Discrete Differential Geometry

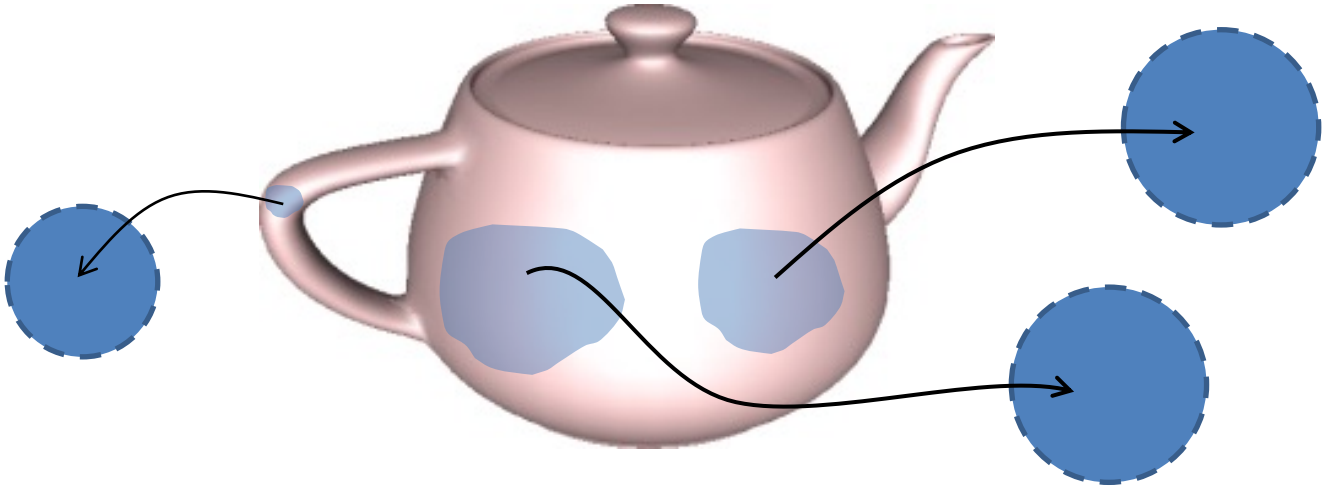
Prof Cengiz Öztireli

UNIVERSITY OF CAMBRIDGE

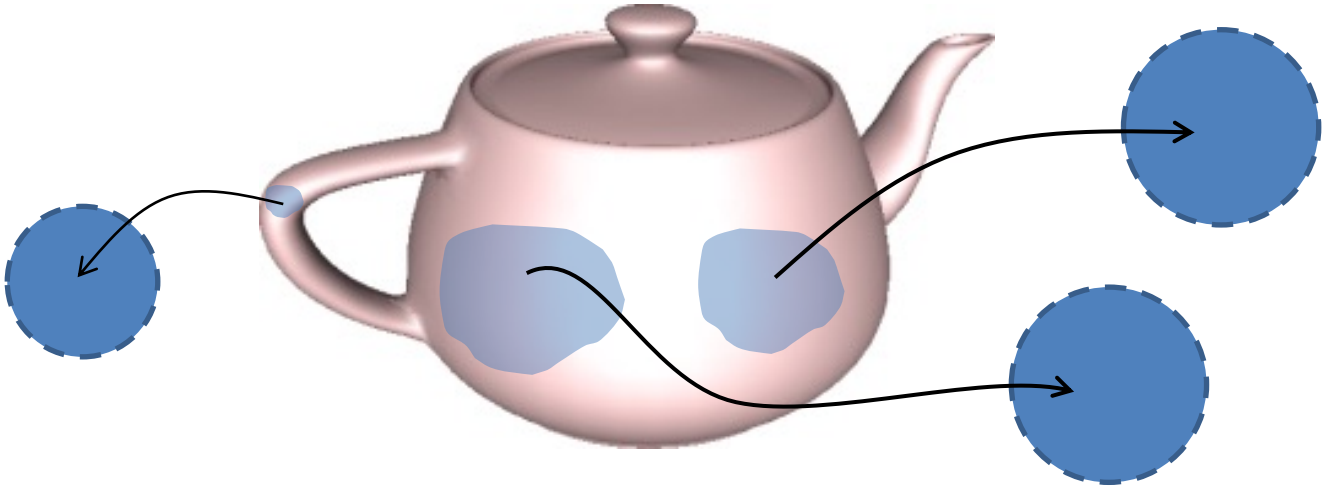
Manifolds

Informally, if we can take any local patch from the surface, and flatten it somehow and possibly with some distortion into a disk on a plane, we call that surface a manifold.

- A surface is a closed **2-manifold** if it is locally homeomorphic to a disk everywhere



- For every point x in M, there is an **open** ball $B_x(r)$ of radius r centered at x such that $M \cap B_x(r)$ is homeomorphic to an open disk



Similarly, for a manifold with boundary, there are some neighborhoods, which we can map to half-disks. These correspond to the patches that intersect the boundary.

- Each boundary point is homeomorphic to a half-disk

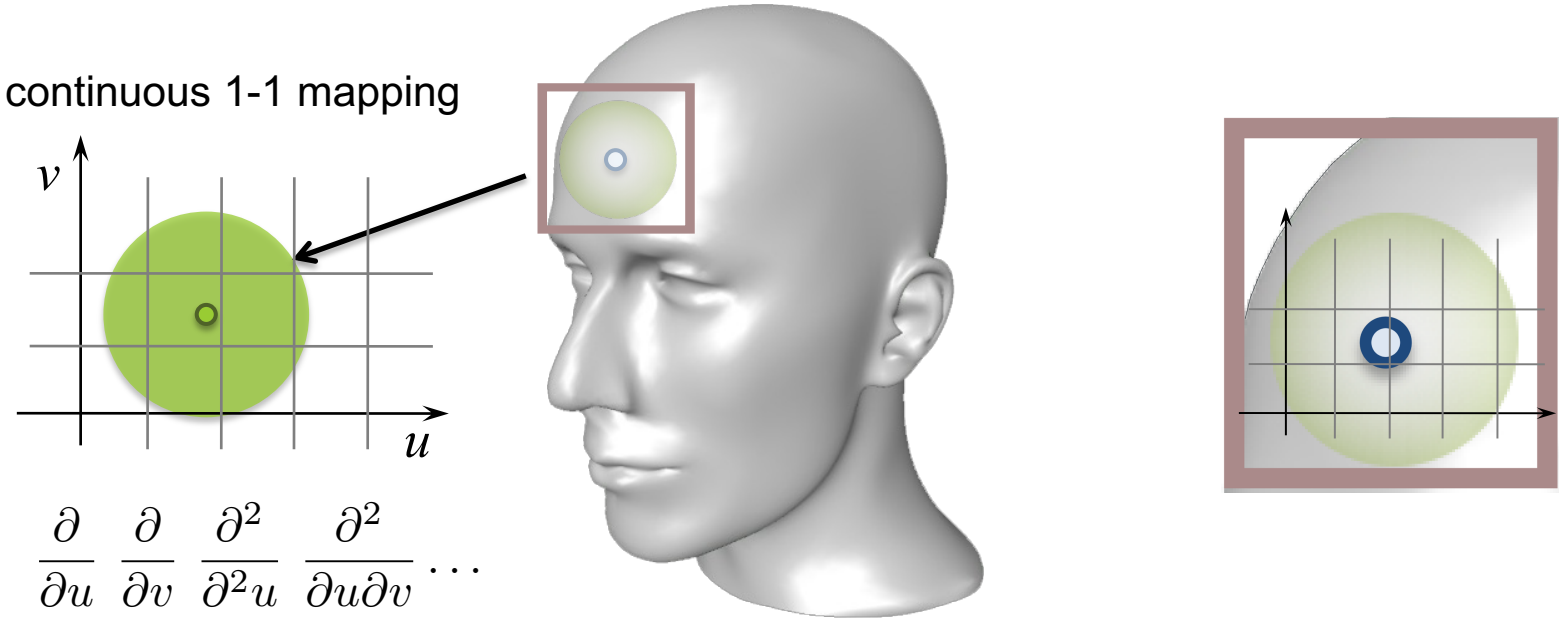
Differential Geometry Basics

For manifolds, we can define differential geometry.

It explains local properties of surfaces.

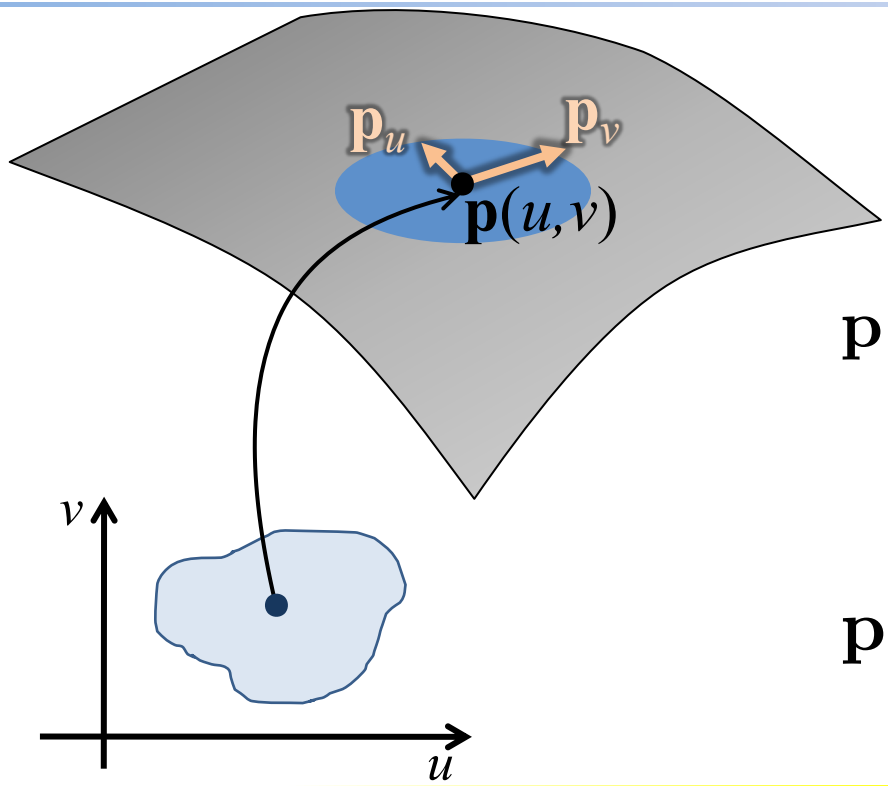
Let's first assume we have a local parametrization with u and v.

Things that can be discovered by local observation



Local Coordinates

As we have seen before, with such a parametrization, we can go on and define the tangent vectors spanning the tangent plane around the point $\mathbf{p}$.



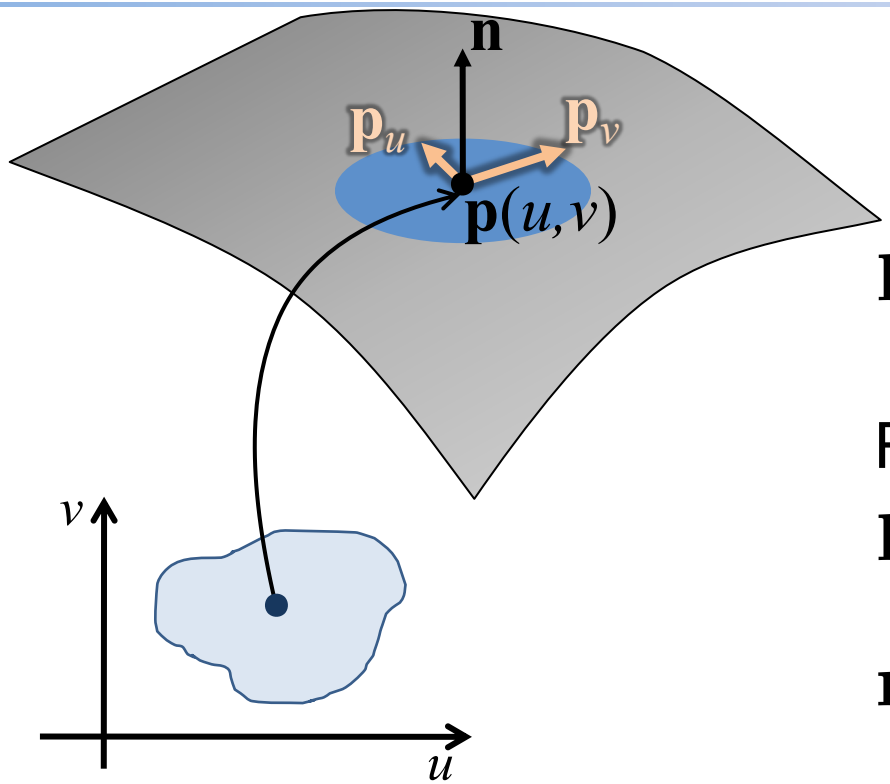
$$\mathbf{p}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in \mathbb{R}^2$$

$$\mathbf{p}_u = \frac{\partial \mathbf{p}(u,v)}{\partial u}, \quad \mathbf{p}_v = \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

Surface Normal

The surface normal is defined as the vector orthogonal to the tangent plane.

Note that this slide is exactly the same as the one we had for parametric surfaces. The difference here is that we have a local parametric surface and hence it applies to all kinds of manifold surfaces.



$$\mathbf{p}_u = \frac{\partial \mathbf{p}(u,v)}{\partial u}, \quad \mathbf{p}_v = \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

Regular parametrization:

$$\mathbf{p}_u \times \mathbf{p}_v \neq 0$$

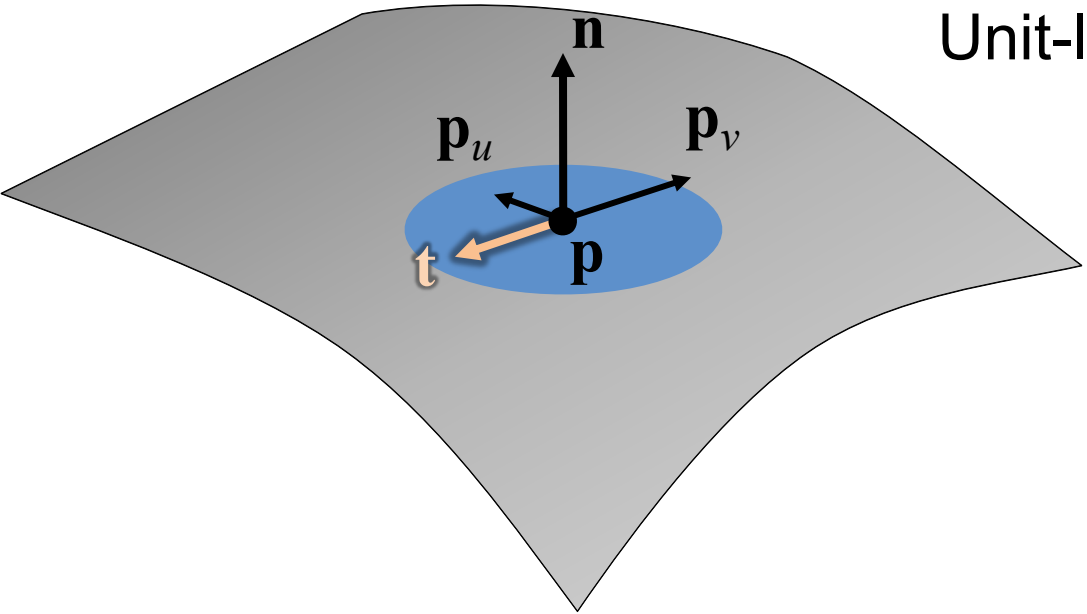
$$\mathbf{n}(u,v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}$$

Normal Curvature

Now we can define what curvature is.

Imagine you have a vector rotating on the plane.

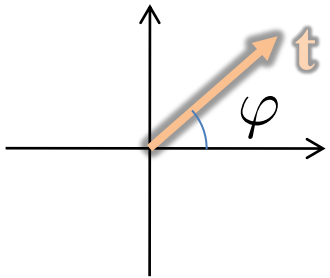
This can be represented with a simple formula if we assume that the two basis vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ are orthogonal.



Unit-length $\mathbf{t}$ in the tangent plane

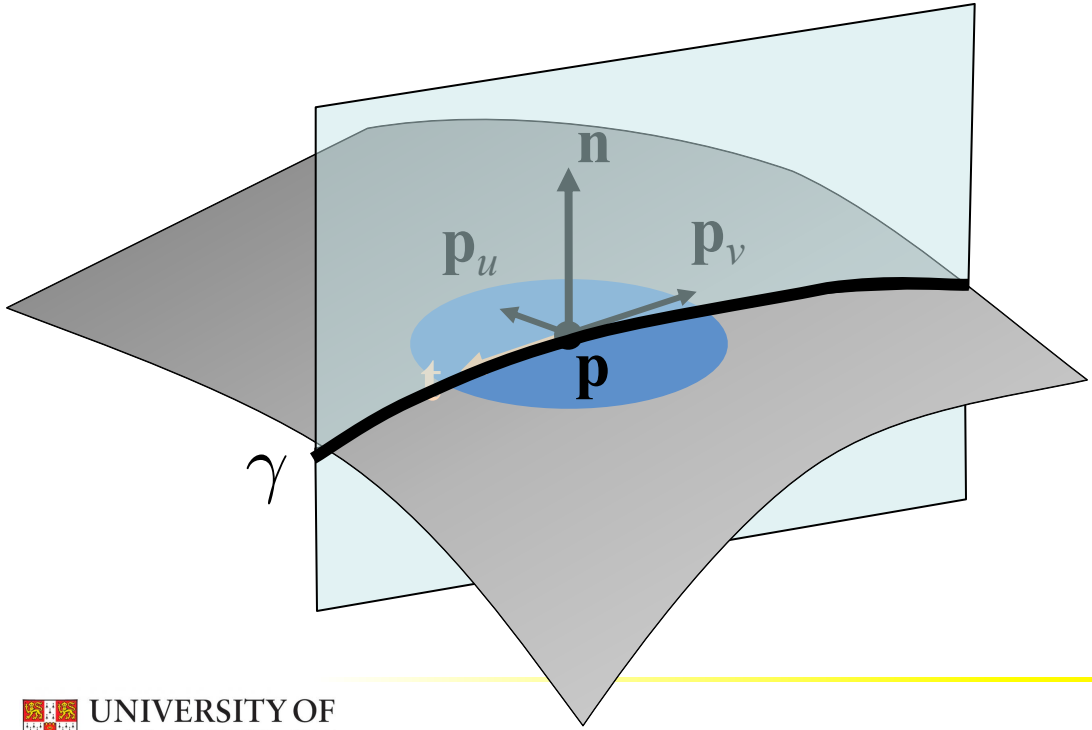
If $\mathbf{p}_u$ and $\mathbf{p}_v$ are orthogonal:

$$\mathbf{t} = \cos\varphi \frac{\mathbf{p}_u}{\|\mathbf{p}_u\|} + \sin\varphi \frac{\mathbf{p}_v}{\|\mathbf{p}_v\|}$$



Normal Curvature

There is a curve formed by the intersection of the surface and a plane spanned by the vectors $\mathbf{t}$ and $\mathbf{n}$.
Normal curvature is the curvature of that curve at the point $\mathbf{p}$.
Normal curvature is thus direction dependent.



The curve γ is the intersection of the surface with the plane through $\mathbf{n}$ and $\mathbf{t}$.

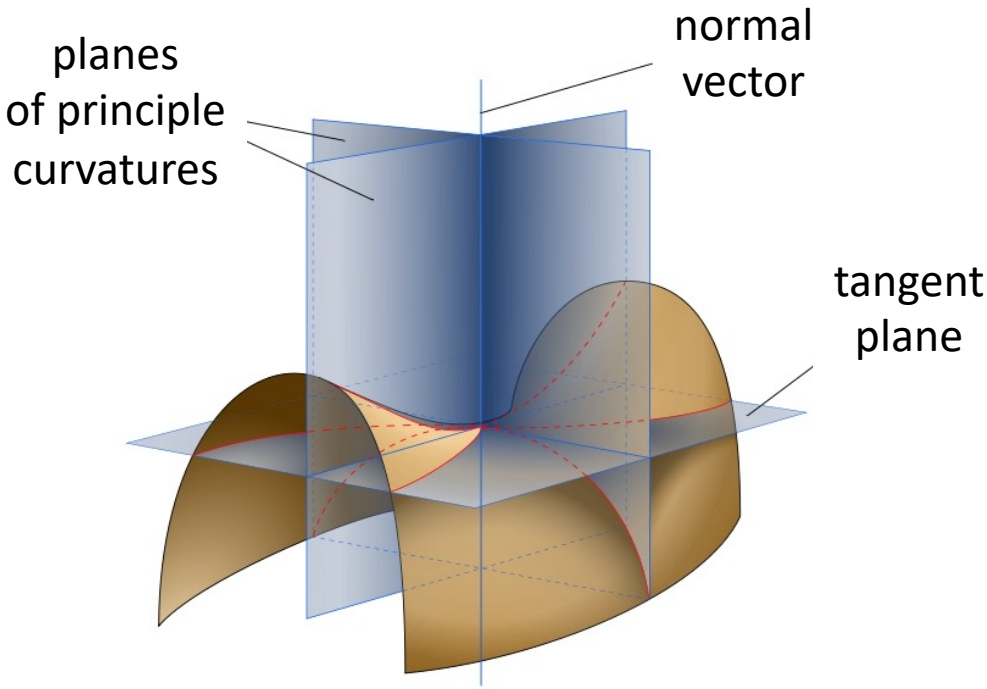
Normal curvature:

$$\kappa_n(\varphi) = \kappa_n(\gamma(\mathbf{p}))$$

- Principal curvatures
  - Minimal curvature $\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$
  - Maximal curvature $\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$
- Mean curvature $H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\varphi) d\varphi$
- Gaussian curvature $K = \kappa_1 \cdot \kappa_2$

Principle Directions

Recall that principle curvatures are the minimum and maximum normal curvatures.
The directions that correspond to these curvatures are the principal directions.
Euler's theorem says: we can write any curvature as a linear combination of the principle curvatures.
The angle here is between the direction for the curvature and that for the minimum normal curvature.



Euler's Theorem:
Planes of principal curvature are **orthogonal** and independent of parameterization.

$$\kappa_n(\varphi) = \kappa_1 \cos^2 \varphi + \kappa_2 \sin^2 \varphi$$

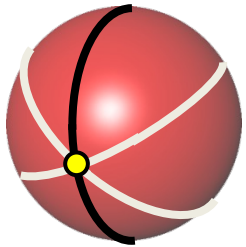
$$\varphi = \text{angle with } \mathbf{t}_1$$

Local Shape by Curvatures

We can characterize surfaces *locally* via curvatures. E.g. isotropic, i.e. the same curvature in all directions.

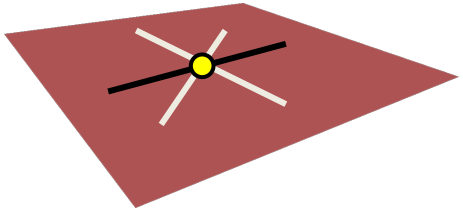
Isotropic:
all directions are
principal directions

spherical (umbilical)



$K > 0, \kappa_1 = \kappa_2$

planar

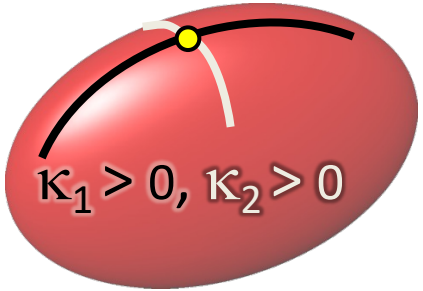


$K = 0$

Local Shape by Curvatures

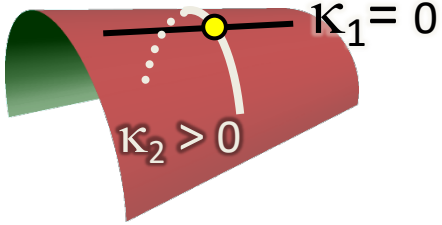
Anisotropic:
2 distinct principal directions

elliptic



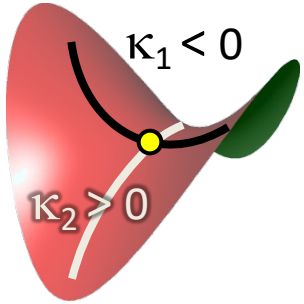
$K > 0$

parabolic



$K = 0$

hyperbolic

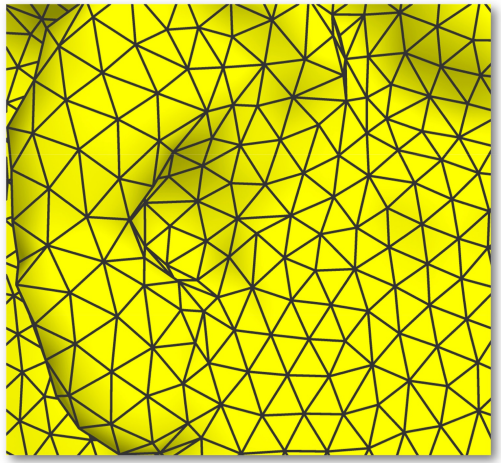


$K < 0$

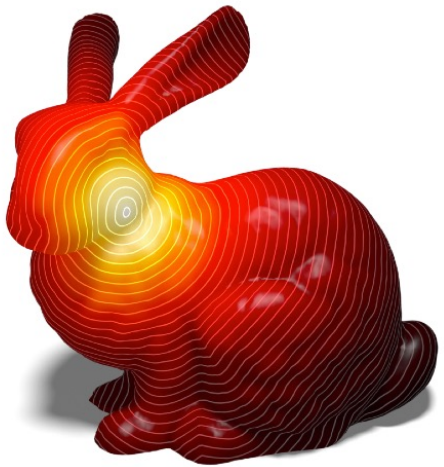
So far, we have talked about surface normal and curvatures for smooth manifold surfaces.
In computer graphics, we often have discrete surfaces, e.g. triangle meshes.
How do we define these differential quantities then? We will talk about two methods: local and global and focus on the latter.

- Approximate surface normal and curvature via

Local surface approximation



Global: discrete Laplace-Beltrami

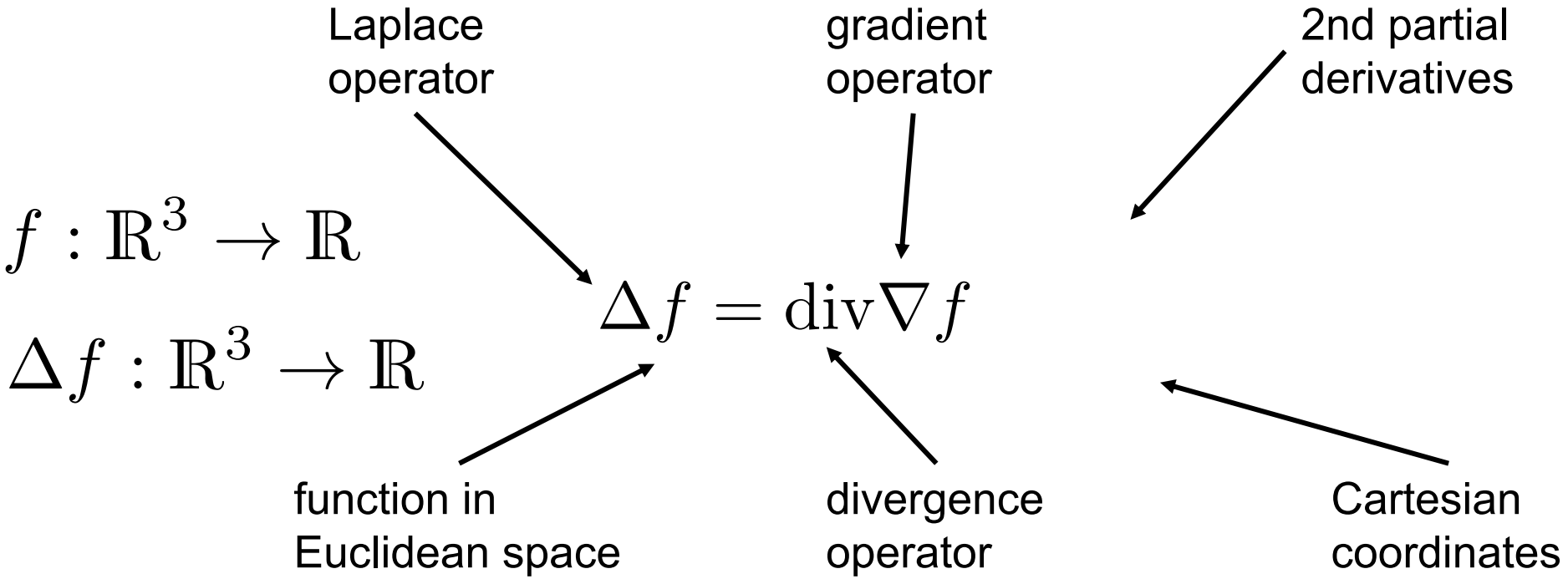


Laplace Operator

Let's start with the global method.

We first need to define the Laplace operator in a Euclidean space.

When applied to a function, it is equivalent to first applying the gradient and then the divergence operators.



Laplace Operator

Recall the definitions of gradient and divergence.

If you apply these, you simple get sum of the second derivatives, assuming the function lives in a Euclidean space.

$$f : \mathbb{R}^3 \to \mathbb{R} \qquad \Delta f : \mathbb{R}^3 \to \mathbb{R}$$

$$\Delta f = \mathrm{div} \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

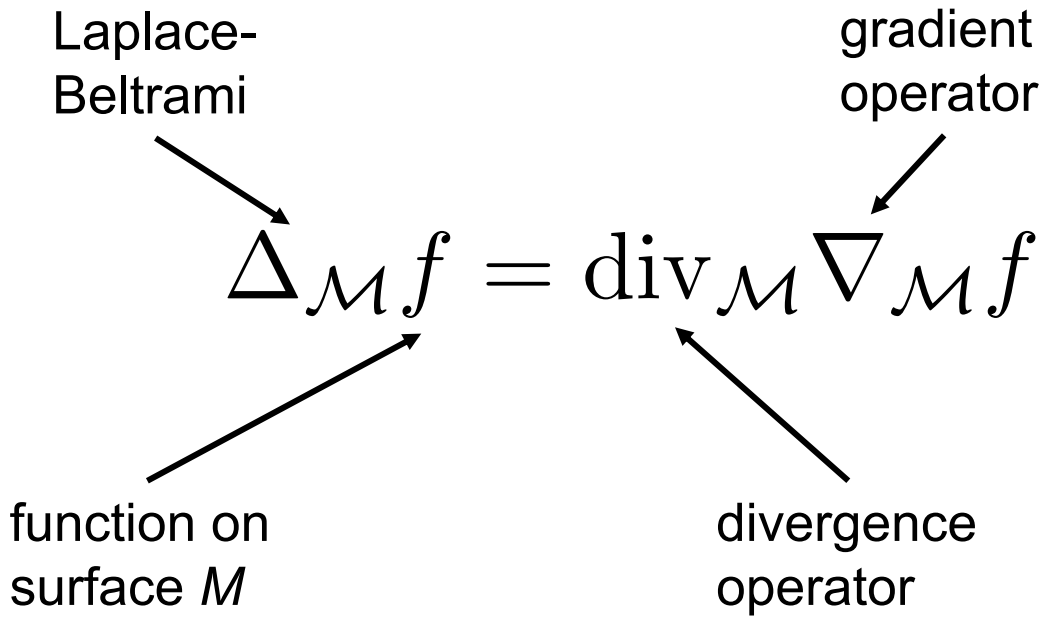
$$\mathrm{grad} f = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \qquad \mathrm{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

Laplace-Beltrami Operator

The Laplace operator has an extension to manifold surfaces.
Let's assume we have a function that lives on the surface, which is not a Euclidean space in general.
We then have the same operators defined a bit differently. This is a generalization of the Laplace operator.

- Extension to manifold surfaces

$$f : \mathcal{M} \to \mathbb{R}$$
$$\Delta f : \mathcal{M} \to \mathbb{R}$$



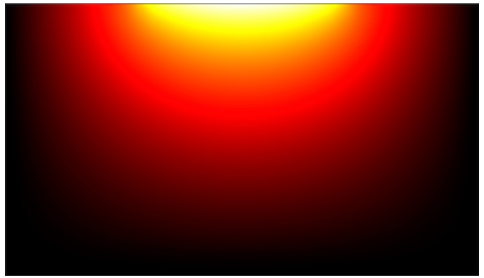
Laplace-Beltrami Operator

As a side note, this is a very important operator to solve differential equations on surfaces.

An example is how heat diffuses on a surface. You can already see from the figures that this is related to curvature and local surface characteristics.

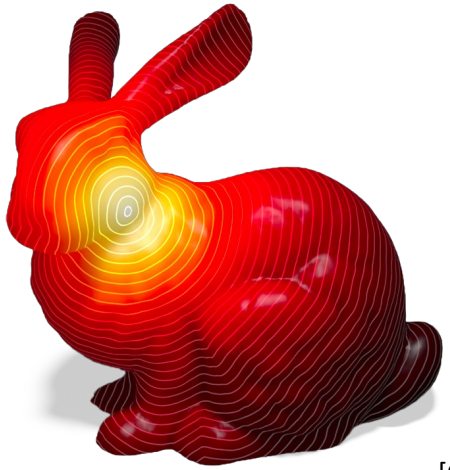
In general, this operator is very important for geometry processing. If interested, you may read here.

- Example: heat equation



$$\Delta f = 0$$

$$\text{s.t. } f|_{\partial\Omega} = f_0$$



[Crane et al. 2013]

$$\Delta_{\mathcal{M}} f = 0$$

$$\text{s.t. } \textit{boundary conditions}$$

A very interesting property of this operator is: if you apply it to the coordinate function, i.e. the coordinates of a point on the surface in 3D, you get the scaled surface normal at that point. The scaling is exactly -2 times the mean curvature. We will not go into the proof.

- Apply to coordinate function

$$f(x, y, z) = x \quad \mathbf{p} = (x, y, z)$$

Laplace-Beltrami

gradient operator

mean curvature

$$\Delta_{\mathcal{M}}\mathbf{p} = \operatorname{div}_{\mathcal{M}}\nabla_{\mathcal{M}}\mathbf{p} = -2H\mathbf{n} \in \mathbb{R}^3$$

function on surface *M*

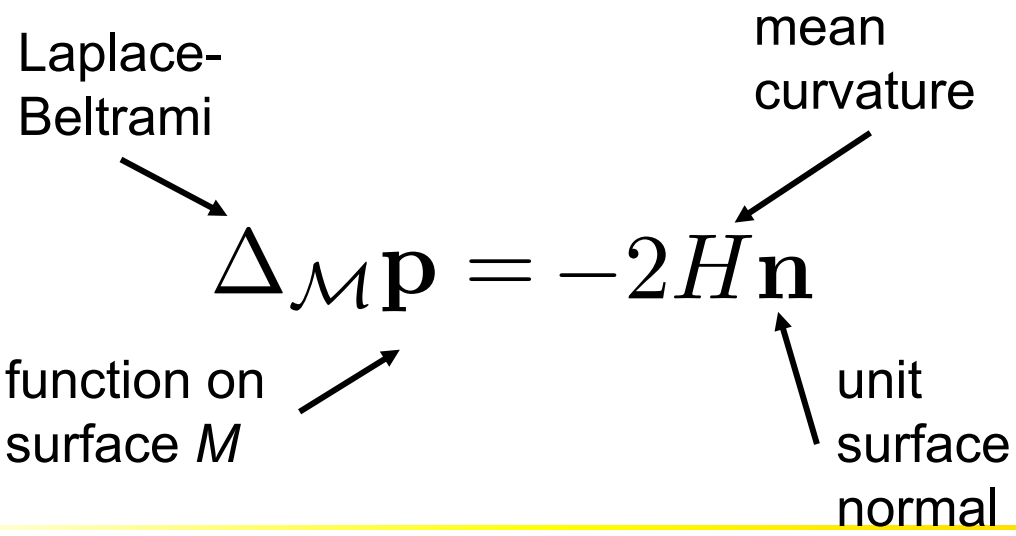
divergence operator

unit surface normal

Now we can see why this operator is so important: it provides us a way to define curvature and normal in terms of the application of it to the coordinate function. If we can define the application to the coordinate function on a discrete surface, we get the mean curvature and surface normal.

- Apply to coordinate function

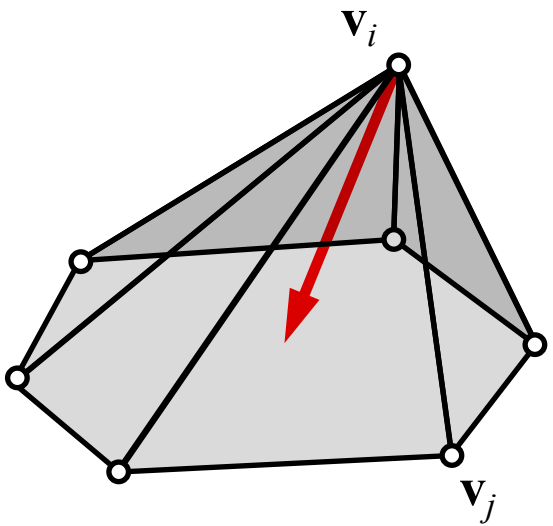
$$f(x, y, z) = x \quad \mathbf{p} = (x, y, z)$$



Discrete Laplace-Beltrami

For a mesh, a very simple definition is: for a vertex, take the mean of the difference vectors to the neighbors. This is the same as averaging the neighboring vertex locations and subtracting from the location of the vertex. This gives us an approximation of -2$H\mathbf{n}$.

$$\Delta_{\mathcal{M}}\mathbf{p} = -2H\mathbf{n}$$



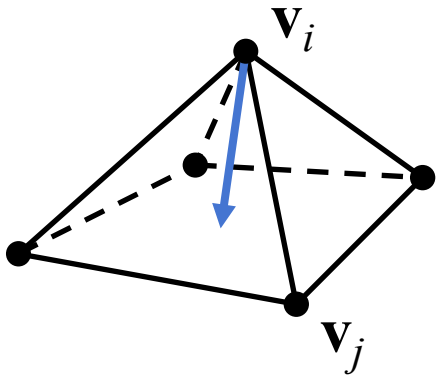
$$L_u(\mathbf{v}_i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\mathbf{v}_j - \mathbf{v}_i)$$

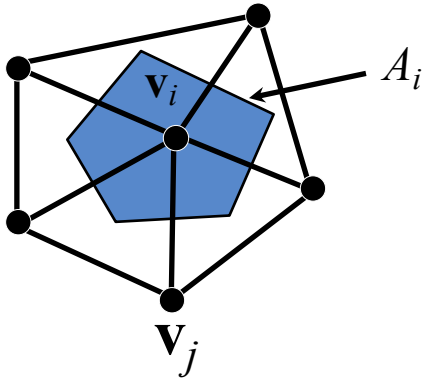
$$= \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{v}_j \right) - \mathbf{v}_i$$

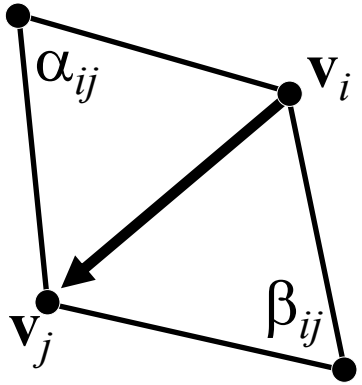
Discrete Laplace-Beltrami

For a general mesh, we need to be more careful with the weights to get a good approximation.
One general scheme for triangular meshes is using the so-called cotangent weights.
For a given vertex, an angle is defined per neighbor as below. We define the area A_i in the next slide.

$$L_c(\mathbf{v}_i) = \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$





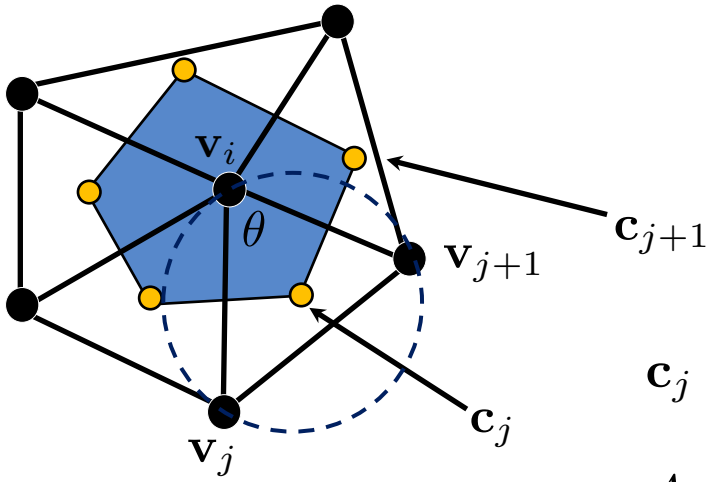


Discrete Laplace-Beltrami

The area A_i has a slightly complex definition.

It is a sum of each of the areas defined for each neighboring vertex.

Below, $\triangle$ defines a triangle.



$$\mathbf{c}_j = \begin{cases} \text{circumcenter of } \triangle(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_{j+1}) & \text{if } \theta < \pi/2 \\ \text{midpoint of edge } (\mathbf{v}_j, \mathbf{v}_{j+1}) & \text{if } \theta \geq \pi/2 \end{cases}$$

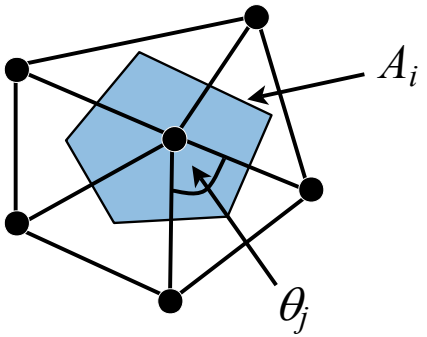
$$A_i = \sum_j \text{Area}\left(\triangle(\mathbf{v}_i, \mathbf{c}_j, \mathbf{c}_{j+1})\right)$$

- Mean curvature (sign according to normal)

$$|H(\mathbf{v}_i)| = \|L_c(\mathbf{v}_i)\|/2$$

- Gaussian curvature

$$K(\mathbf{v}_i) = \frac{1}{A_i}(2\pi - \sum_j \theta_j)$$

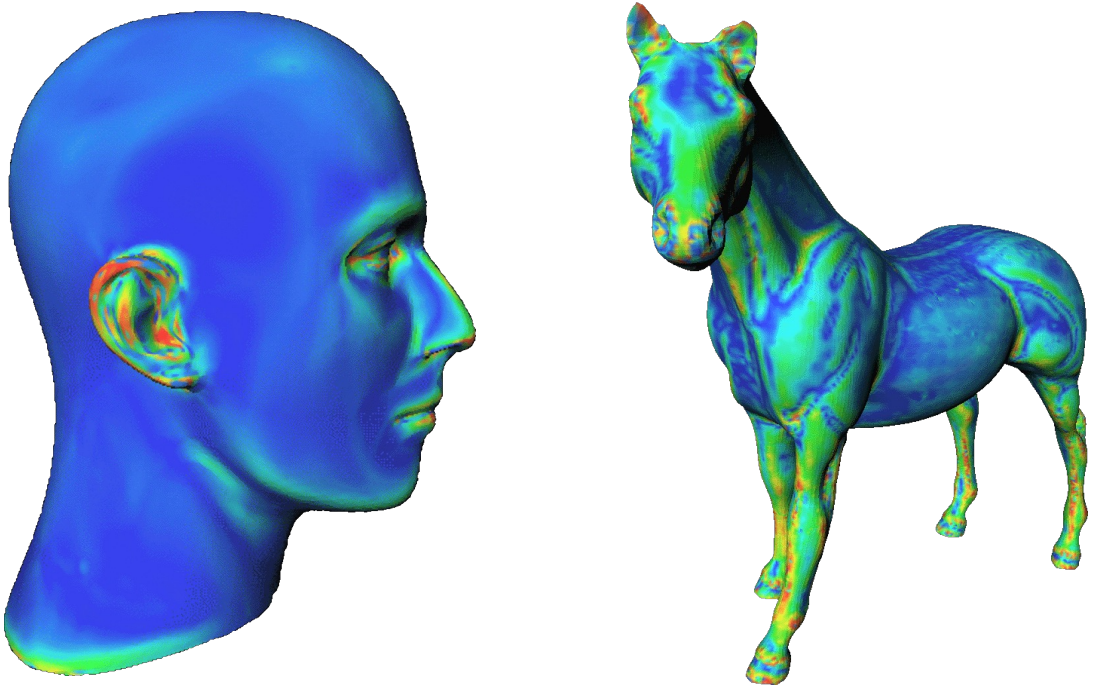


- Principal curvatures

$$\kappa_1 = H - \sqrt{H^2 - K} \qquad \kappa_2 = H + \sqrt{H^2 - K}$$

Discrete Curvatures

Mean Curvature



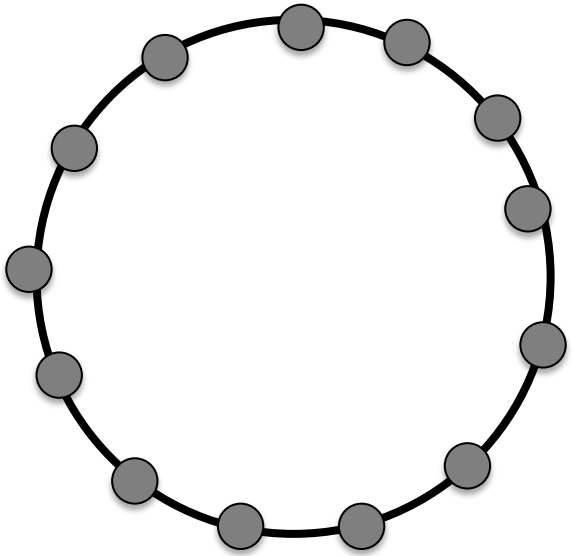
Discrete Laplace-Beltrami

We have considered meshes as a discrete surface so far.

We can do the same for graphs and point clouds: define an approximation of the application of the Laplace-Beltrami operator to the coordinate function and approximate the curvatures.

In this case, the approximation is with Gaussian weights instead of cotangent weights.

- # Extension to graphs and point clouds



$$h_t(x_i, x_j) = e^{-d(x_i,x_j)/t}$$

$$= e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/t}$$

$$L_g(x_i) = \frac{1}{\sum_{j=1}^{n} h_t(x_i, x_j)} \sum_{j=1}^{n} h_t(x_i, x_j)(x_j - x_i)$$

- Extension to graphs and point clouds