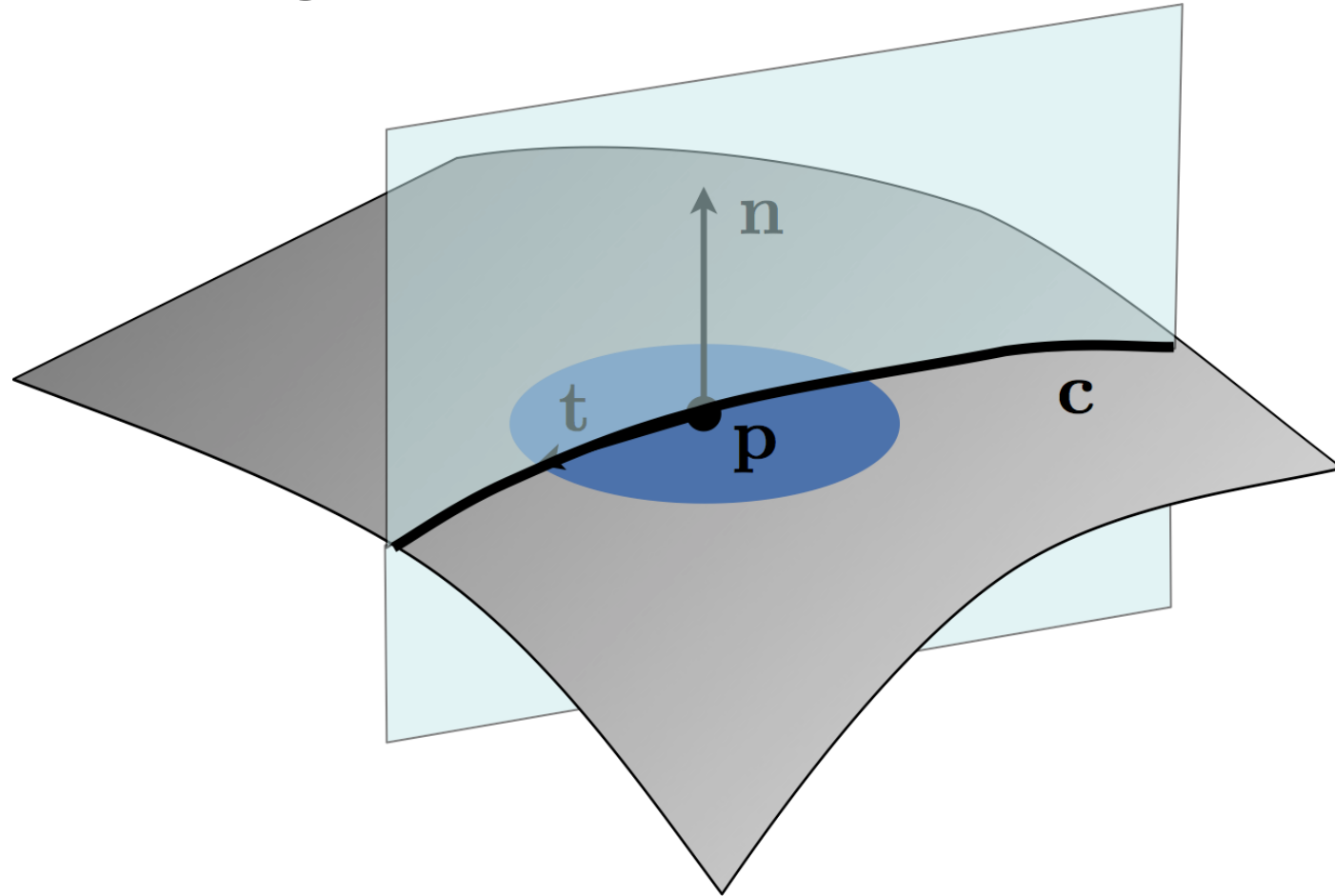
Consider a tangent vector oriented with an angle ϕ with respect to the parametric plane



$$\mathbf{t} = \cos\phi \frac{\mathbf{x}_u}{\|\mathbf{x}_u\|} + \sin\phi \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|}$$

Curvature

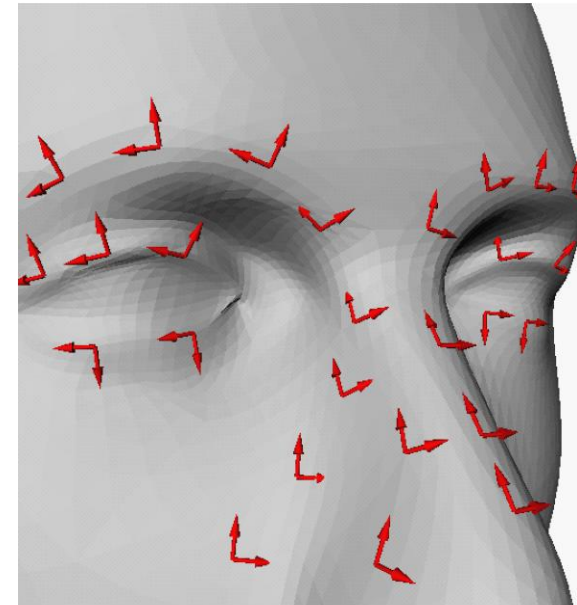
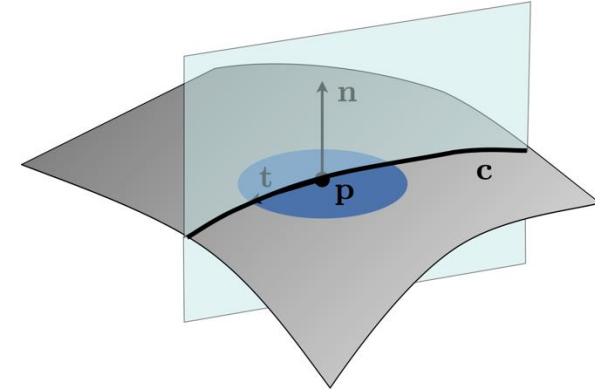
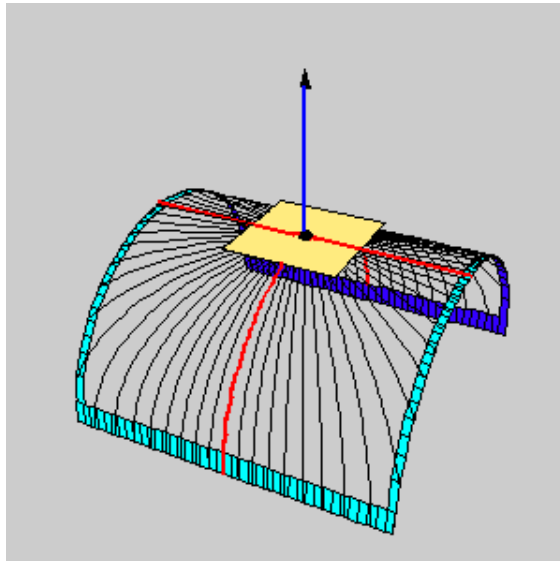
Consider the plane along $\mathbf{n}$, $\mathbf{t}$ and the 2D curve $\mathbf{c}$ defined on it



Main curvature directions

For each direction $\mathbf{t}$, we define a curvature value k .

Let's consider the two directions $\mathbf{k}_1$ and $\mathbf{k}_2$ where the curvature values k_1 and k_2 are **maximum** and **minimum**



Min max curvature

- **Euler theorem:**
the two directions of min
and max k_1 and k_2 are
perpendicular

Note: Curvature is
represented by **two** 3D
vectors, but there are nice
ways to summarize this info



min curvature



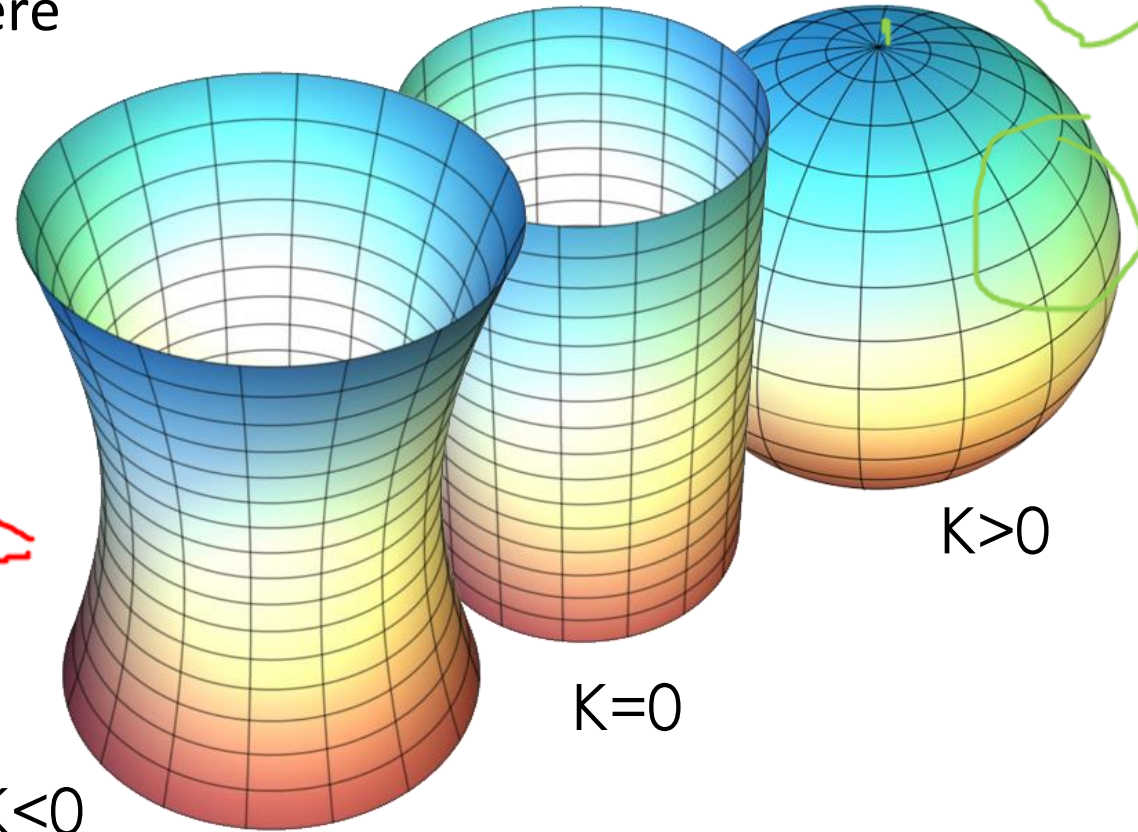
max curvature

Gaussian curvature

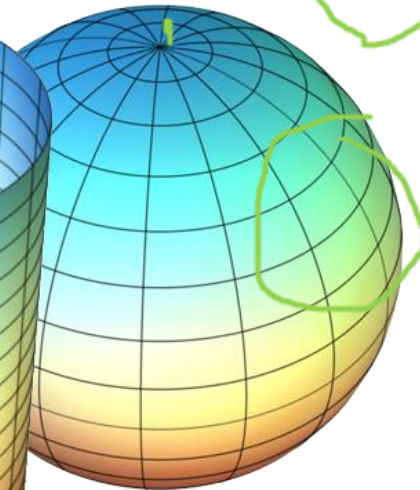
- Defined as $K = k_1 \cdot k_2$
 - >0 when the surface is a sphere
 - 0 if locally flat
 - <0 for hyperboloids



$K < 0$



$K = 0$



$K > 0$

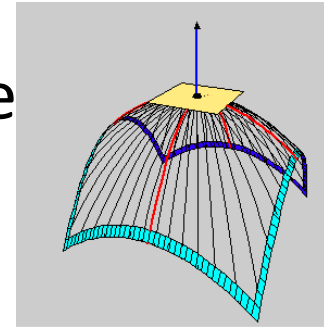


Gaussian curvature

- A point x on the surface is called

- **elliptic** if $K > 0$

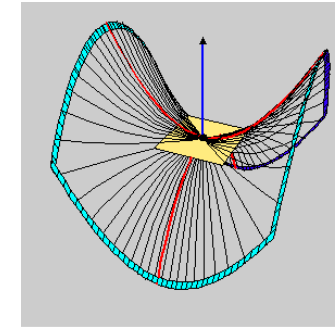
- (k_1 and k_2 have the same sign)



elliptic

- **hyperbolic** if $K < 0$

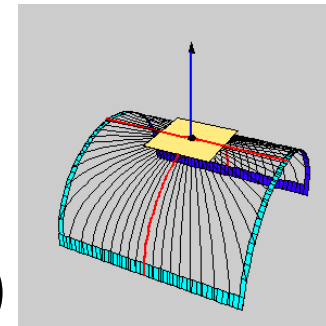
- (k_1 and k_2 have opposite sign)



hyperbolic

- **parabolic** if $K = 0$

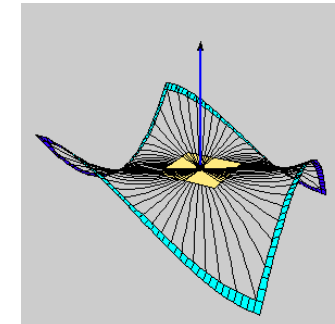
- (exactly one of k_1 and k_2 is zero)



parabolic

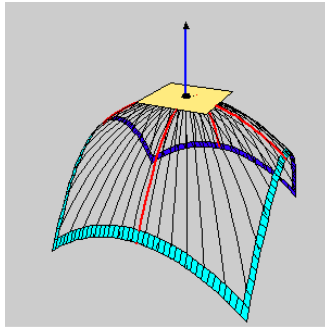
- **planar** if $K = 0$

- (equivalently $k_1=k_2=0$).

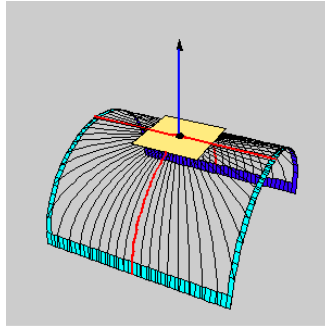


planar

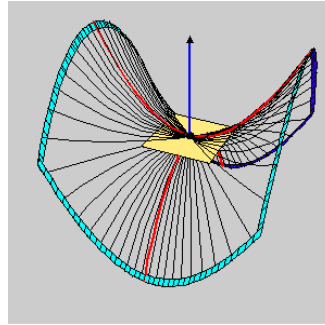
Different classes distributed on the surface



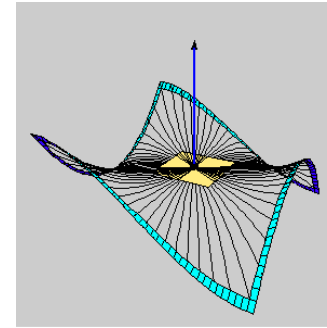
elliptic



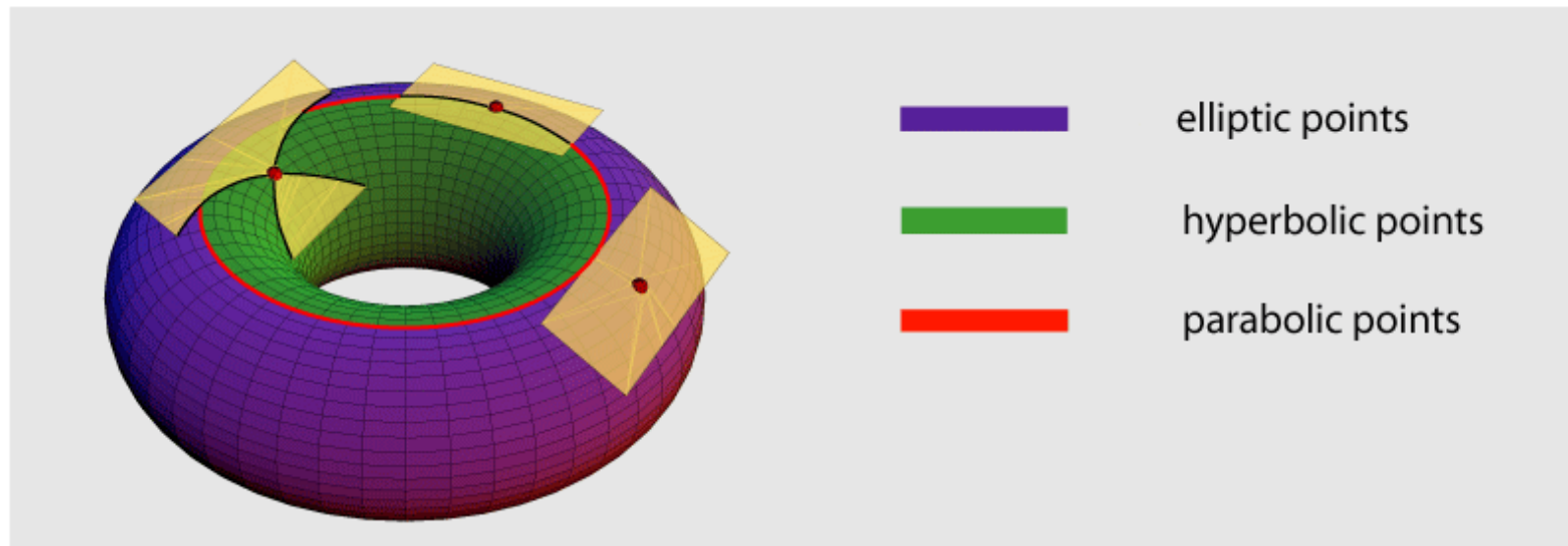
parabolic



hyperbolic

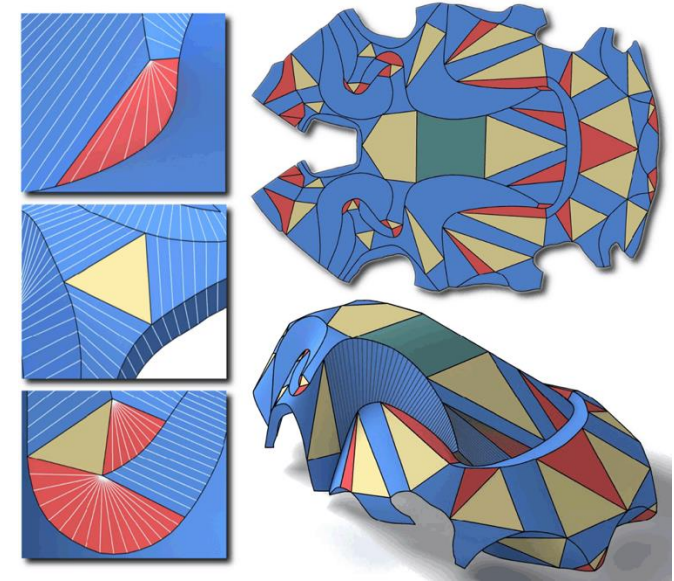
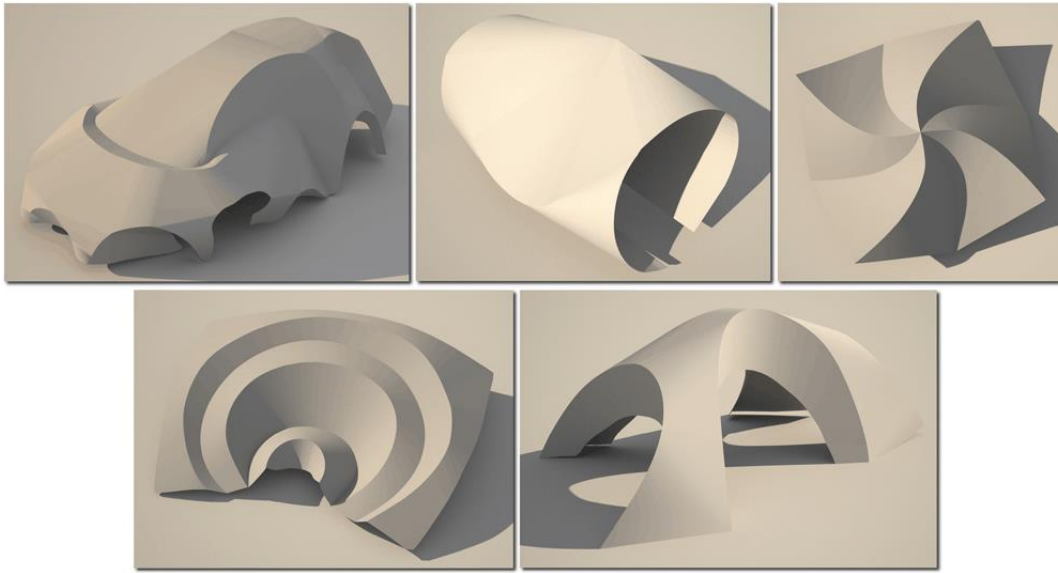


planar



Developable surfaces

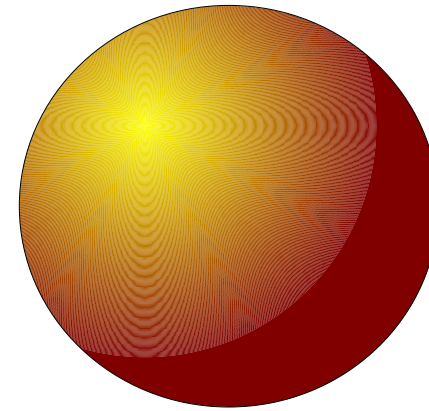
- Developable surface $\Leftrightarrow K = 0$
- Flattening introduce no distortion



Gaussian Curvature: intrinsic / extrinsic

- Gaussian curvature is an **intrinsic** properties of the surface (even if we defined in an extrinsic way)
- It is possible to determine it by moving on the surface keeping the geodesic distance constant to a radius r and measuring the circumference $C(r)$:

$$K = \lim_{r \rightarrow 0} \frac{6\pi r - 3C(r)}{\pi r^3}$$



$K > 0$



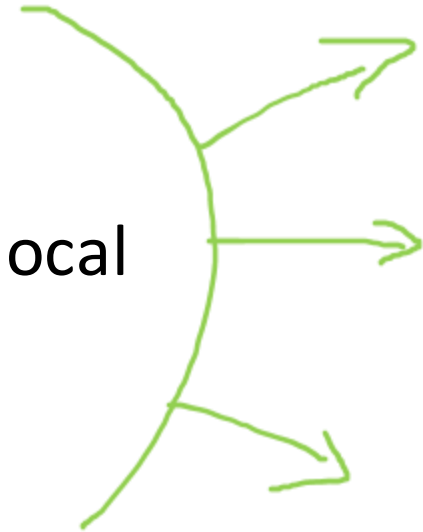
$K < 0$

Mean Curvature

- $H = (k_1 + k_2) / 2$

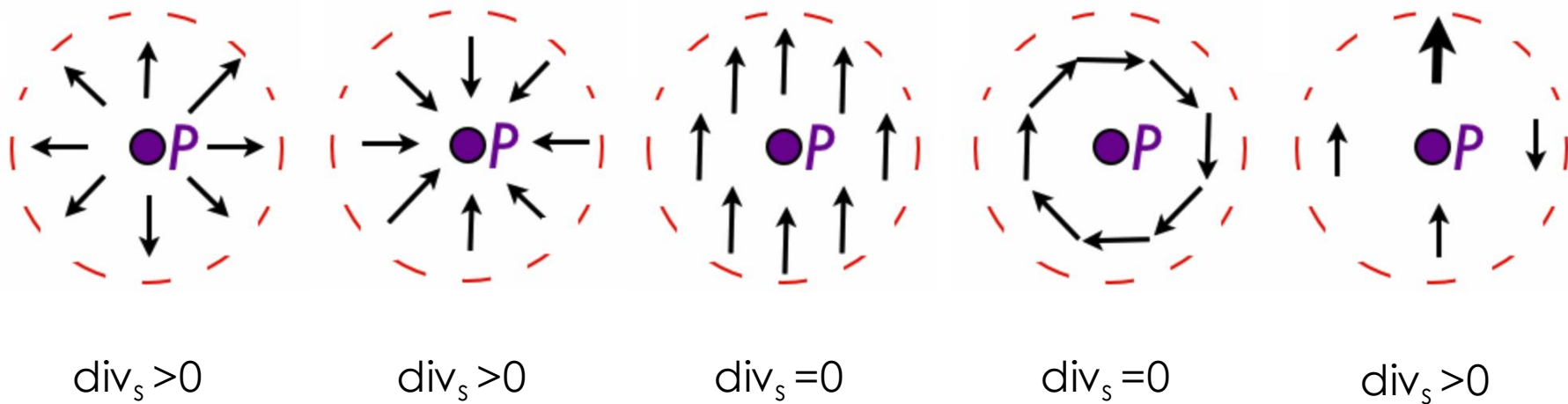
- Measure the **divergence** of the normal in a local neighborhood of the surface

- The **divergence** div_s is an operator that measures a vector field's tendency to originate from or converge upon a given point



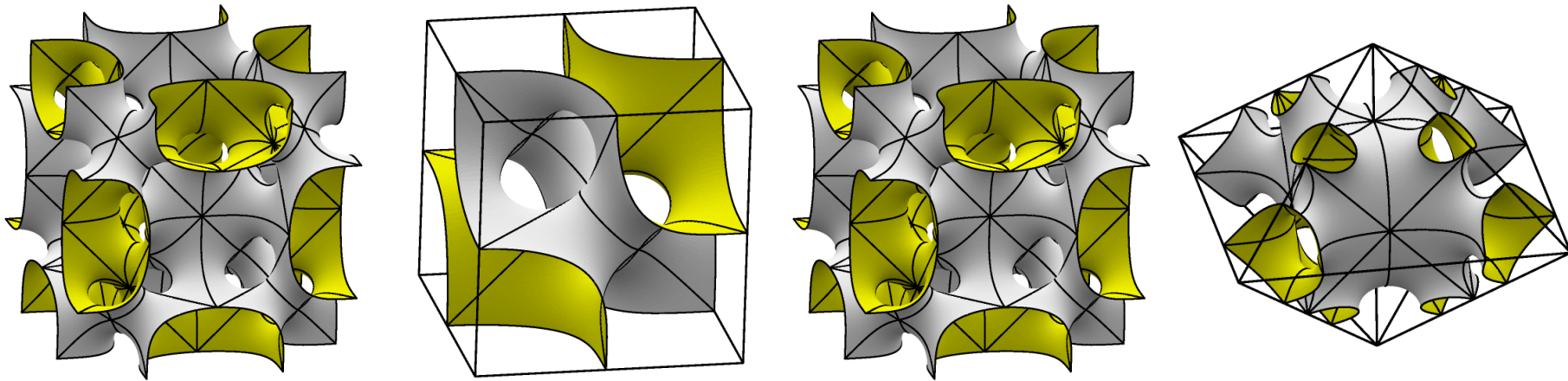
Divergence

- Imagine a vector field represents water flow:
 - If div_s is a **positive** number, then water is **flowing out** of the point.
 - If div_s is a **negative** number, then water is **flowing into** the



Minimal surface and minimal area surfaces

- A surface is **minimal** iff $H=0$ everywhere
- All surfaces of minimal AREA (subject to boundary constraints) have $H=0$ (not always true the opposite!)
- The surface tension of an interface, like a soap bubble, is proportional to its mean curvature



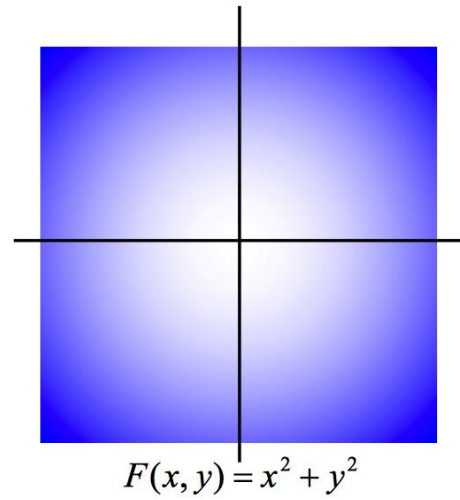
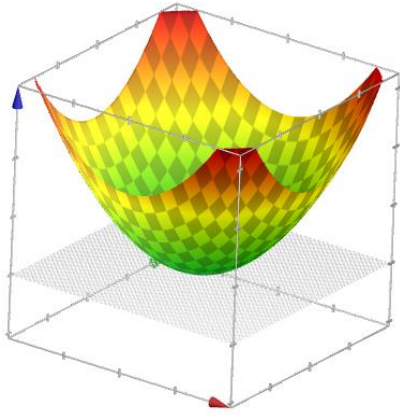
Some math.... Gradient and divergence

- Given a function $F: \mathbb{R}^2 \rightarrow \mathbb{R}$ (our surface) the **gradient** of F is the vector field $\nabla F: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by the partial derivatives:

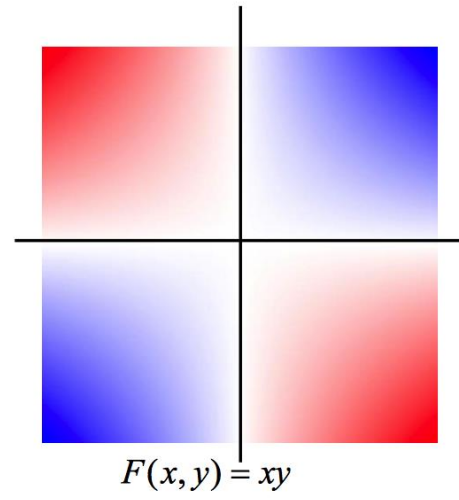
$$\nabla F(x, y) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right)$$

- **Intuitively:** At the point p_0 , the vector $\nabla F(p_0)$ points in the **direction of greatest change of F .**

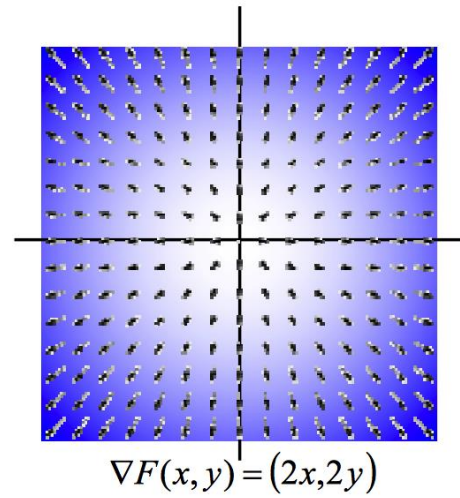
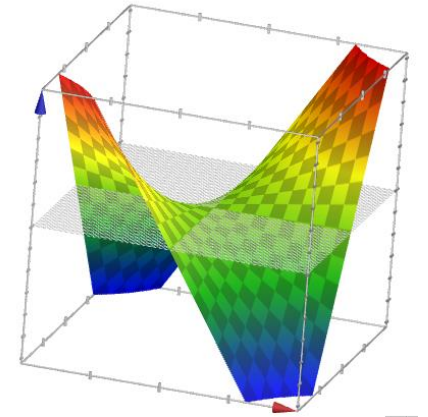
Some math.... Gradient and divergence



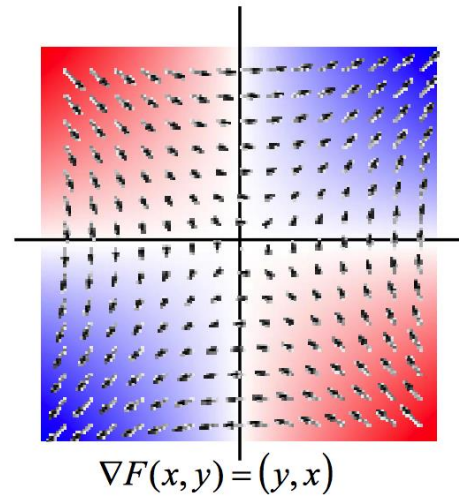
$$F(x, y) = x^2 + y^2$$



$$F(x, y) = xy$$



$$\nabla F(x, y) = (2x, 2y)$$



$$\nabla F(x, y) = (y, x)$$

Some math.... Gradient and divergence

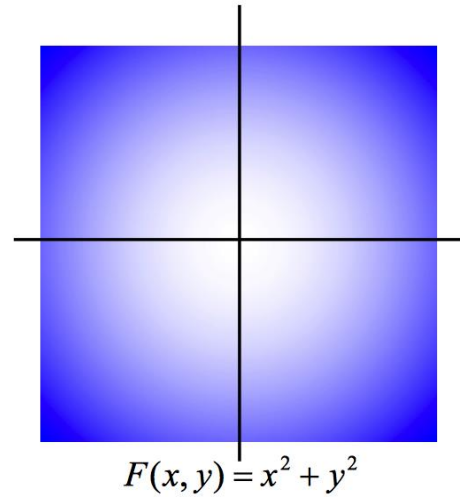
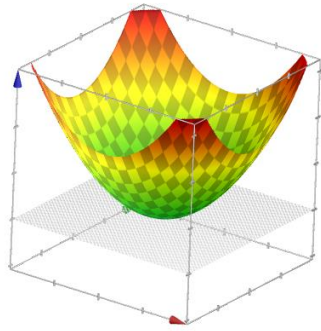
- Given a function $\mathbf{F}(F_1, F_2): \mathbb{R}^2 \rightarrow \mathbb{R}^2$ the **divergence** of $\mathbf{F}$ is the function **div**: $\mathbb{R}^2 \rightarrow \mathbb{R}$ defined as:

$$\text{div } \mathbf{F}(x, y) = \partial F_1 / \partial x + \partial F_2 / \partial y$$

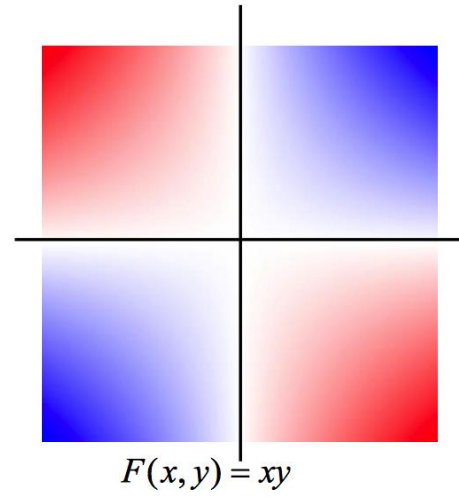
- **Intuitively**: At the point p_0 , the divergence $\text{div } \mathbf{F}(p_0)$ is a measure of the extent to which the flow (de)compresses at p_0 .

Some math.... Gradient and divergence

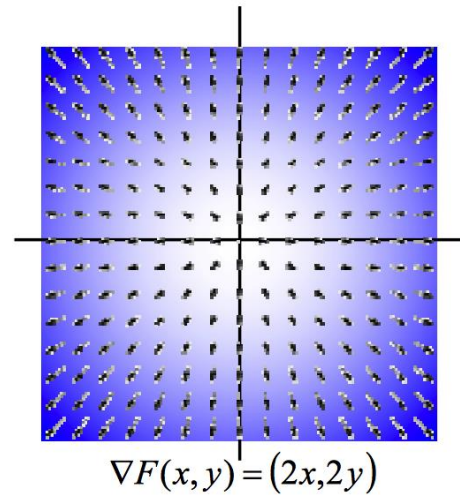
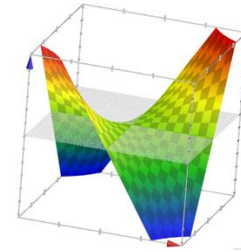
• E



$$F(x, y) = x^2 + y^2$$

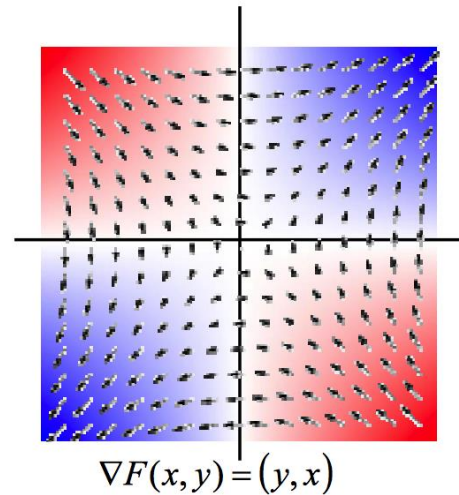


$$F(x, y) = xy$$



$$\nabla F(x, y) = (2x, 2y)$$

$$\text{div } \nabla F(x, y) = 4$$



$$\nabla F(x, y) = (y, x)$$

$$\text{div } \nabla F(x, y) = 0$$

Some math.... Laplacian

• Given a function $F(F_1, F_2): \mathbb{R}^2 \rightarrow \mathbb{R}$
the **Laplacian** of F is the function $\Delta F: \mathbb{R}^2 \rightarrow \mathbb{R}$ defined by the divergence of the gradient of the partial derivatives:

$$\Delta F = \text{div}(\nabla F(x, y)) = \partial^2 F / \partial x^2 + \partial^2 F / \partial y^2$$

• **Intuitively:** The Laplacian of F at the point p_0 measures the extent to which the value of F at p_0 differs from the average value of F its neighbors.

Discrete Differential Operators

- Assumption: Meshes are piecewise linear approximations of smooth surfaces
- Approach: Approximate differential properties at point x as spatial average over local mesh neighborhood $N(x)$, where typically
 - x = mesh vertex
 - $N(x)$ = n -ring neighborhood (or local geodesic ball)

Discrete Laplacian

- Uniform discretization

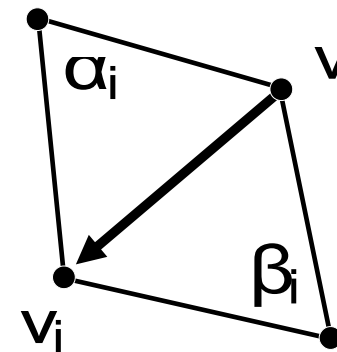
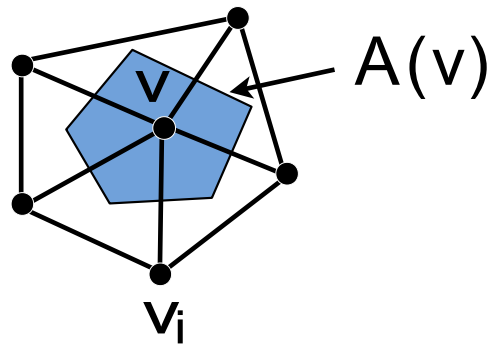
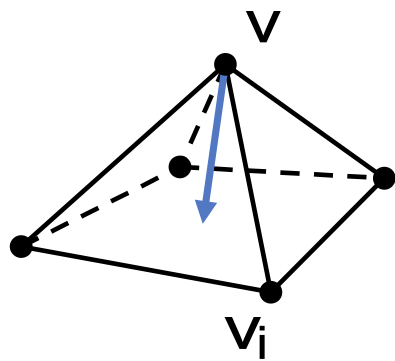
$$\Delta_{uni} f(v) := \frac{1}{|\mathcal{N}_1(v)|} \sum_{v_i \in \mathcal{N}_1(v)} (f(v_i) - f(v))$$

- depends only on connectivity $\rightarrow$ simple and efficient
- bad approximation for irregular triangulations

Discrete Laplacian

Cotangent formula

$$\Delta_{\mathcal{S}} f(v) := \frac{2}{A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot \alpha_i + \cot \beta_i) (f(v_i) - f(v))$$



Discrete Curvatures

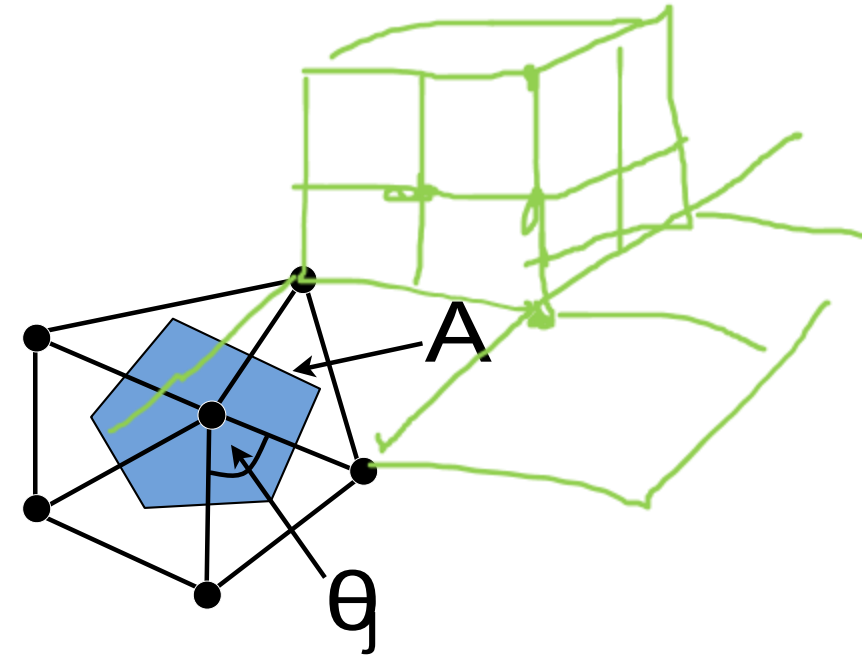
- Mean Curvature $H = \|\Delta_S \mathbf{x}\|$

- Gaussian Curvature $G = (2\pi - \sum_j \theta_j) / A$

- Principal Curvatures

$$\kappa_1 = H + \sqrt{H^2 - G}$$

$$\kappa_2 = H - \sqrt{H^2 - G}$$



Gaussian curvature on a triangle mesh

❖ It's the *angle defect* over the area

❖

$$\kappa_G(v_i) = \frac{1}{3A} \left(2\pi - \sum_{t_j \text{ adj } v_i} \theta_j \right)$$

❖ **Gauss-Bonnet Theorem:** The integral of the Gaussian Curvature on a closed surface depends on the Euler number

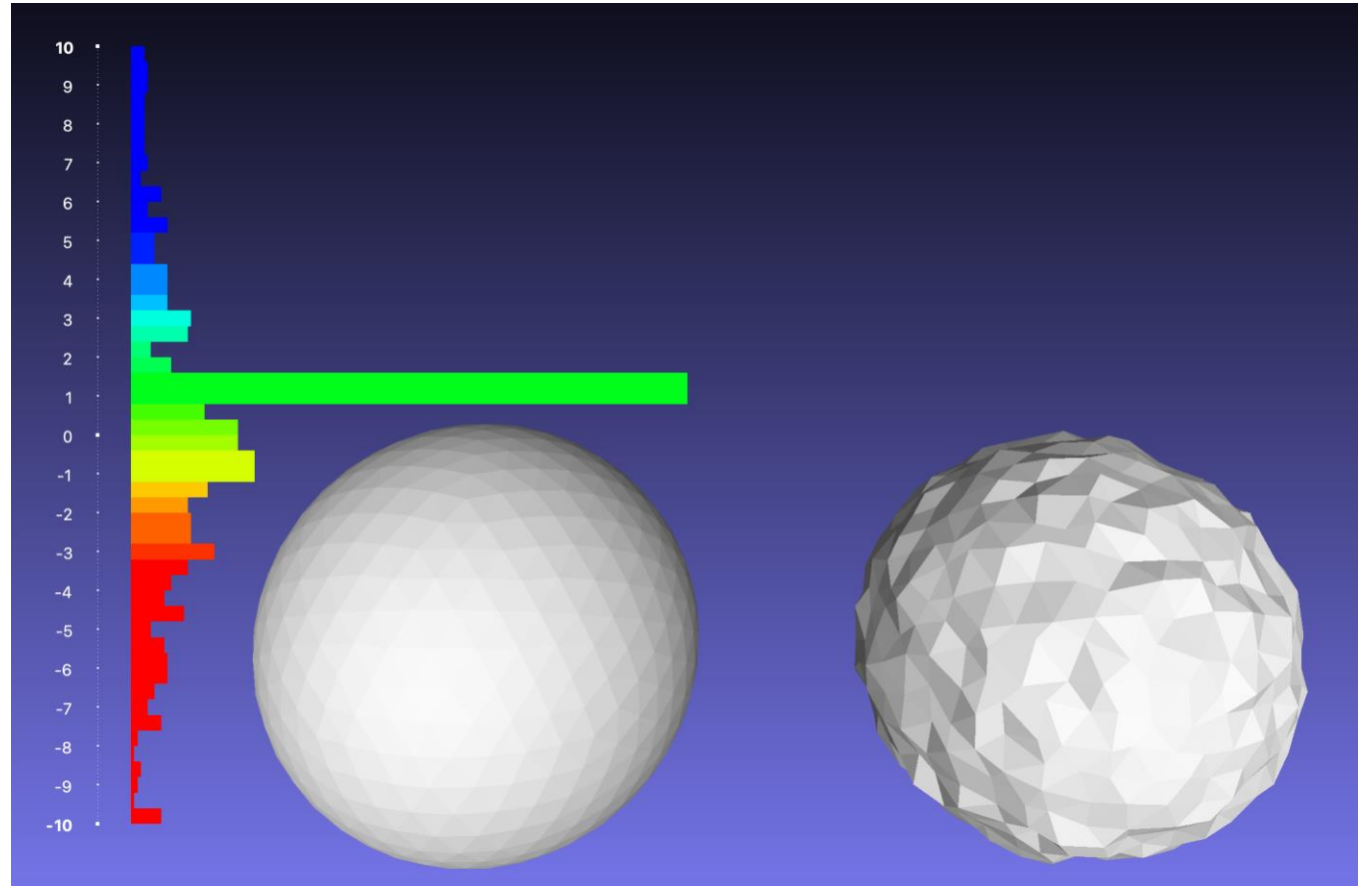
$$\int_S \kappa_G = 2\pi \chi$$



Discrete Curvatures

Problems:

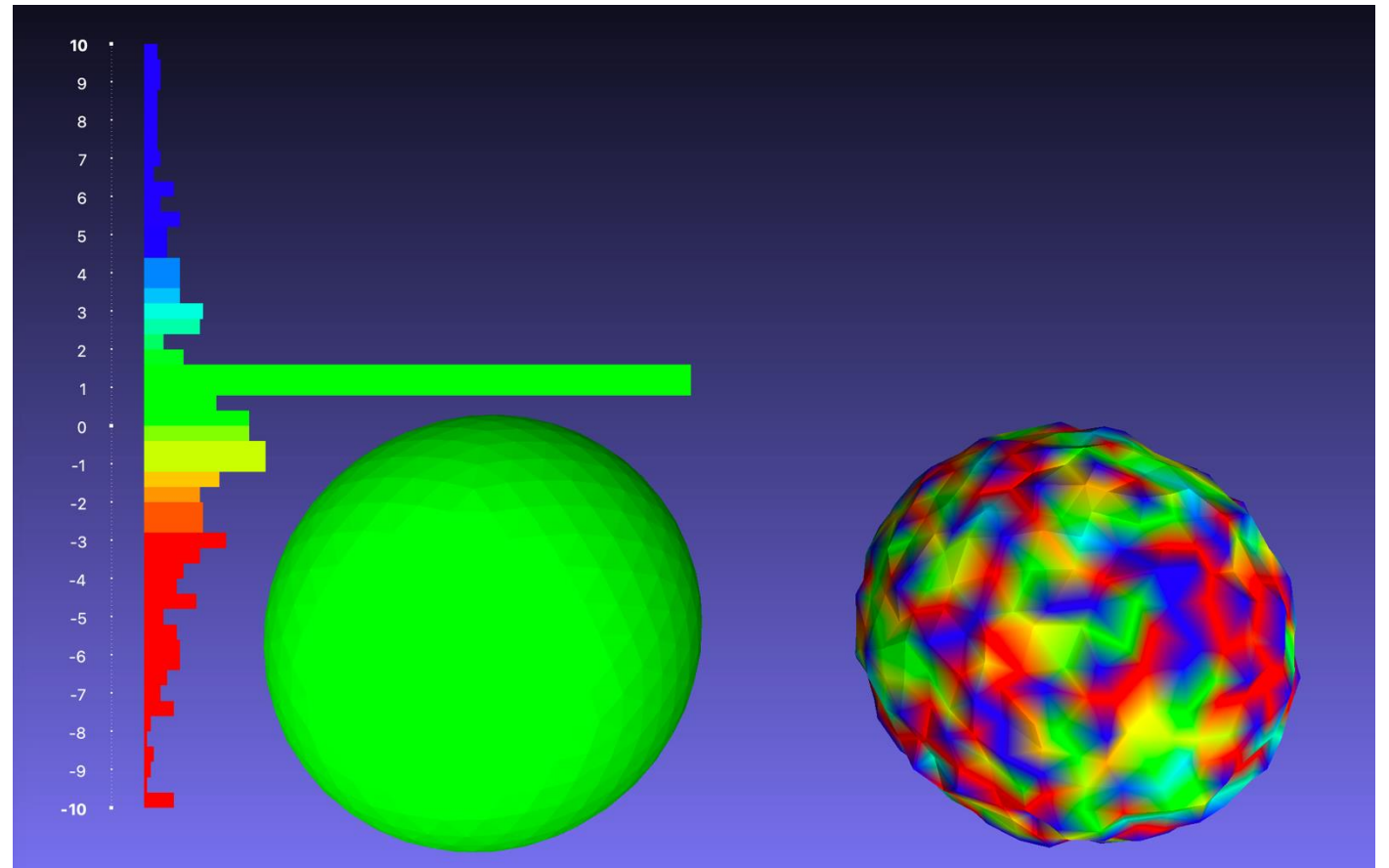
- Depends on triangulation!
- Very sensitive to Noise...



Discrete Curvatures

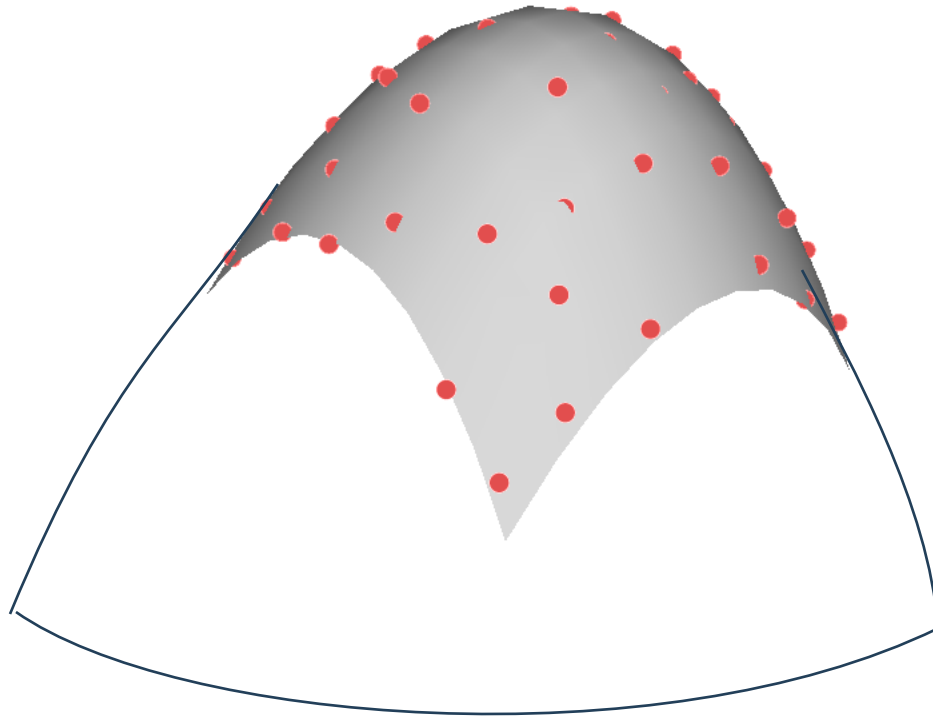
Problems:

- Depends on triangulation!
- Very sensitive to Noise...



Curvature via Surface Fitting

- Main idea: forget the connectivity
- Consider only some vertices and fits a quadric over some of them.
- Use this analytic function to compute the differential properties

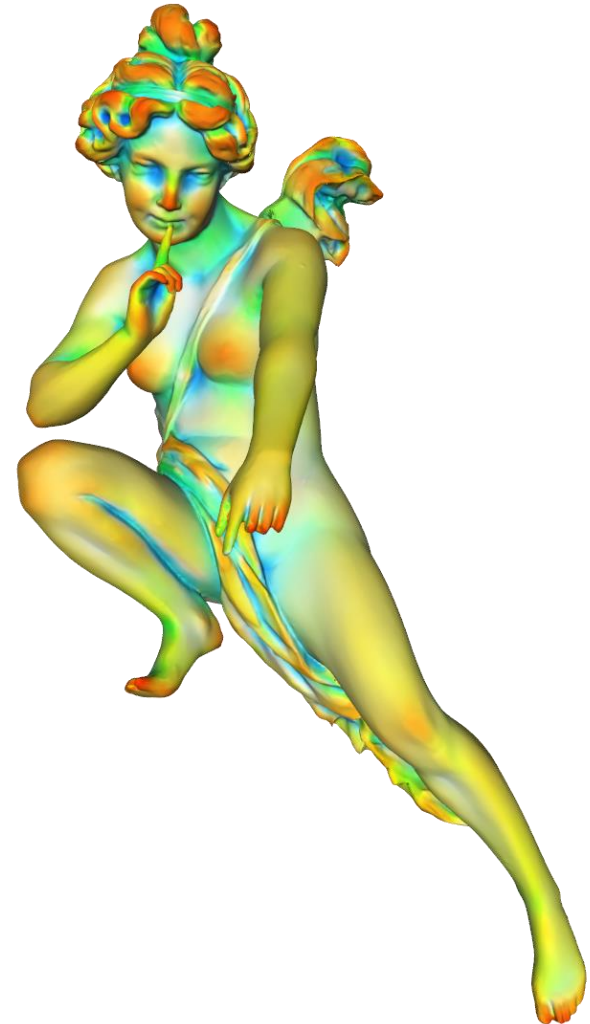


Curvature via Surface Fitting

- The radius r of the neighborhood of each point p is used as a scale parameter
 - 1. gather all faces in a local neighborhood of radius r

$$\mathbf{w} = \frac{1}{n_v} \sum_{i=1}^n \mathbf{n}_i$$

- 2. set an axis
- where n_v is the number of vertices gathered and n_i is the surface normal at each such vertex

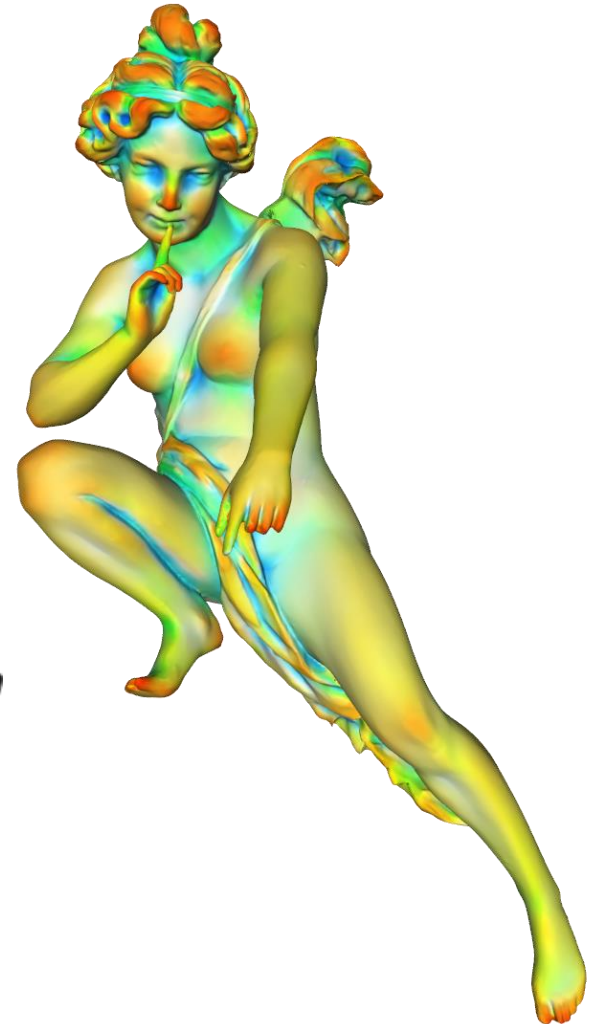


Curvature via Surface Fitting

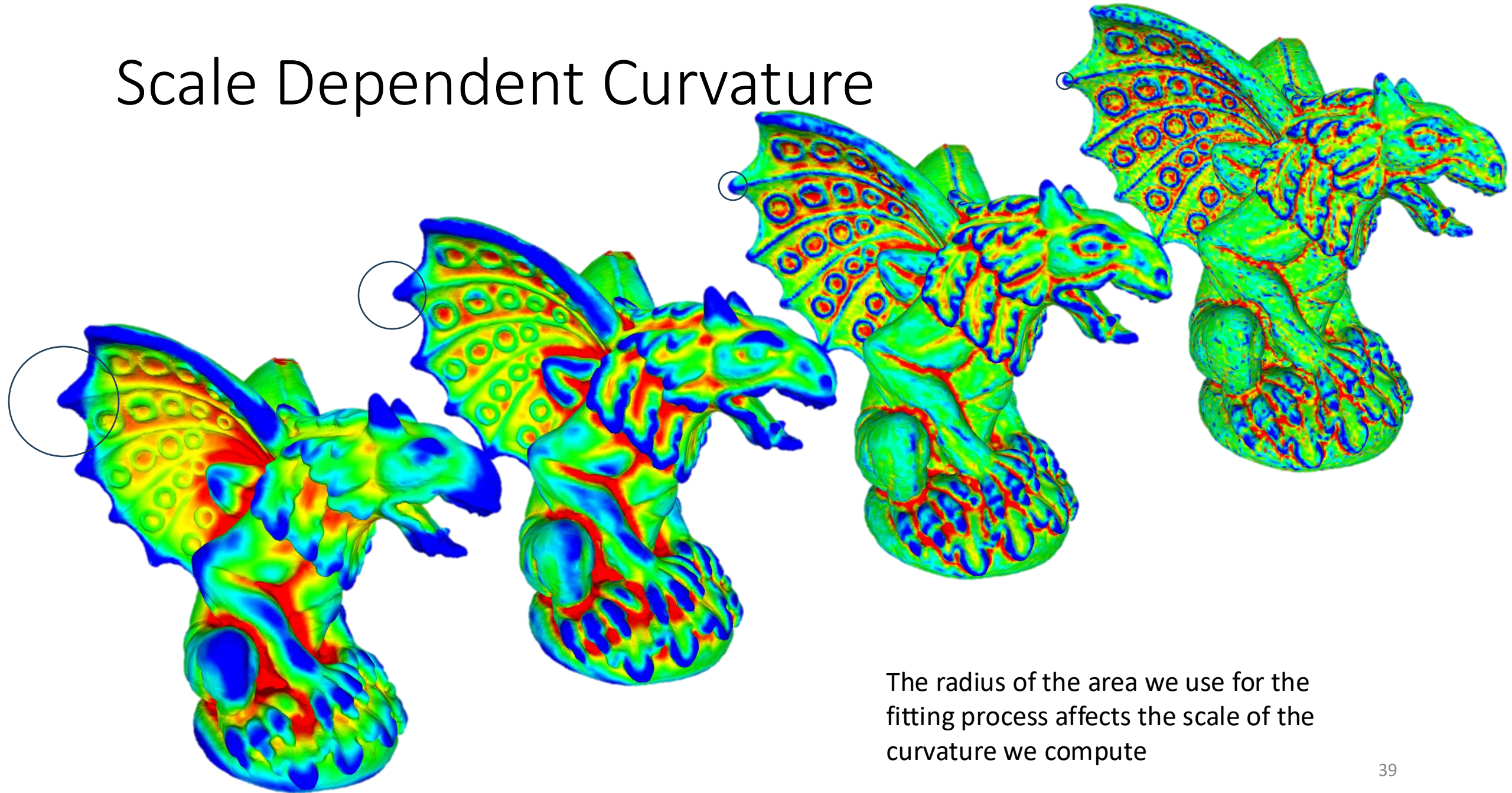
3. discard all vertices v_i such that $n_i \cdot w < 0$
4. set a local frame (u, v, w) where u and v are any two orthogonal unit vectors lying on the plane orthogonal to w , and such that the frame is right-handed
5. express all vertices of the neighborhood in such a local frame with origin at p
6. fit to these points a polynomial of degree two through p (least squares fitting)

$$f(u, v) = au^2 + bv^2 + cuv + du + ev$$

Curvatures at p are computed **analytically via first and second fundamental forms** of f at the origin



Scale Dependent Curvature



The radius of the area we use for the fitting process affects the scale of the curvature we compute

Screen Space Mean Curvature

```
// License: CC0 (http://creativecommons.org/publicdomain/zero/1.0/)  
#extension GL_OES_standard_derivatives : enable
```

```
varying vec3 normal;  
varying vec3 vertex;
```

```
void main() {  
    vec3 n = normalize(normal);  
  
    // Compute curvature  
    vec3 dx = dFdx(n);  
    vec3 dy = dFdy(n);  
    vec3 xneg = n - dx;  
    vec3 xpos = n + dx;  
    vec3 yneg = n - dy;  
    vec3 ypos = n + dy;  
    float depth = length(vertex);  
    float curvature = (cross(xneg, xpos).y - cross(yneg, ypos).x) * 4.0 / depth;  
  
    // Compute surface properties  
    vec3 light = vec3(0.0);  
    vec3 ambient = vec3(curvature + 0.5);  
    vec3 diffuse = vec3(0.0);  
    vec3 specular = vec3(0.0);  
    float shininess = 0.0;  
  
    // Compute final color  
    float cosAngle = dot(n, light);  
    gl_FragColor.rgb = ambient +  
        diffuse * max(0.0, cosAngle) +  
        specular * pow(max(0.0, cosAngle), shininess);  
}
```

- Known effect as Cavity Shading

