252-0538-00L, Spring 2025

Shape Modeling and Geometry Processing

Normal Estimation in Point Clouds

INTERACTIVE GEOMETRY LAB

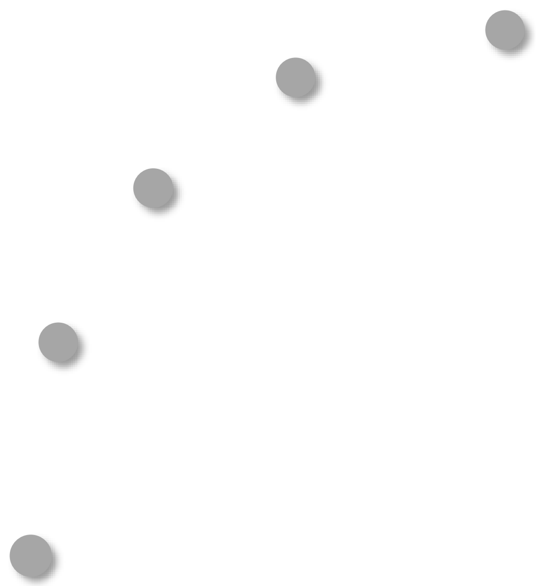
March 12, 2025

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Normal Estimation

Goal:

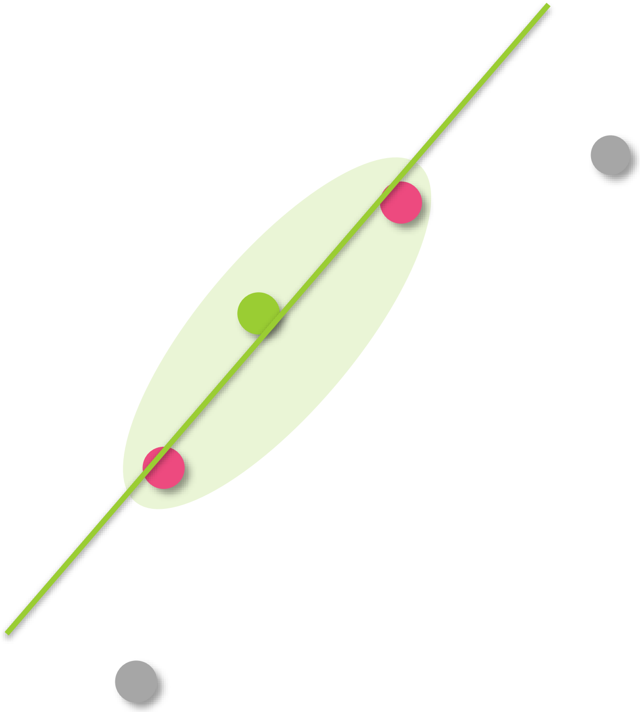
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$



Normal Estimation

Goal:

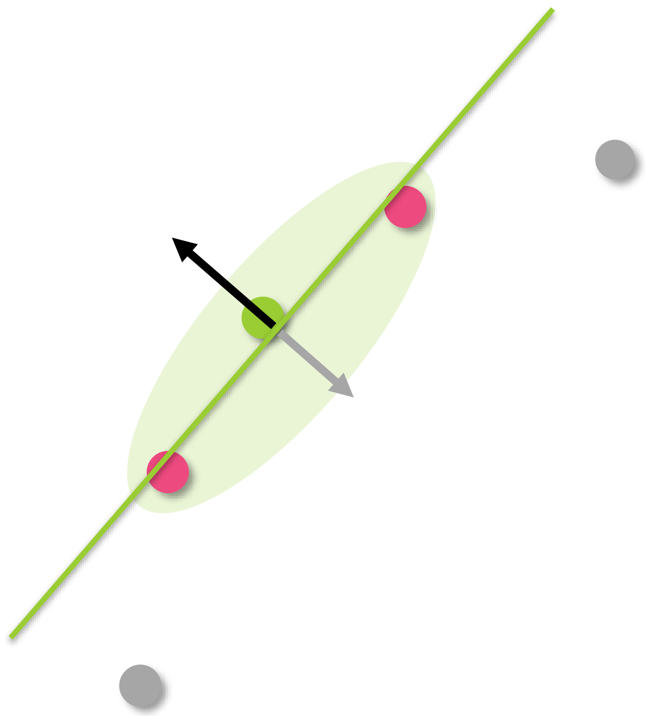
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

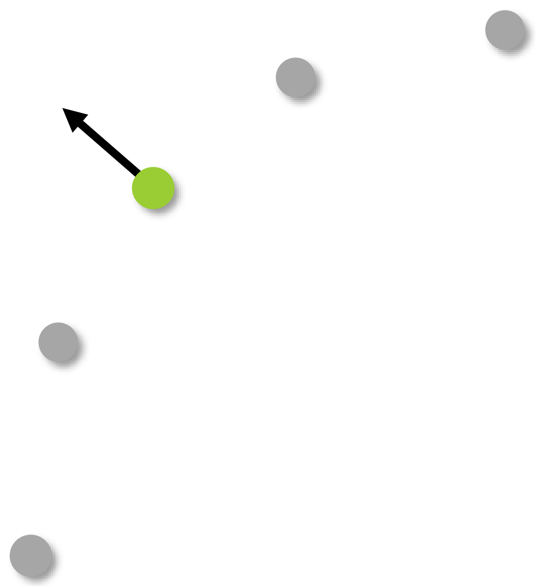
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

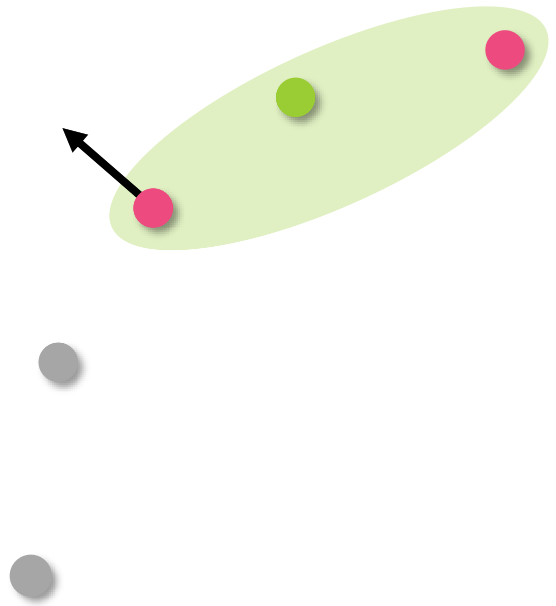
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

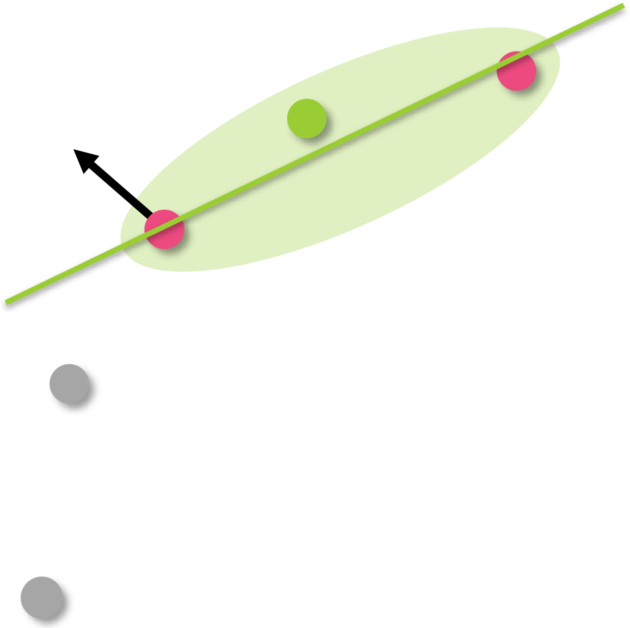
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

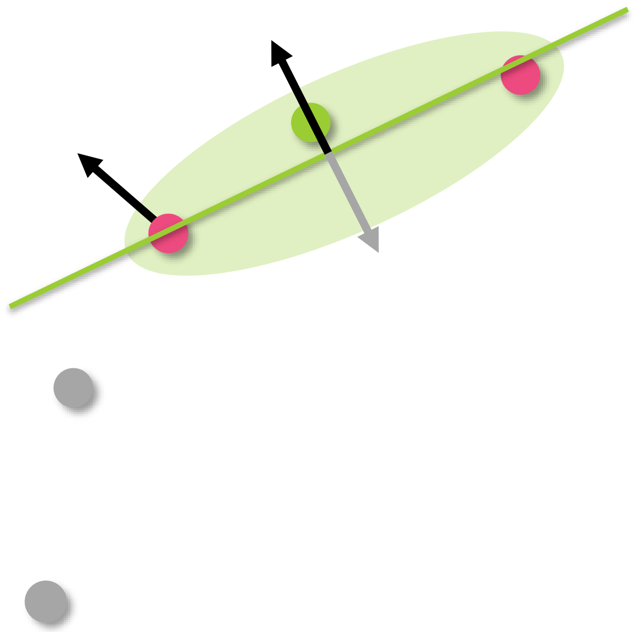
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

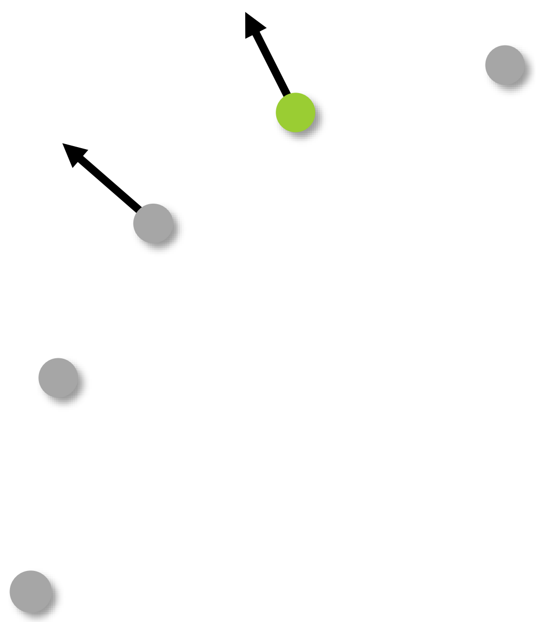
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

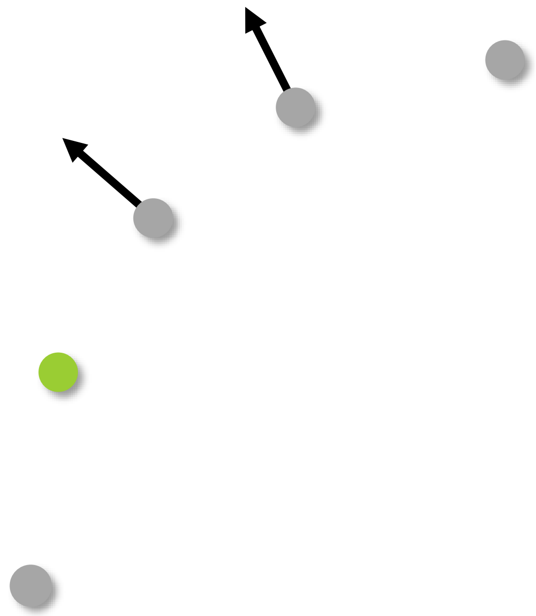
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

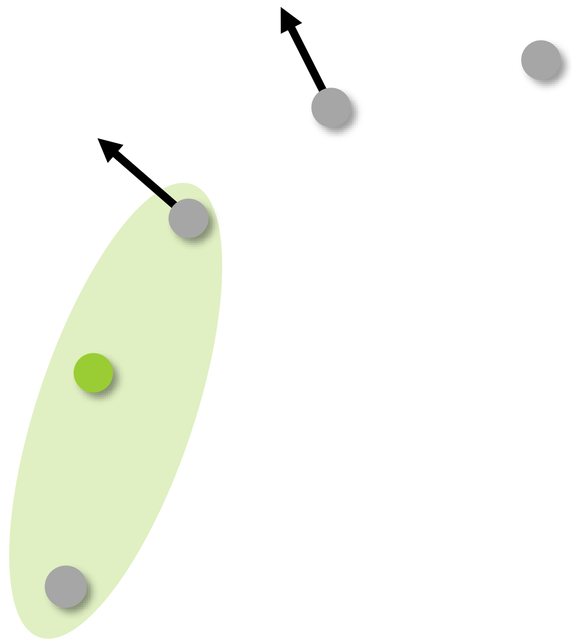
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

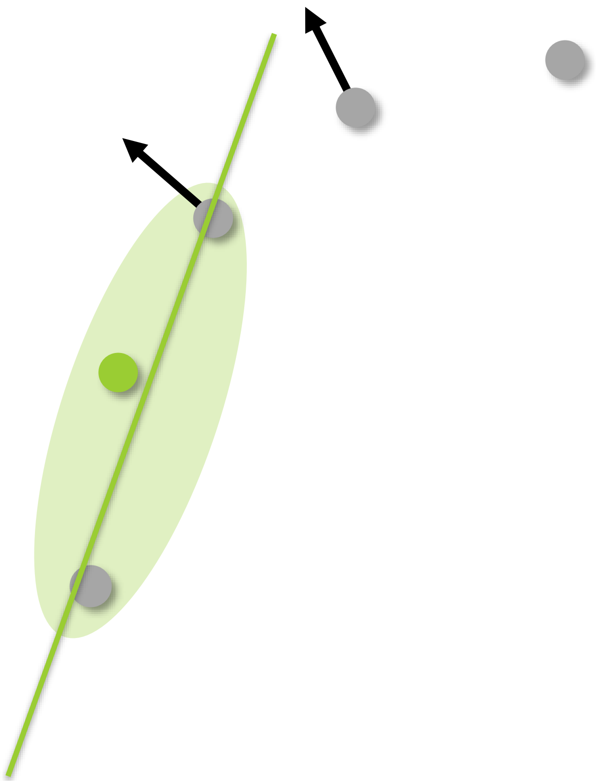
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

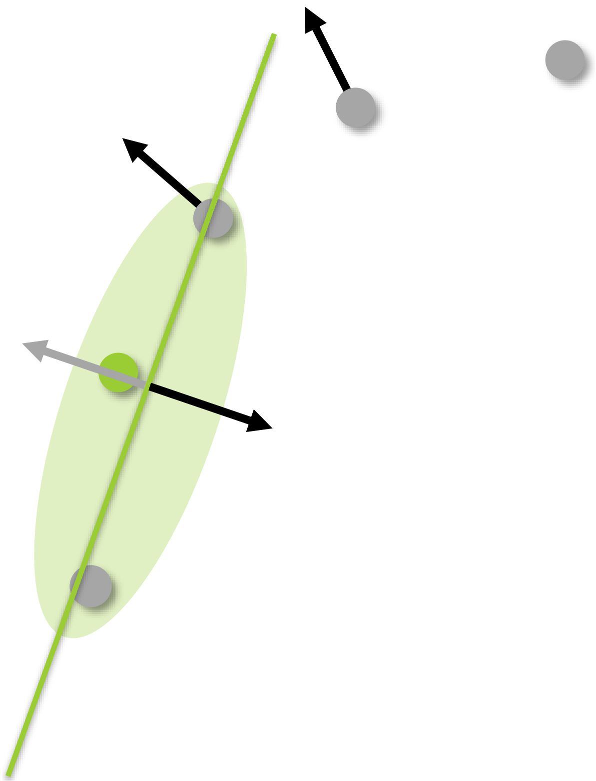
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

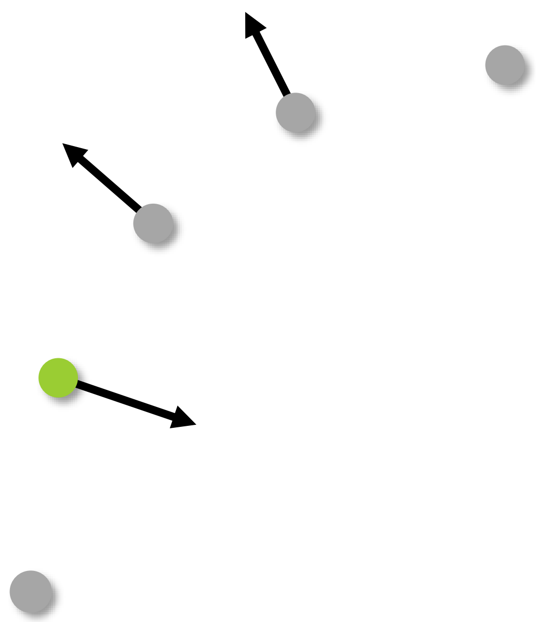
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

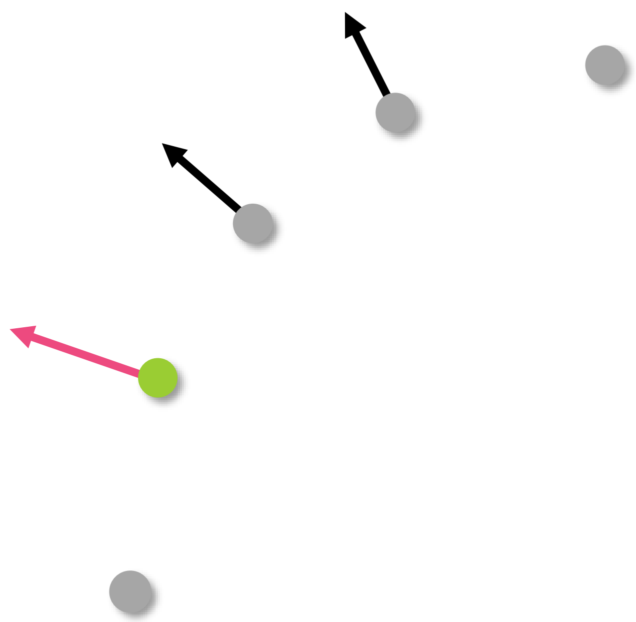
- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane



Normal Estimation

Goal:

- Assign a normal vector $\mathbf{n}$ at each point cloud point $\mathbf{x}$
  - Estimate the direction by fitting a local plane
  - Find consistent global orientation by propagation (spanning tree)

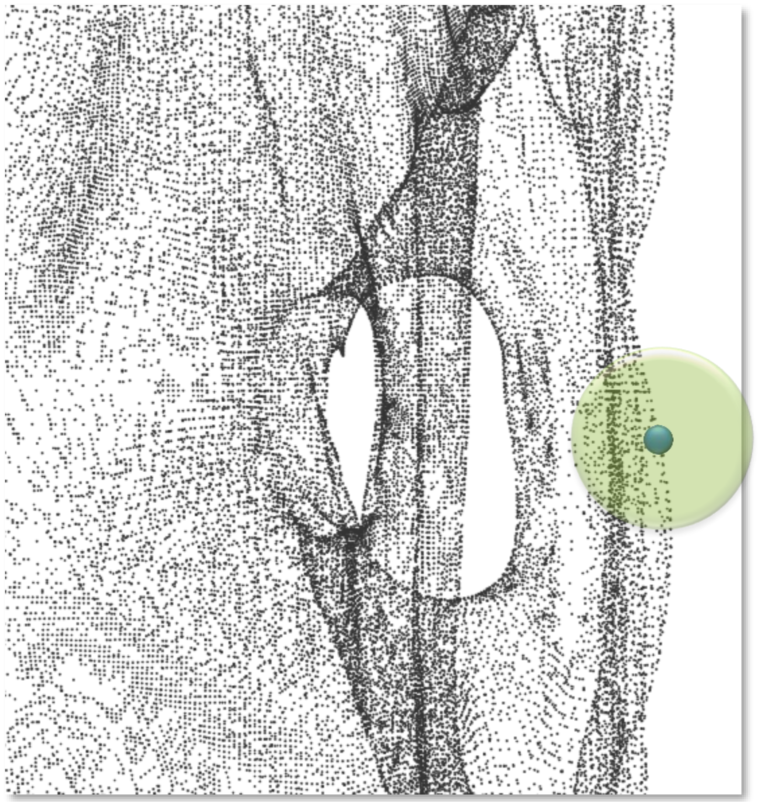


Local Plane Fitting

- For each point $\mathbf{x}$ in the cloud, pick n nearest neighbors, or all points in r-ball:

$$\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$$

$$\Rightarrow \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$



Local Plane Fitting

- For each point $\mathbf{x}$ in the cloud, pick n nearest neighbors, or all points in r-ball:

$$\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$$

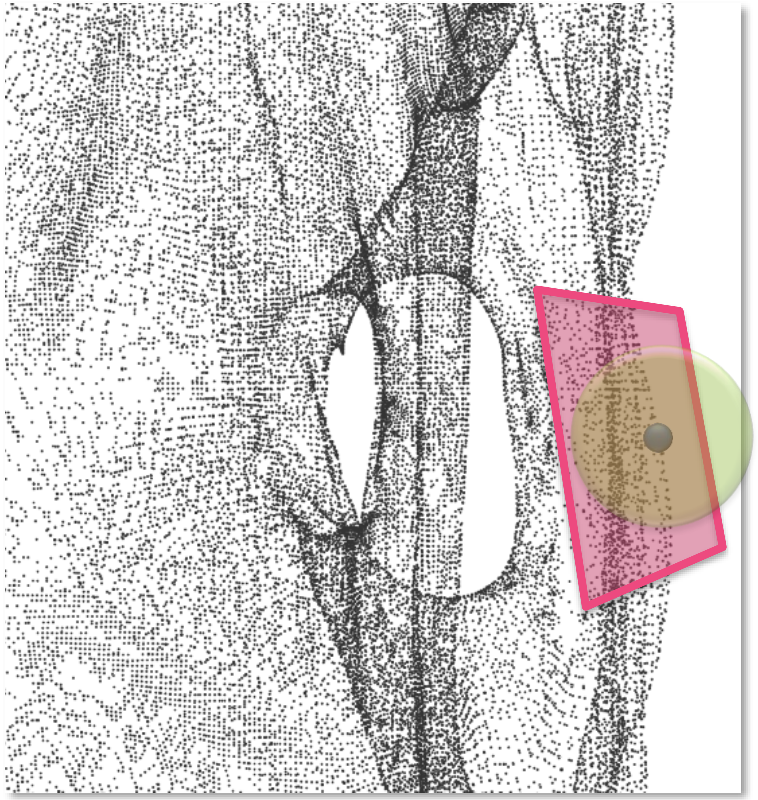
$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$$

- Find a **plane Π** that minimizes the sum of squared distances:

$$\min \sum_{i=1}^{n} \mathrm{dist}(\mathbf{x}_i, \Pi)^2$$

best fitting plane

But how we do this?



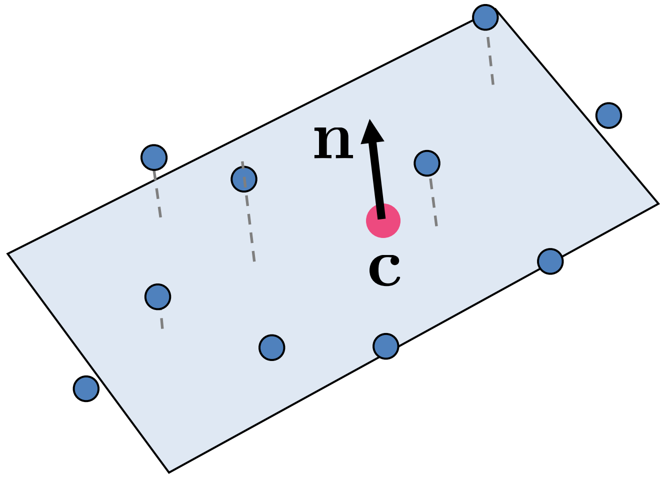
Notations

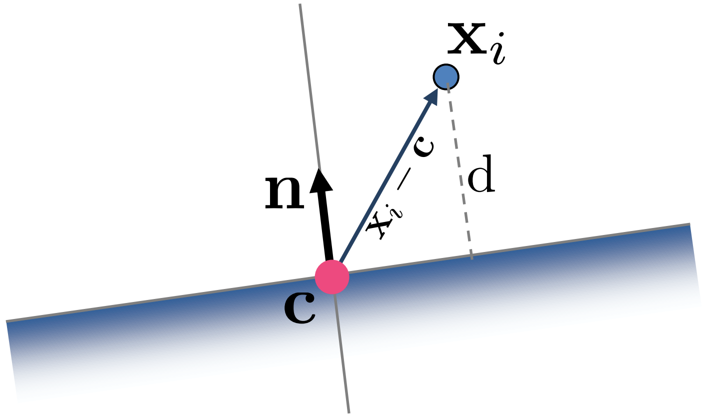
- Input points: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

- Looking for a (hyper) **plane** passing through $\mathbf{c}$ with normal $\mathbf{n}$ s.t.

$$\min_{\mathbf{c},\mathbf{n},\|\mathbf{n}\|=1} \sum_{i=1}^{n} \left((\mathbf{x}_i - \mathbf{c})^\top \mathbf{n} \right)^2$$

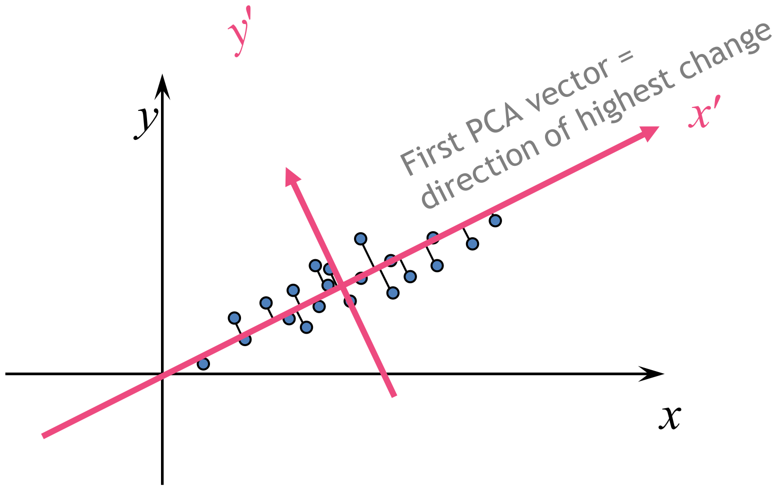
sum of squared distances from plane

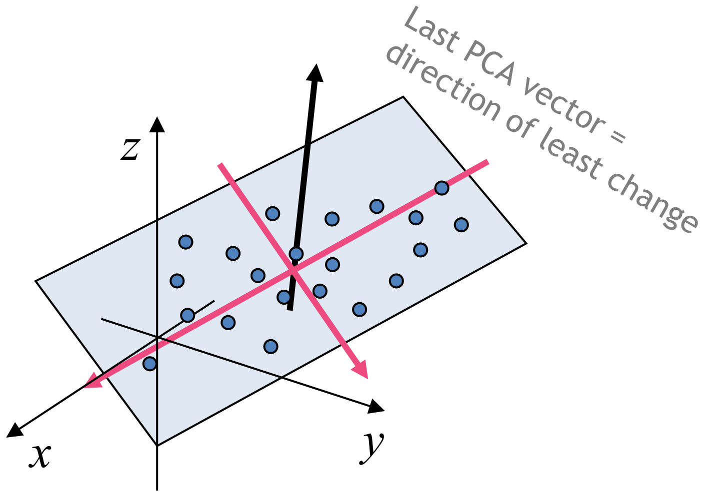




Principal Component Analysis (PCA)

- PCA finds an orthogonal basis that best represents a given data set





- PCA finds the best approximating line/plane/linear subspace in terms of

$$\sum distances^2$$

Best-fit PCA plane - basic recipe

- Input: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (column vectors)
- Compute centroid = plane origin $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$
- Compute $d \times d$ scatter matrix S

$$\mathbf{Y} = (\mathbf{y}_1\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n)$$

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{c}$$

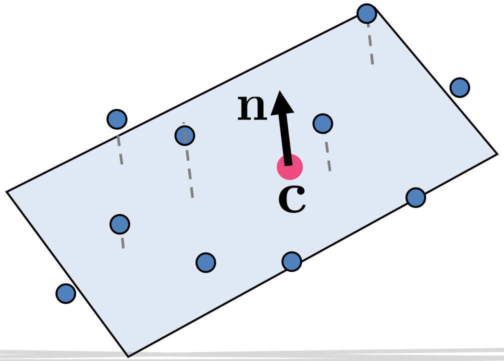
$$\mathbf{S} = \mathbf{Y}\mathbf{Y}^\mathsf{T}$$

$[d \times d] = [d \times n][n \times d]$ matrix

Encodes all change correlations! 💡

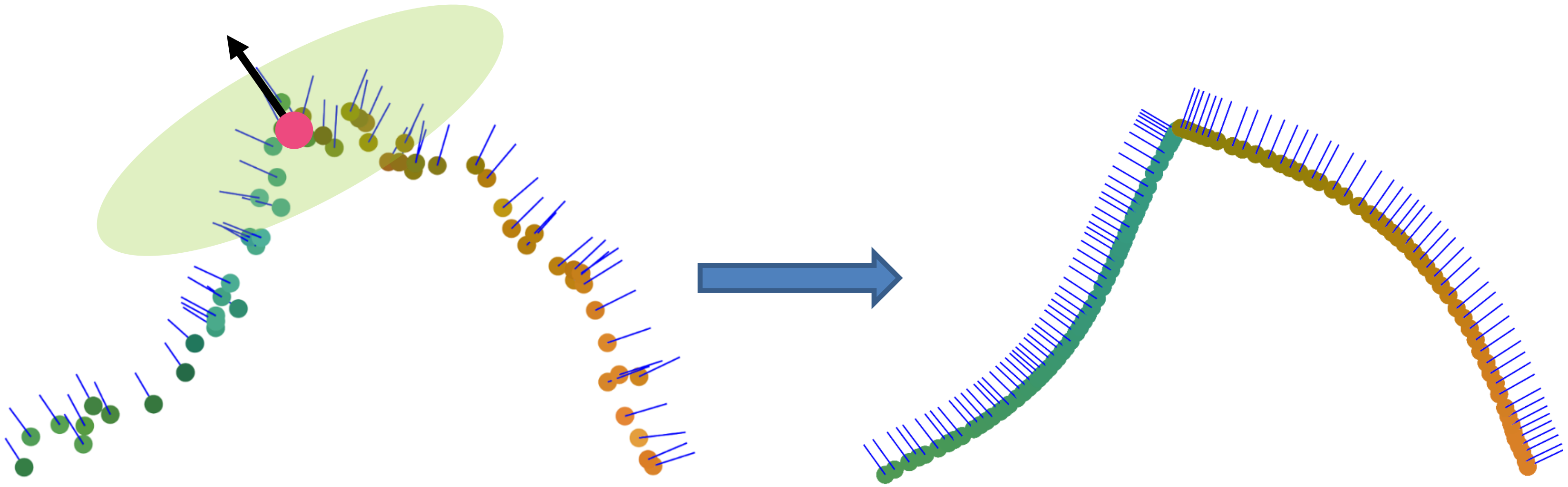
- Plane normal $\mathbf{n}$ is the eigenvector of $\mathbf{S}$ with the smallest eigenvalue

$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^\mathsf{T}$$



Direction of smallest change! 😎

Normal Smoothing



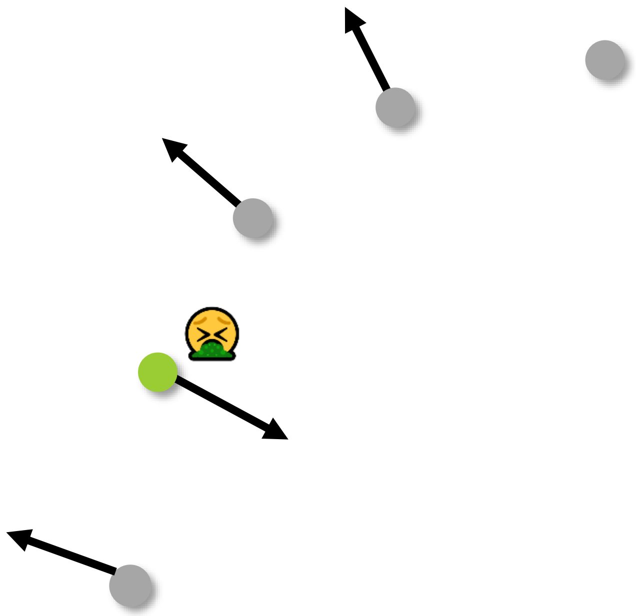
Bilateral (anisotropic) normal smoothing is often used.

https://doc.cgal.org/latest/Point_set_processing_3/index.html

- ⚠ Beware of noise – weighted PCA for better robustness.
- ⚠ Beware of sharp edges, e.g., intersection of two planes.

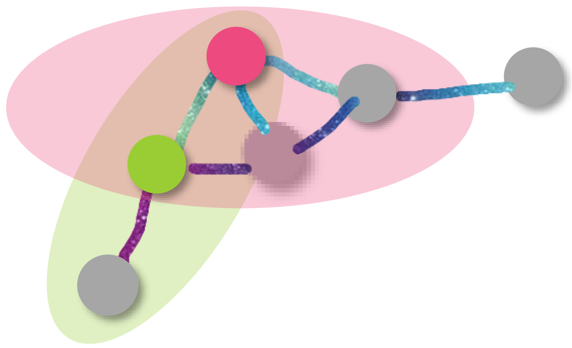
Normal Orientation

- PCA may return arbitrarily oriented eigenvectors
- Wish to orient consistently
  - Neighboring points should have similar normals



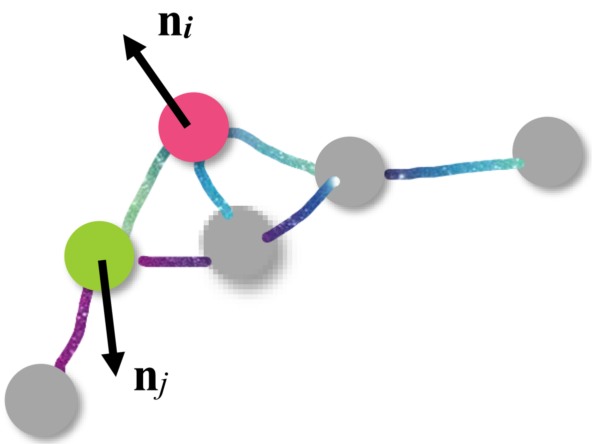
Normal Orientation

- Build graph connecting neighboring points
  - Edge (i,j) exists if $\mathbf{x}_i \in \mathrm{kNN}(\mathbf{x}_j)$ or $\mathbf{x}_j \in \mathrm{kNN}(\mathbf{x}_i)$



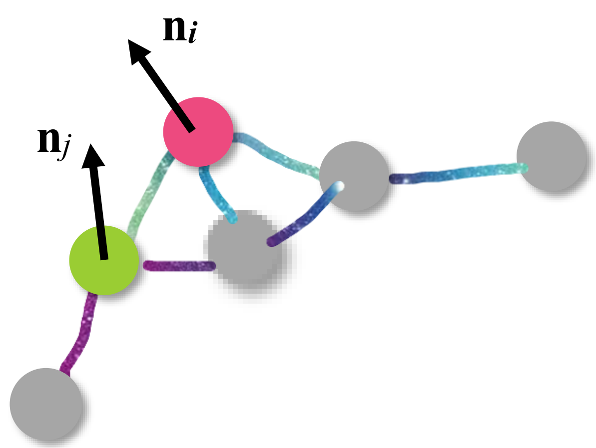
Normal Orientation

- Build graph connecting neighboring points
- Propagate normal orientation through graph connectivity
  - For neighbors $\mathbf{x}_i$, $\mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^\top \mathbf{n}_j < 0$



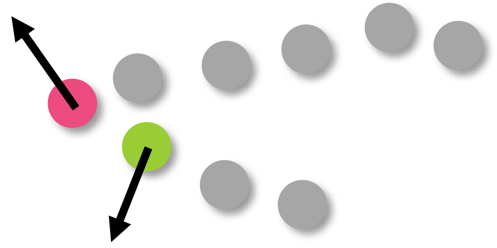
Normal Orientation

- Build graph connecting neighboring points
- Propagate normal orientation through graph connectivity
  - For neighbors $\mathbf{x}_i$, $\mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^\top \mathbf{n}_j < 0$



Normal Orientation

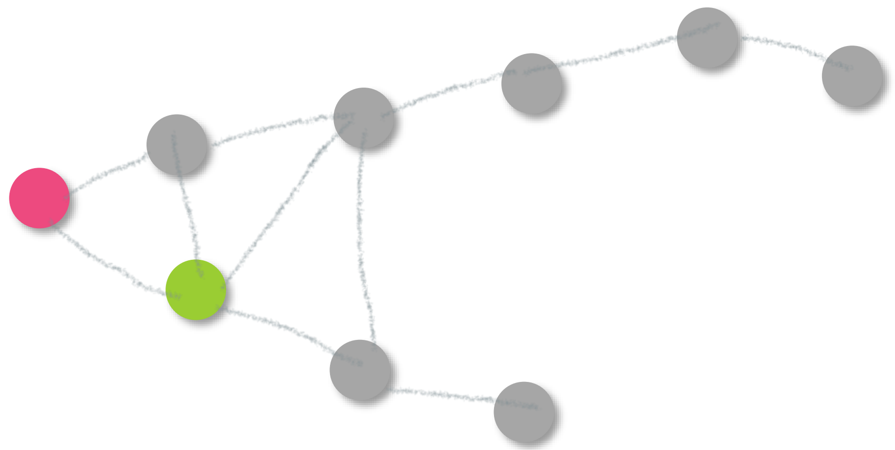
- Build graph connecting neighboring points
- Propagate normal orientation through graph connectivity
  - For neighbors $\mathbf{x}_i, \mathbf{x}_j$: Flip $\mathbf{n}_j$ if $\mathbf{n}_i^\top \mathbf{n}_j < 0$
  - Fails at sharp edges/corners 😭



Normal Orientation

- Build graph connecting neighboring points
- ~~Propagate normal orientation through graph connectivity~~
- Propagate along “safe” paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

$$w_{ij} = 1 - |\mathbf{n}_i^\top \mathbf{n}_j|$$

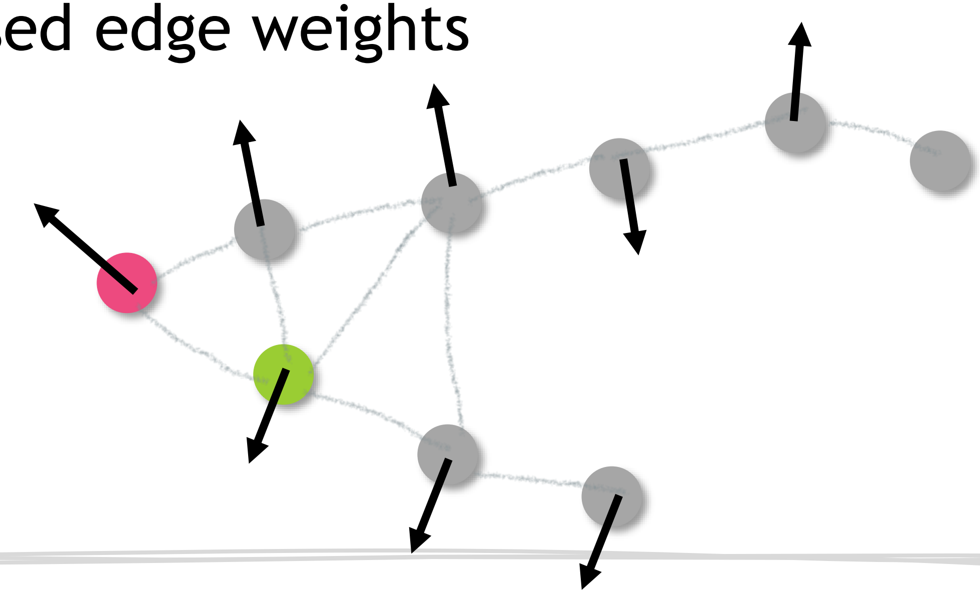


http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

Normal Orientation

- Build graph connecting neighboring points
- ~~Propagate normal orientation through graph connectivity~~
- Propagate along "safe" paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

$$w_{ij} = 1 - |\mathbf{n}_i^\top \mathbf{n}_j|$$

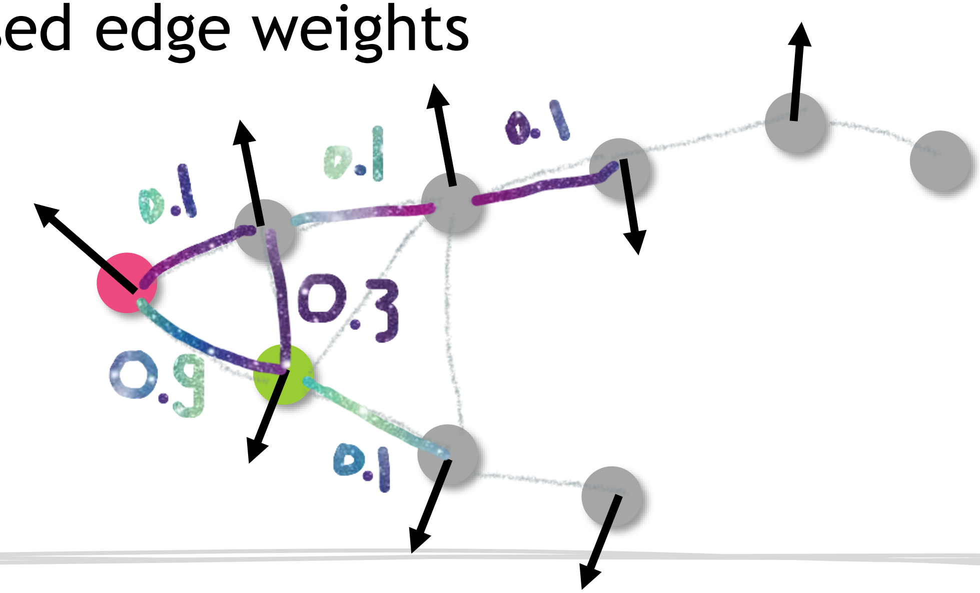


http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

Normal Orientation

- Build graph connecting neighboring points
- ~~Propagate normal orientation through graph connectivity~~
- Propagate along "safe" paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

$$w_{ij} = 1 - |\mathbf{n}_i^\top \mathbf{n}_j|$$

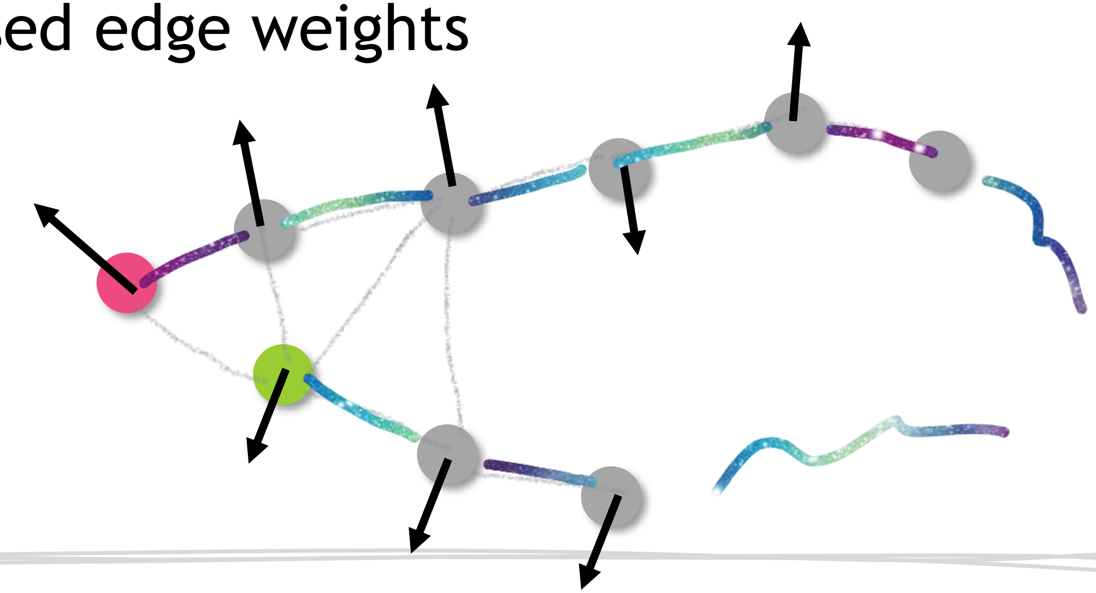


http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

Normal Orientation

- Build graph connecting neighboring points
- ~~Propagate normal orientation through graph connectivity~~
- Propagate along “safe” paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

$$w_{ij} = 1 - |\mathbf{n}_i^{\mathsf{T}} \mathbf{n}_j|$$



This algorihtm is way better, but still not „perfect“ at all sharp corners.

http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf

Shape Modeling and Geometry Processing

(Discrete) Differential Geometry

Planar Curves

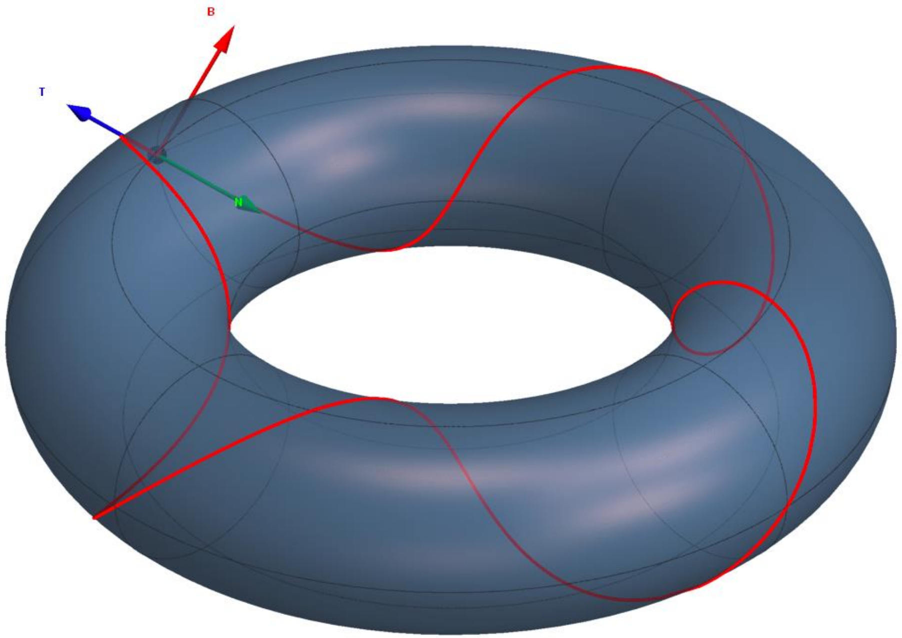
INTERACTIVE GEOMETRY LAB

March 12, 2025

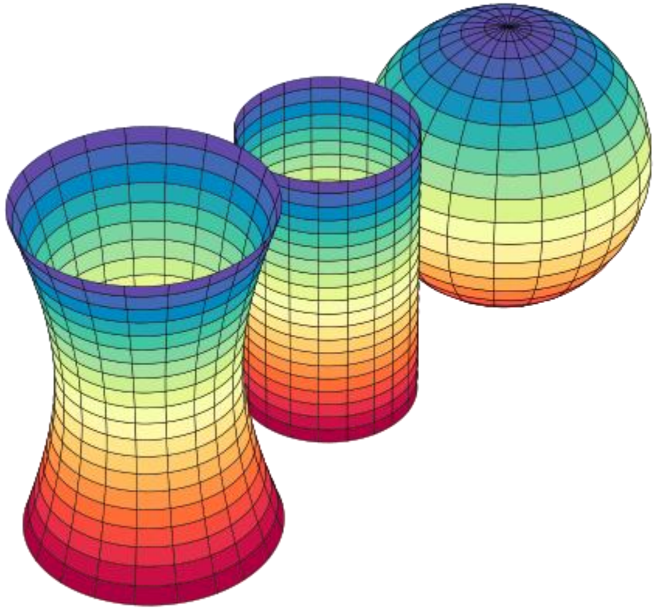
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Differential Geometry

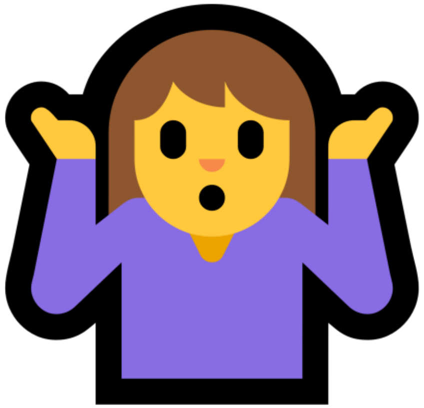
- Language to analyze:



Curves



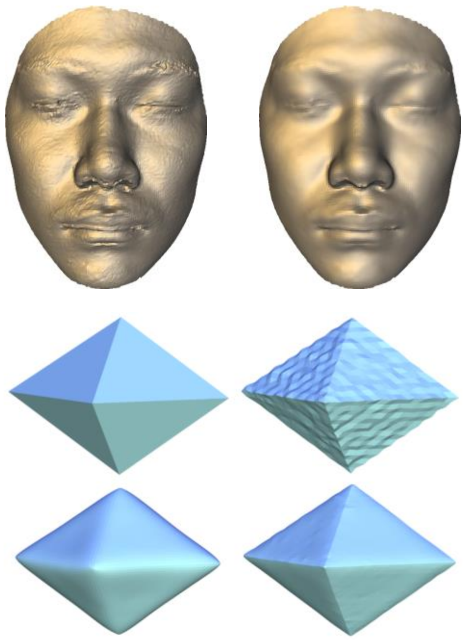
Surfaces



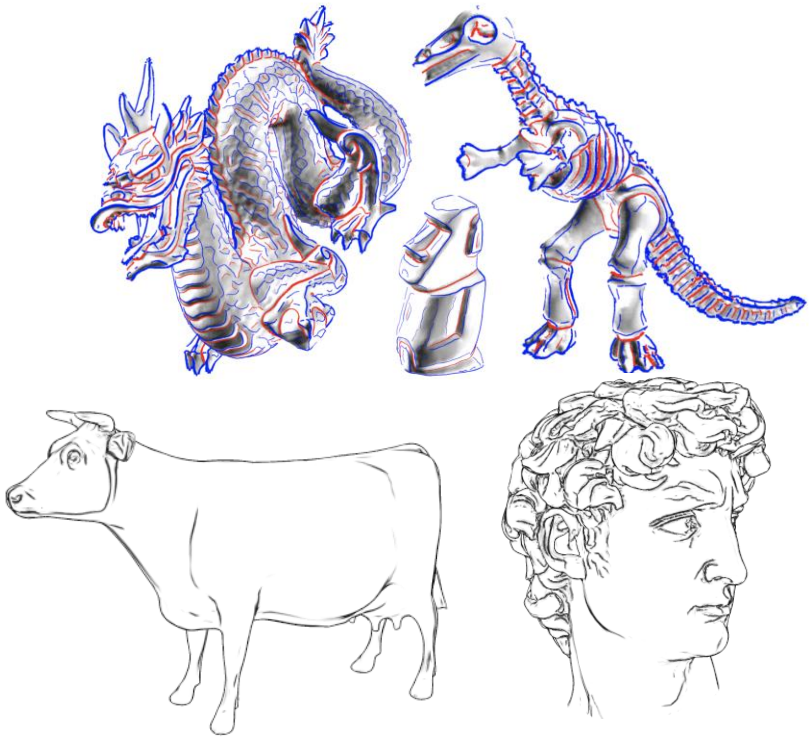
K-dimensional manifolds
in N-dimensional space

Differential Geometry – Motivation

- Describe and analyze geometric characteristics of shapes



Smoothness



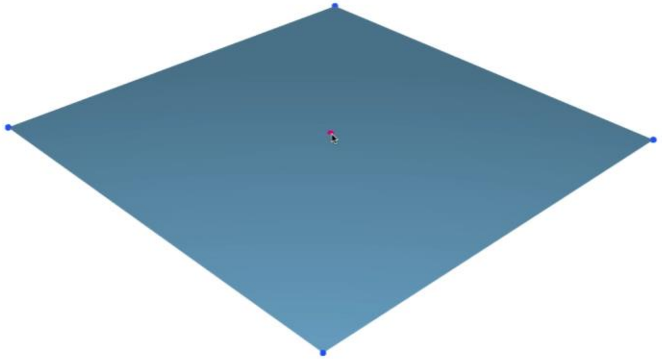
Segmentation

feature detection

Differential Geometry – Motivation

- Describe and analyze geometric characteristics of shapes

Parameterizations

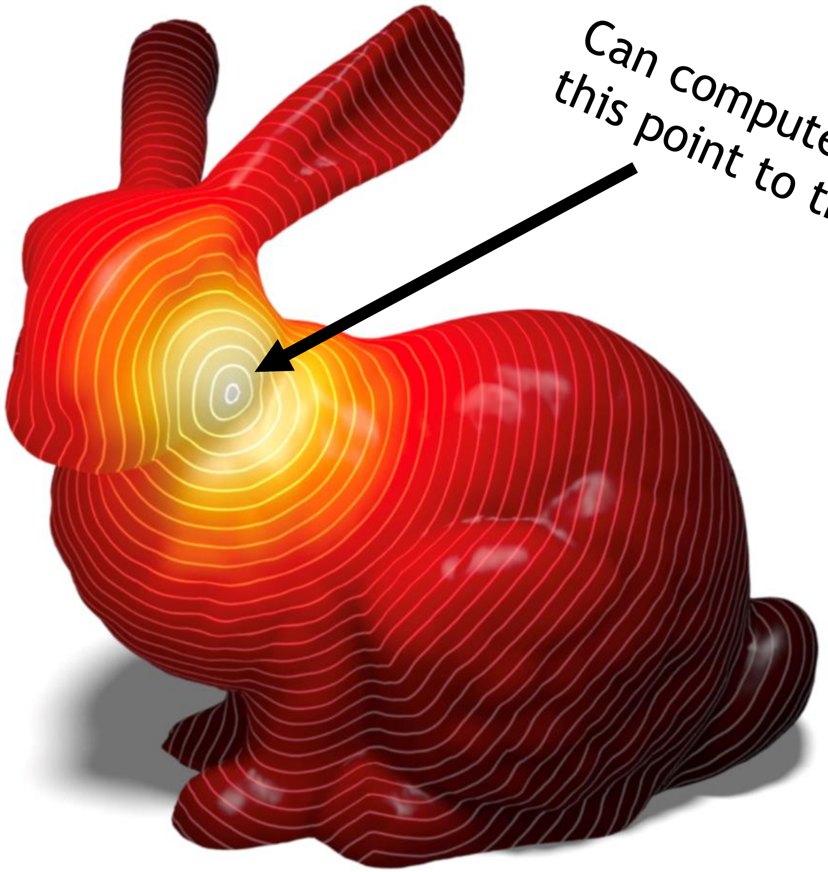


Deformations

Geometric properties – distance

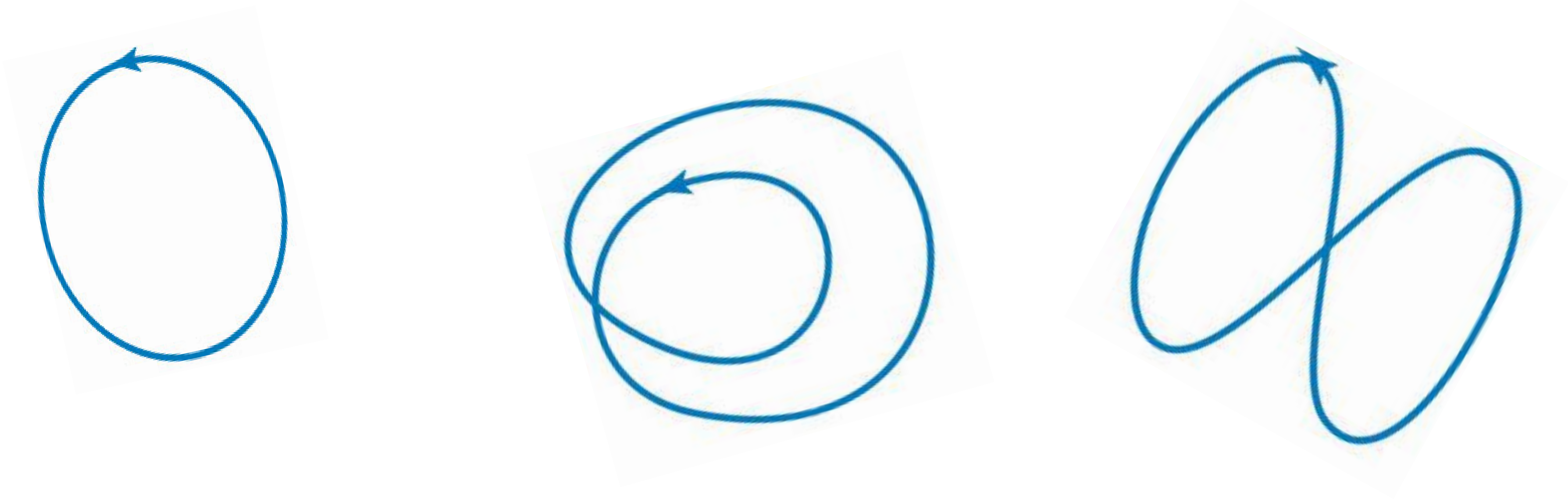
Poke a bunny with a hot needle

See how the heat distributes after a millisecond.



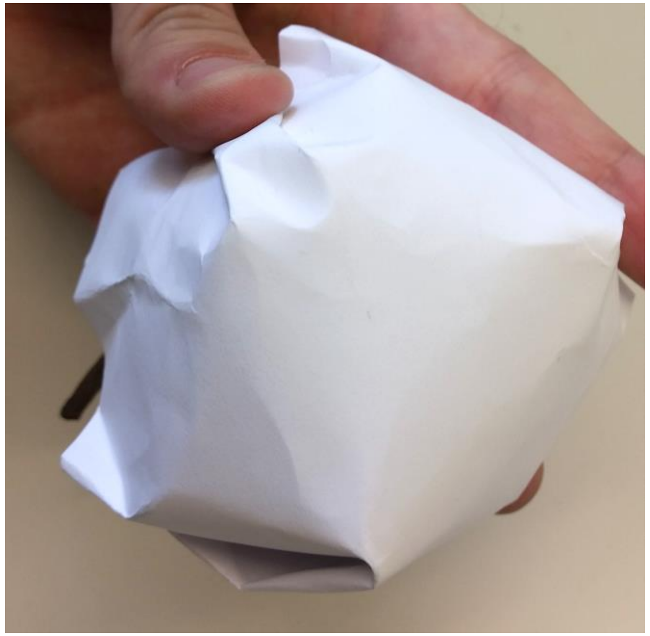
Geodesic in Heat
[Crane et al. 2013]

Geometric properties – curvature



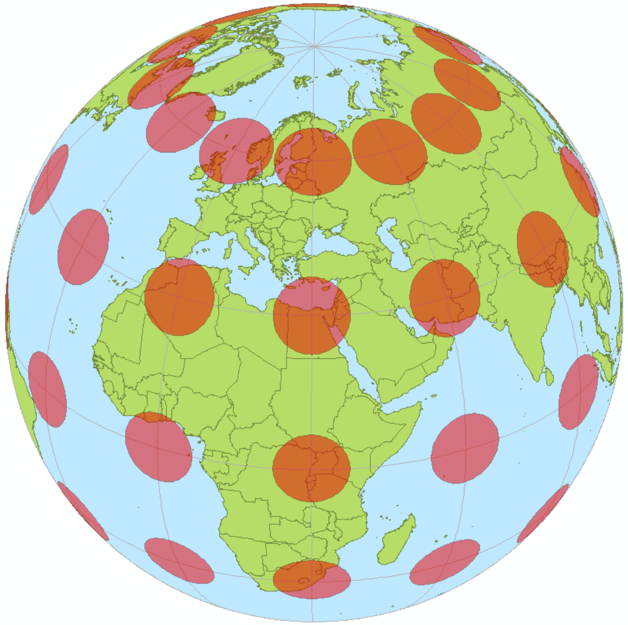
The bending of curves

Surface curvature

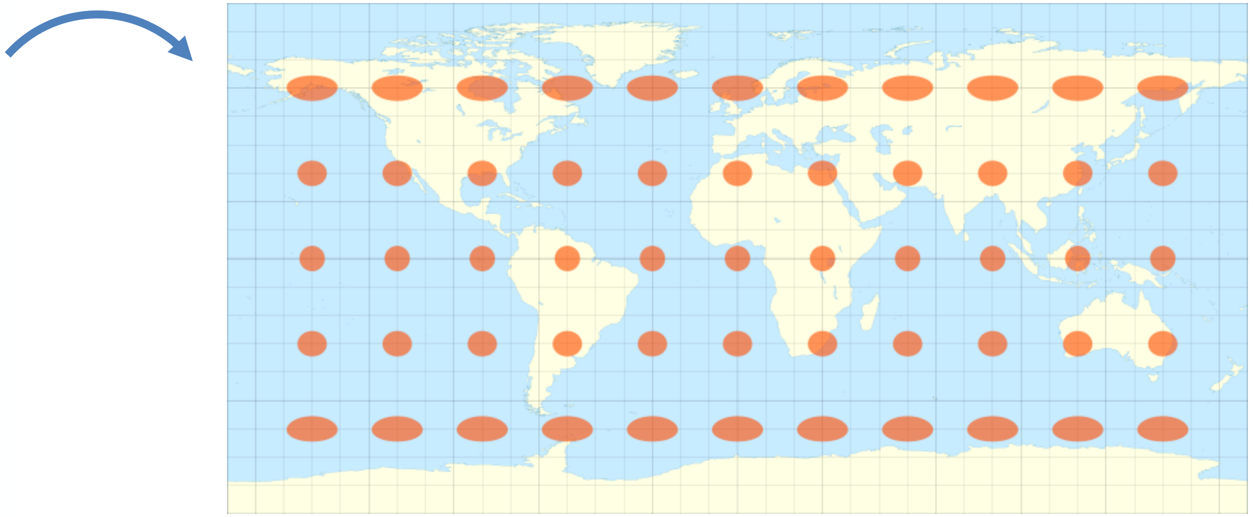


The bending of surfaces

Surface curvature



Stretch ✗

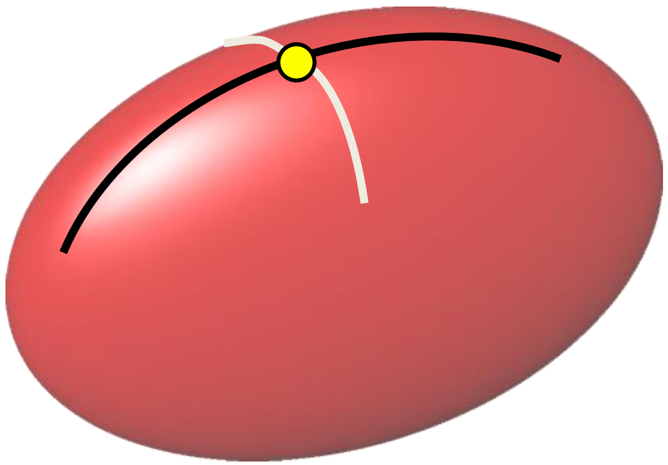


Mappings are strongly affected by bending!

Classification of surfaces by curvature

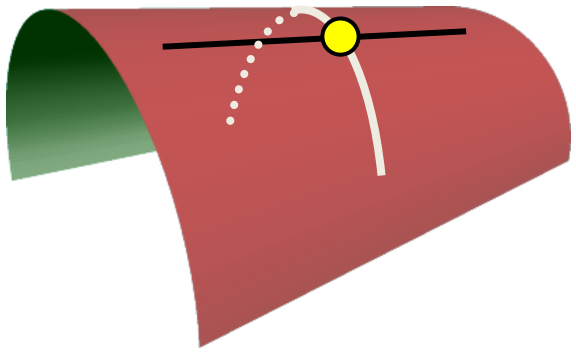
Only 3 classes:

elliptic



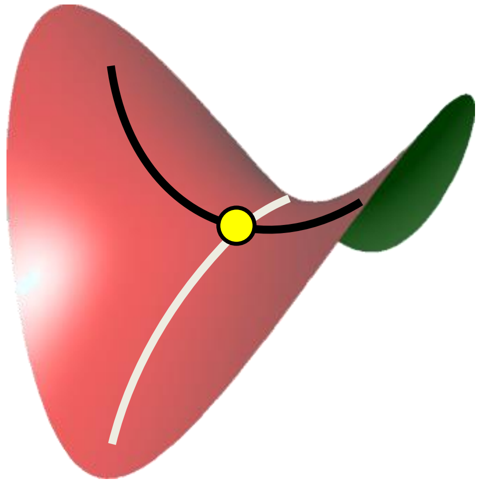
Bending "together"

parabolic



On side direction
is not bending

hyperbolic

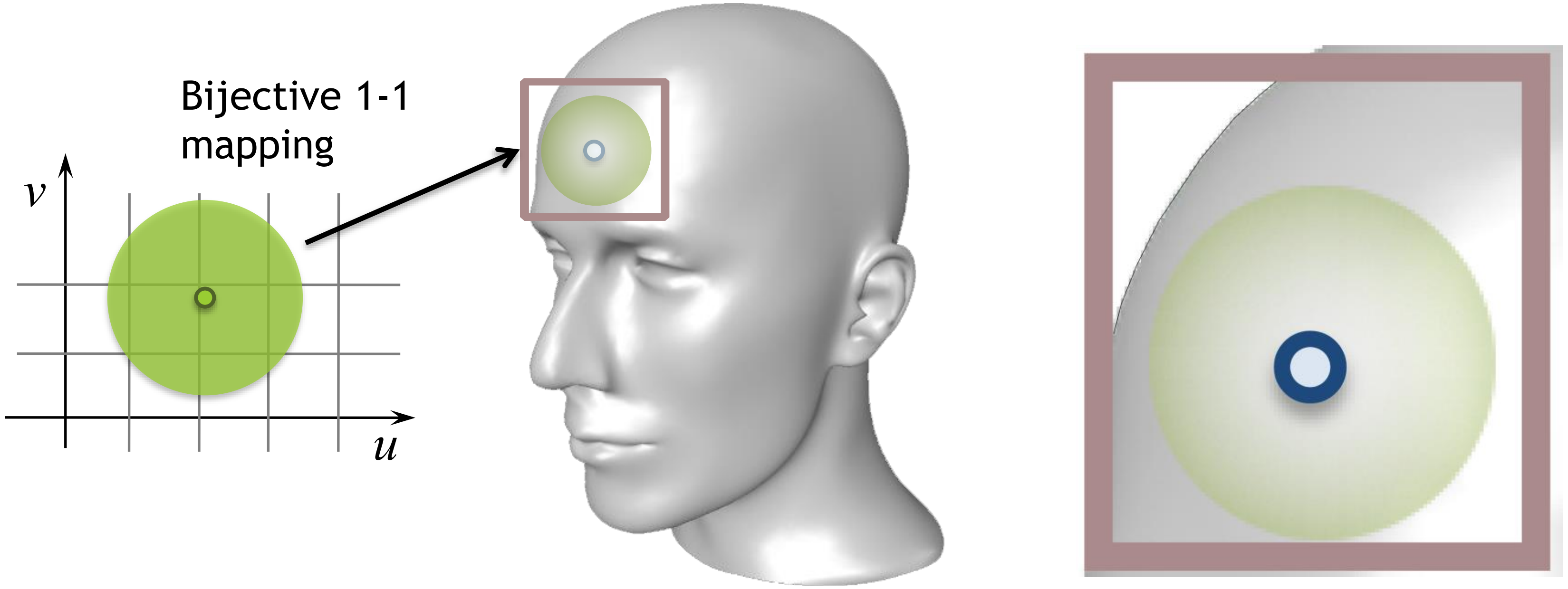


Bending
"away from each other"

Differential Geometry Basics

- What does it mean to be “manifold”?

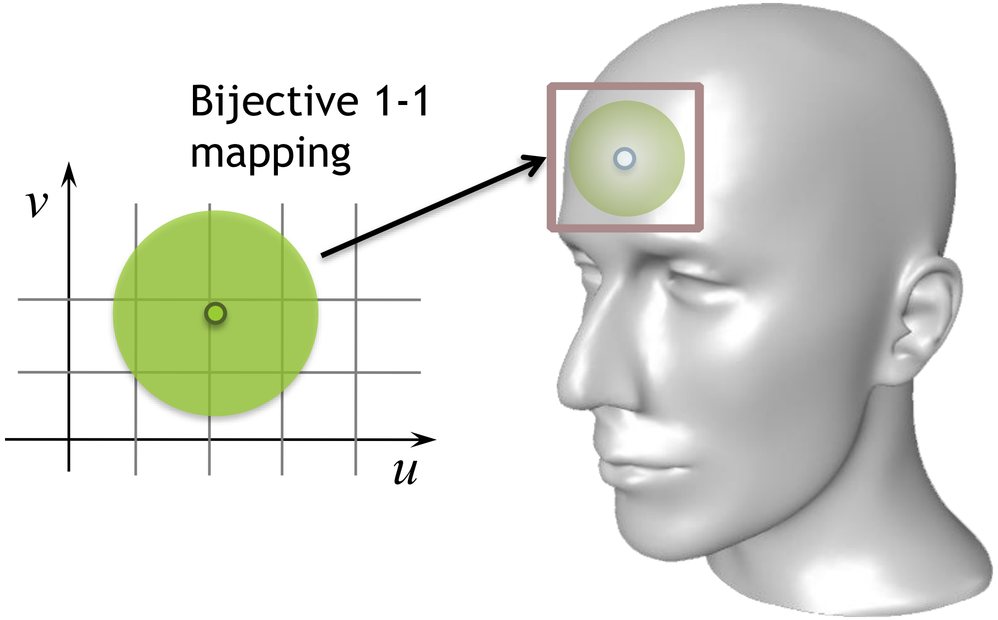
Local neighborhoods are mappings of tiny discs (or lines)



Differential Geometry Basics

- What does it mean to be “manifold”?

Local neighborhoods are mappings of tiny discs (or lines)



The tiny mappings allow us to compute:

- Derivatives
- Tangents
- Normals
- Curvature
- Angles
- Distances
- Etc…

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

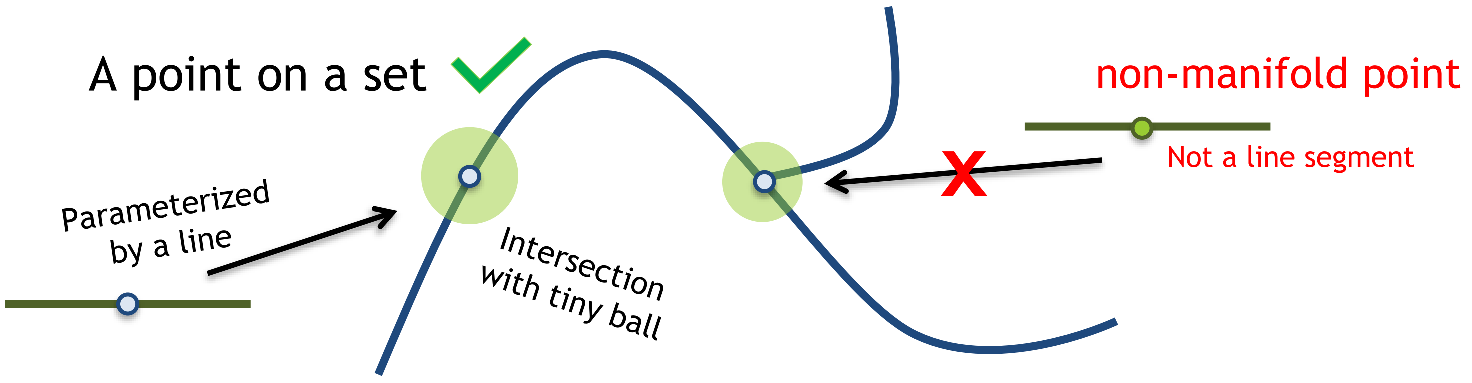
DG – Curves (1-dimensional manifolds)

Differential Geometry Basics

- Geometry of manifolds
- Local neighborhoods are parametrized by tiny lines (or discs)

Definition:

The neighbourhood is parameterizable by a line segment
⇔
1-dimensional manifold point



Definition:

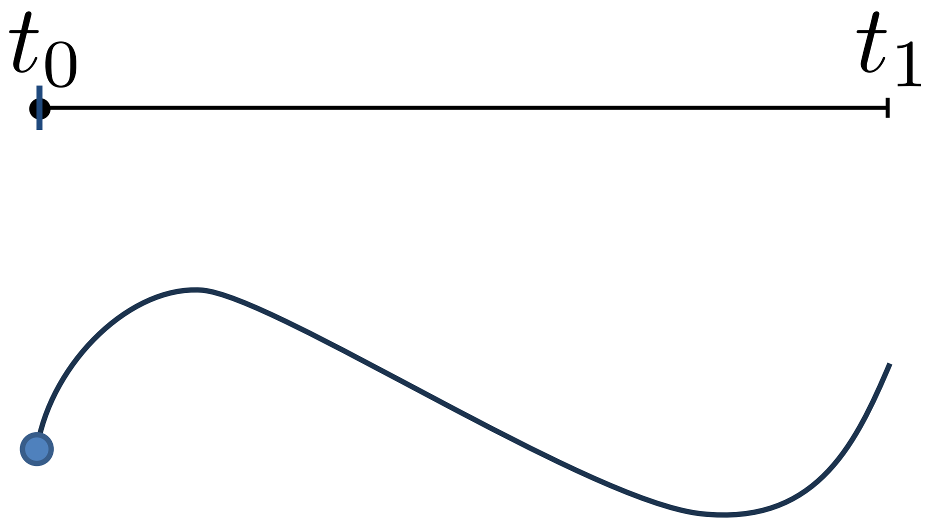
A geometry is manifold ⇔ all its points are manifold

Curves

- Parameterization in 2D: $$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \; t \in [t_0, t_1]$$

- $\mathbf{p}(t)$ must be continuous

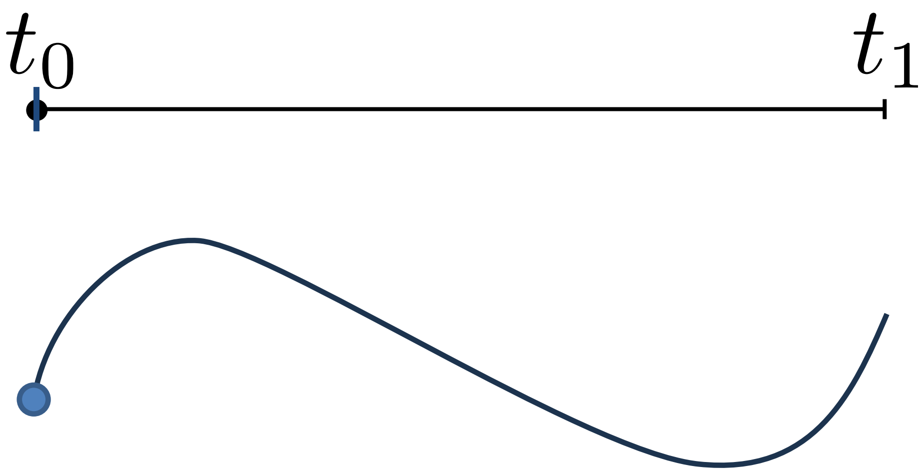
  no jumps!



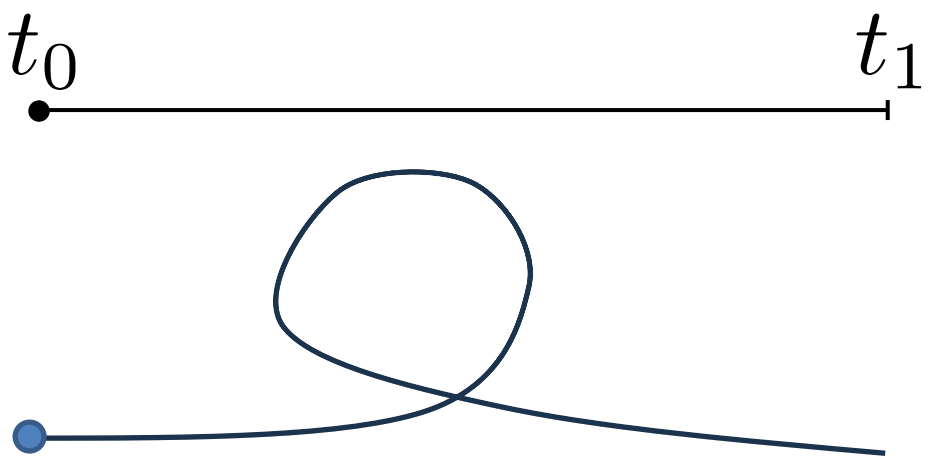
Properties of planar curves

- Length, curvature, tangents, normals

embedding *vs.* immersion



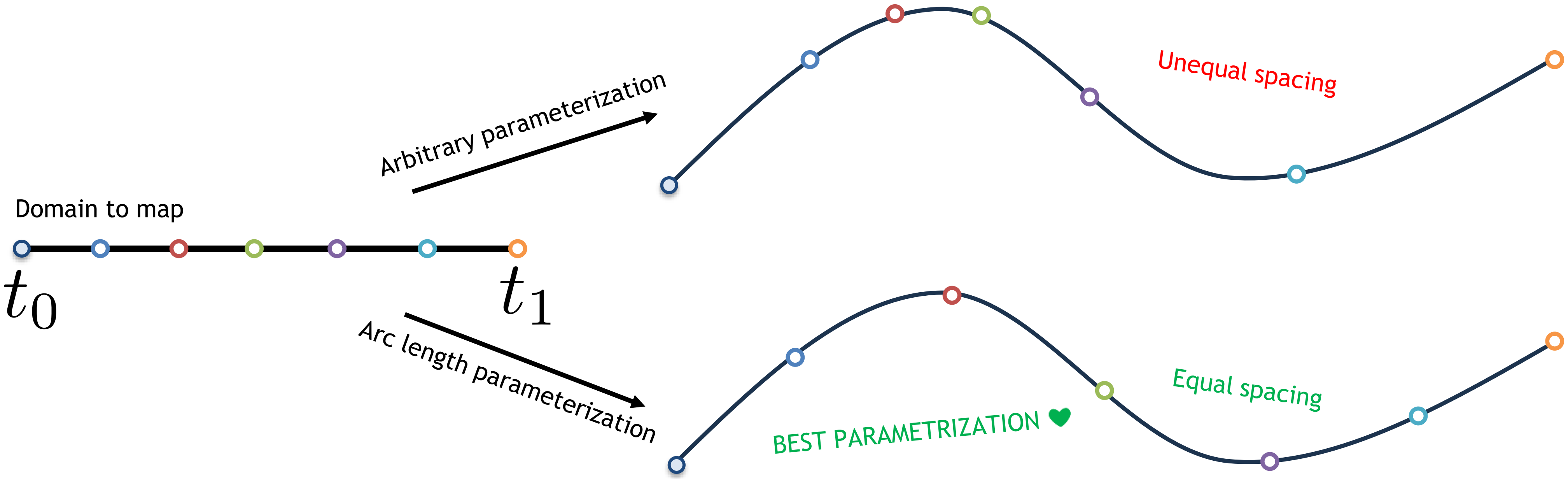
Parameterizable and manifold



Globally parameterizable,
but *not* necessarily manifold

Arc Length Parameterization

- Same curve has many different parameterizations!
- **Arc-length**: equal, unit pace of the parameter along the curve

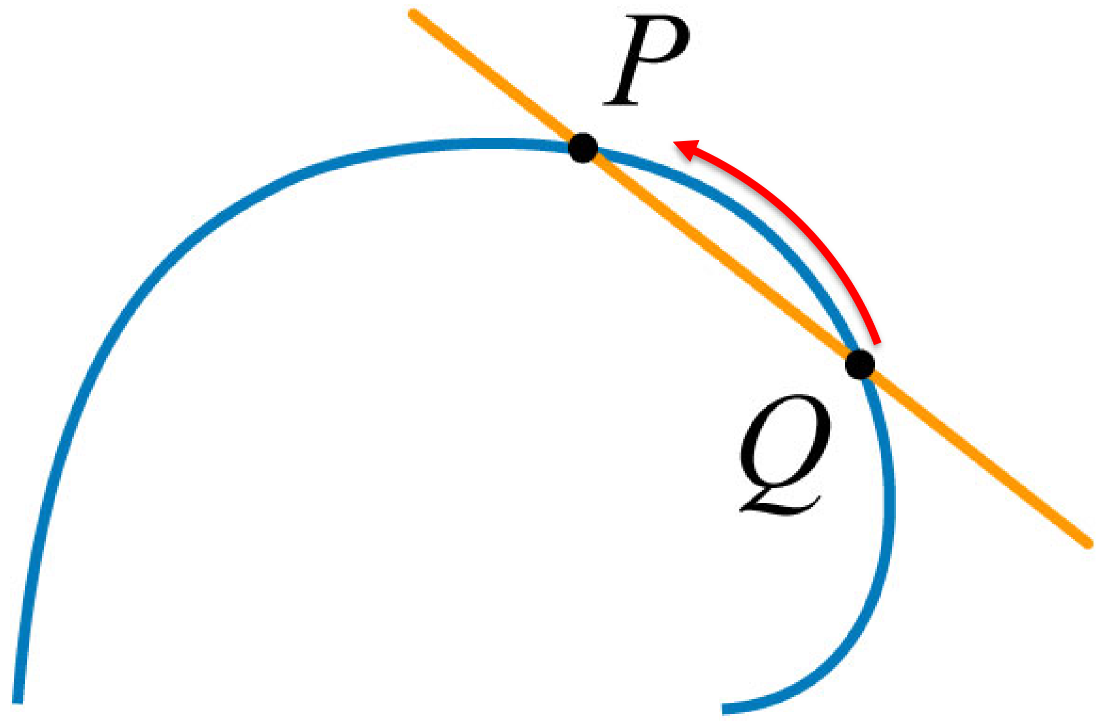


Same curves but different mappings!

Different derivative properties

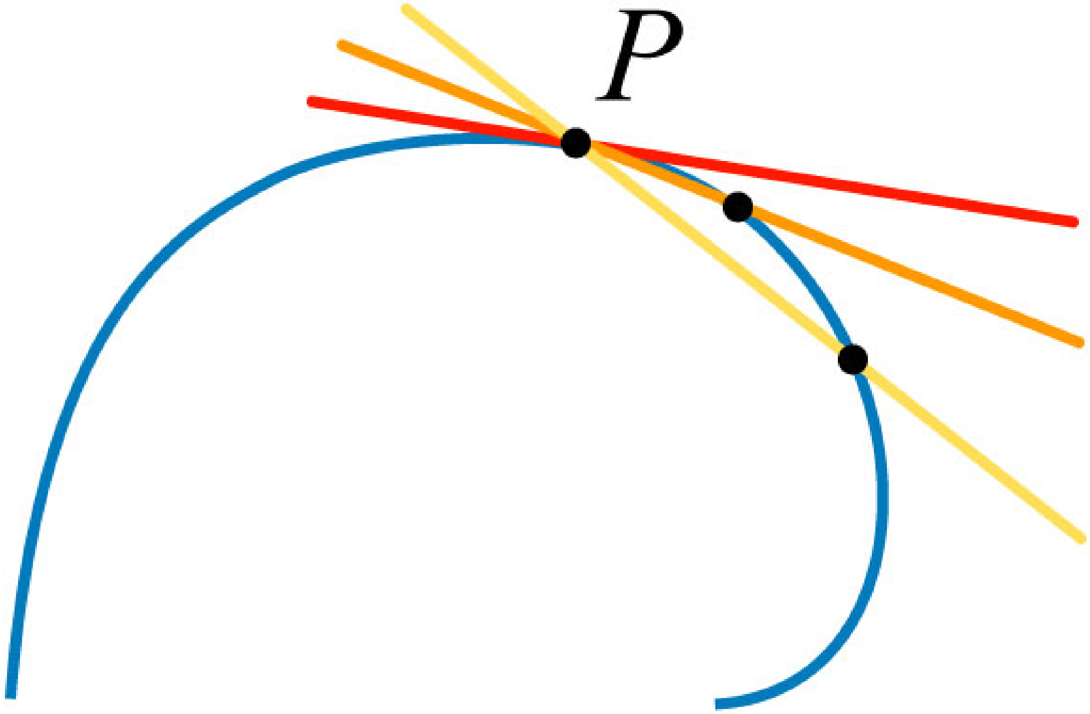
Secant

- A line through two points on the curve.



Tangent

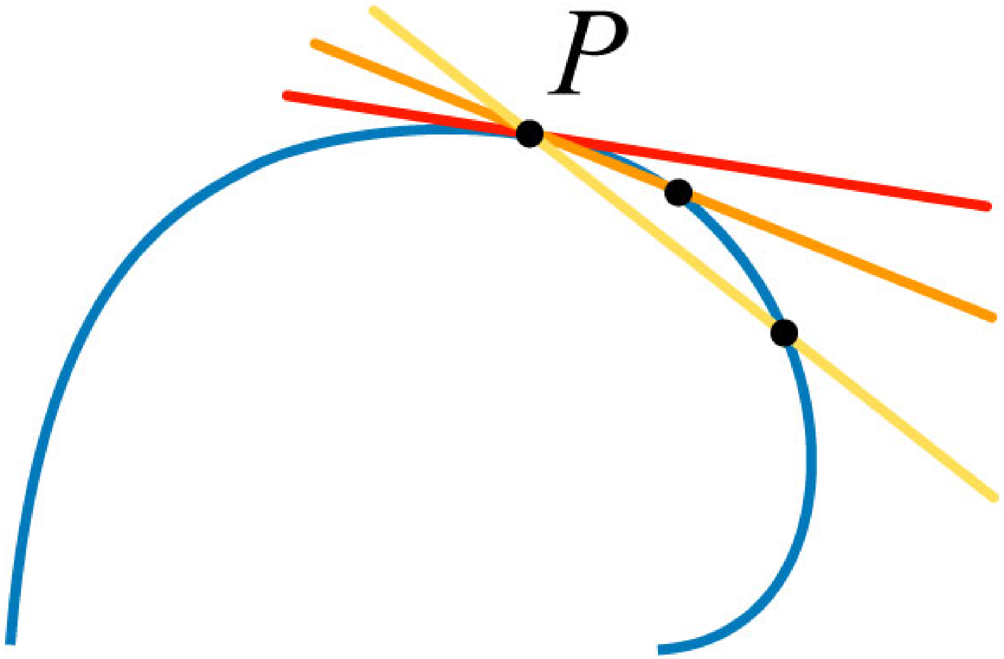
- The limiting secant as the two points come together.



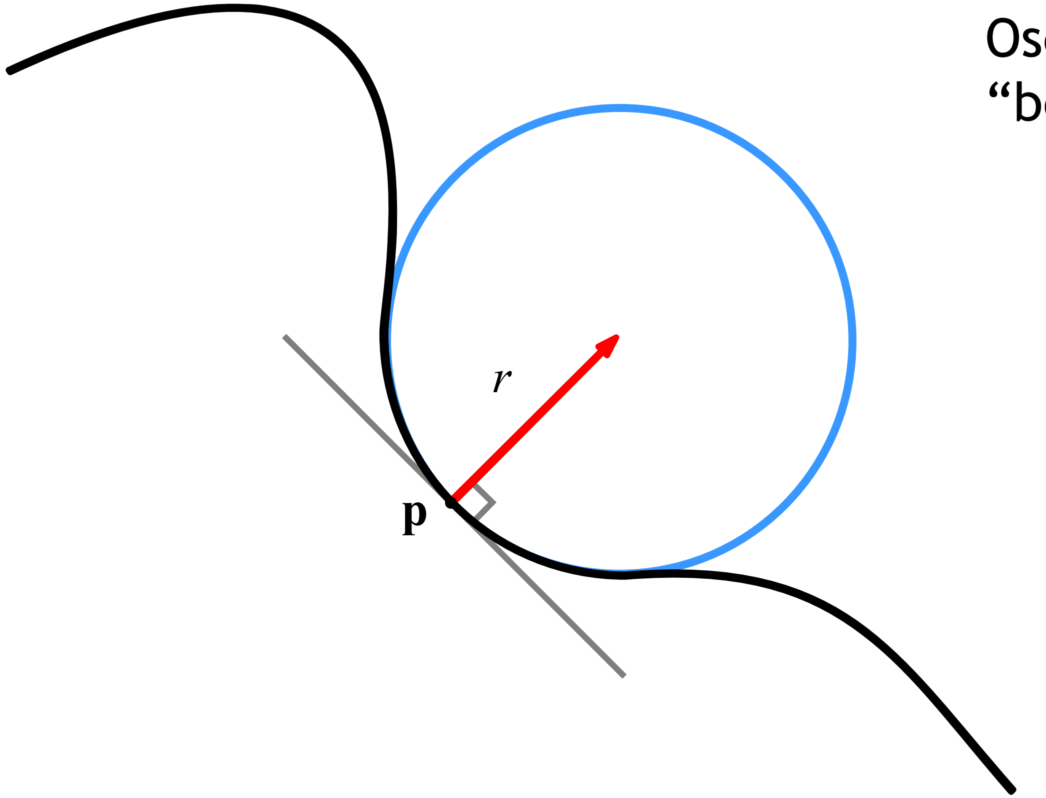
Secant and Tangent – Parametric Form

- Secant: line through $\mathbf{p}(t) - \mathbf{p}(s)$
- Tangent vector: $\mathbf{p}'(t) = (x'(t), y'(t), \ldots)^{\mathsf{T}}$

parameterization is arc-length
$\Leftrightarrow \| \mathbf{p}'(t) \| = 1$



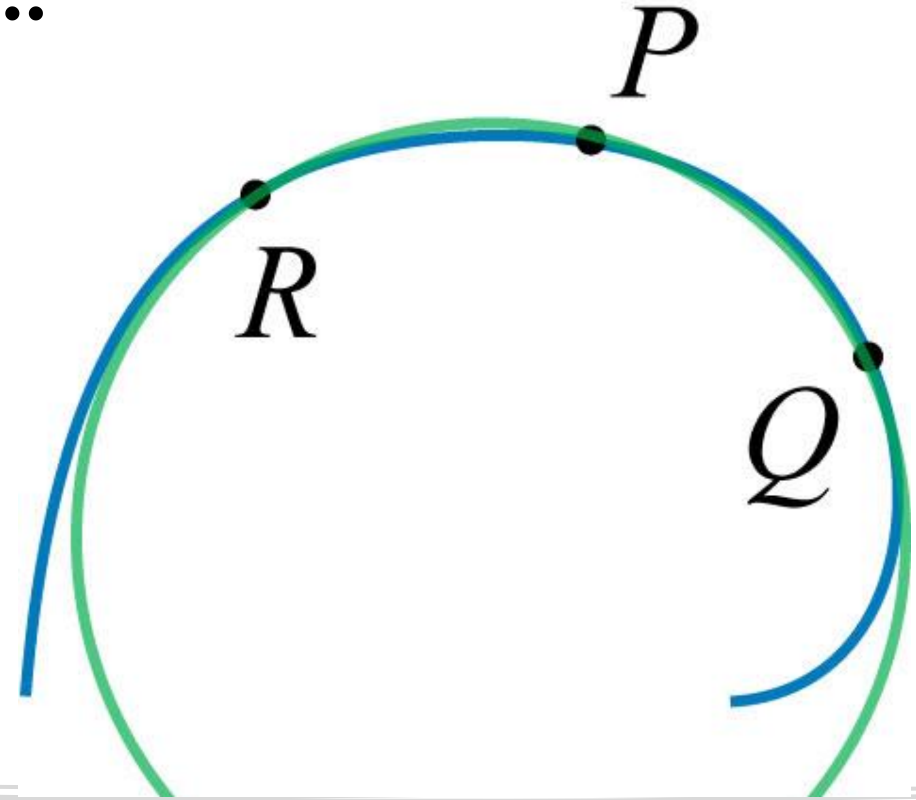
Tangent, normal, radius of curvature



Osculating circle
“best fitting circle”

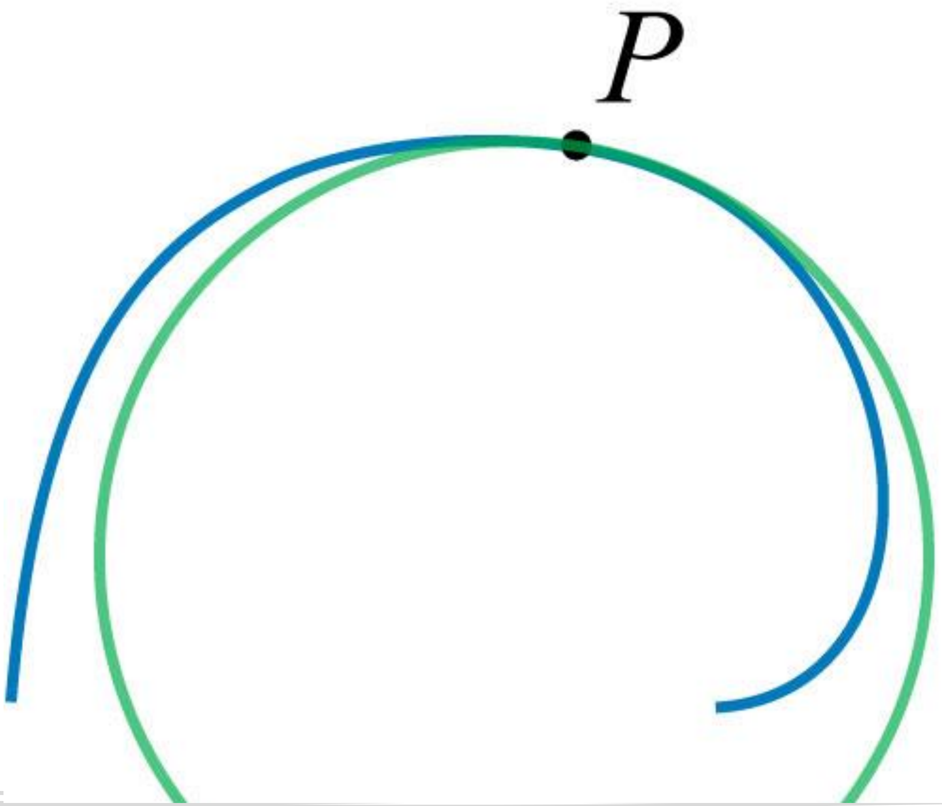
Circle of Curvature

- Consider the circle passing through three points on the curve…



Circle of Curvature

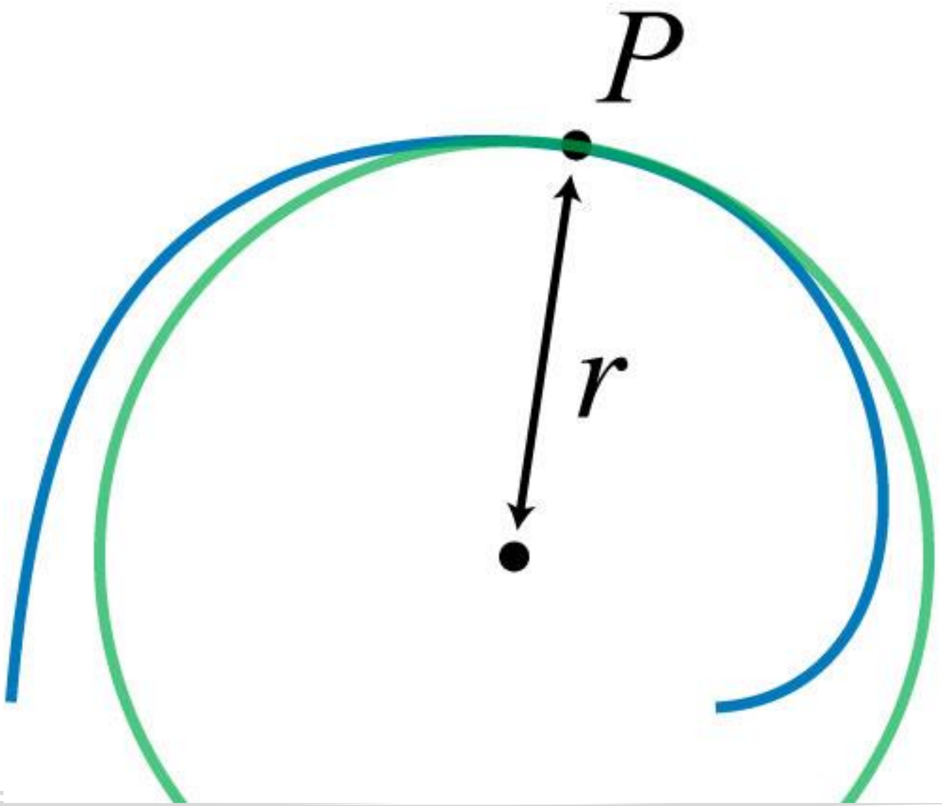
- …the limiting circle as three points come together.



Radius of Curvature, r

Curvature

$$\kappa = \frac{1}{r}$$

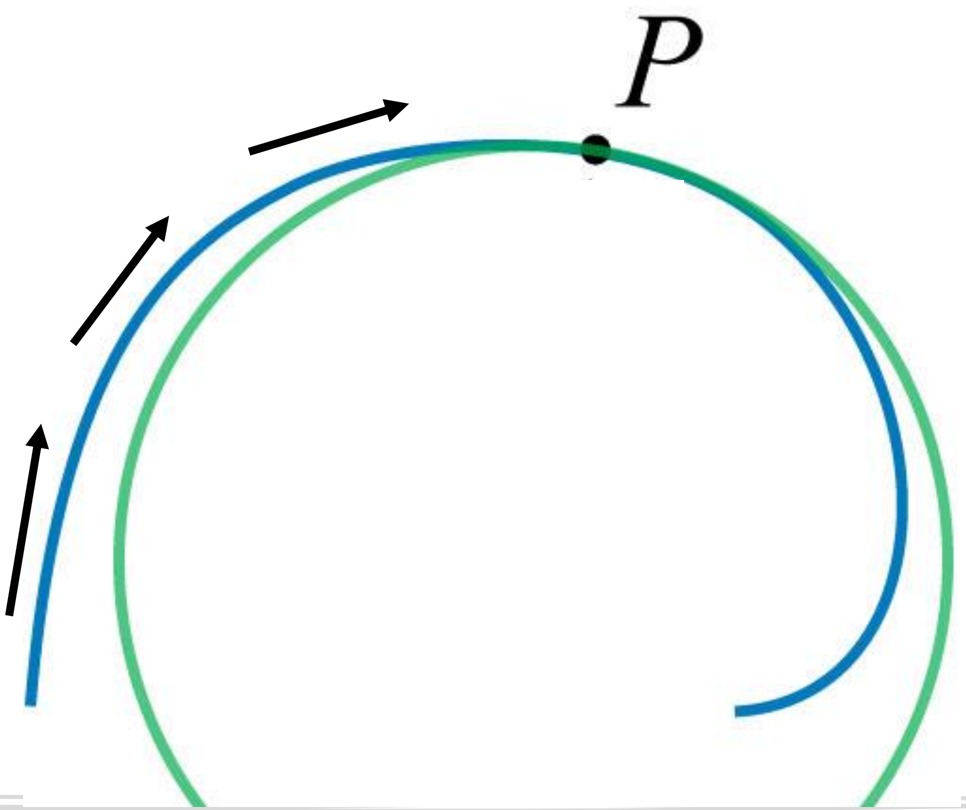


Signed Curvature

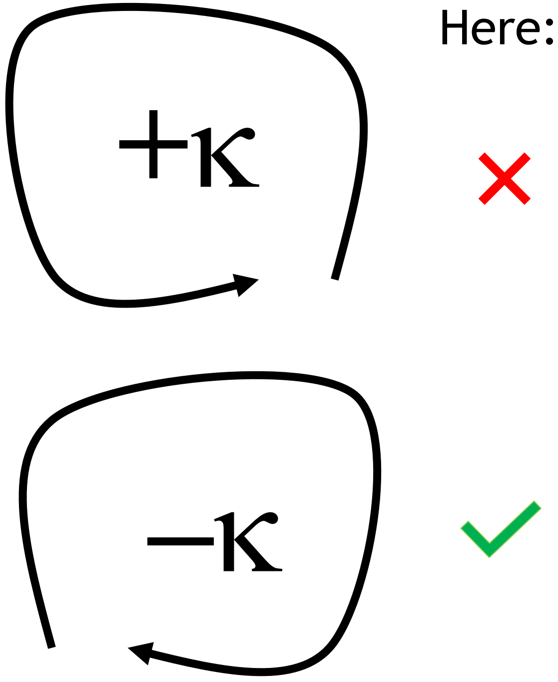
- Traversal along curve

Curvature

$$\kappa = \frac{1}{r}$$



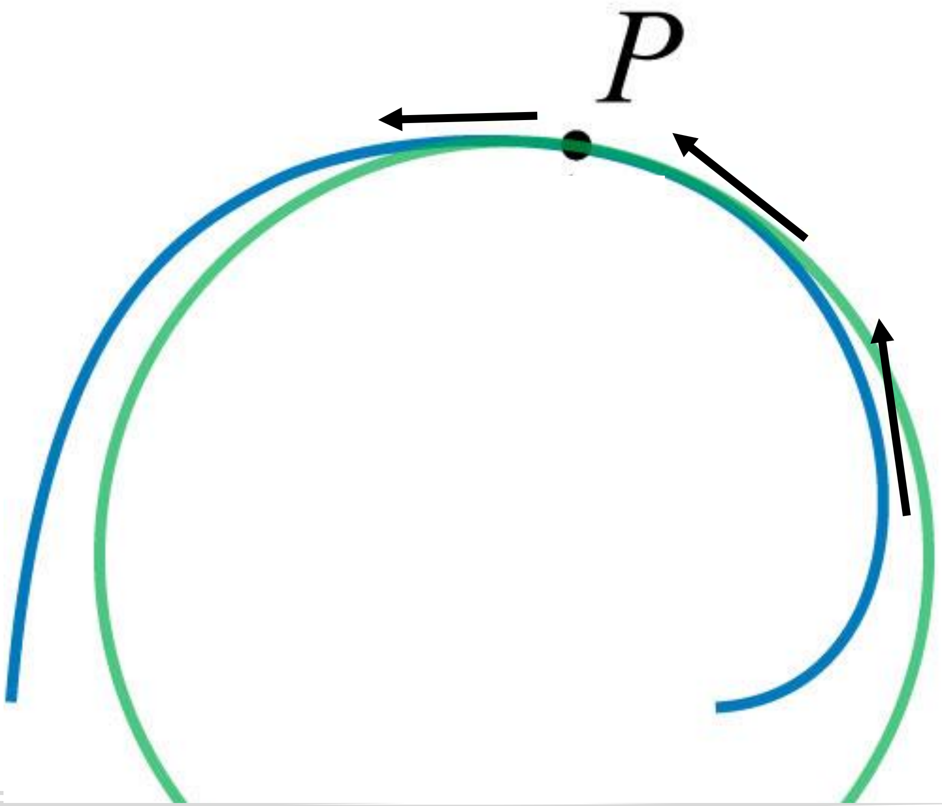
Here:



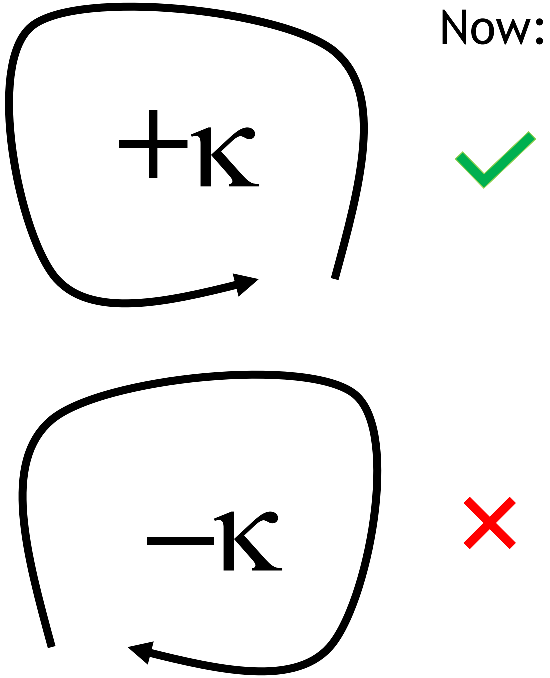
Signed Curvature

- Traversal along curve

Flipping direction flips curvature for planar curves!



Now:

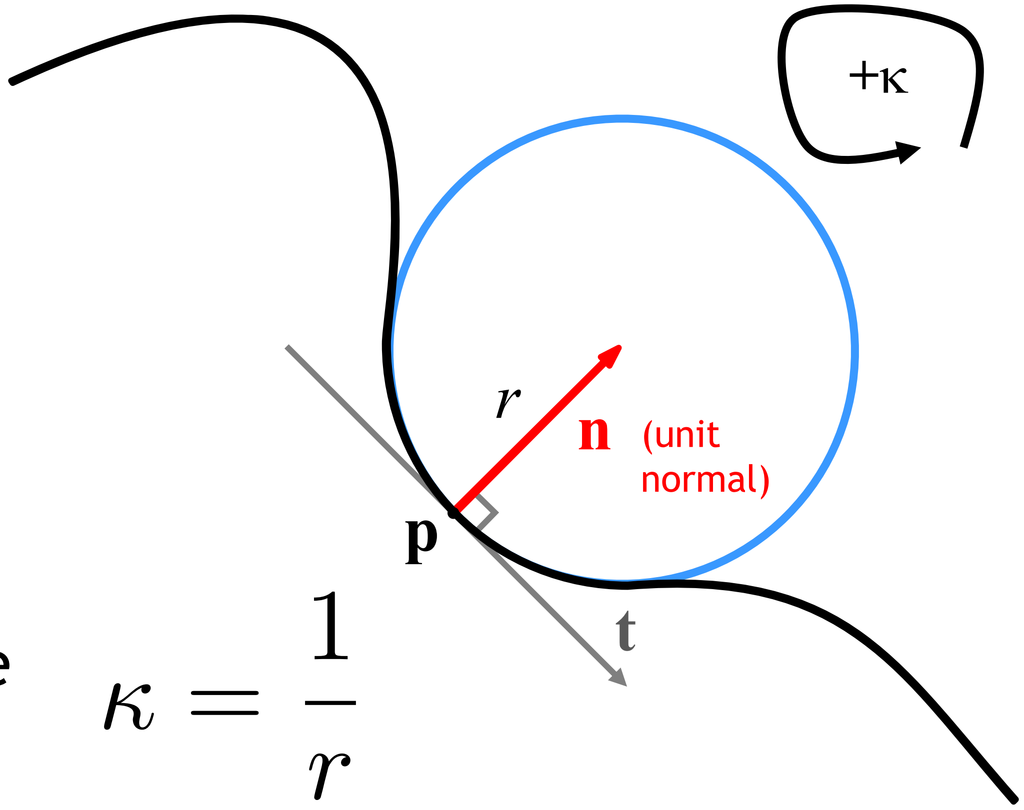


Curvature in arc-length parameterization

- Curvature κ corresponds to the rate of change of the tangent $\mathbf{t}$ (size of its derivative)

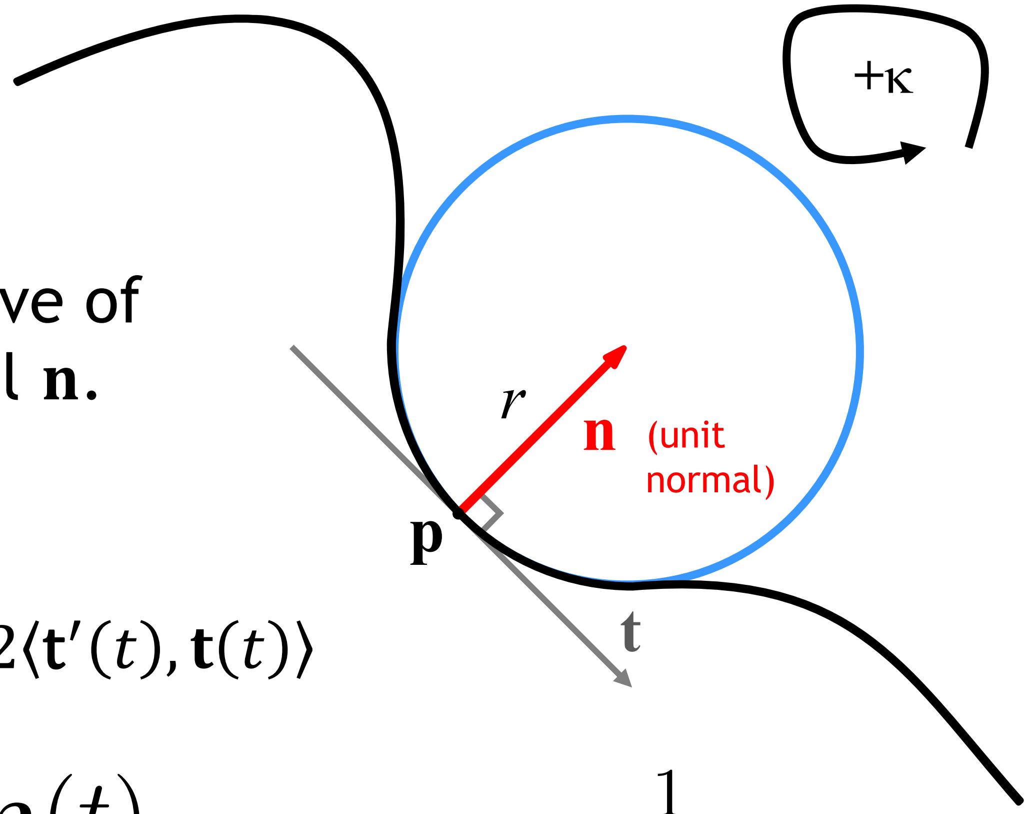
- Curvature is inversely proportional to the osculating circle radius r

$$\kappa = \frac{1}{r}$$



Curvature in arc-length parameterization

- In arc-length parameterization, the derivative of the tangent $\mathbf{t}$, is parallel to the curve normal $\mathbf{n}$.



Quick proof:

$$\langle \mathbf{t}(t), \mathbf{t}(t) \rangle = |t|^2 = 1 \quad \Rightarrow 0 = 1' = \langle \mathbf{t}(t), \mathbf{t}(t) \rangle' = 2\langle \mathbf{t}'(t), \mathbf{t}(t) \rangle$$

$$\mathbf{p}''(t) = (\mathbf{p}'(t))' = \mathbf{t}'(t) = \kappa(t)\, \mathbf{n}(t)$$

$$\kappa = \frac{1}{r}$$

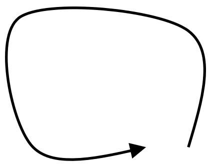
The Curvature Normal

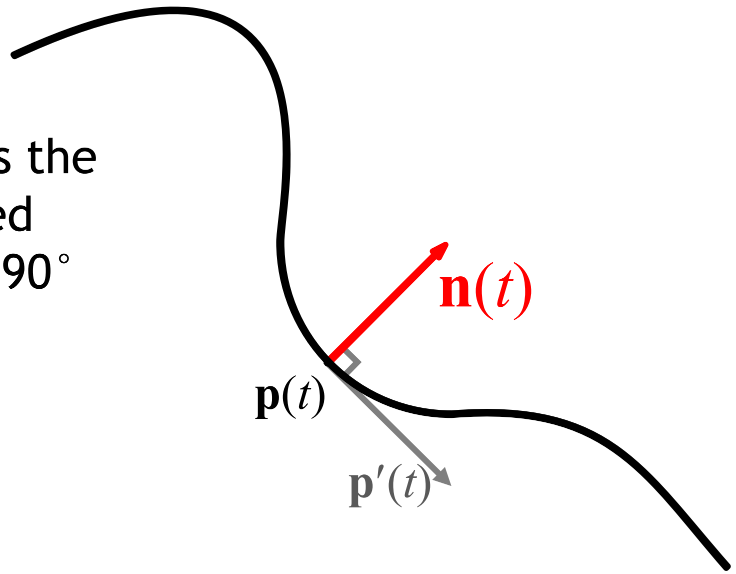
- When t is arc-length parameter

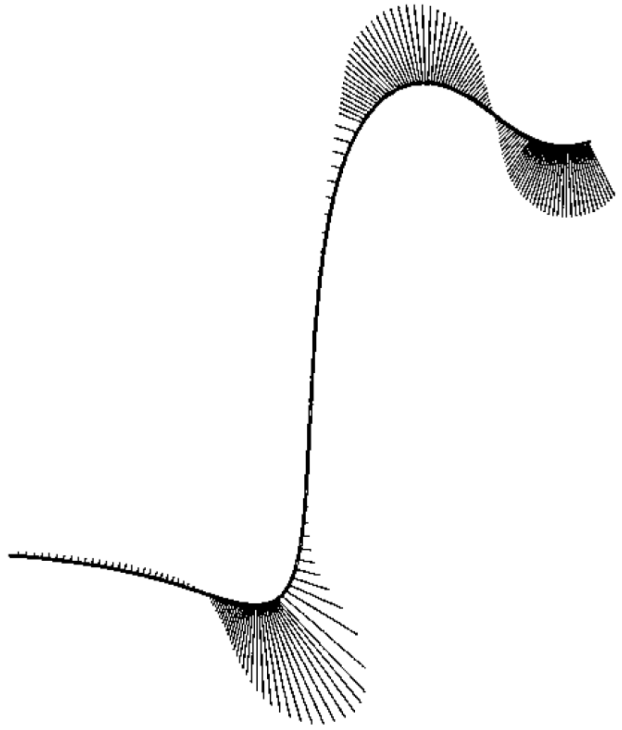
$$\mathbf{p}''(t) = \kappa(t)\,\mathbf{n}(t)$$

Definition:

For planar curves, $\mathbf{n}$ is the unit tangent rotated counterclockwise by 90°







"A multiresolution framework for variational subdivision", Kobbelt and Schröder, ACM TOG 17(4), 1998

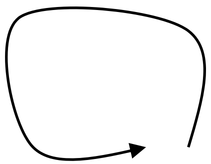
The Curvature Normal

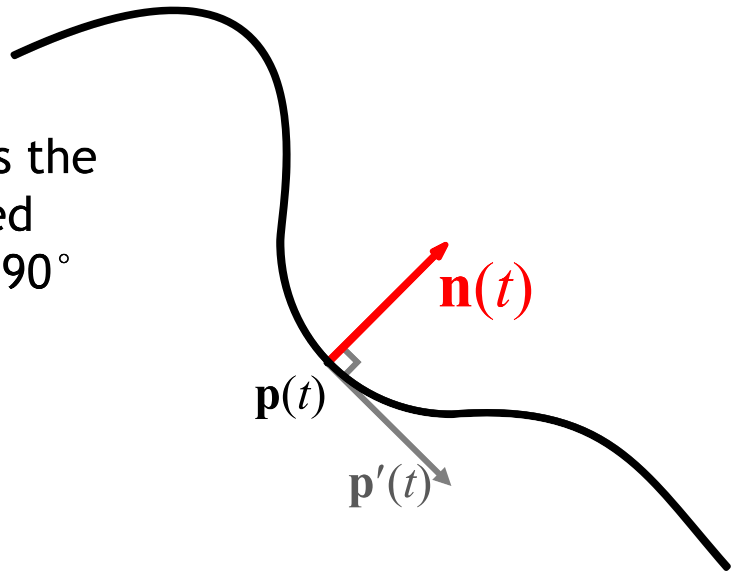
- When t is arc-length parameter

$$\mathbf{p}''(t) = \kappa(t)\,\mathbf{n}(t)$$

Definition:

For planar curves, $\mathbf{n}$ is the unit tangent rotated counterclockwise by 90°





> ***Theorem:***
> The curvature **defines** the planar curve **shape**, up to rotation and translation!

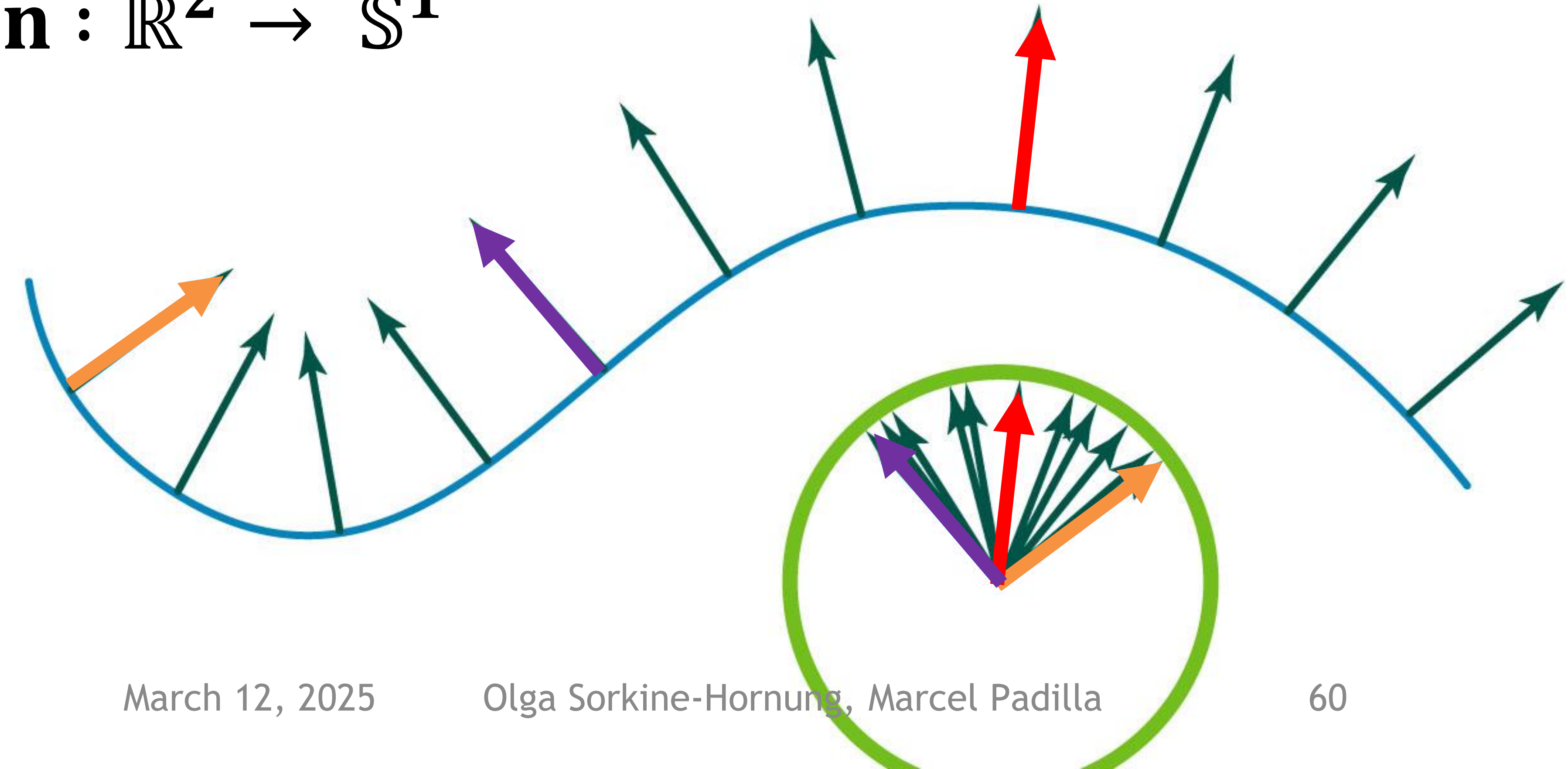
All these curves have the exact same curvature

(Because it is a non-vectorial property of the derivative.)

Gauss map $\mathbf{n}(\mathbf{p})$

- Maps points on a curve to the normal at that point.

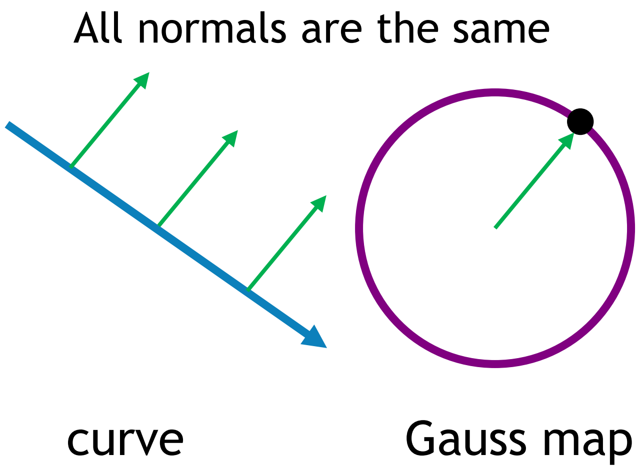
$$\mathbf{n} : \mathbb{R}^2 \rightarrow \mathbb{S}^1$$

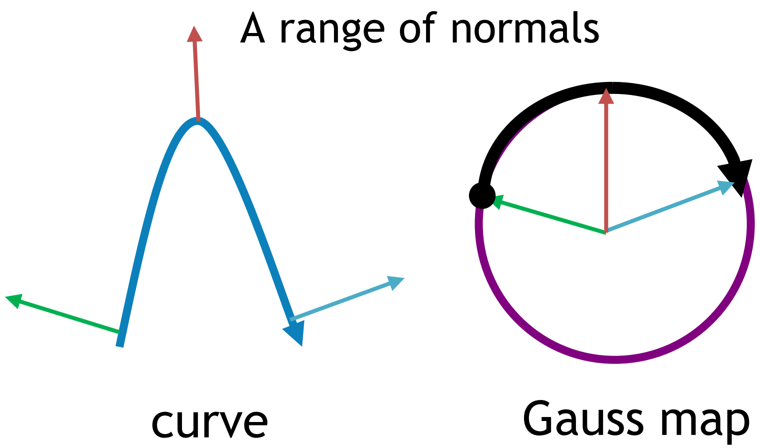


Curvature = change in normal direction

- Assuming arc length t
  - Absolute curvature: $\kappa(t) = \|\mathbf{n}'(t)\|$
  - Signed curvature: $\mathbf{n}'(t) = -\kappa(t)\,\mathbf{t}(t)$

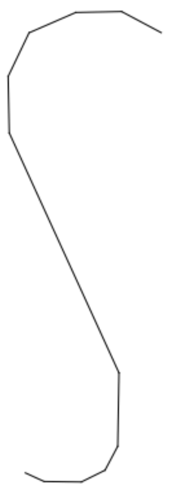
Parameter-free view via the Gauss map





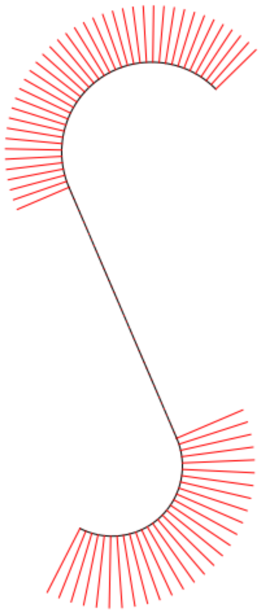
Curvature Normal – Examples

G^0



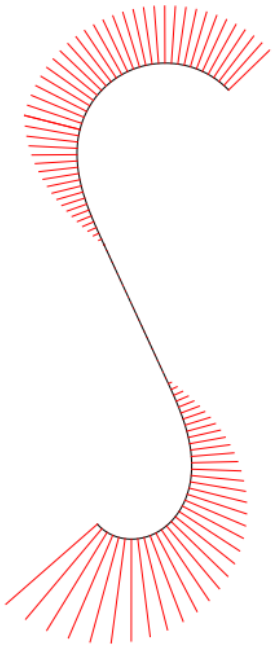
No continuous tangents

G^1



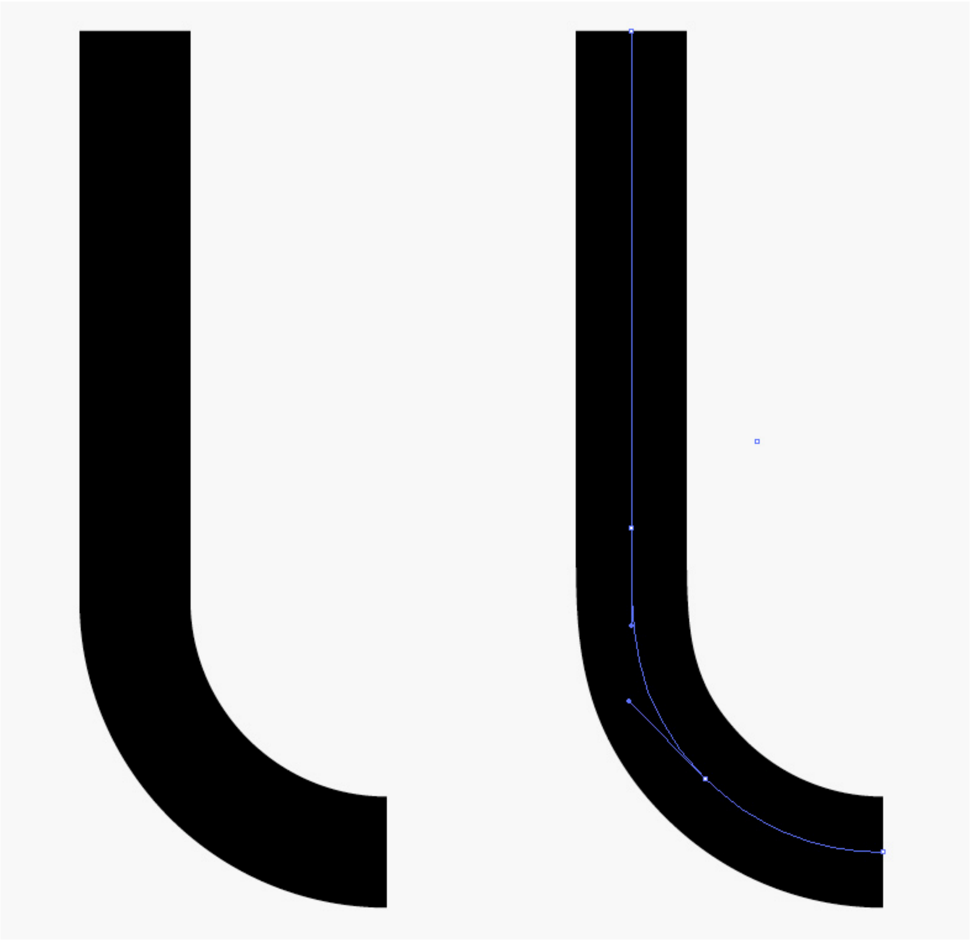
Continuous tangents, but no continuous curvature

G^2



smooth

Smoothness Example



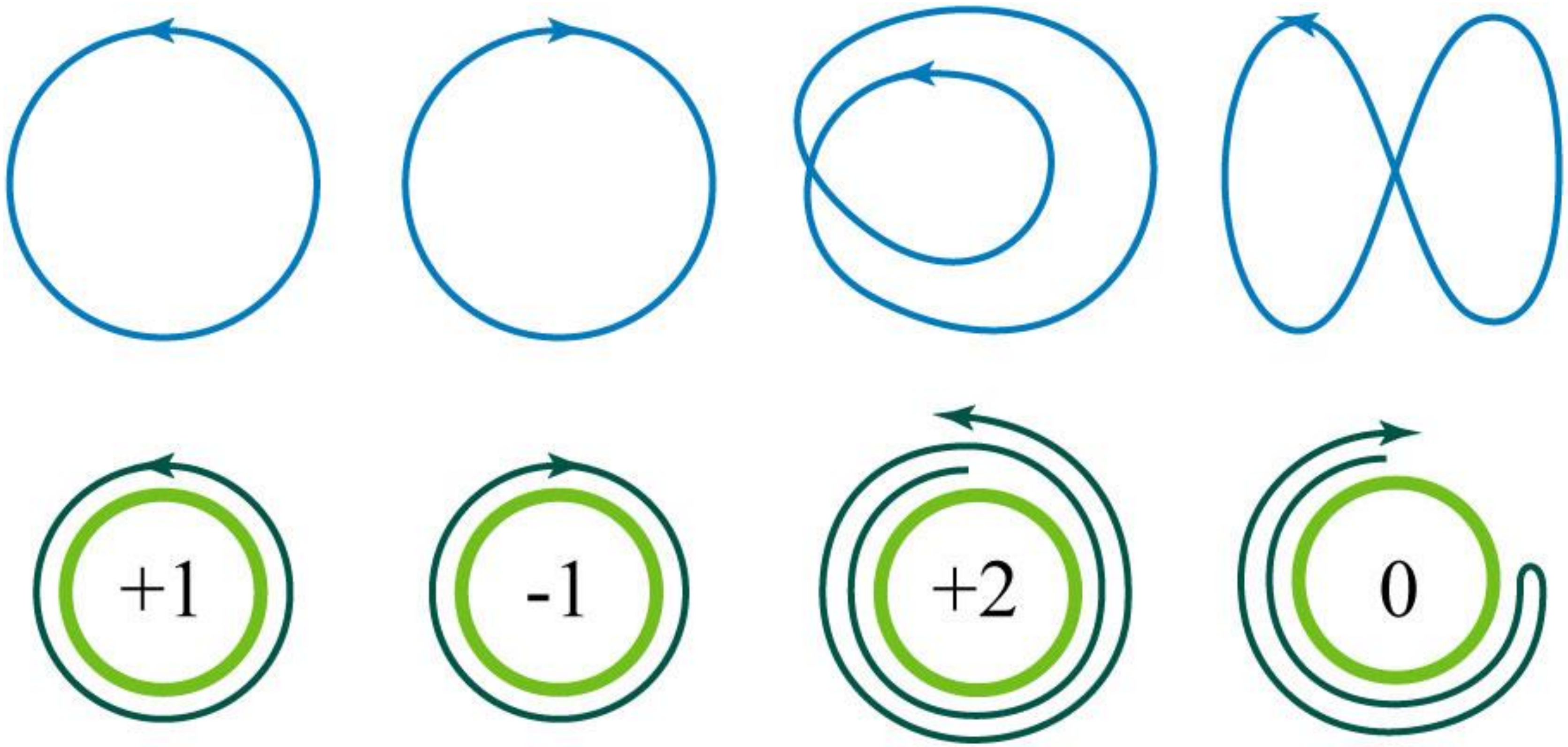
Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = ?$$

Curvature and Topology – Turning Number, k

- Number of orbits in Gauss image



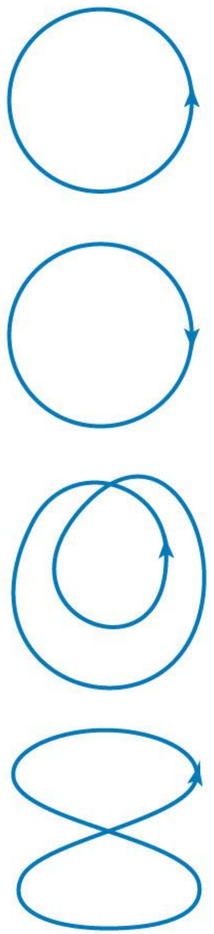
Curvature and Topology

> ***Turning Number Theorem:***
> For a closed curve,
> the integral of curvature is
> an integer multiple of 2π.

$$\int_\gamma \kappa \, dt = 2\pi k$$

Interpretation: If you want to drive back to the start, your total curvature / steering needs to match the number of loops times 2π.

$$\int_\gamma \kappa \, dt =$$



$+2\pi$

-2π

$+4\pi$

0

Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = 2\pi$$

Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = 4\pi$$

Total Curvature

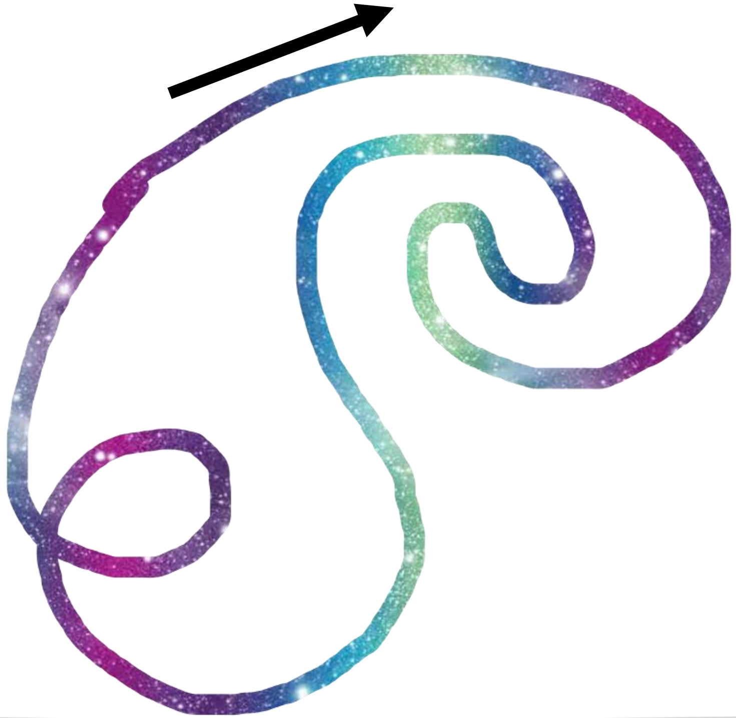
- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = 2\pi$$

Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = -4\pi$$



Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = 4\pi$$

Total Curvature

- What is the total curvature of the following curve?

$$\int_\gamma \kappa \, dt = 0$$

Thank you

INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich