Discrete Differential Geometry

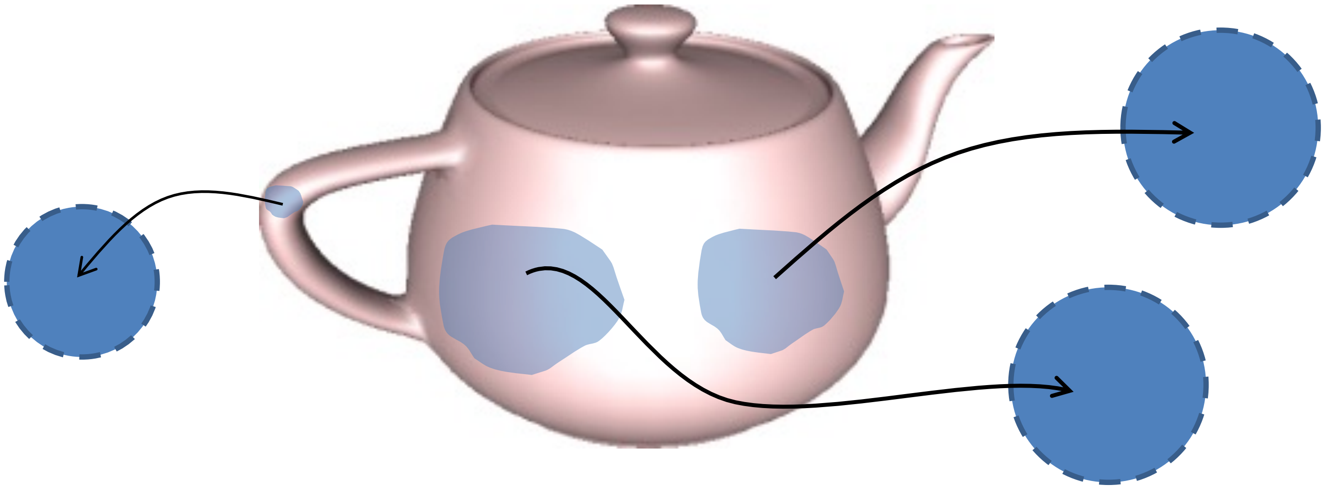
Prof Cengiz Öztireli

UNIVERSITY OF CAMBRIDGE

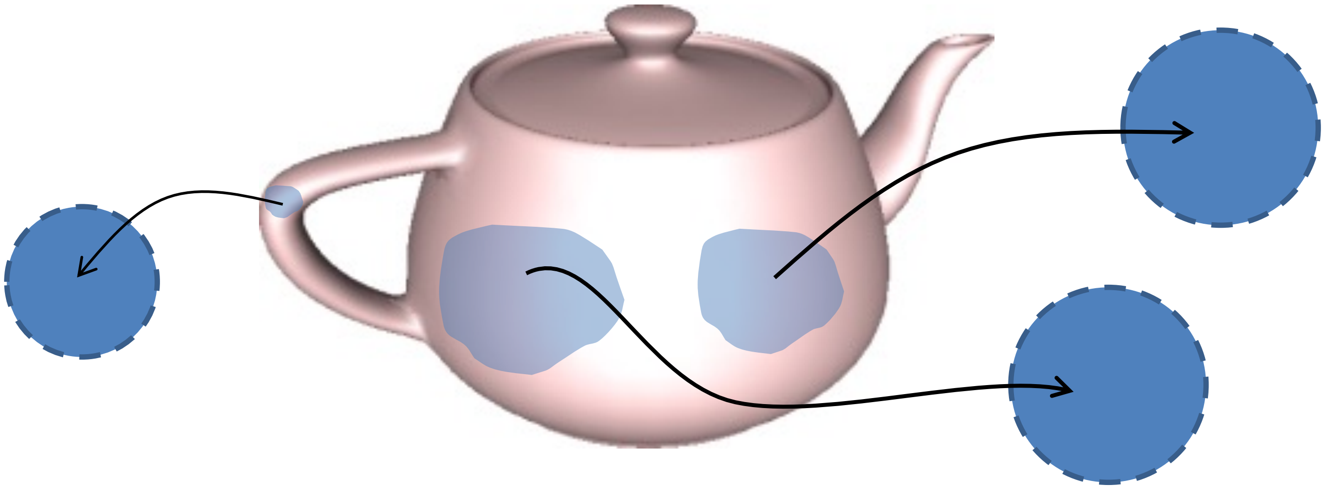
Manifolds

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



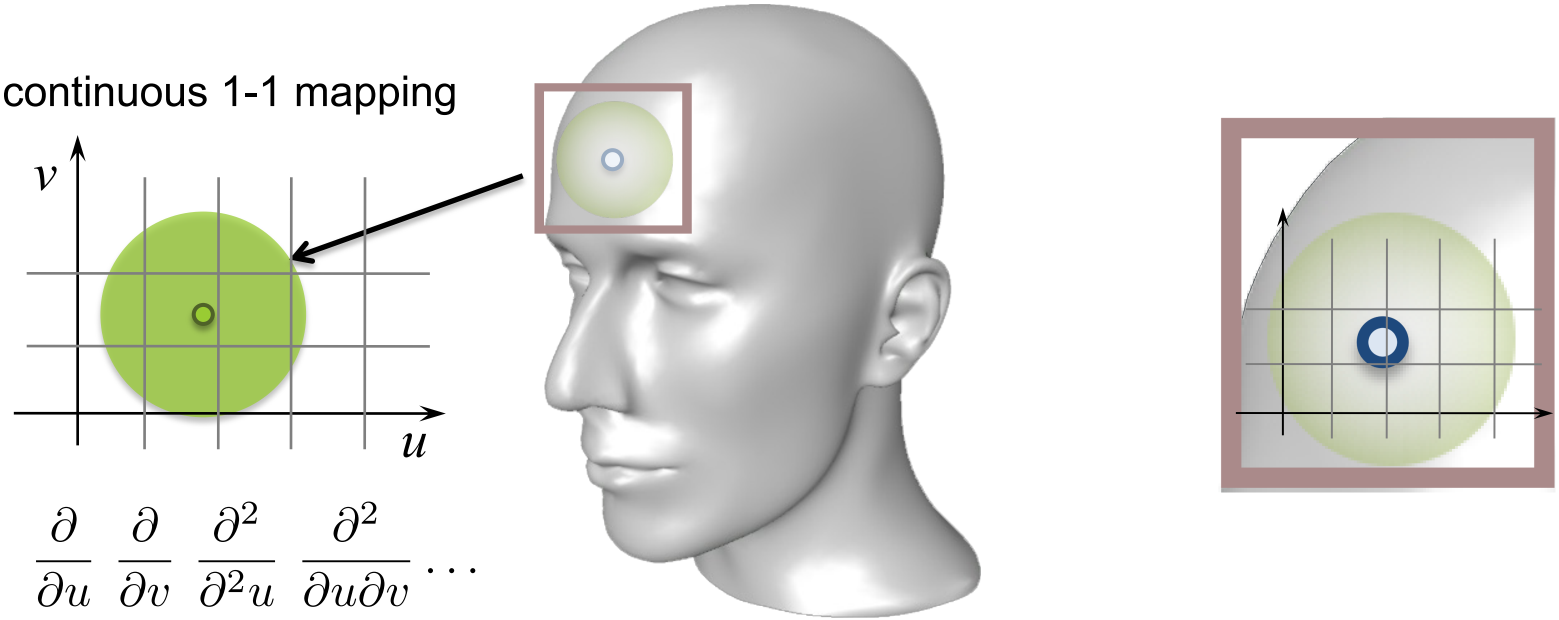
Manifolds
Similarly, for a manifold with boundary, there are some neighborhoods, which we can map to half-disks. These correspond to the patches that intersect the boundary.

- 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.

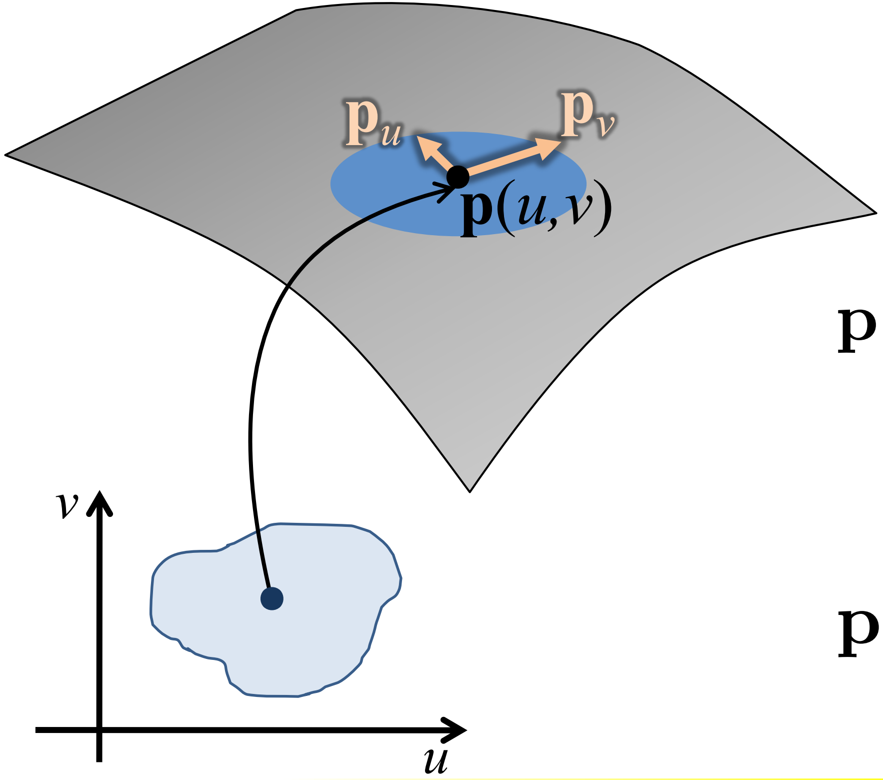
Things that can be discovered by local observation



$$\frac{\partial}{\partial u} \quad \frac{\partial}{\partial v} \quad \frac{\partial^2}{\partial^2 u} \quad \frac{\partial^2}{\partial u \partial v} \quad \cdots$$

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 $\mathbf{p}$.



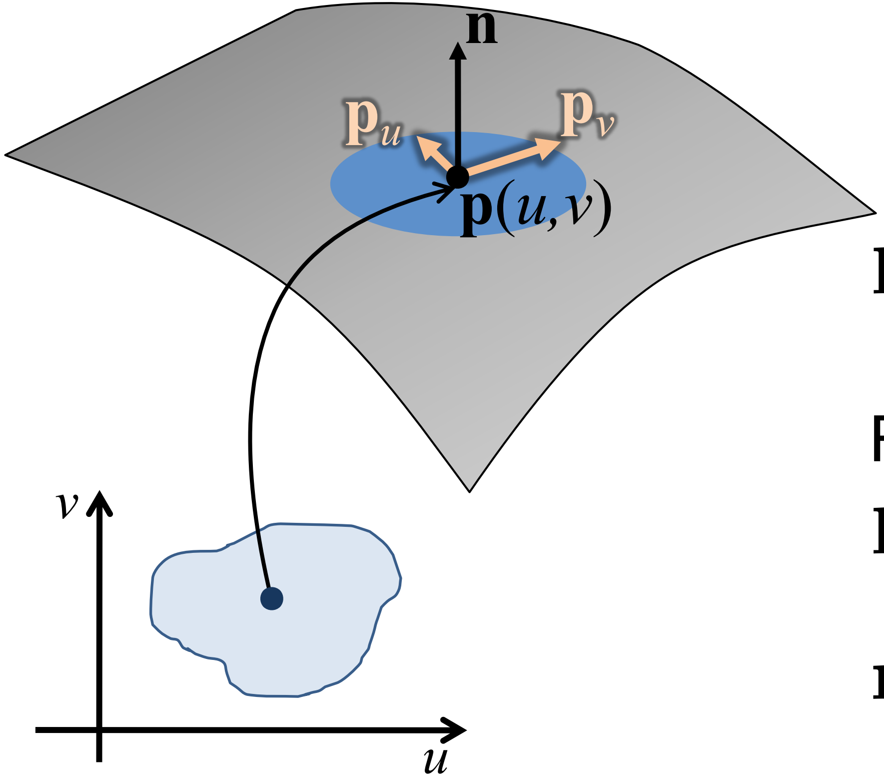
$$\mathbf{p}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in \mathbb{R}^2$$

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

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:

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

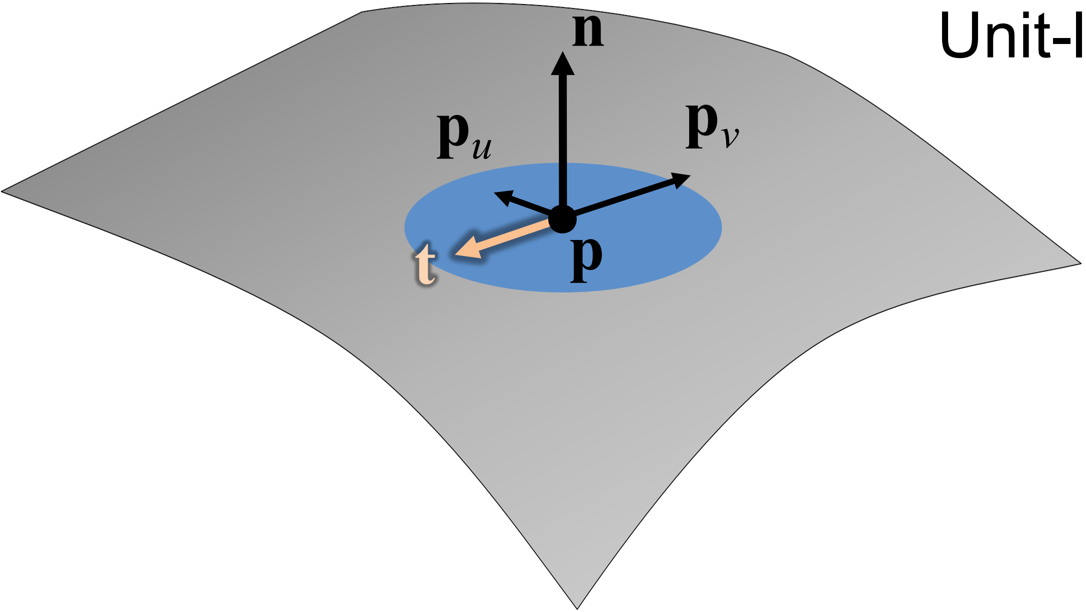
$$\mathbf{n}(u,v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}$$

Normal Curvature

Now we can define what curvature is.

Imagine you have a vector rotating on the plane.

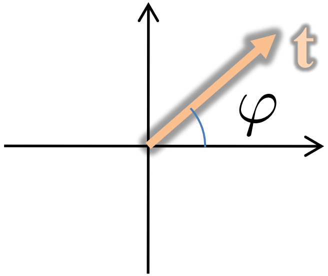
This can be represented with a simple formula if we assume that the two basis vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ are orthogonal.



Unit-length **t** in the tangent plane

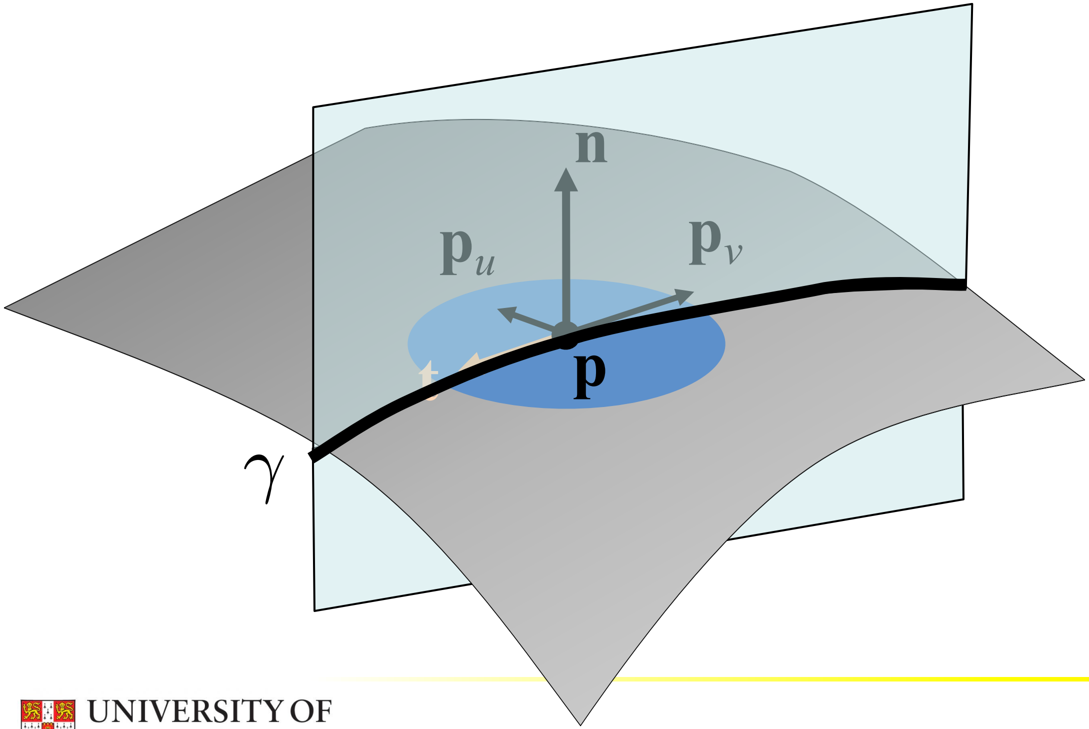
If $\mathbf{p}_u$ and $\mathbf{p}_v$ are orthogonal:

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



Normal Curvature

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



The curve γ is the intersection of the surface with the plane through **n** and **t**.

Normal curvature:

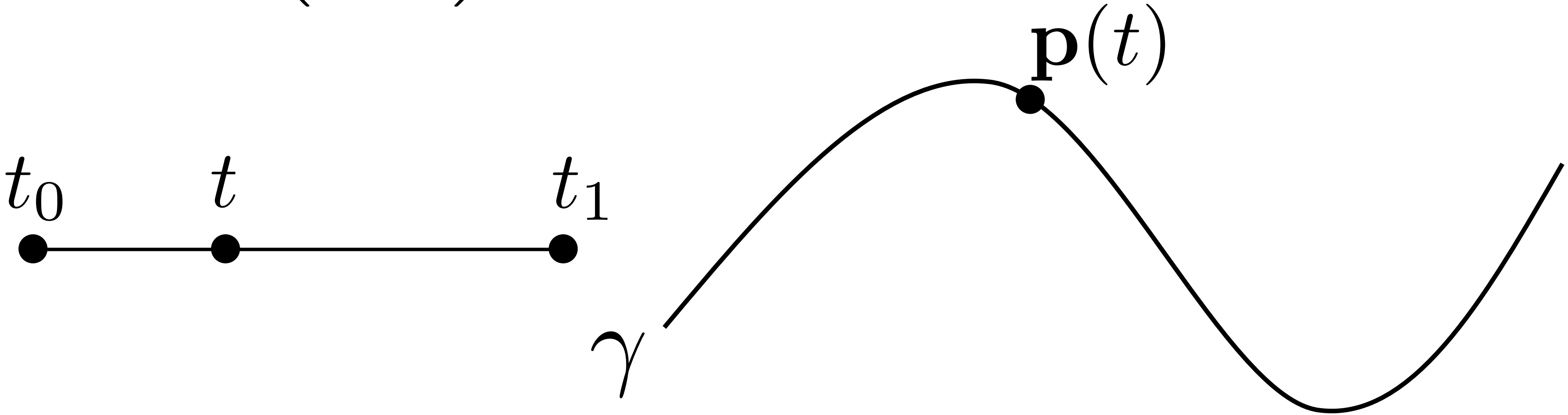
$$\kappa_n(\varphi) = \kappa_n(\gamma(\mathbf{p}))$$

Planar Curves

The definition of normal curvature tells us we can instead work with planar curves for each direction.

A planar curve is defined on a plane i.e. no third dimension. It can be parametrized by a single parameter t.

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \; t \in [t_0, t_1]$$



Planar Curves

A given curve can have many parametrisations.

Imagine walking on the curve. You can adjust your speed as you wish to go from one end to the other end.

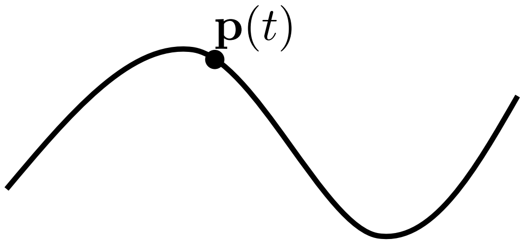
We choose a specific parametrisation, where the absolute value of the derivative of the curve w.r.t. t is a constant.

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \ t \in [t_0, t_1]$$

Same curve has infinitely many parametrisations.

We can assume arc-length parametrization:

$$\left\| \frac{\partial \mathbf{p}(t)}{\partial t} \right\| = 1$$



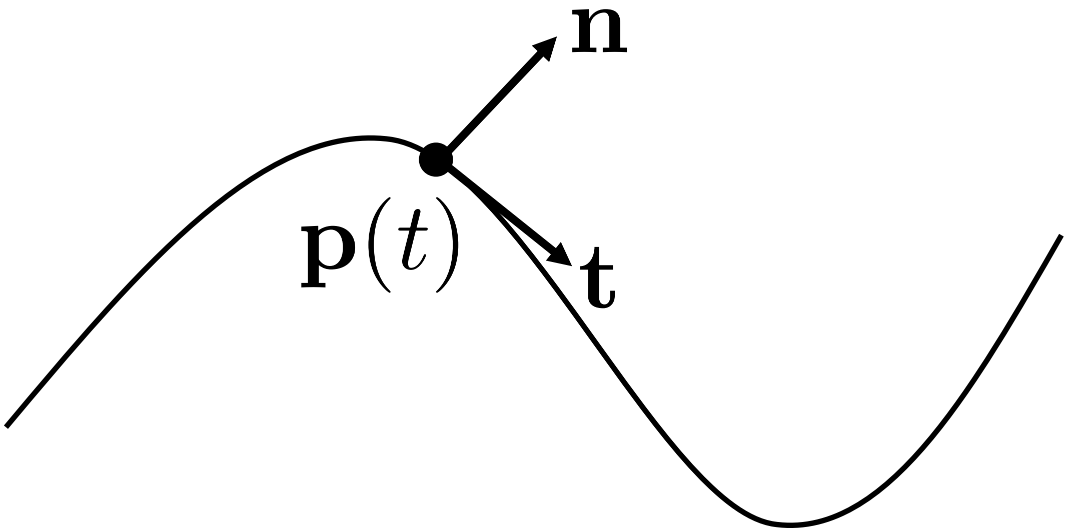
Planar Curves

With such a so-called arc-length parametrisation, we can define the tangent vector, normal vector, and curvature in terms of the first and second derivatives of the curve with simplified forms.

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \ t \in [t_0, t_1]$$

$$\mathbf{t} = \frac{\partial \mathbf{p}(t)}{\partial t}$$ unit length

$$\frac{\partial^2 \mathbf{p}(t)}{\partial t^2} = \kappa(t)\mathbf{n}(t)$$

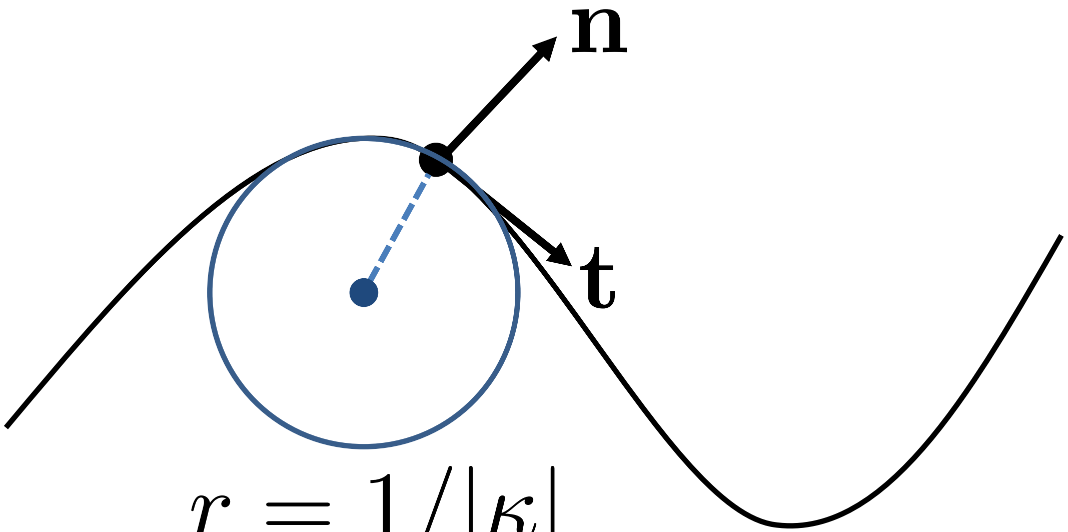


UNIVERSITY OF CAMBRIDGE

In some works, you may see the term osculating or best fitting circle. This is a circle with a radius equal to the inverse of the absolute value of the curvature that is passing through the point $\mathbf{p}$.
It is really the circle that fits best in the local region around $\mathbf{p}$. Curvature gives us a second-order approximation.

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \; t \in [t_0, t_1]$$

Best fitting circle
(osculating circle)

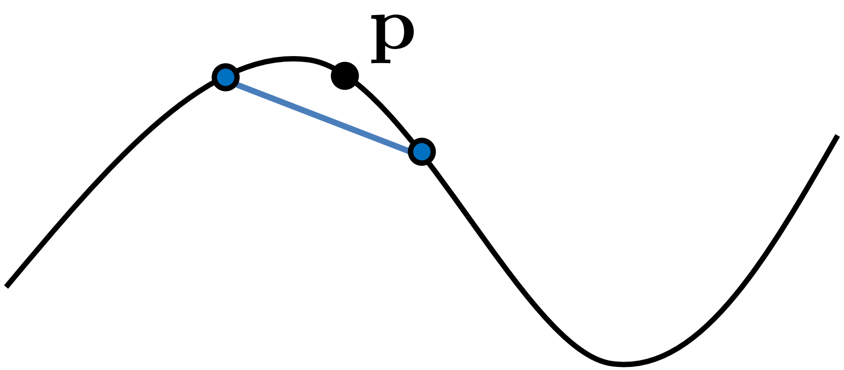


$$r = 1/|\kappa|$$

Planar Curves

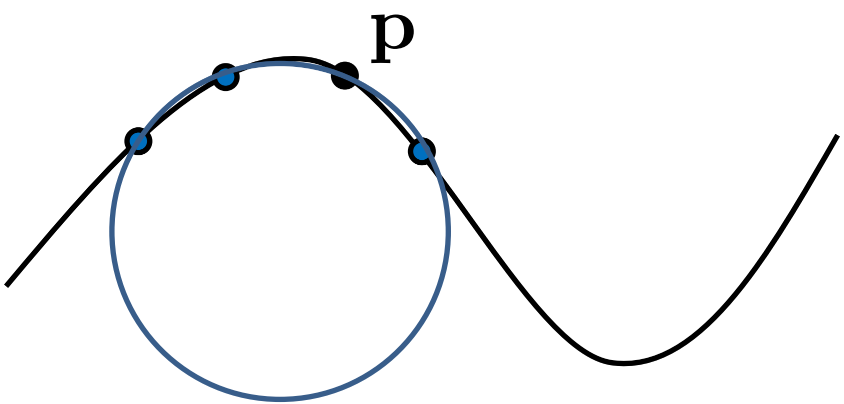
One way to think about differential properties is starting from a set of points that are finitely apart and converging them towards a given point. For the tangent at **p**, imagine drawing a line segment between two points and making those points converge towards **p**. The resulting line segment's direction gives you the tangent vector.

Tangent and osculating circle as local limits



The same applies to osculating circles. In this case, we start from three points to define a circle. We then converge those points towards $\mathbf{p}$. The final circle is the best fitting circle at $\mathbf{p}$.

Tangent and osculating circle as local limits

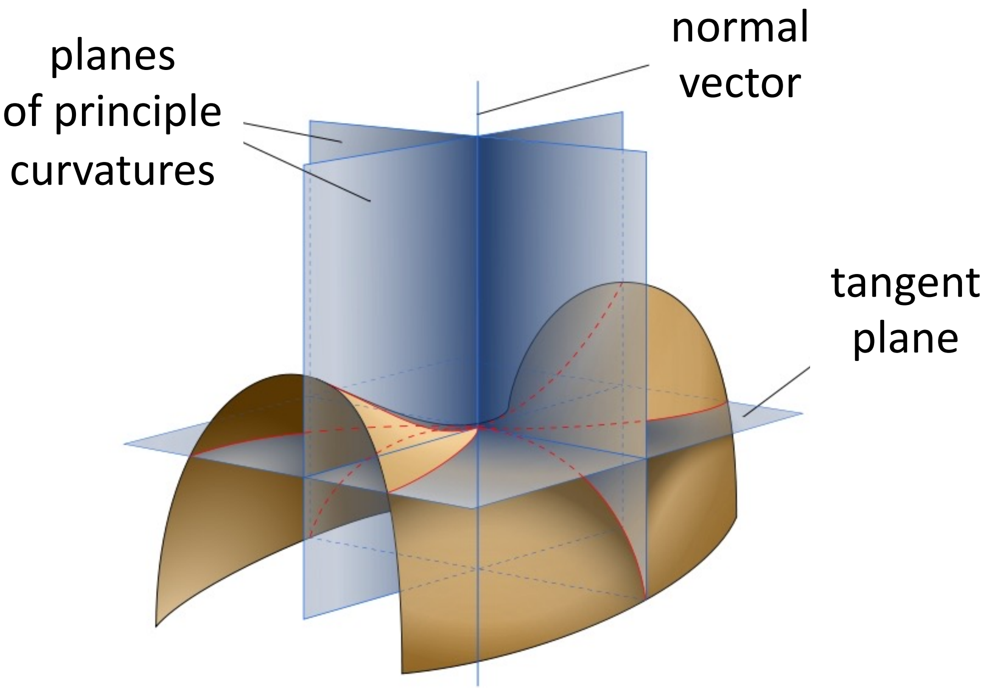


Now we know how to compute the normal curvature for each direction.
Other curvature-related definitions are derived from the normal curvature.

- 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.

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

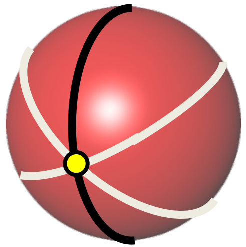
$\varphi =$ angle with $\mathbf{t}_1$

Local Shape by Curvatures

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

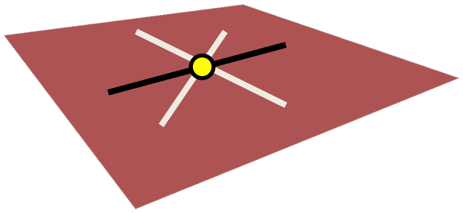
Isotropic:
all directions are principal directions

spherical (umbilical)



$$K > 0, \kappa_1 = \kappa_2$$

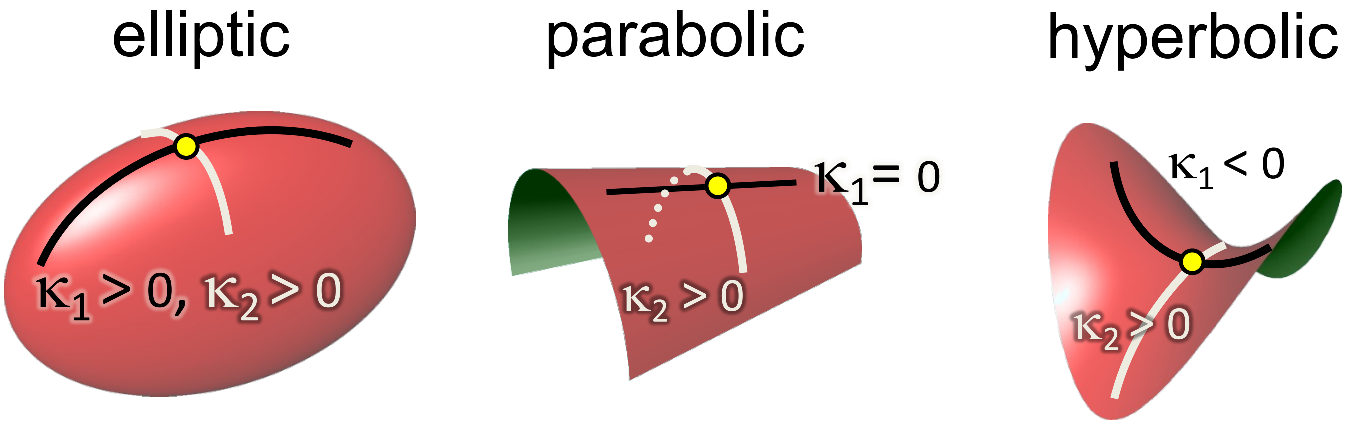
planar



$$K = 0$$

Local Shape by Curvatures

Anisotropic:
2 distinct principal directions



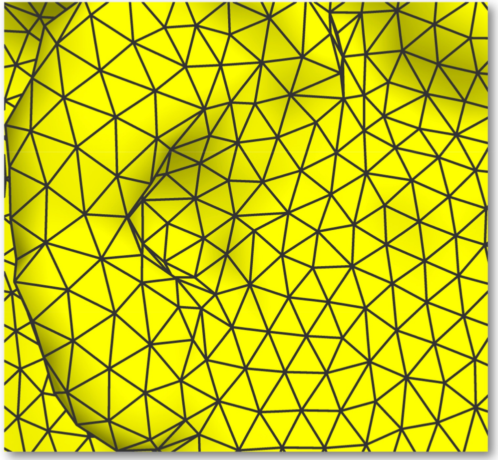
$K > 0$ $K = 0$ $K < 0$

Discrete Differential Geometry

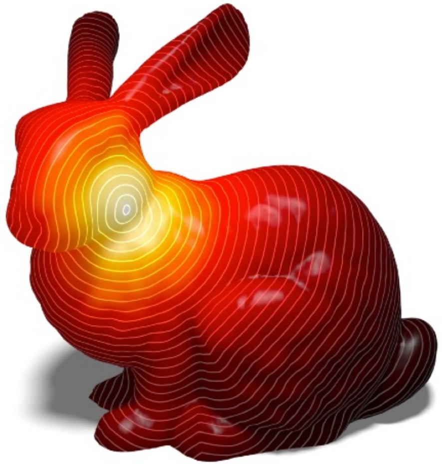
So far, we have talked about surface normal and curvatures for smooth manifold surfaces.
In computer graphics, we often have discrete surfaces, e.g. triangle meshes.
How do we define these differential quantities then? There are local and global methods.

- Approximate surface normal and curvature via

Local surface approximation



Global: discrete Laplace-Beltrami



Laplace-Beltrami Operator

Let's start with the global method based on the Laplace-Beltrami operator. This operator is defined on a manifold surface. The key is approximating this operator when applied to the coordinate function $\mathbf{p}$ i.e. the location of each point on a discrete surface.

- Apply to coordinate function

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

Laplace-Beltrami

mean curvature

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

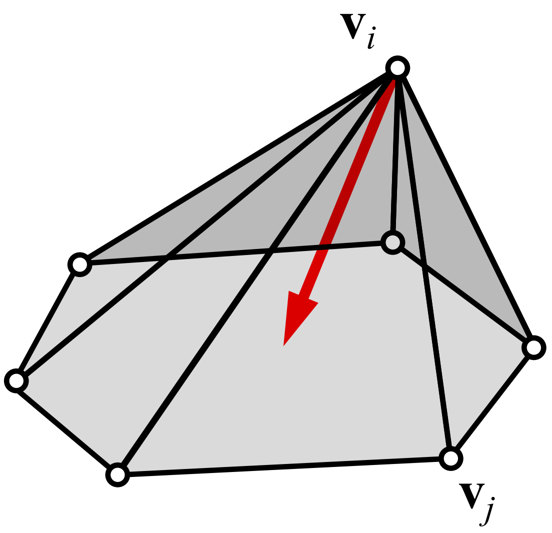
function on surface *M*

unit surface normal

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 it from the location of the vertex. This gives us an approximation of -2H**n**, where H is the mean curvature and **n** is the surface normal.

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

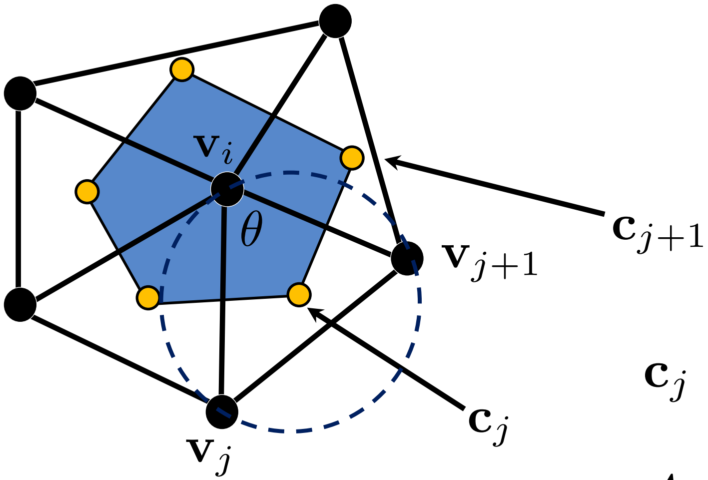


$$L_u(\mathbf{v}_i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\mathbf{v}_j - \mathbf{v}_i)$$

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

Discrete Curvatures

We need two more definitions before we can get to curvatures on discrete surfaces, the area A_i and the angle θ.
The A_i is the sum of each of the areas defined for each neighboring vertex.
Below, $\triangle$ defines a triangle.



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

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

Discrete Curvatures

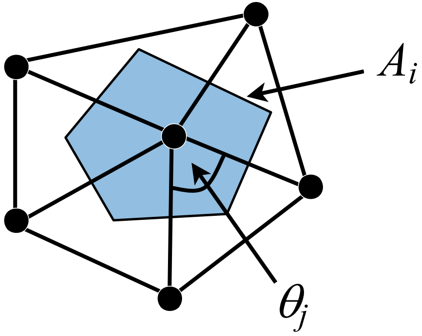
With these, we can now compute approximations of curvatures on discrete surfaces.
Mean curvature is estimated by the norm of the discrete Laplace-Beltrami operator applied to the coord. function.
Gaussian curvature is estimated (stated without proof) with a formula involving the area and angle we defined.
Principle curvatures can then be obtained from the mean and Gaussian curvatures.

- Mean curvature (sign according to normal)

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

- Gaussian curvature

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

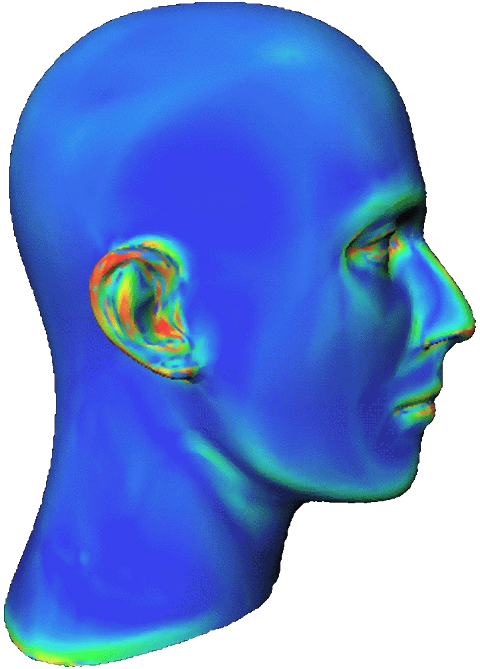


- Principal curvatures

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

Discrete Curvatures

Mean Curvature



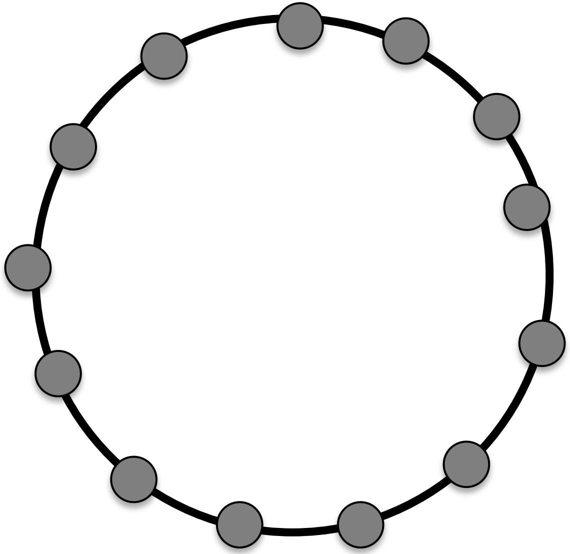
Discrete Laplace-Beltrami

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.

- Extension to graphs and point clouds



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

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

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

Discrete Laplace-Beltrami

This approximation allows us to capture complex domains. In fact, any domain that can be point-sampled.

- Extension to graphs and point clouds