

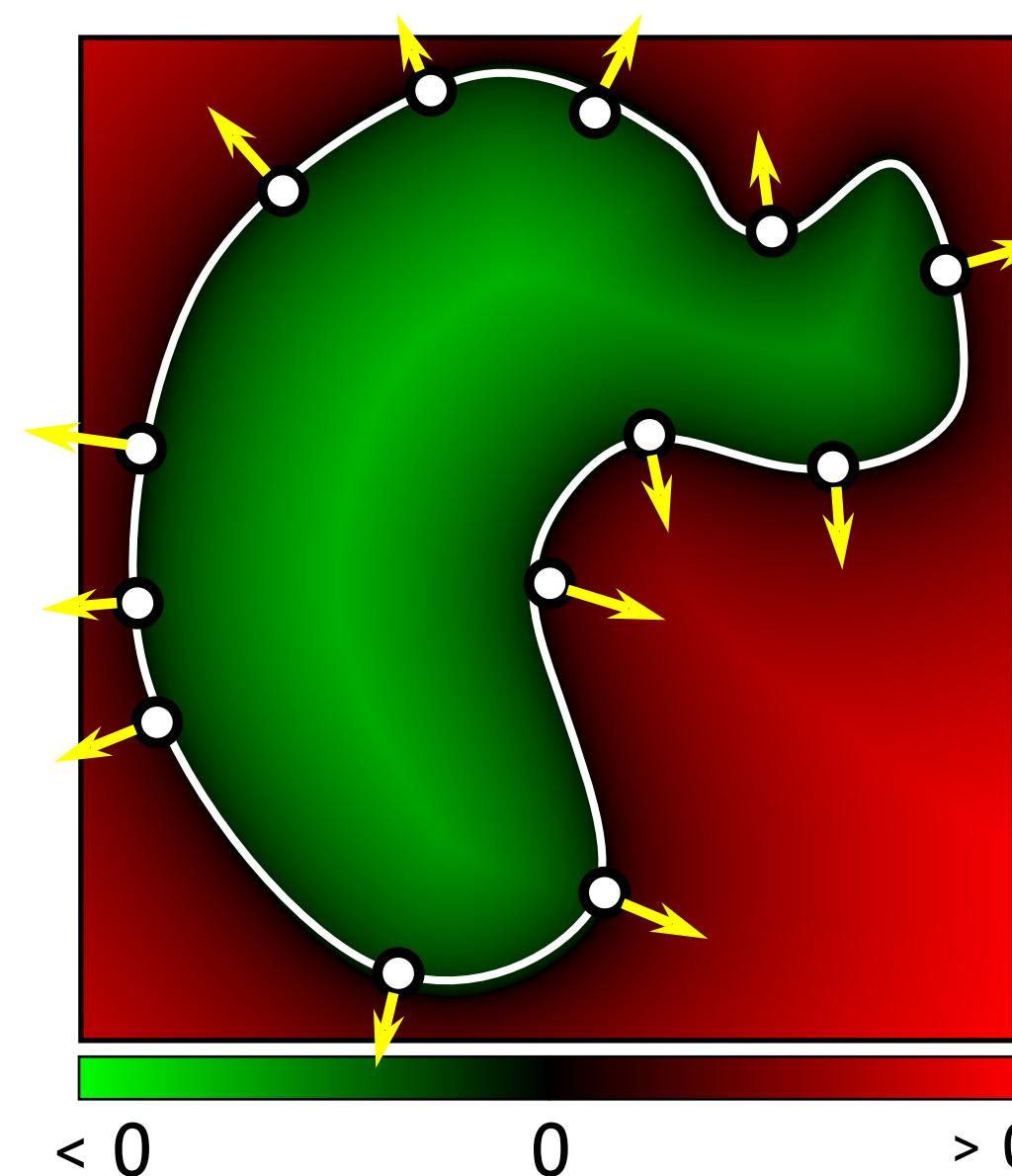
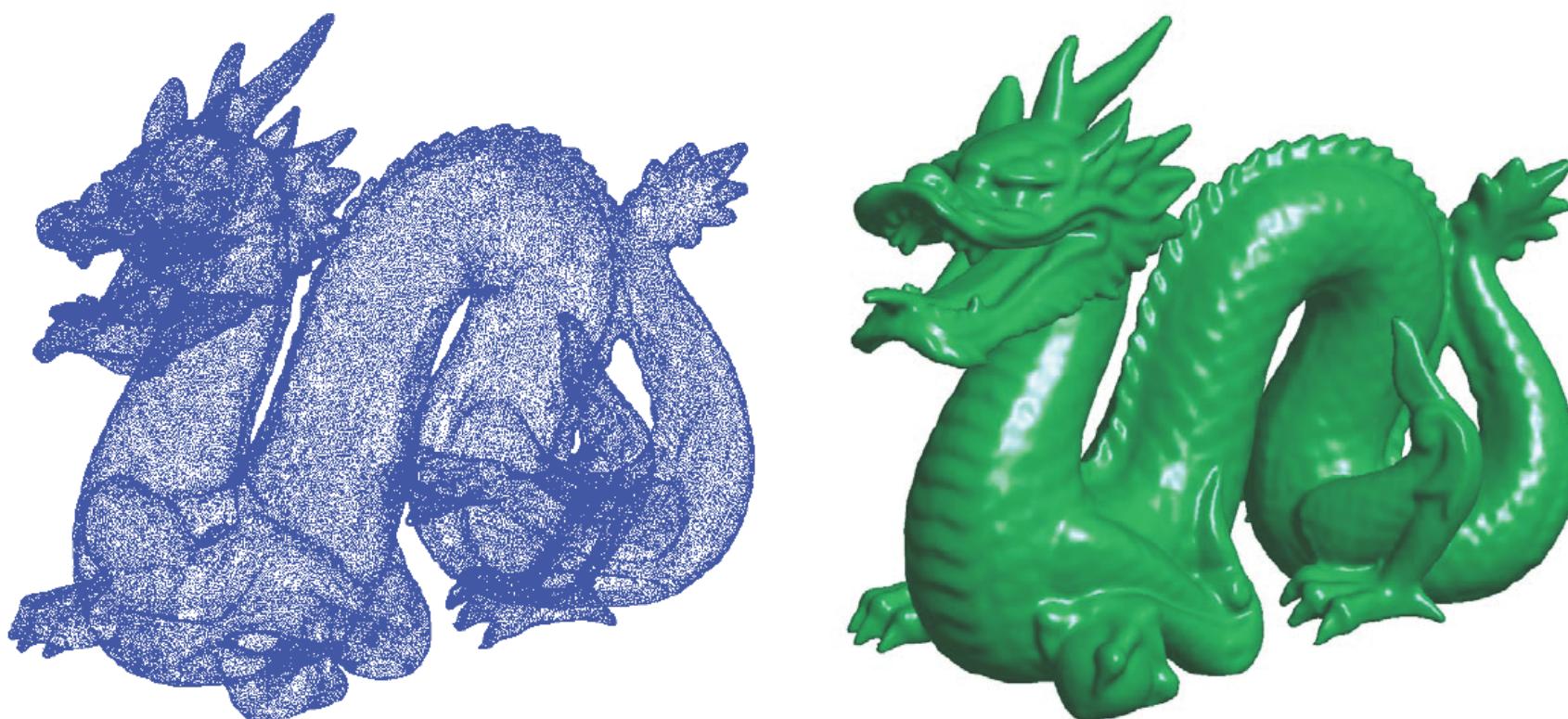
04 - Normal Estimation, Curves

Acknowledgements: Olga Sorkine-Hornung
CSCI-GA.3033-018 - Geometric Modeling - Daniele Panozzo

Normal Estimation

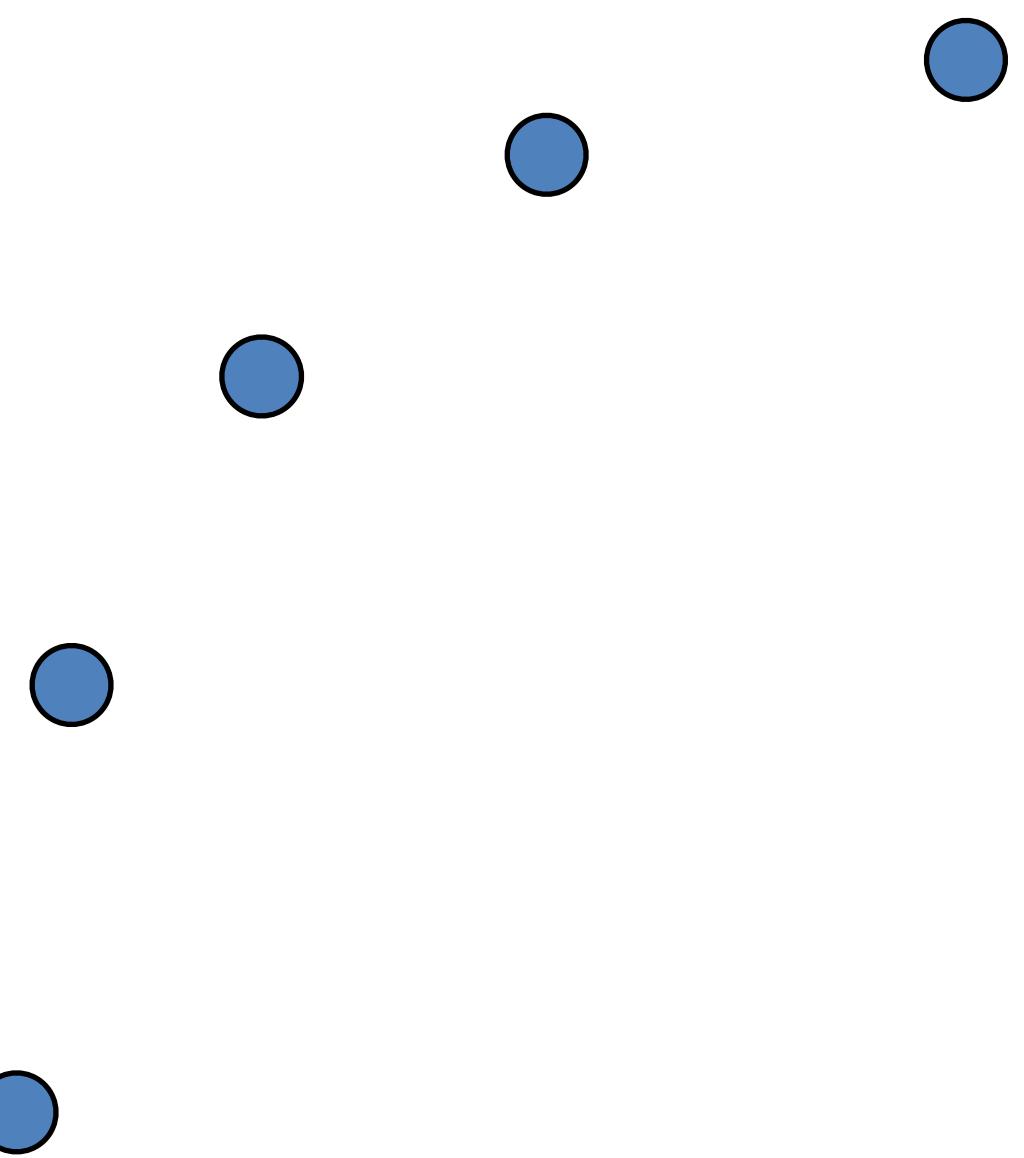
Implicit Surface Reconstruction

- Implicit function from point clouds
- Need consistently oriented normals



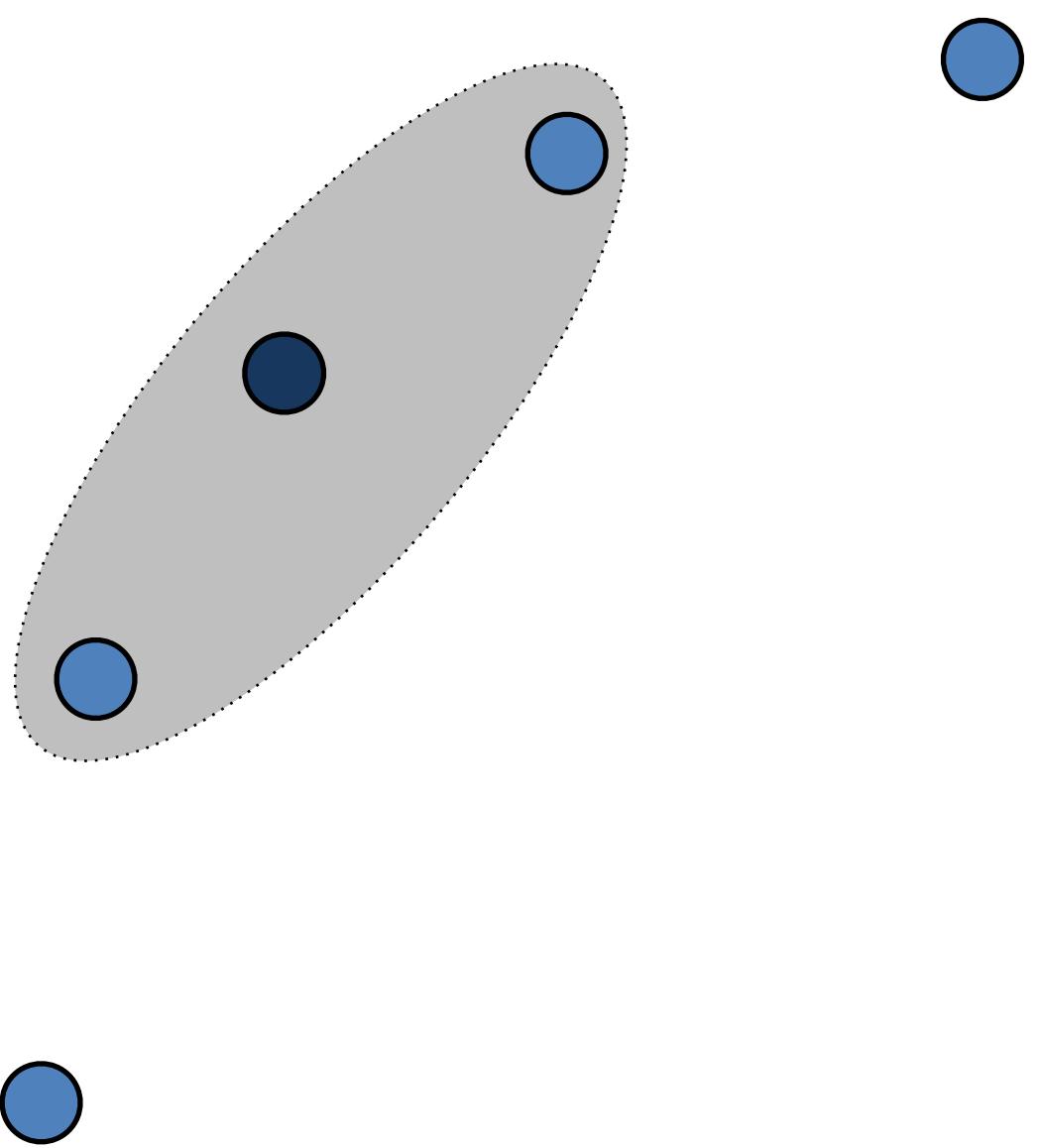
Normal Estimation

- 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)



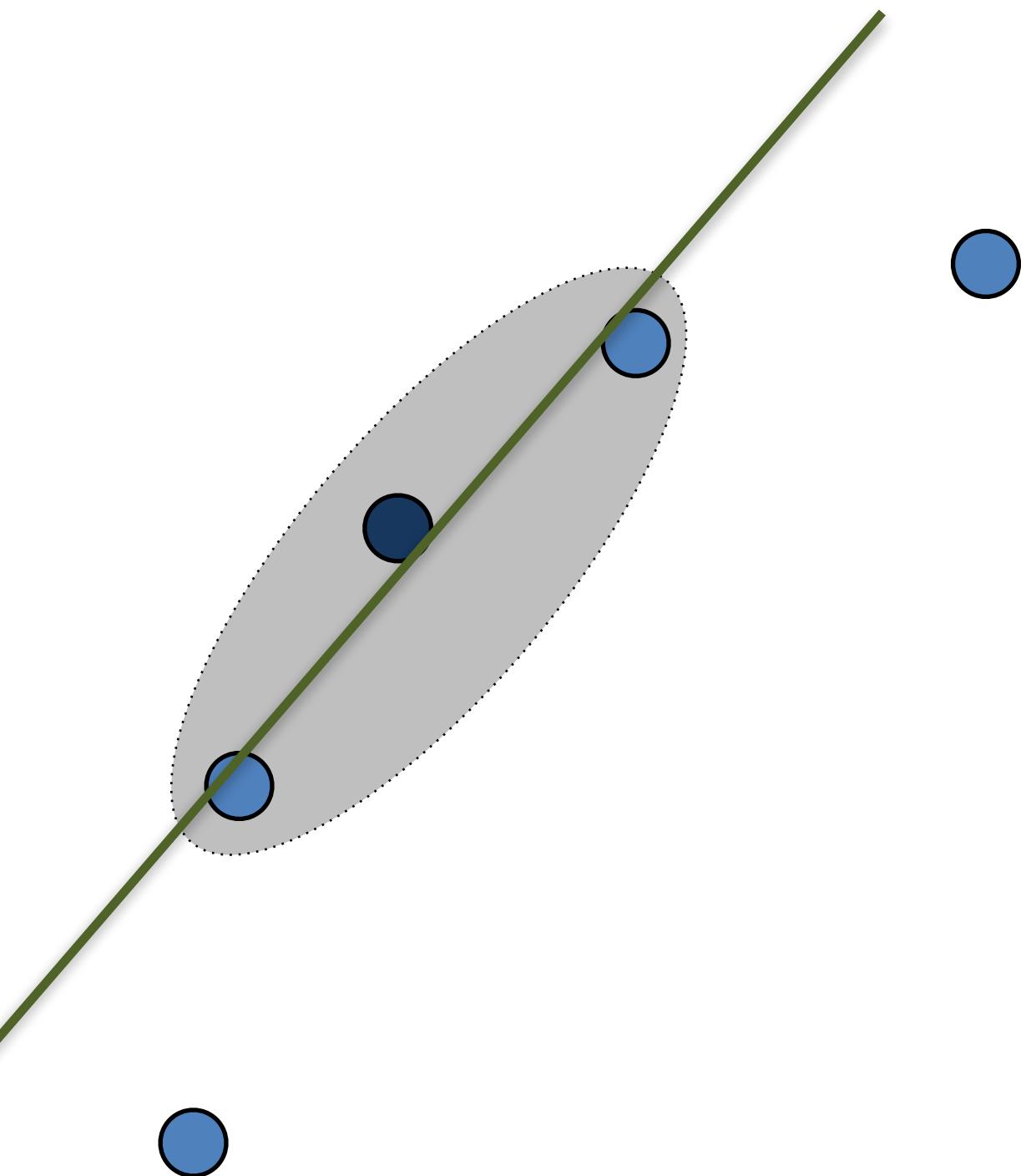
Normal Estimation

- 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)



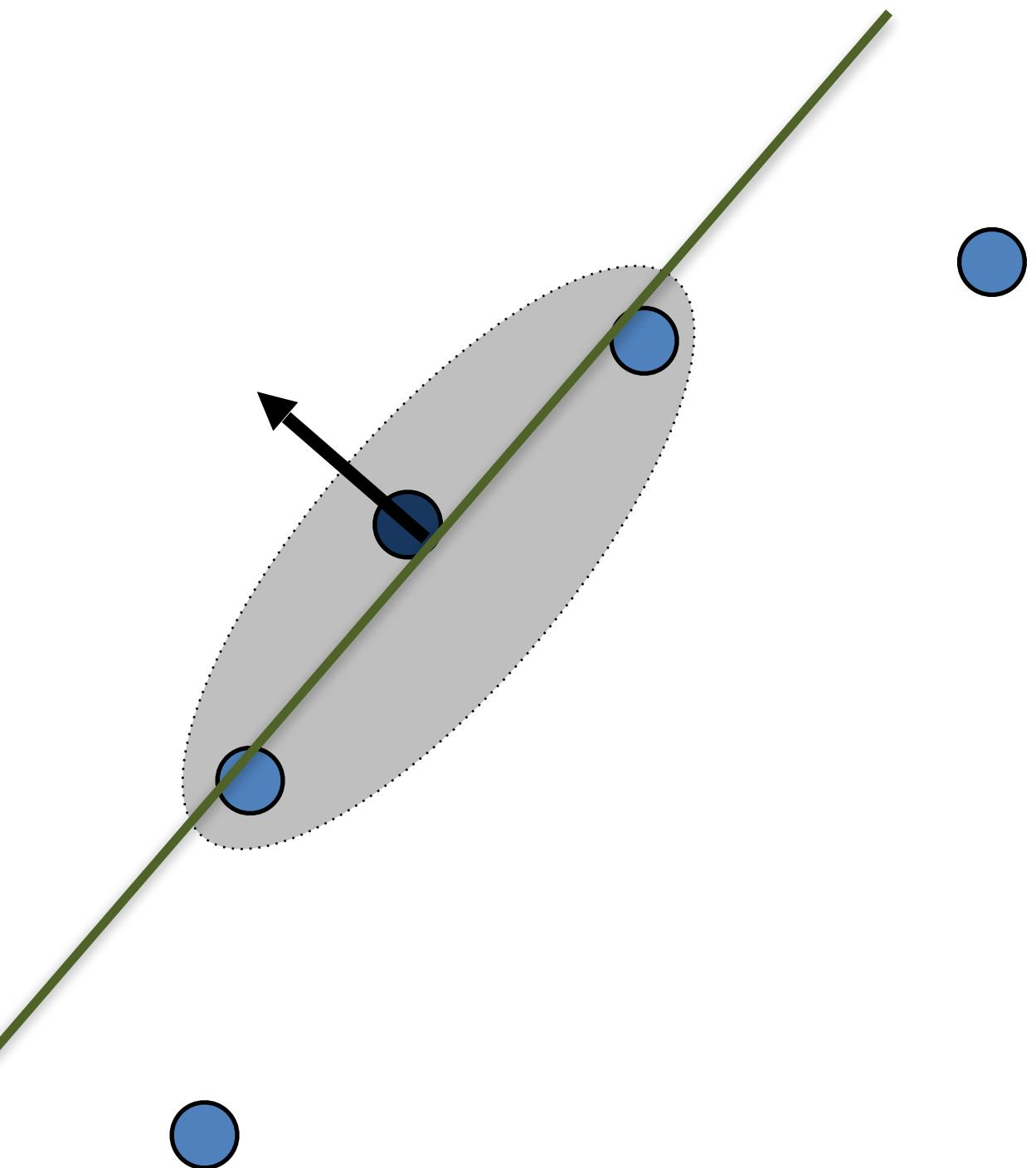
Normal Estimation

- 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)



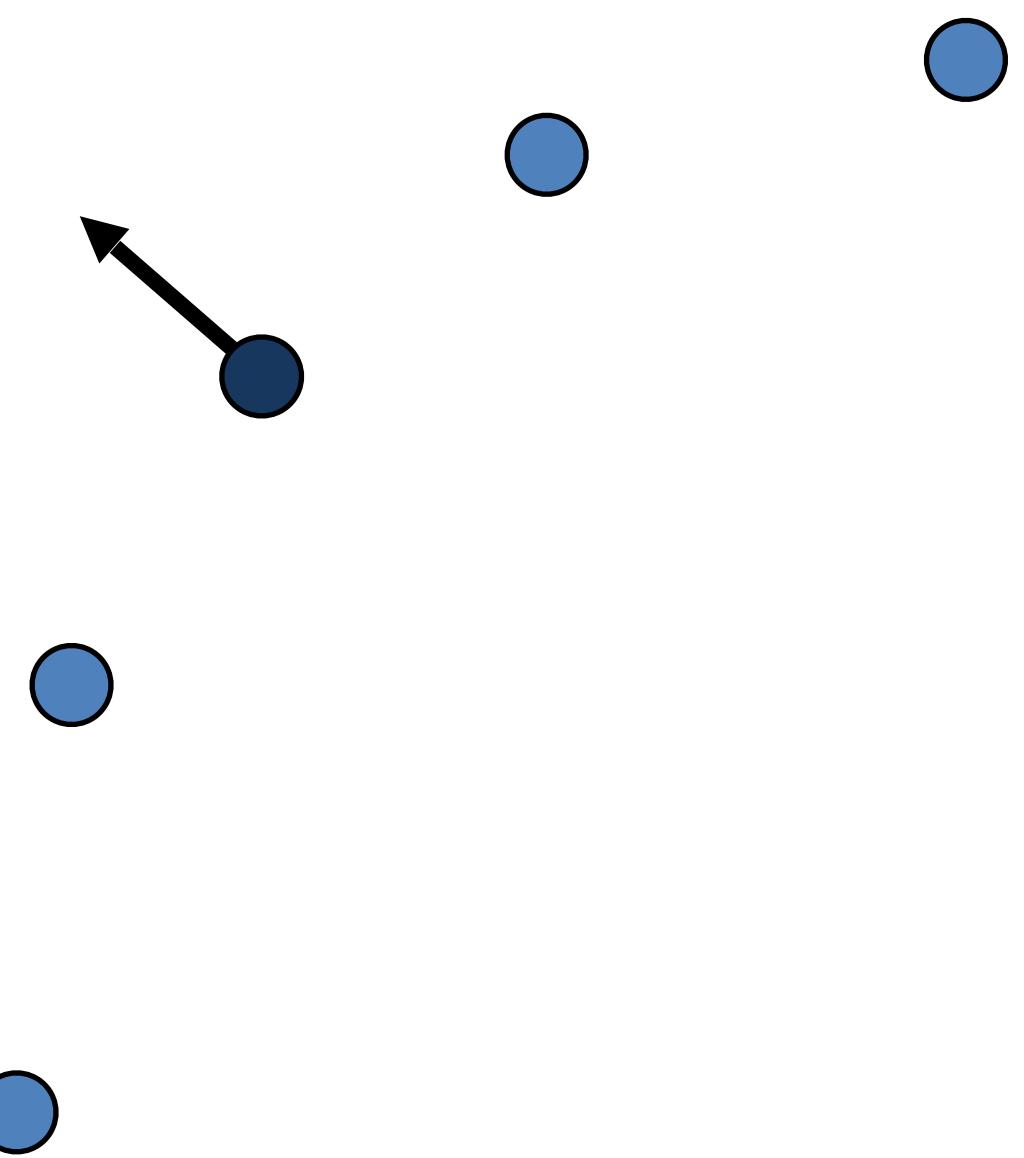
Normal Estimation

- 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)



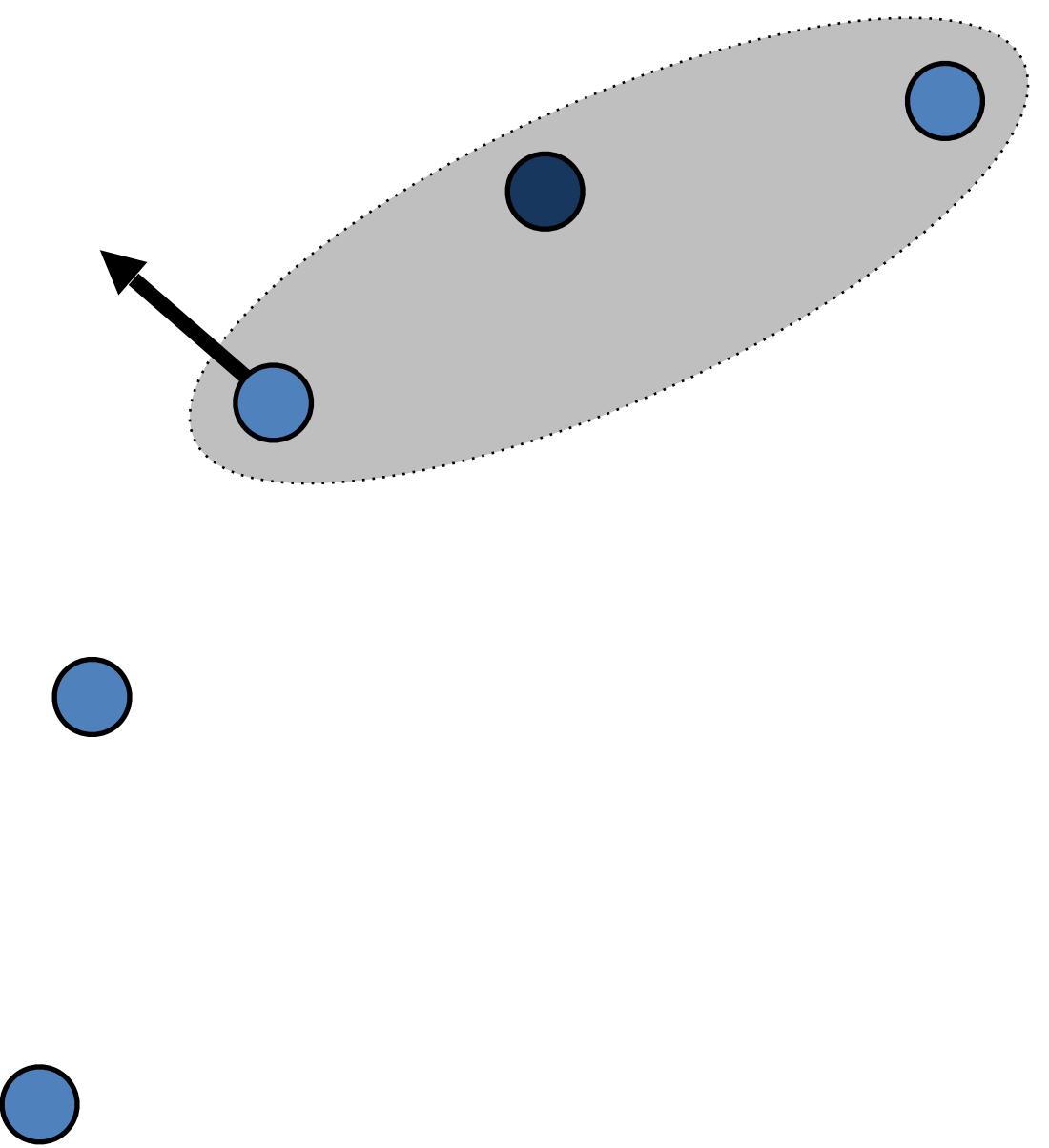
Normal Estimation

- 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)



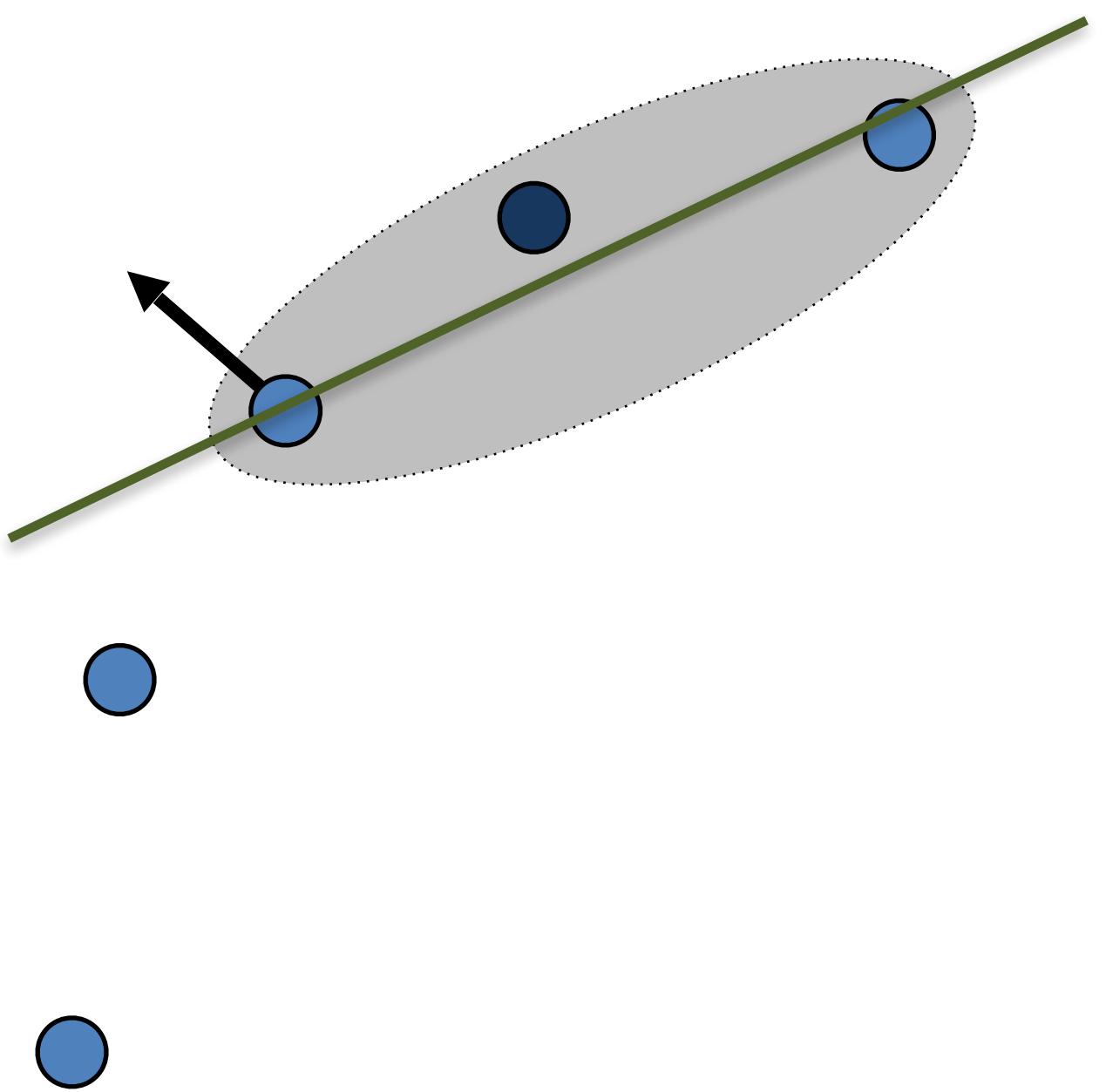
Normal Estimation

- 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)



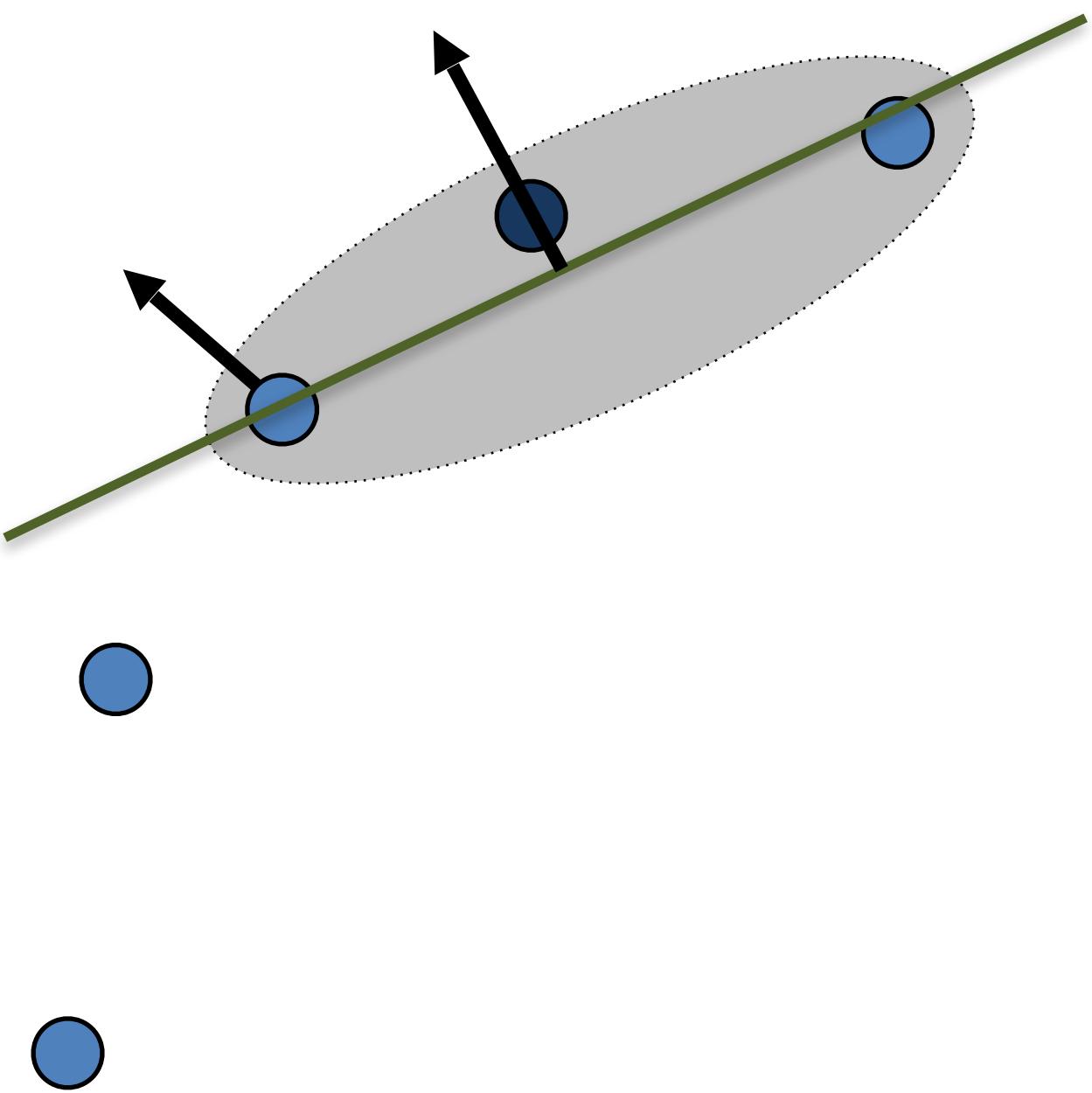
Normal Estimation

- 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)



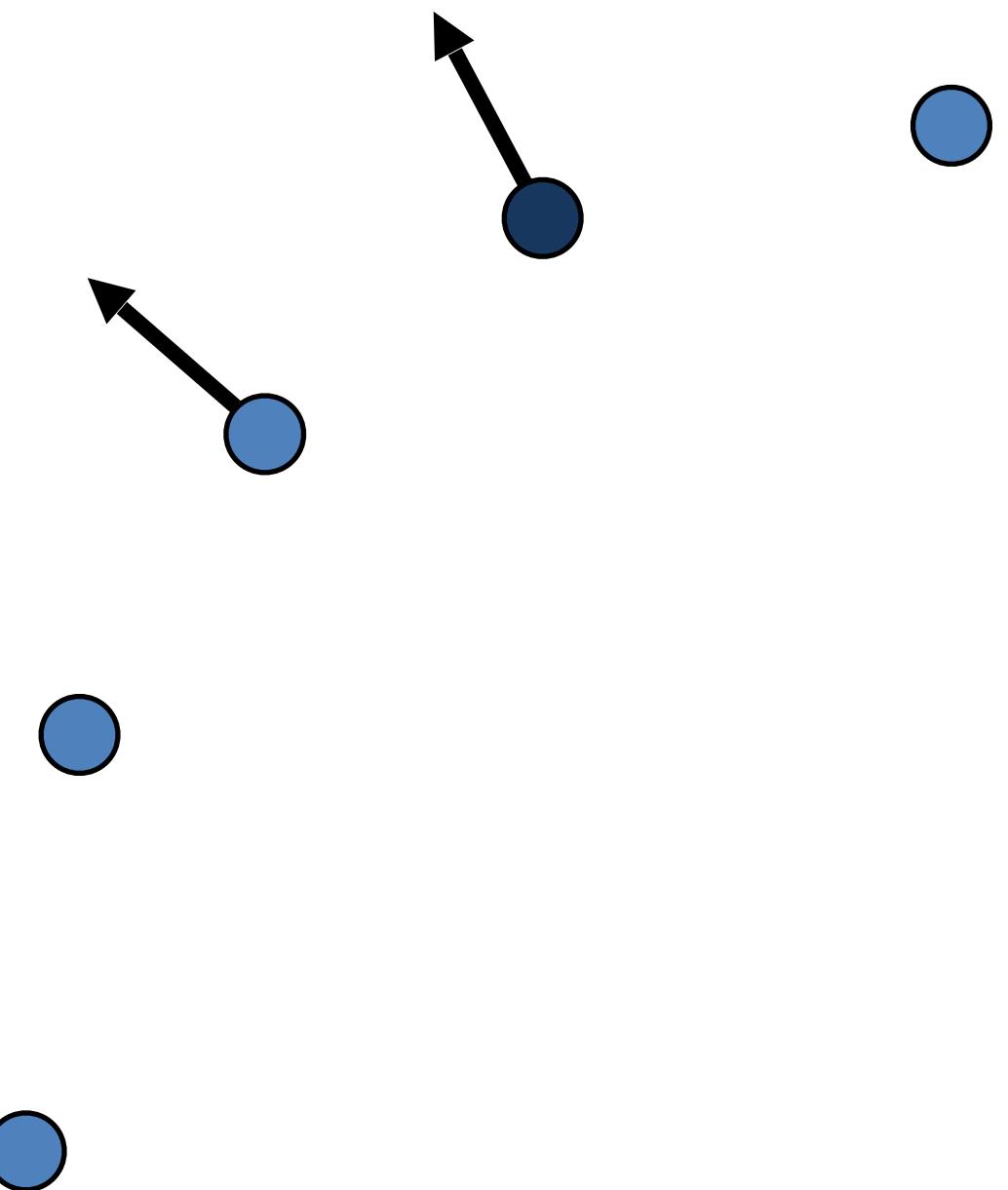
Normal Estimation

- 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)



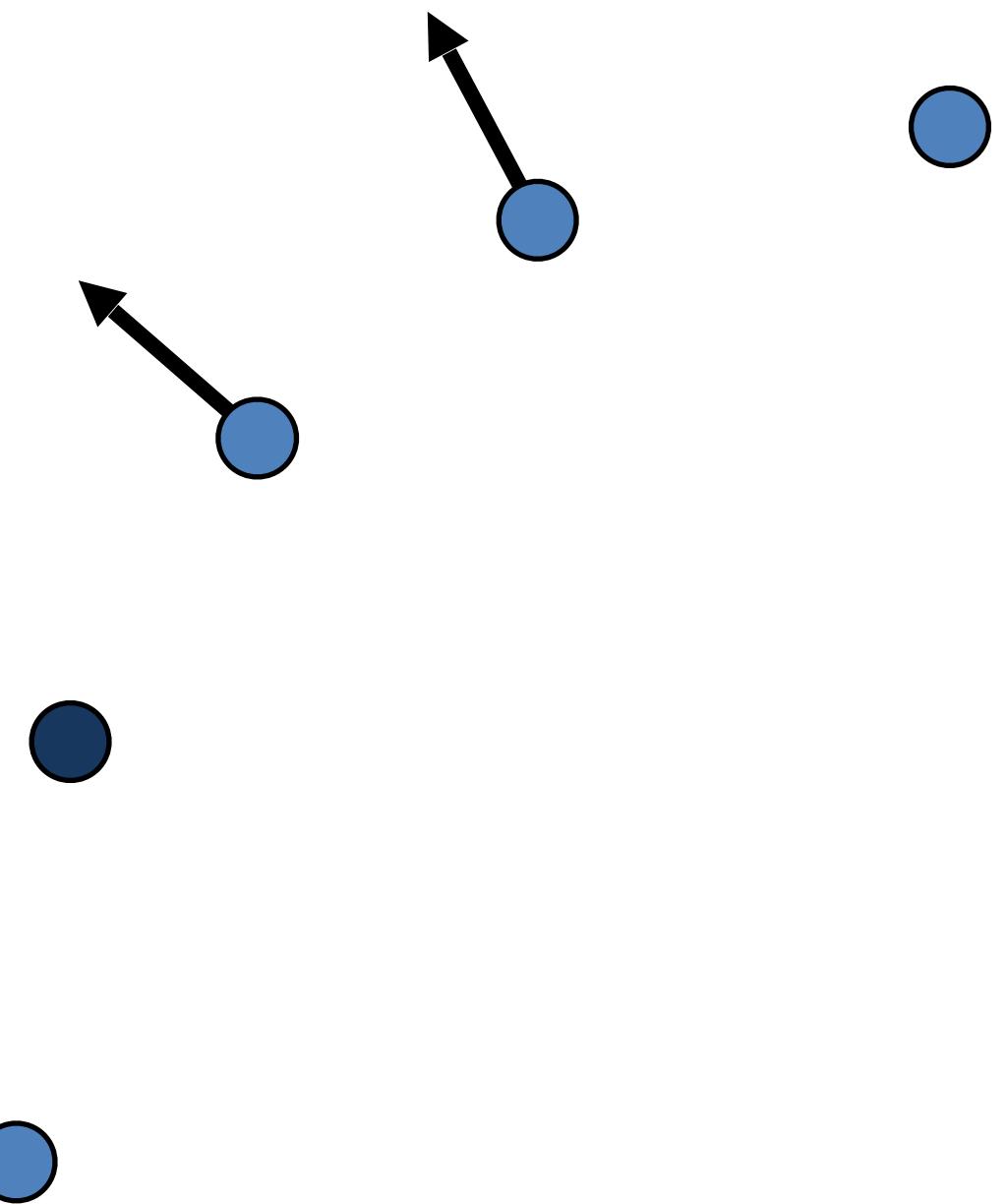
Normal Estimation

- 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)



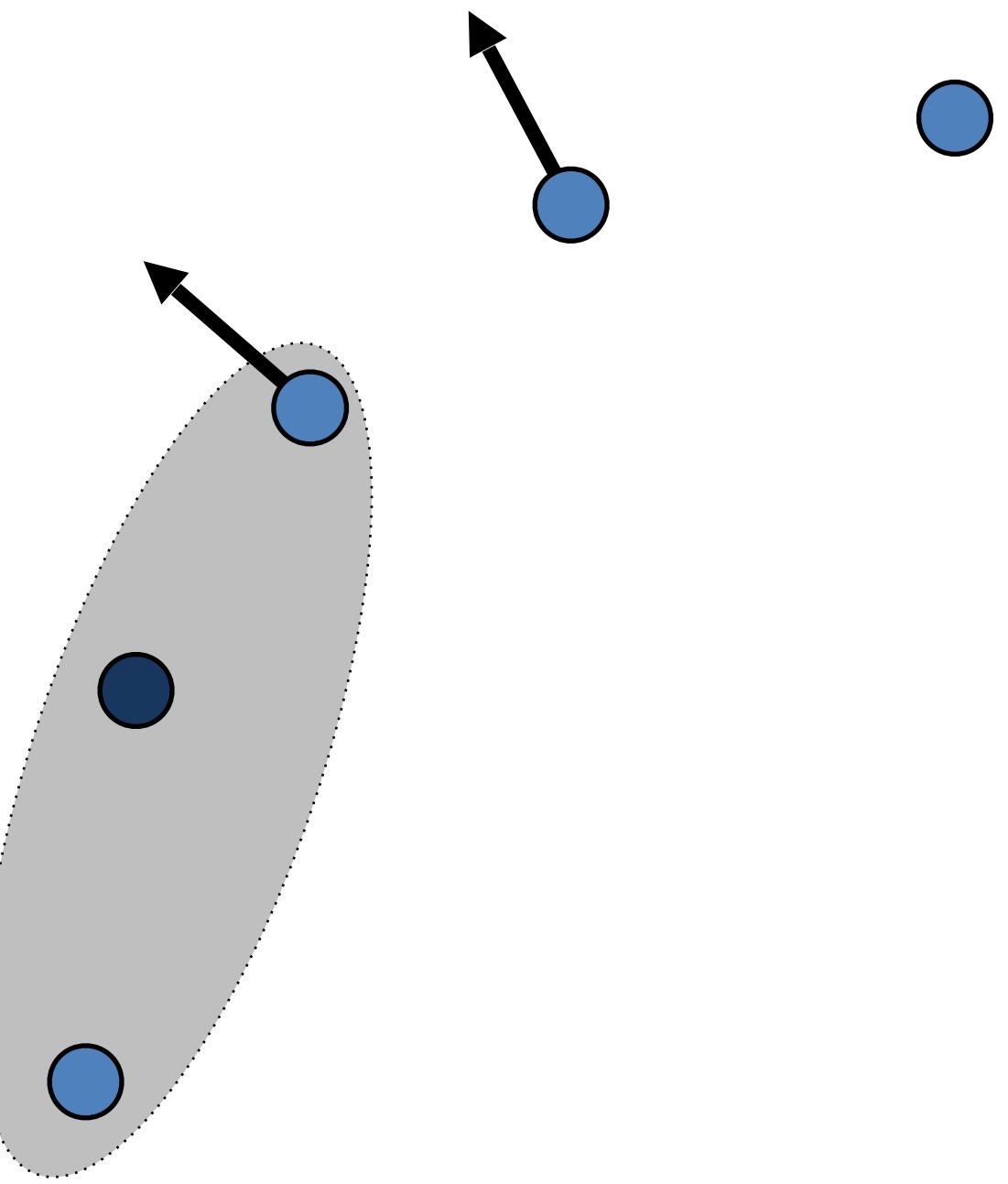
Normal Estimation

- 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)



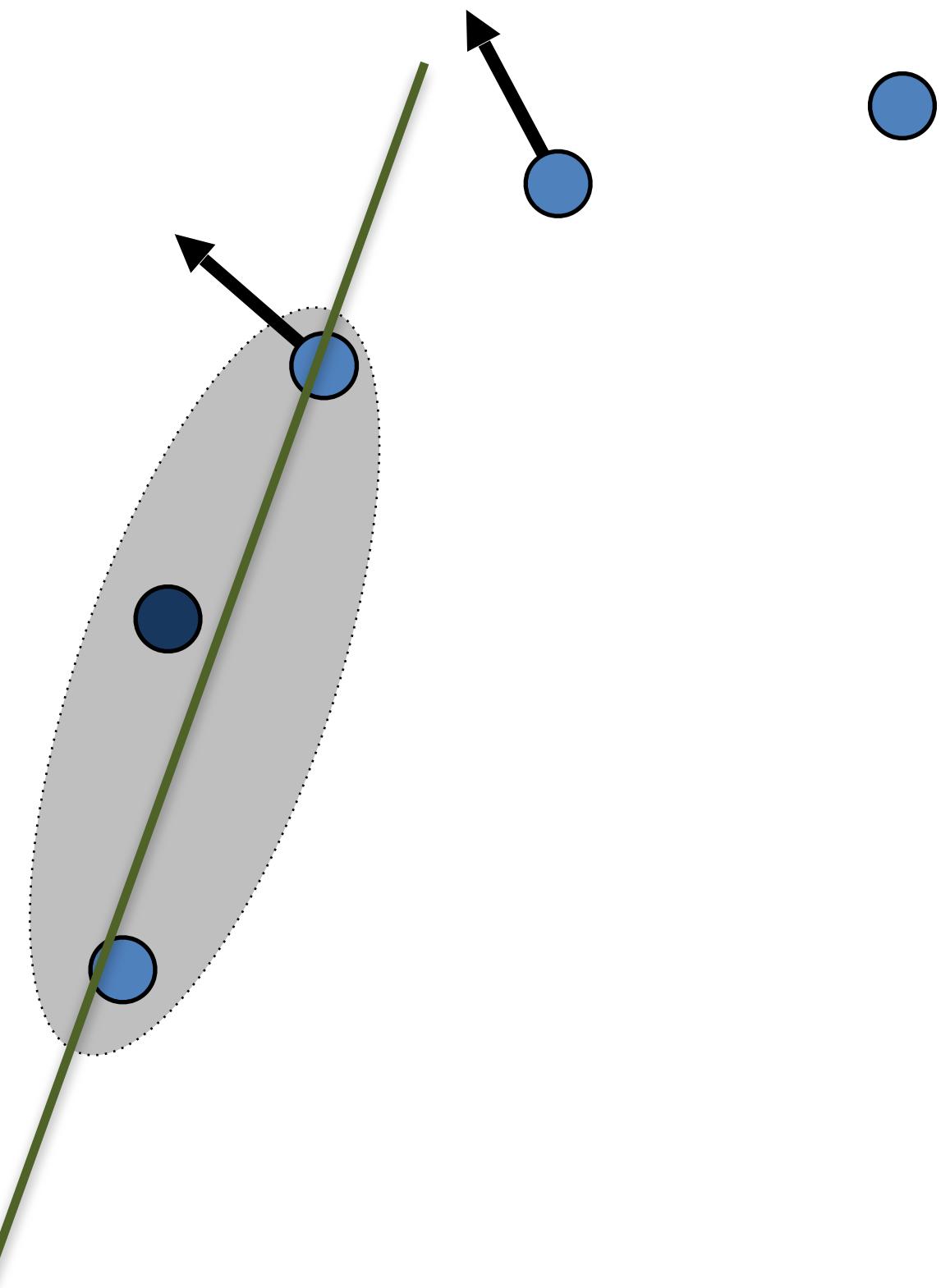
Normal Estimation

- 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)



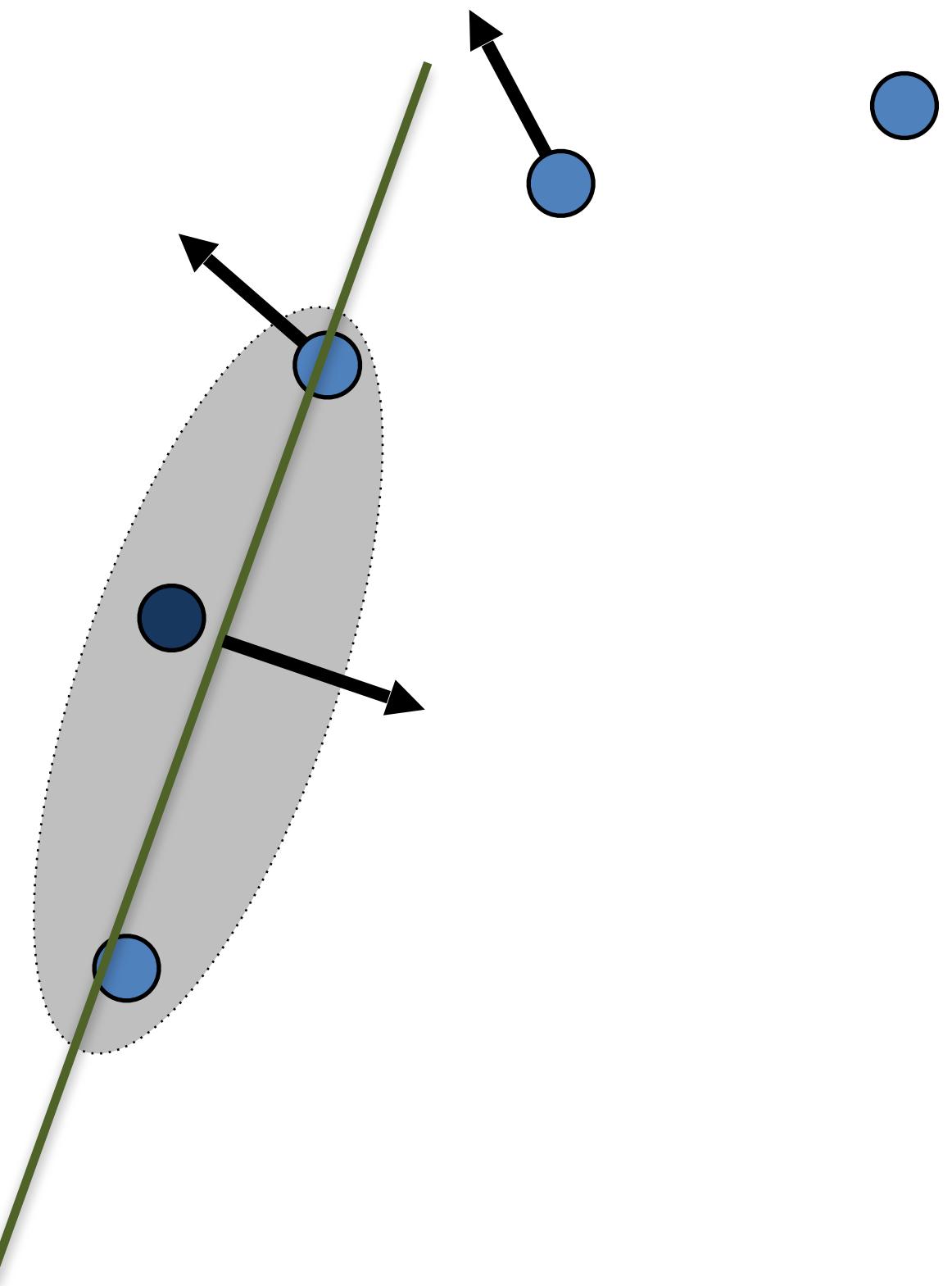
Normal Estimation

- 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)



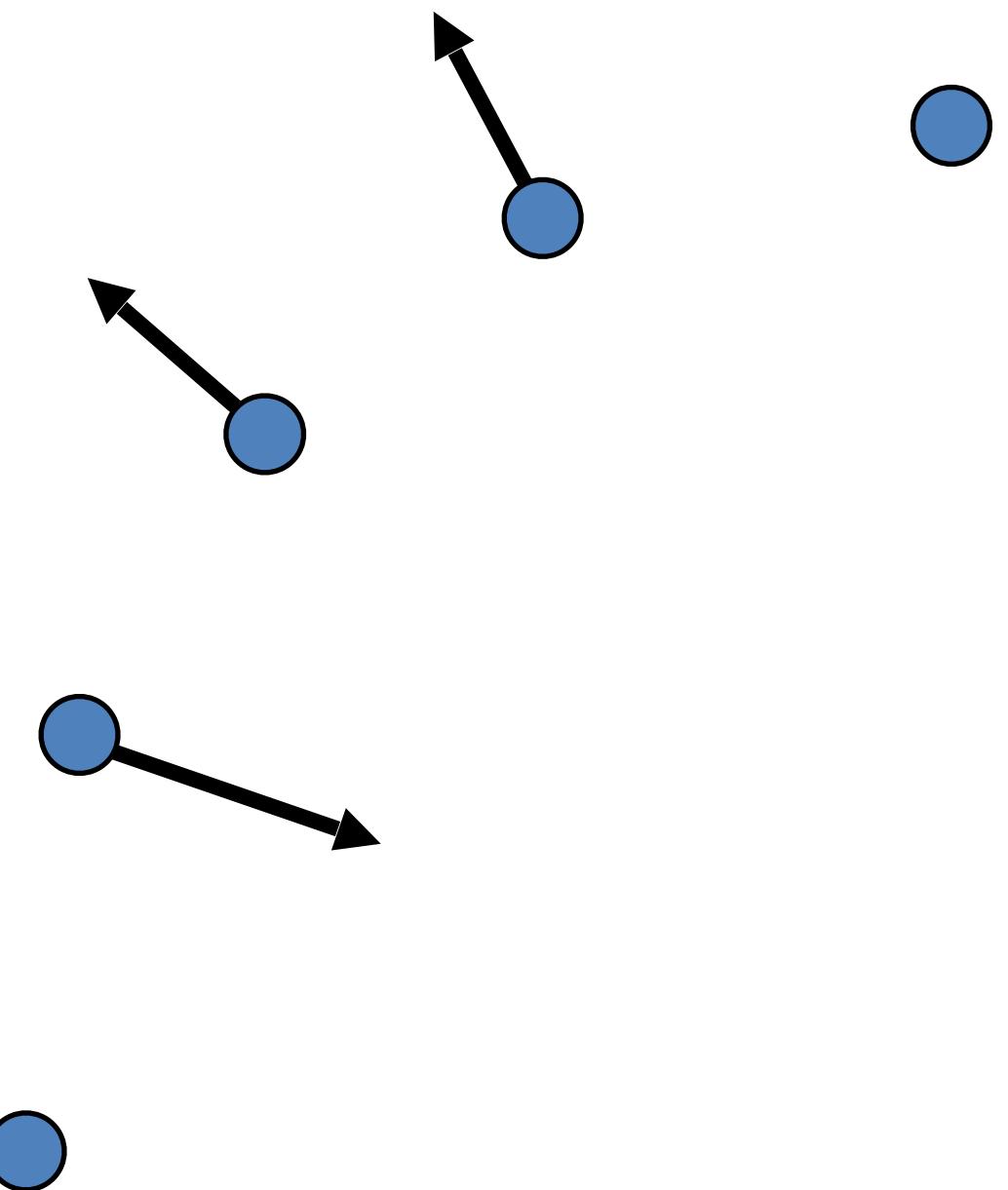
Normal Estimation

- 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)



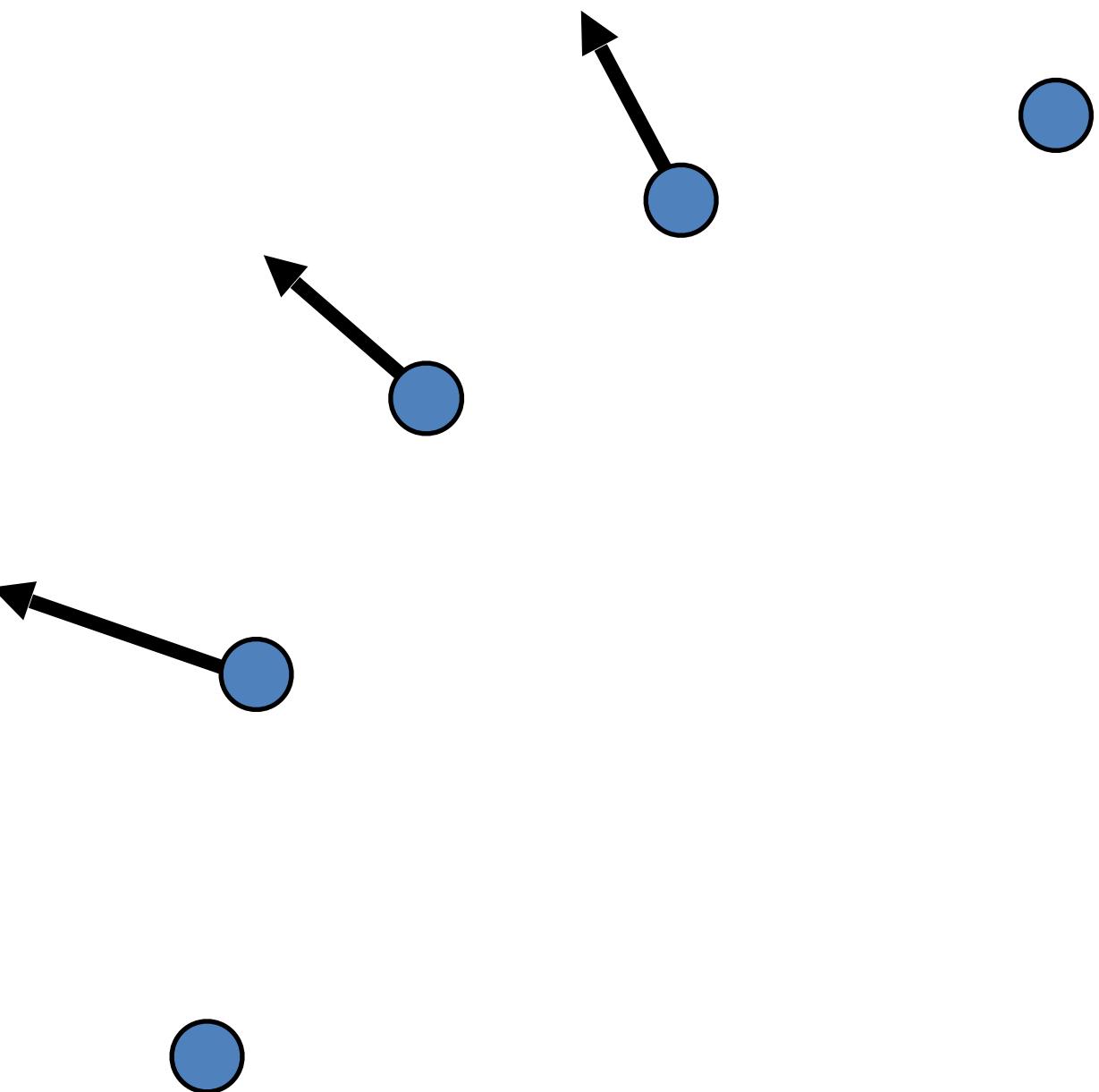
Normal Estimation

- 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)



Normal Estimation

- 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)



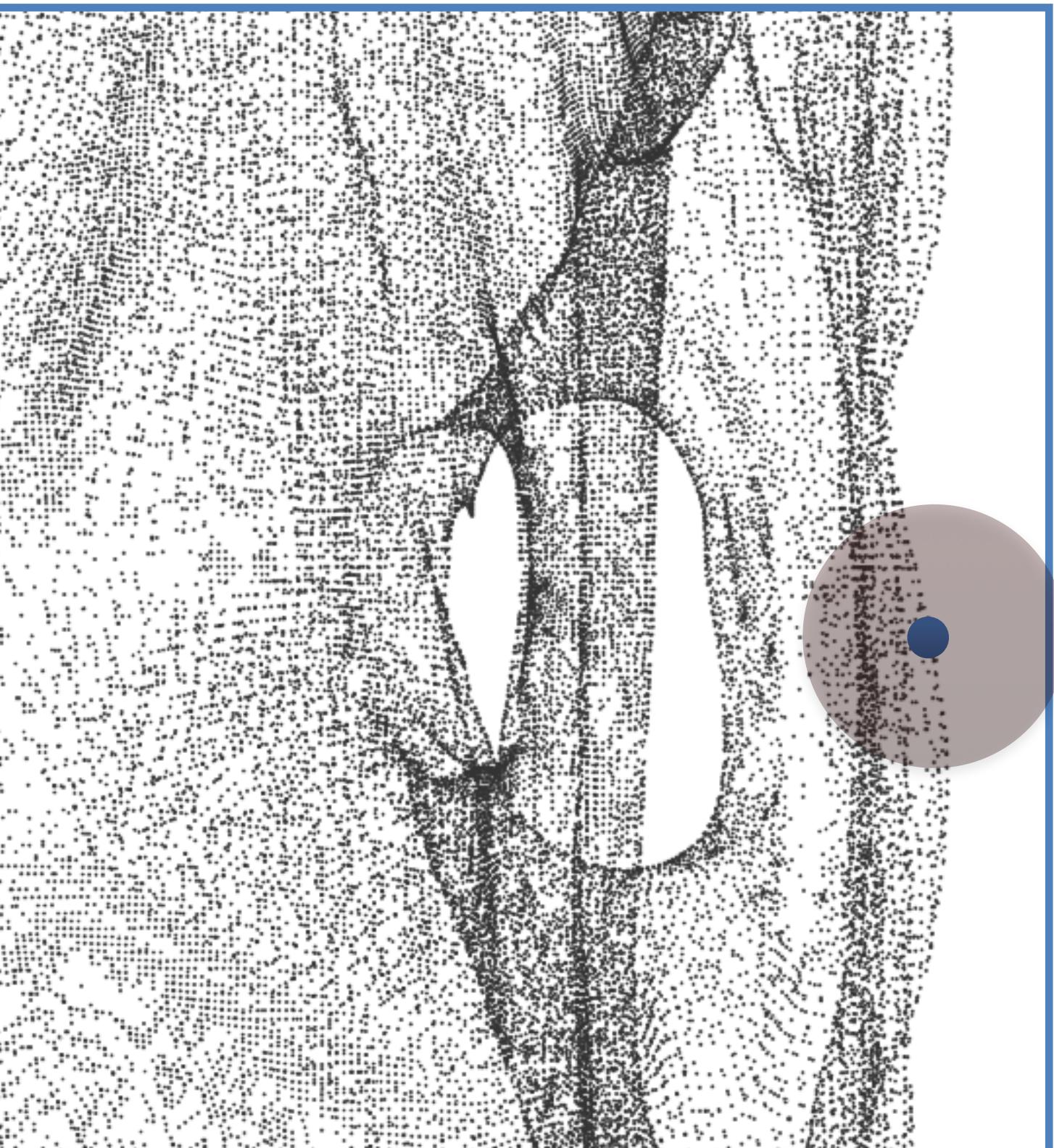
Local Plane Fitting

- For each point \mathbf{x} in the cloud, pick k nearest neighbors or all points in r -ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

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

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

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



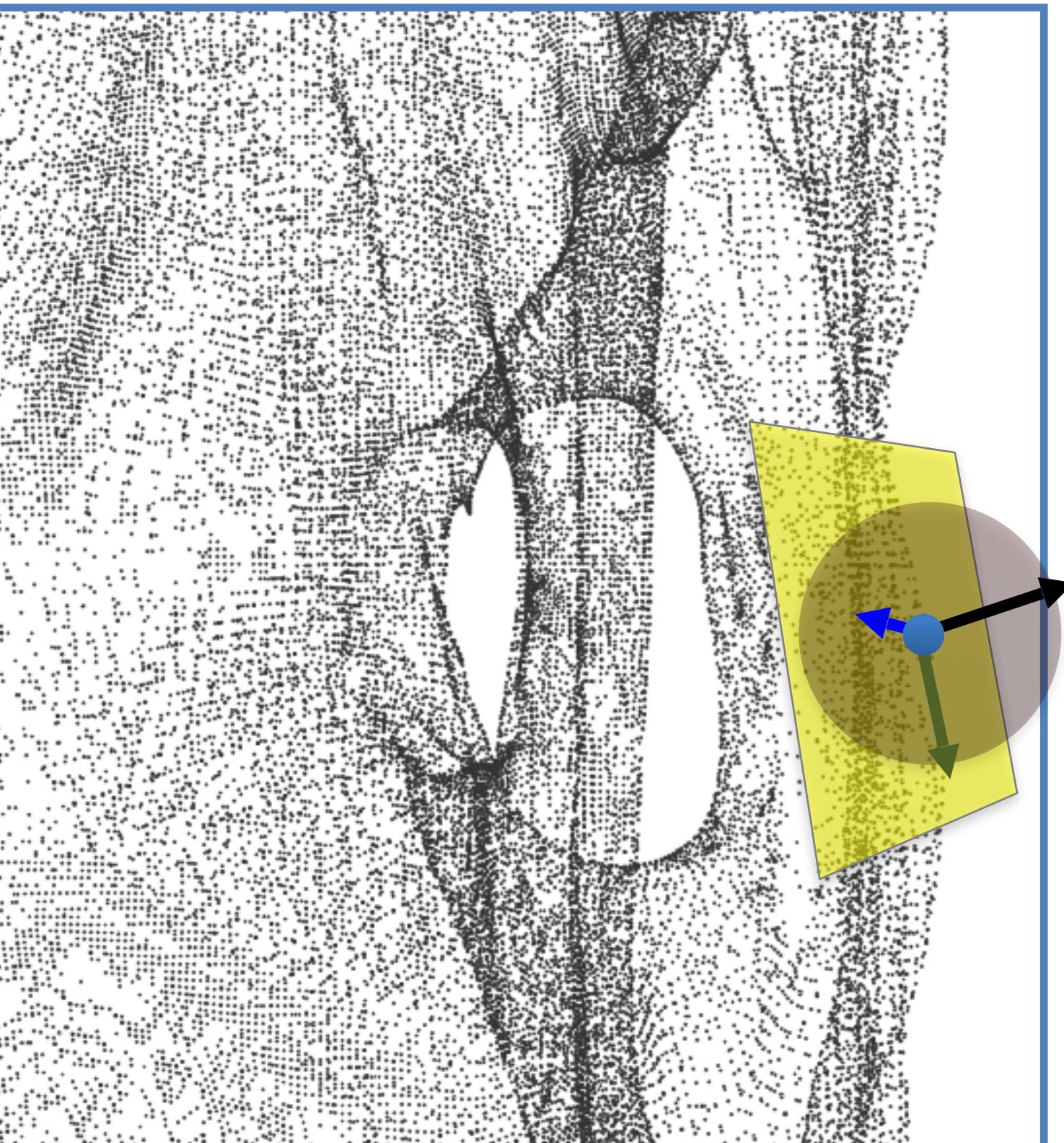
Local Plane Fitting

- For each point \mathbf{x} in the cloud, pick k nearest neighbors or all points in r -ball: $\{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < r\}$

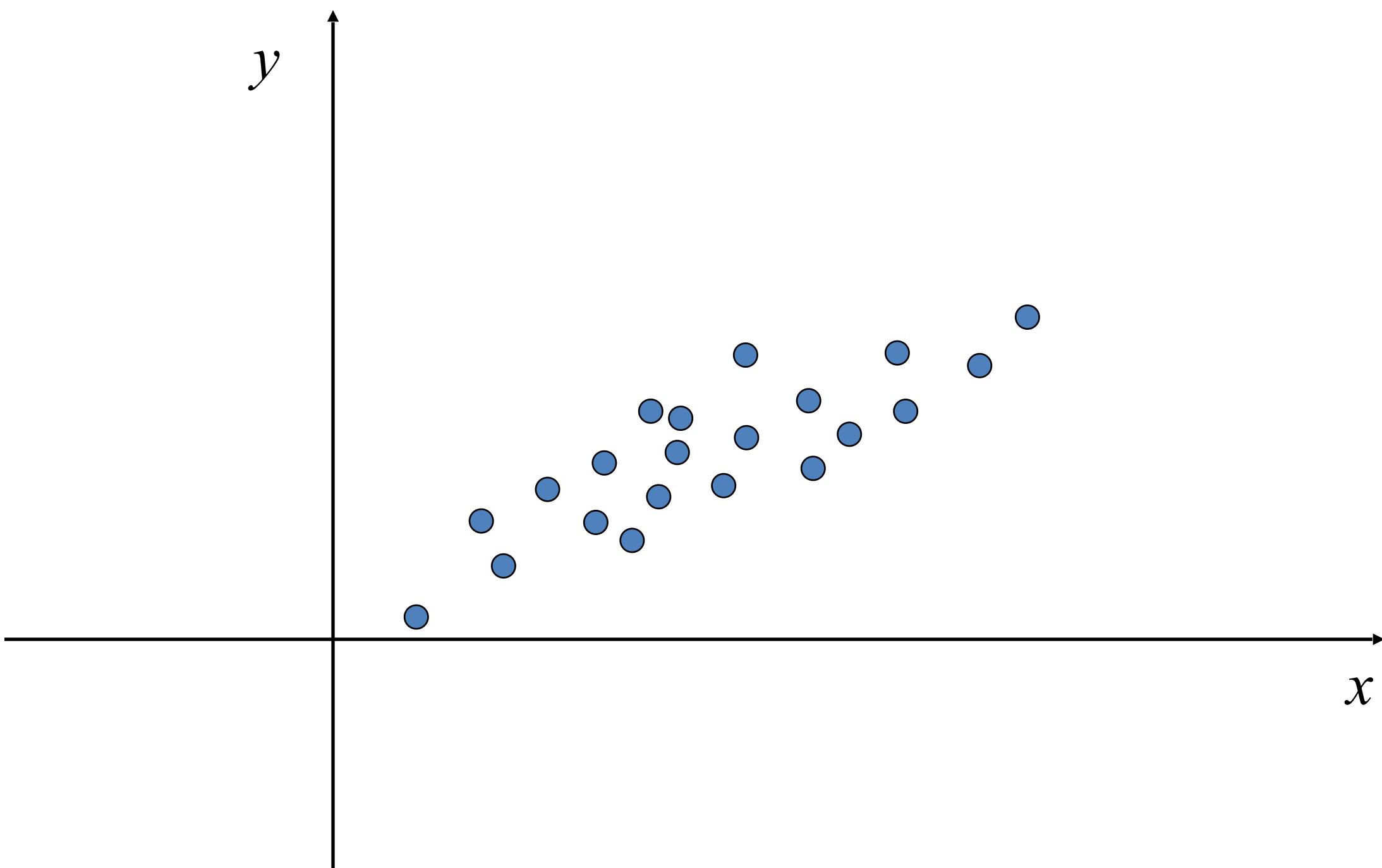
$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$$

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

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

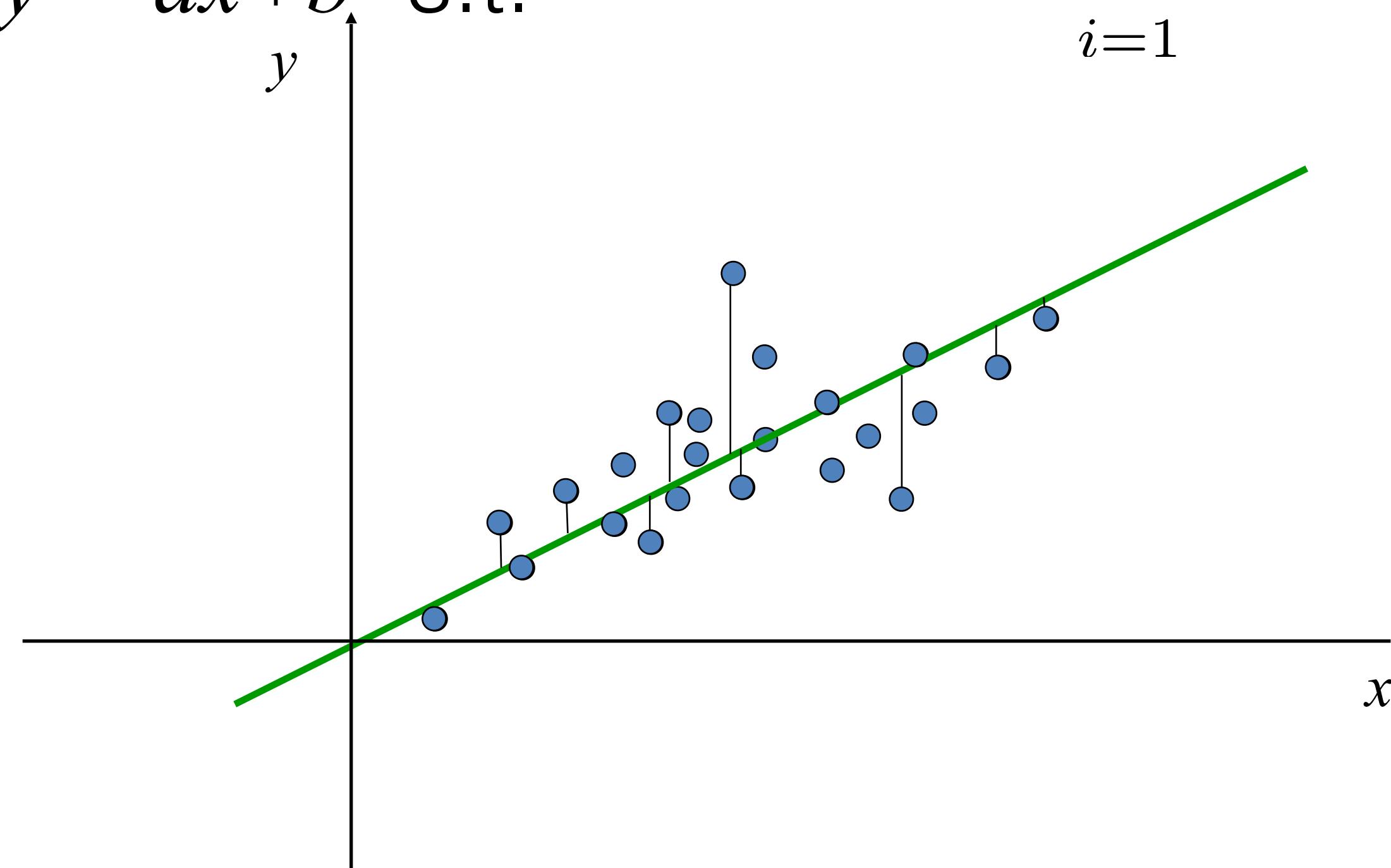


Linear Least Squares?



Linear Least Squares?

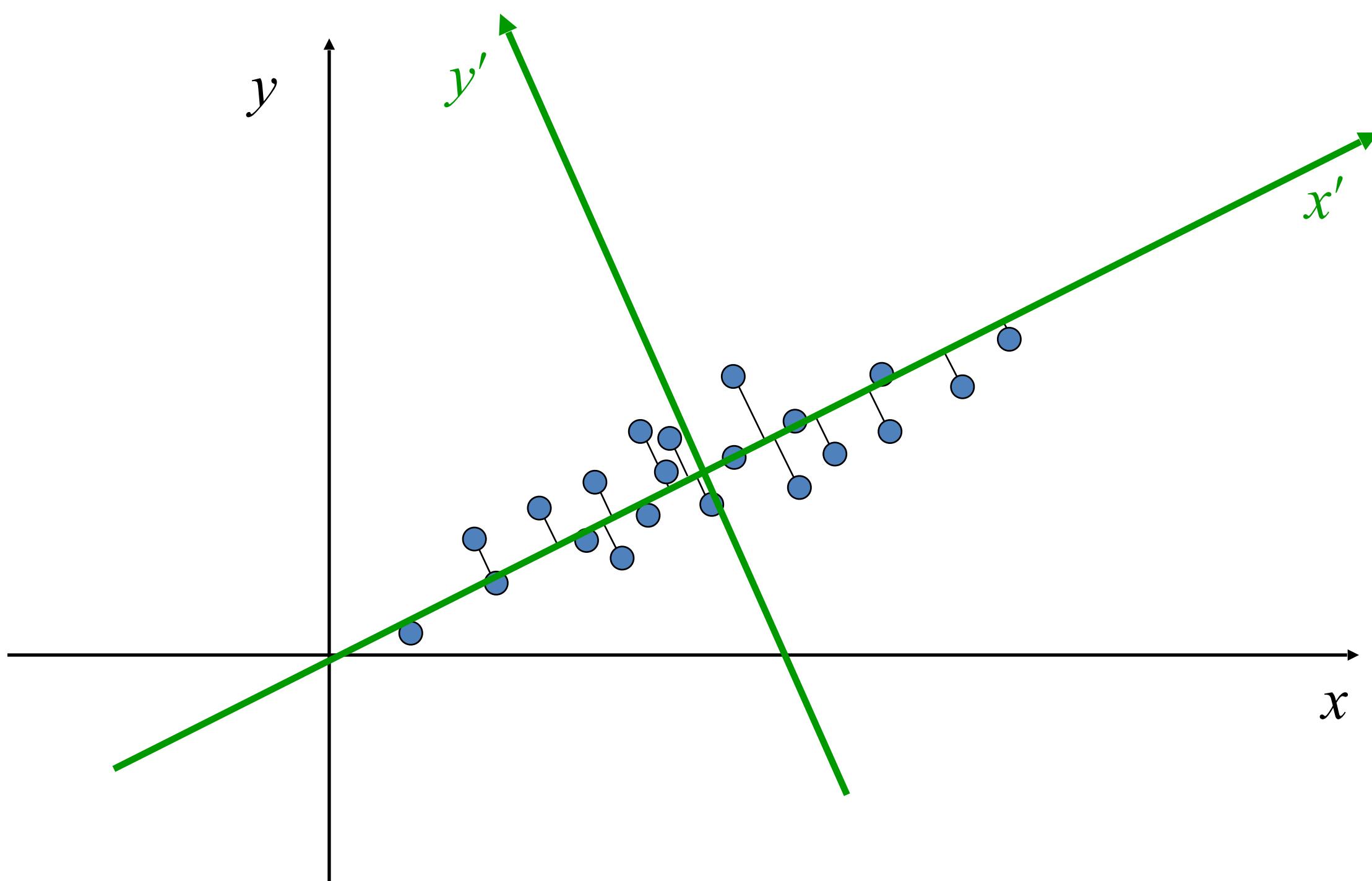
- Find a line $y = ax + b$ s.t.



$$\min \sum_{i=1}^n (y_i - (ax_i + b))^2$$

- But we would like true orthogonal distances!

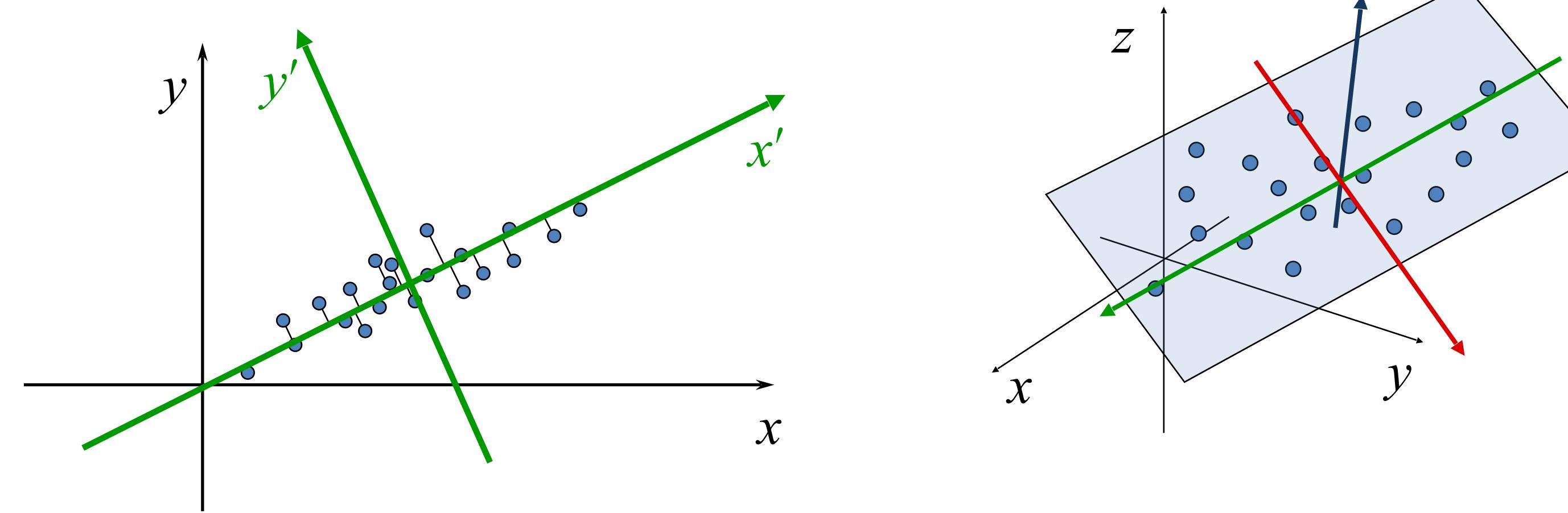
Best Fit with SSD



SSD = sum of squared distances (or differences)

Principle Component Analysis (PCA)

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



- PCA finds the best approximating line/plane/orientation... (in terms of $\sum distances^2$)

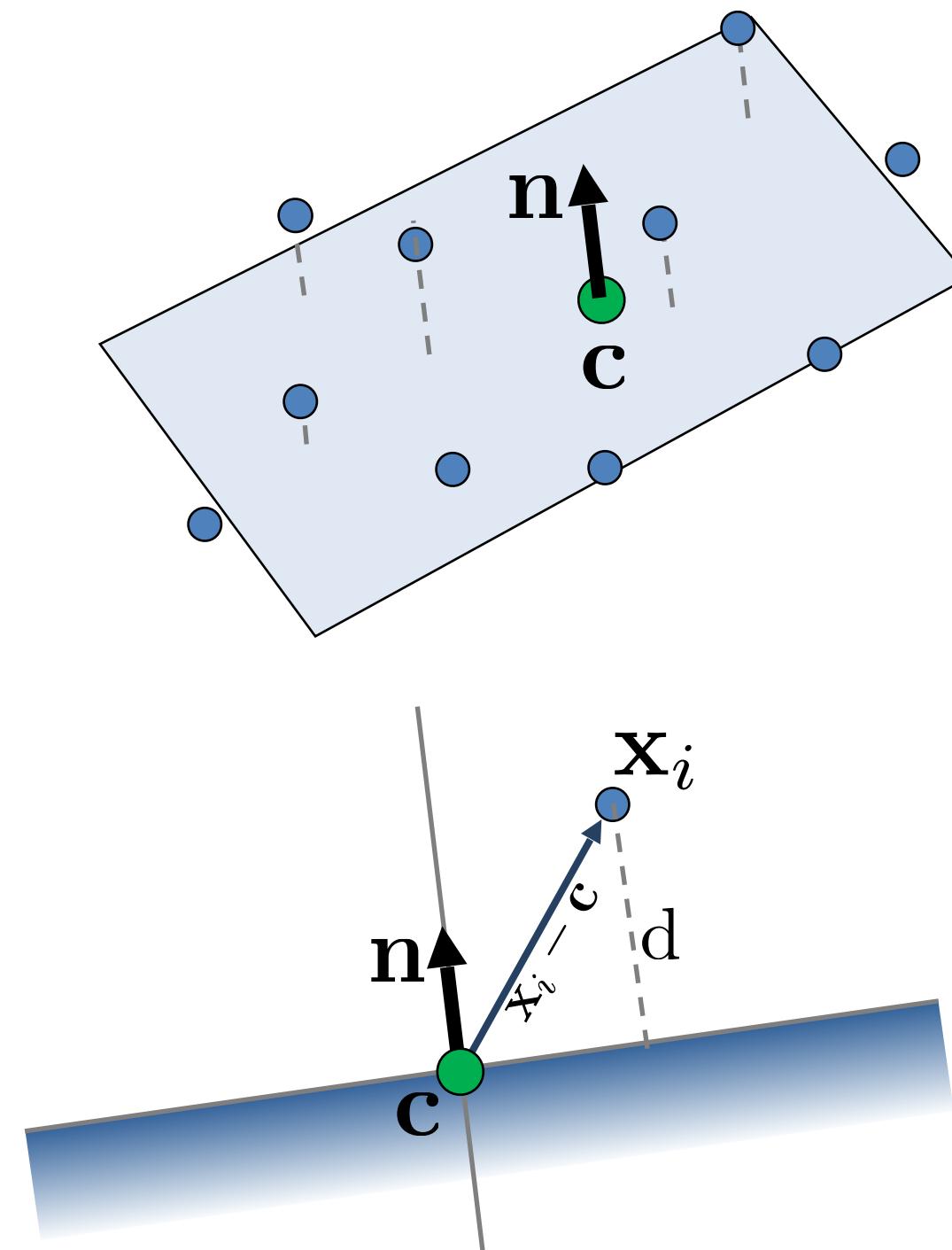
Notations

- Input points:

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$$

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

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



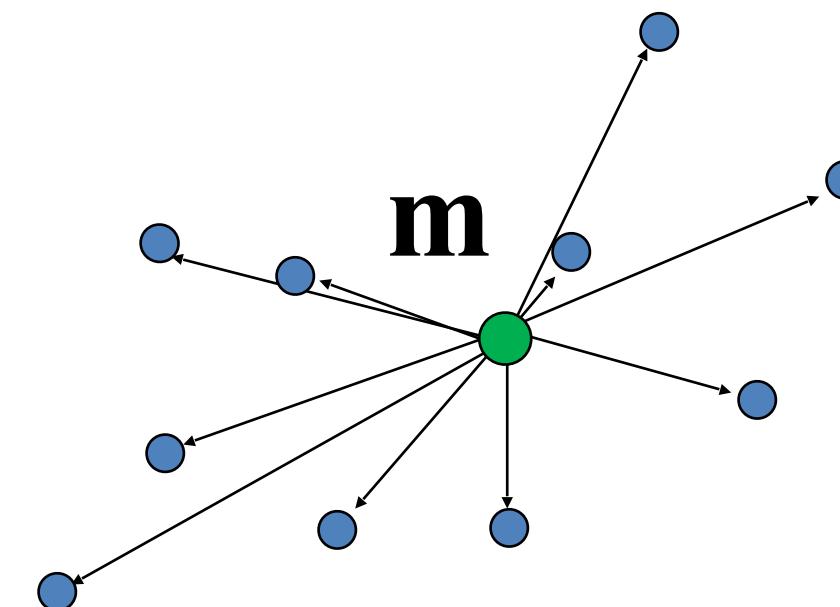
Notations

- Input points:

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$$

- Centroid:

$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$



- Vectors from the centroid:

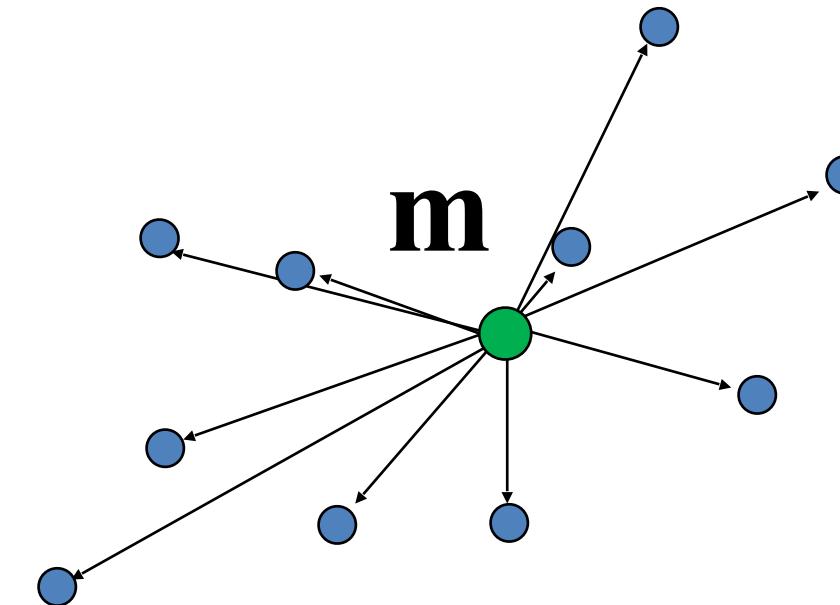
$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$

Centroid: 0-dim Approximation

- It can be shown that:

$$\mathbf{m} = \operatorname{argmin}_{\mathbf{c}} \sum_{i=1}^n ((\mathbf{x}_i - \mathbf{c})^T \mathbf{n})^2$$

$$\mathbf{m} = \operatorname{argmin}_{\mathbf{c}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{c}\|^2$$



$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

- **m** minimizes SSD
- **m** will be the origin of the (hyper)-plane
- Our problem becomes:

$$\min_{\|\mathbf{n}\|=1} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{n})^2$$

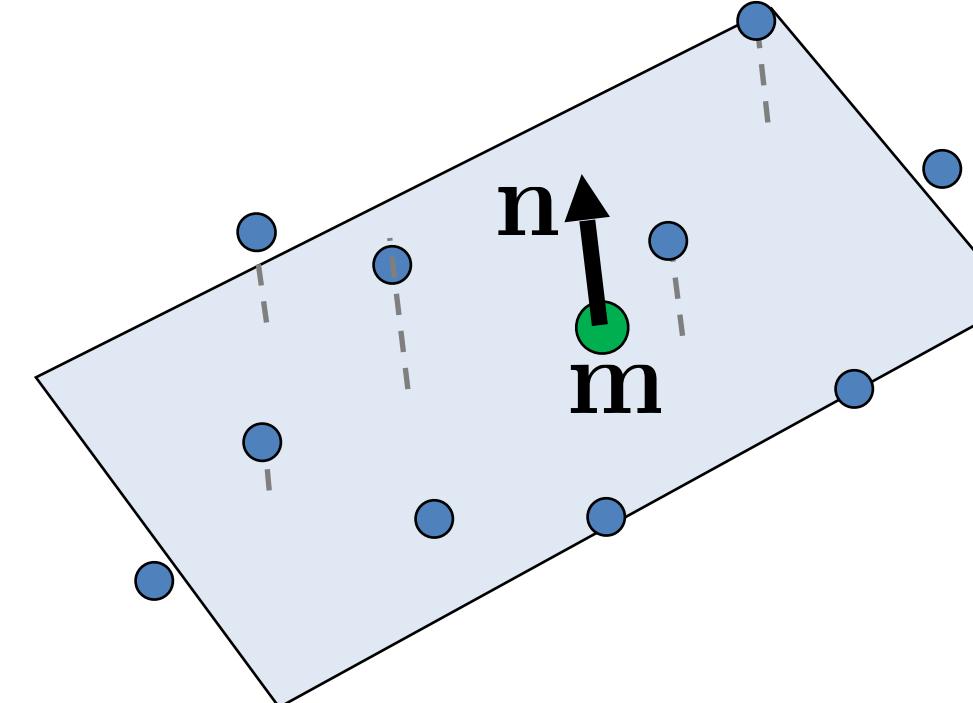
Hyperplane Normal

- Minimize!

$$\begin{aligned} \min_{\mathbf{n}^T \mathbf{n} = 1} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{n})^2 &= \min_{\mathbf{n}^T \mathbf{n} = 1} \sum_{i=1}^n \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} = \\ \min_{\mathbf{n}^T \mathbf{n} = 1} \mathbf{n}^T \left(\sum_{i=1}^n \mathbf{y}_i \mathbf{y}_i^T \right) \mathbf{n} &= \min_{\mathbf{n}^T \mathbf{n} = 1} \mathbf{n}^T (\mathbf{Y} \mathbf{Y}^T) \mathbf{n} \end{aligned}$$

$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$





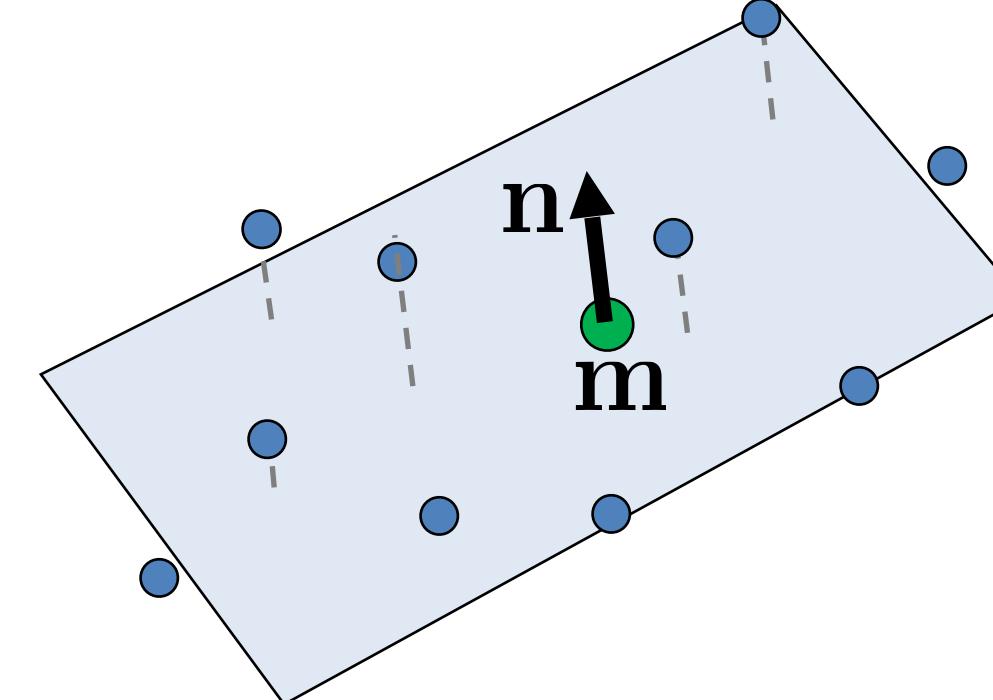
Hyperplane Normal

- Minimize!

$$\begin{aligned} \min_{\mathbf{n}^T \mathbf{n} = 1} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{n})^2 &= \min_{\mathbf{n}^T \mathbf{n} = 1} \sum_{i=1}^n \mathbf{n}^T \mathbf{y}_i \mathbf{y}_i^T \mathbf{n} = \\ \min_{\mathbf{n}^T \mathbf{n} = 1} \mathbf{n}^T \left(\sum_{i=1}^n \mathbf{y}_i \mathbf{y}_i^T \right) \mathbf{n} &= \min_{\mathbf{n}^T \mathbf{n} = 1} \mathbf{n}^T (\mathbf{Y} \mathbf{Y}^T) \mathbf{n} \end{aligned}$$

$\mathbf{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \\ | & | & & | \end{pmatrix}$

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \quad (\mathbf{S} = \mathbf{Y} \mathbf{Y}^T)$$
$$\min f(\mathbf{n}) \quad s.t. \quad \mathbf{n}^T \mathbf{n} = 1$$



Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \quad (\mathbf{S} = \mathbf{Y} \mathbf{Y}^T)$$

$$\min f(\mathbf{n}) \quad s.t. \quad \mathbf{n}^T \mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$

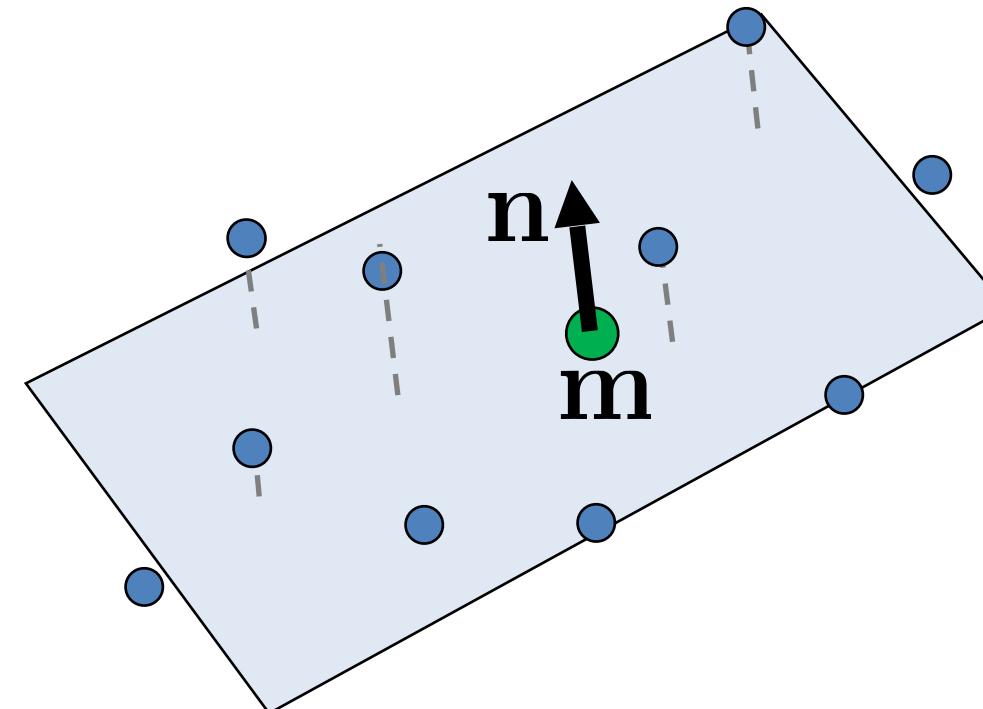
$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = \frac{\partial}{\partial \mathbf{n}} f(\mathbf{n}) - \lambda \frac{\partial}{\partial \mathbf{n}} (\mathbf{n}^T \mathbf{n} - 1)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \mathbf{n}^T \mathbf{n} - 1$$

Matrix Cookbook!
<https://archive.org/details/imm3274>

$$\frac{\partial}{\partial \mathbf{n}} f(\mathbf{n}) - \lambda \frac{\partial}{\partial \mathbf{n}} (\mathbf{n}^T \mathbf{n} - 1) = (\mathbf{S} + \mathbf{S}^T) \mathbf{n} - \lambda (\mathbf{I} + \mathbf{I}^T) \mathbf{n} = 2\mathbf{S}\mathbf{n} - 2\lambda\mathbf{n}$$



Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \quad (\mathbf{S} = \mathbf{Y} \mathbf{Y}^T)$$

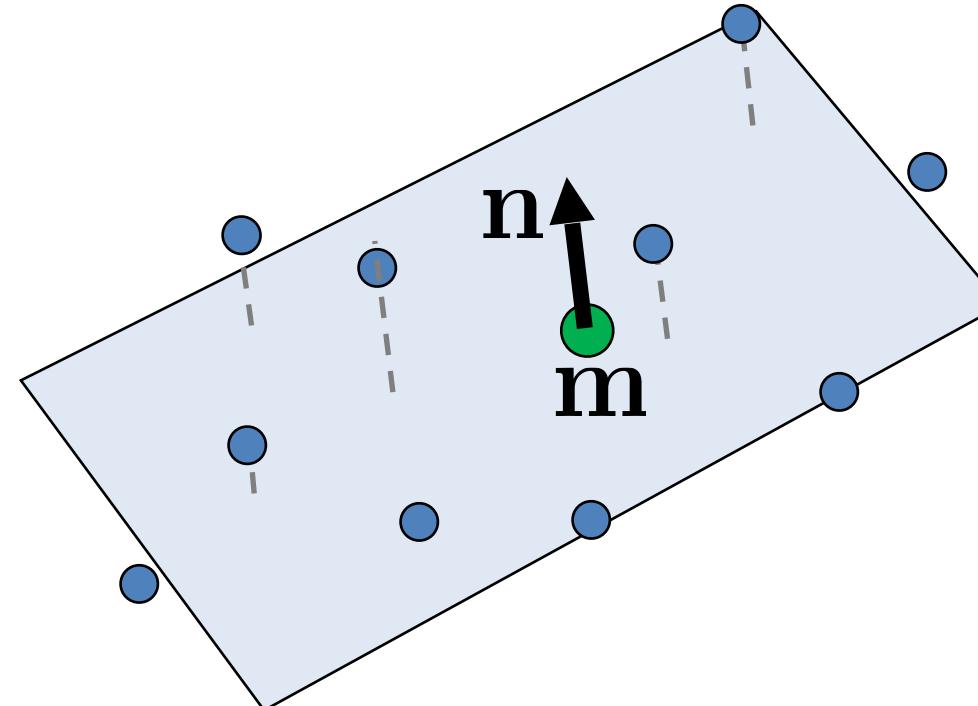
$$\min f(\mathbf{n}) \quad s.t. \quad \mathbf{n}^T \mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$

$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S} \mathbf{n} = \lambda \mathbf{n}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$



Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \quad (\mathbf{S} = \mathbf{Y} \mathbf{Y}^T)$$

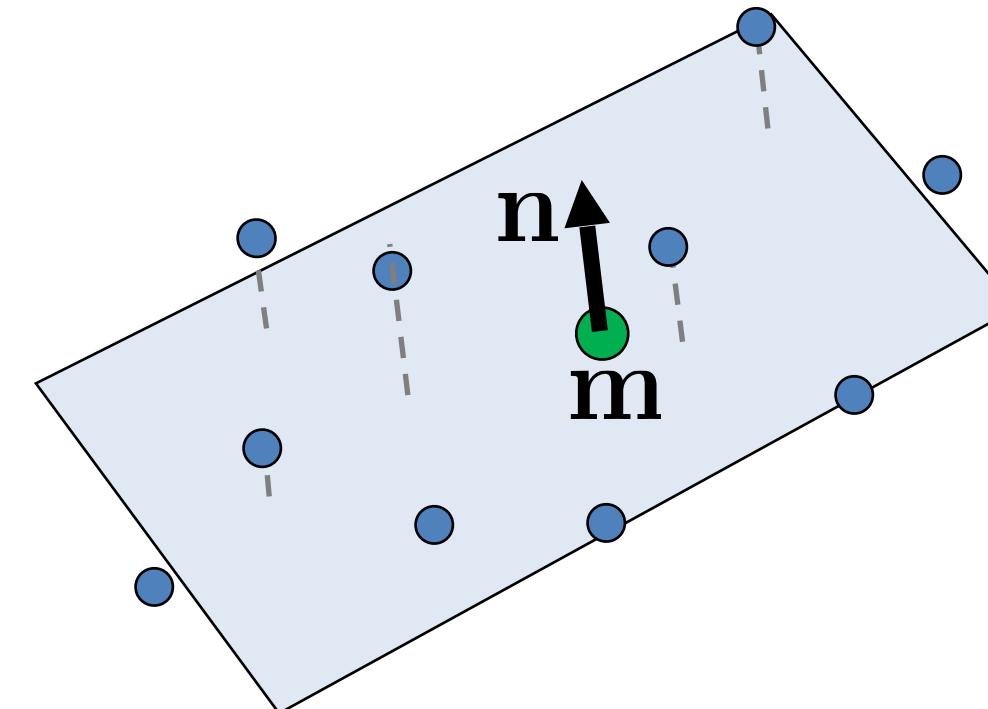
$$\min f(\mathbf{n}) \quad s.t. \quad \mathbf{n}^T \mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$

$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S} \mathbf{n} = \lambda \mathbf{n}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$



What can be said about \mathbf{n} ??

Hyperplane Normal

- Constrained minimization – Lagrange multipliers

$$f(\mathbf{n}) = \mathbf{n}^T \mathbf{S} \mathbf{n} \quad (\mathbf{S} = \mathbf{Y} \mathbf{Y}^T)$$

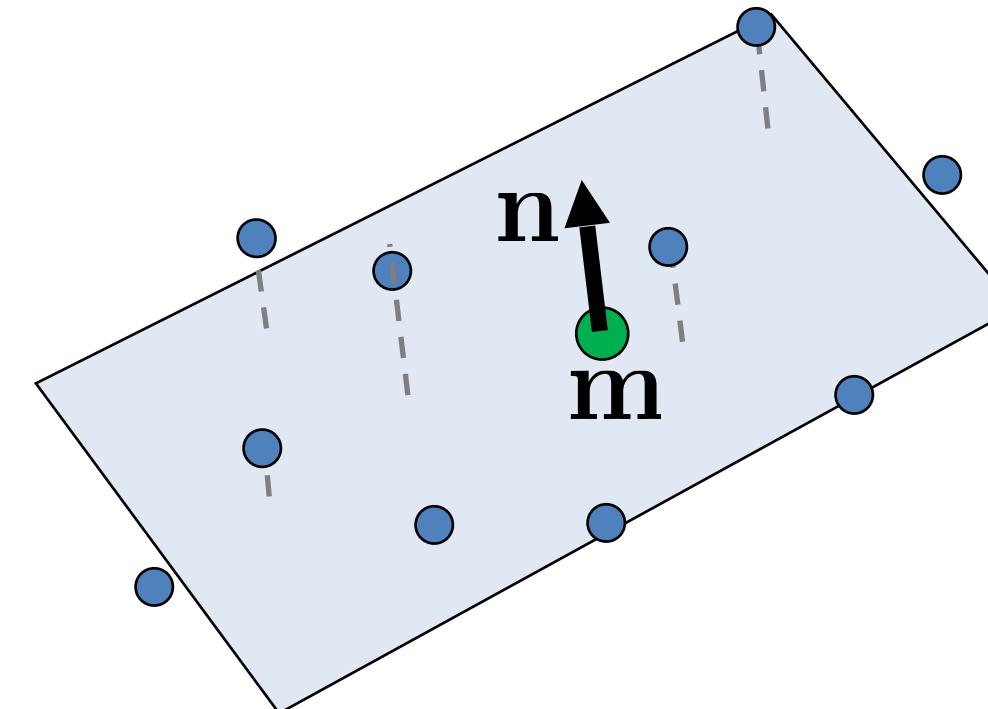
$$\min f(\mathbf{n}) \quad s.t. \quad \mathbf{n}^T \mathbf{n} = 1$$

$$\mathcal{L}(\mathbf{n}, \lambda) = f(\mathbf{n}) - \lambda(\mathbf{n}^T \mathbf{n} - 1)$$

$$\nabla \mathcal{L} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}} = 0 \iff \mathbf{S} \mathbf{n} = \lambda \mathbf{n}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff \mathbf{n}^T \mathbf{n} = 1$$

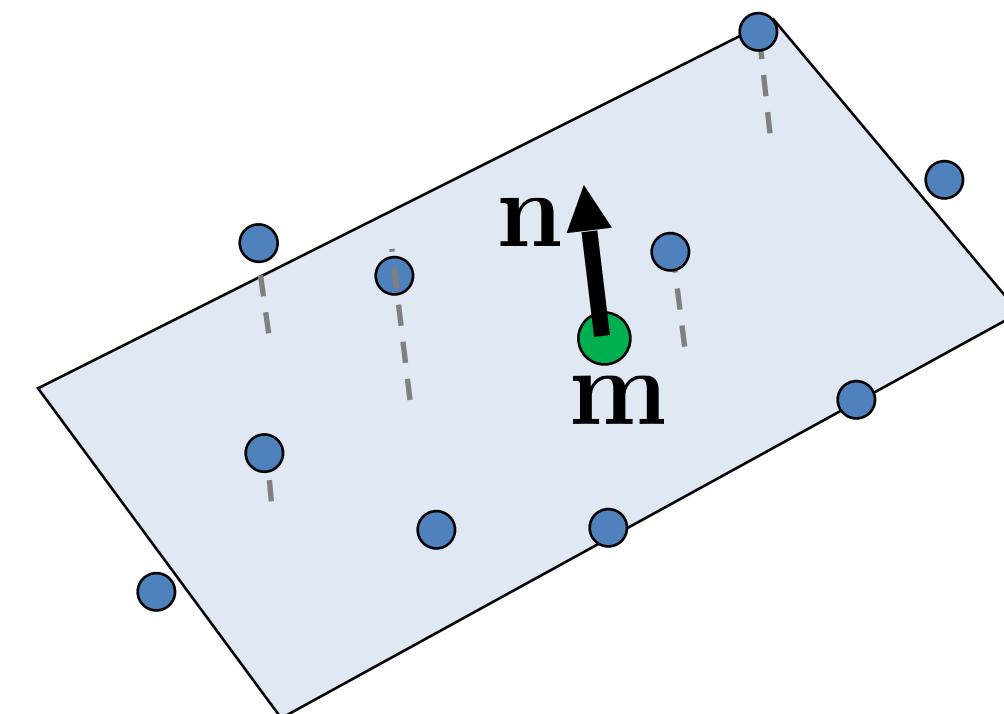


n is the eigenvector of **S** with the smallest eigenvalue

Summary – Best Fitting Plane Recipe

- Input: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d$
- Compute centroid = plane origin $\mathbf{m} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$
- Compute scatter matrix $\mathbf{S} = \mathbf{Y}\mathbf{Y}^T$
 $\mathbf{Y} = (\mathbf{y}_1 \ \mathbf{y}_2 \ \dots \ \mathbf{y}_n)$
 $\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$
- The plane normal \mathbf{n} is the eigenvector of \mathbf{S} with the smallest eigenvalue

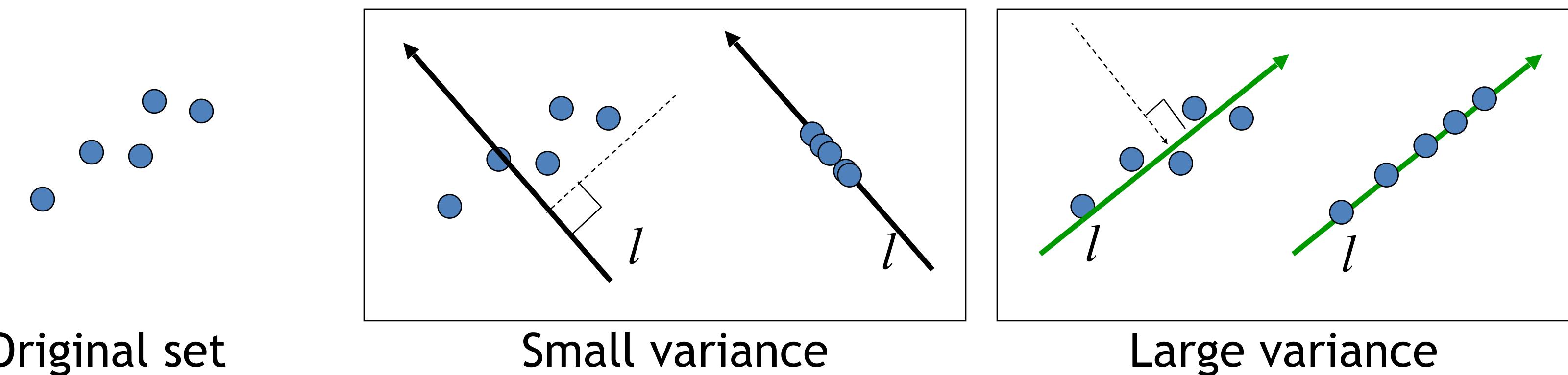
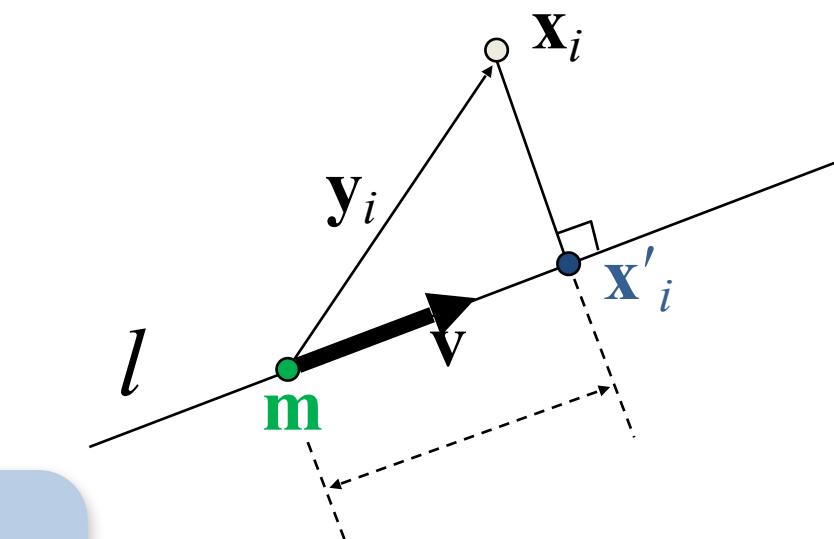
$$\mathbf{S} = \mathbf{V} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix} \mathbf{V}^T$$



What does the Scatter Matrix do?

- Let's look at a line l through the center of mass \mathbf{m} with direction vector \mathbf{v} , and project our points \mathbf{x}_i onto it. The variance of the projected points \mathbf{x}'_i is:

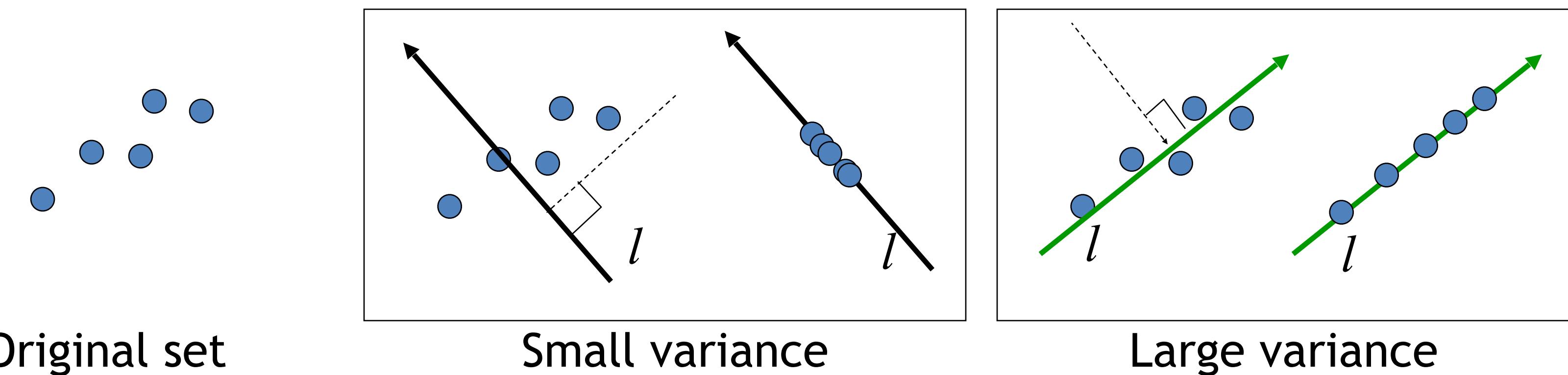
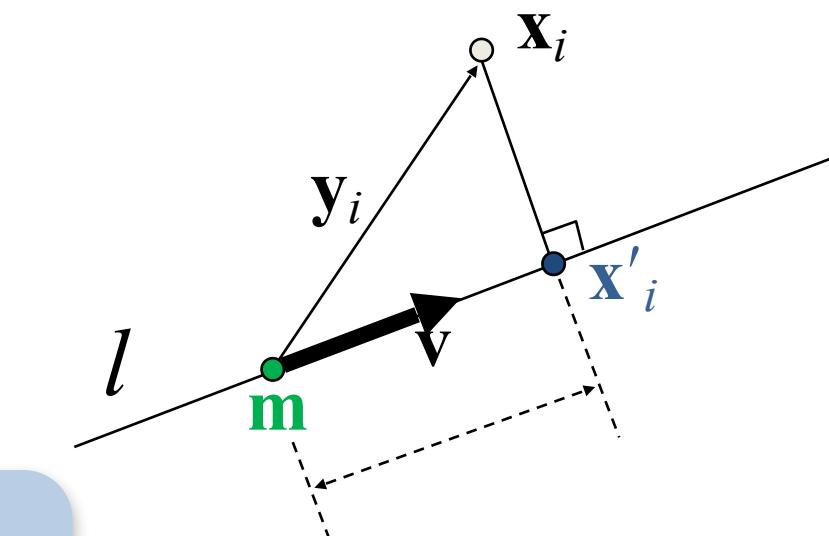
$$\begin{aligned}\text{var}(\mathbf{x}_1, \dots, \mathbf{x}_n; \mathbf{v}) &= \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}'_i - \mathbf{m}\|^2 = \\ &= \frac{1}{n} \sum_{i=1}^n \|(\mathbf{m} + \mathbf{v}^T \mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{v})^2 = \frac{1}{n} \mathbf{v}^T \mathbf{S} \mathbf{v}\end{aligned}$$



What does the Scatter Matrix do?

- The scatter matrix measures the variance of our data points along the direction \mathbf{v}

$$\begin{aligned}\text{var}(\mathbf{x}_1, \dots, \mathbf{x}_n; \mathbf{v}) &= \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}'_i - \mathbf{m}\|^2 = \\ &= \frac{1}{n} \sum_{i=1}^n \|(\mathbf{m} + \mathbf{v}^T \mathbf{y}_i) - \mathbf{m}\|^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{v})^2 = \frac{1}{n} \mathbf{v}^T \mathbf{S} \mathbf{v}\end{aligned}$$

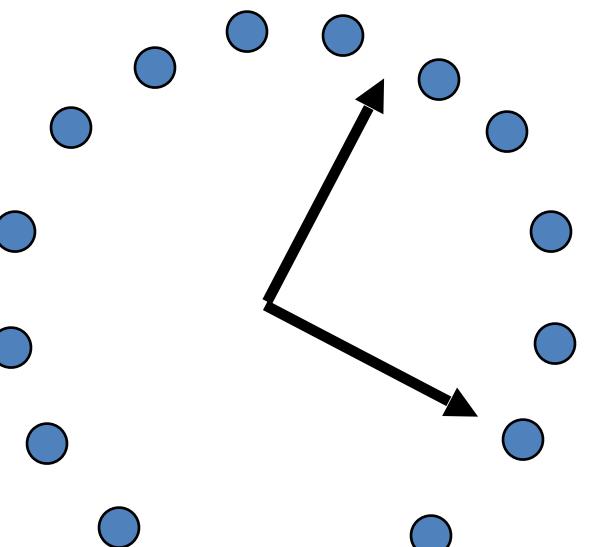


Principal Components

- Eigenvectors of \mathbf{S} that correspond to **big** eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.

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

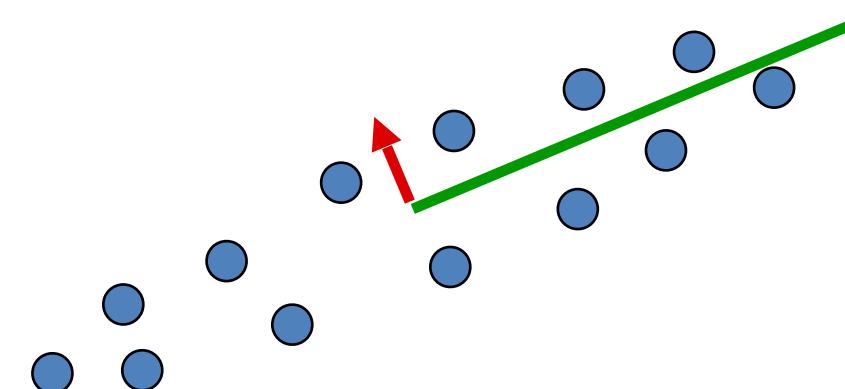
Principal Components



- There's no preferable direction
- S looks like this:

$$S = V \begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix} V^T$$

- Any vector is an eigenvector



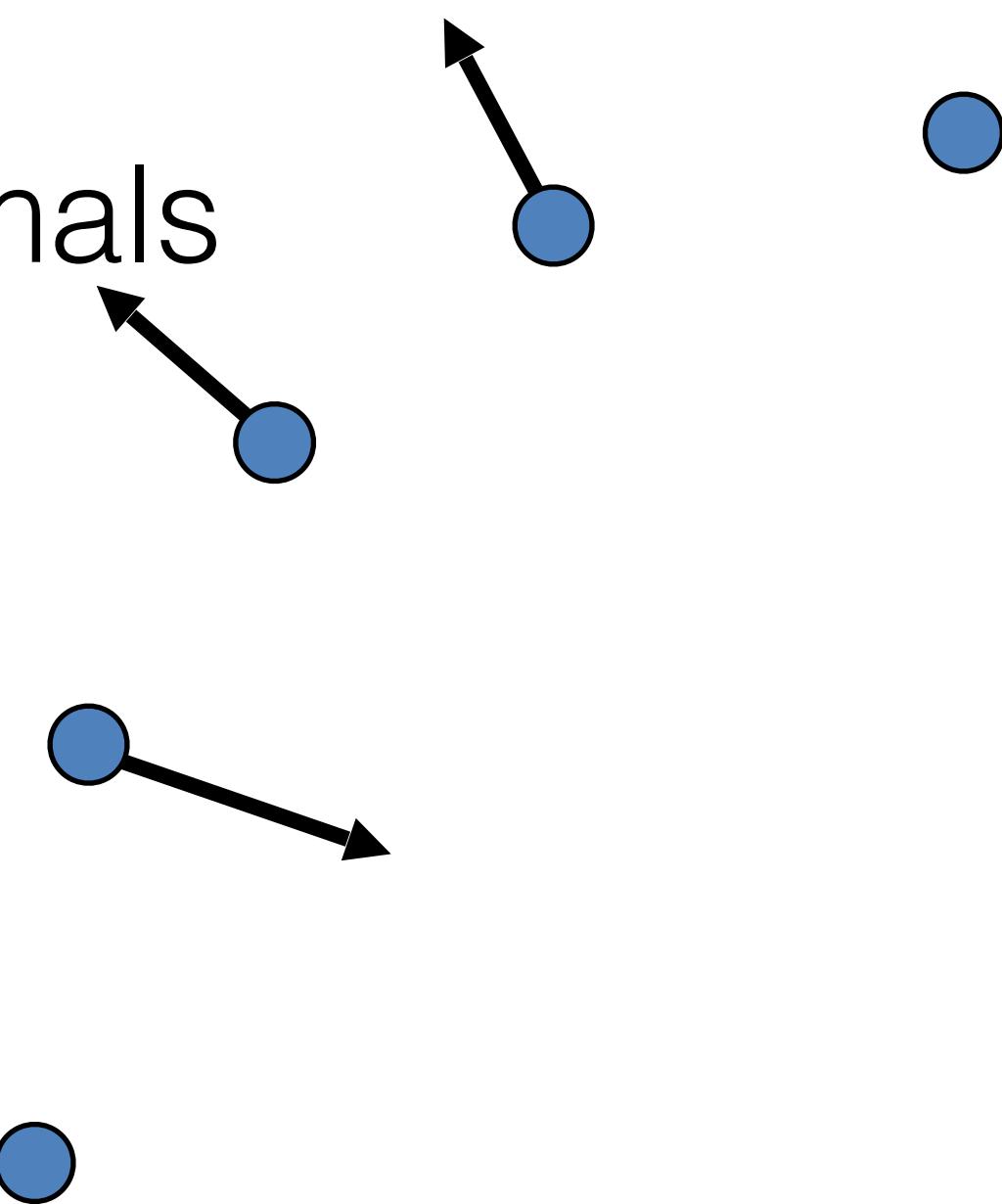
- There's a clear preferable direction
- S looks like this:

$$S = V \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} V^T$$

- μ is close to zero, much smaller than λ

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 $\mathbf{x}_i \in k\text{NN}(\mathbf{x}_j)$ or $\mathbf{x}_j \in k\text{NN}(\mathbf{x}_i)$
- Propagate normal orientation through graph
 - For neighbors $\mathbf{x}_i, \mathbf{x}_j$: Flip \mathbf{n}_j if $\mathbf{n}_i^T \mathbf{n}_j < 0$
 - Fails at sharp edges/corners
- Propagate along “safe” paths (parallel tangent planes)
 - Minimum spanning tree with angle-based edge weights

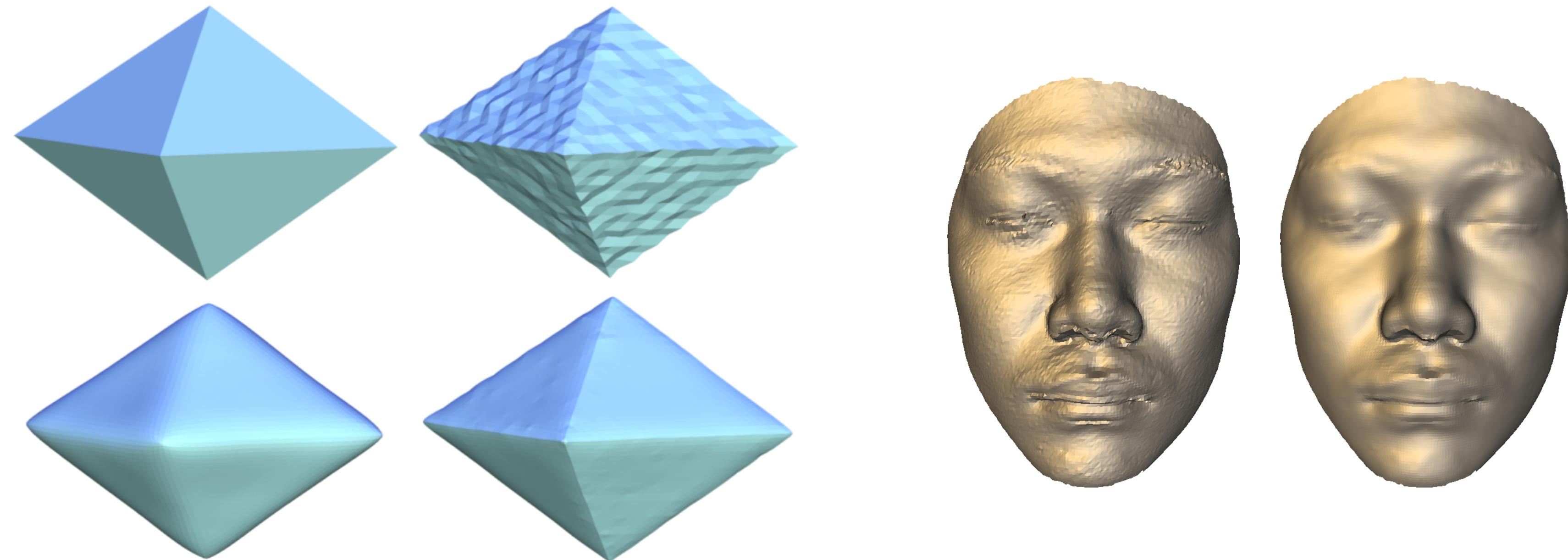
$$w_{ij} = 1 - |\mathbf{n}_i^T \mathbf{n}_j|$$

“Surface reconstruction from unorganized points”, Hoppe et al., SIGGRAPH 1992
<http://research.microsoft.com/en-us/um/people/hoppe/recon.pdf>

Elementary Differential Geometry

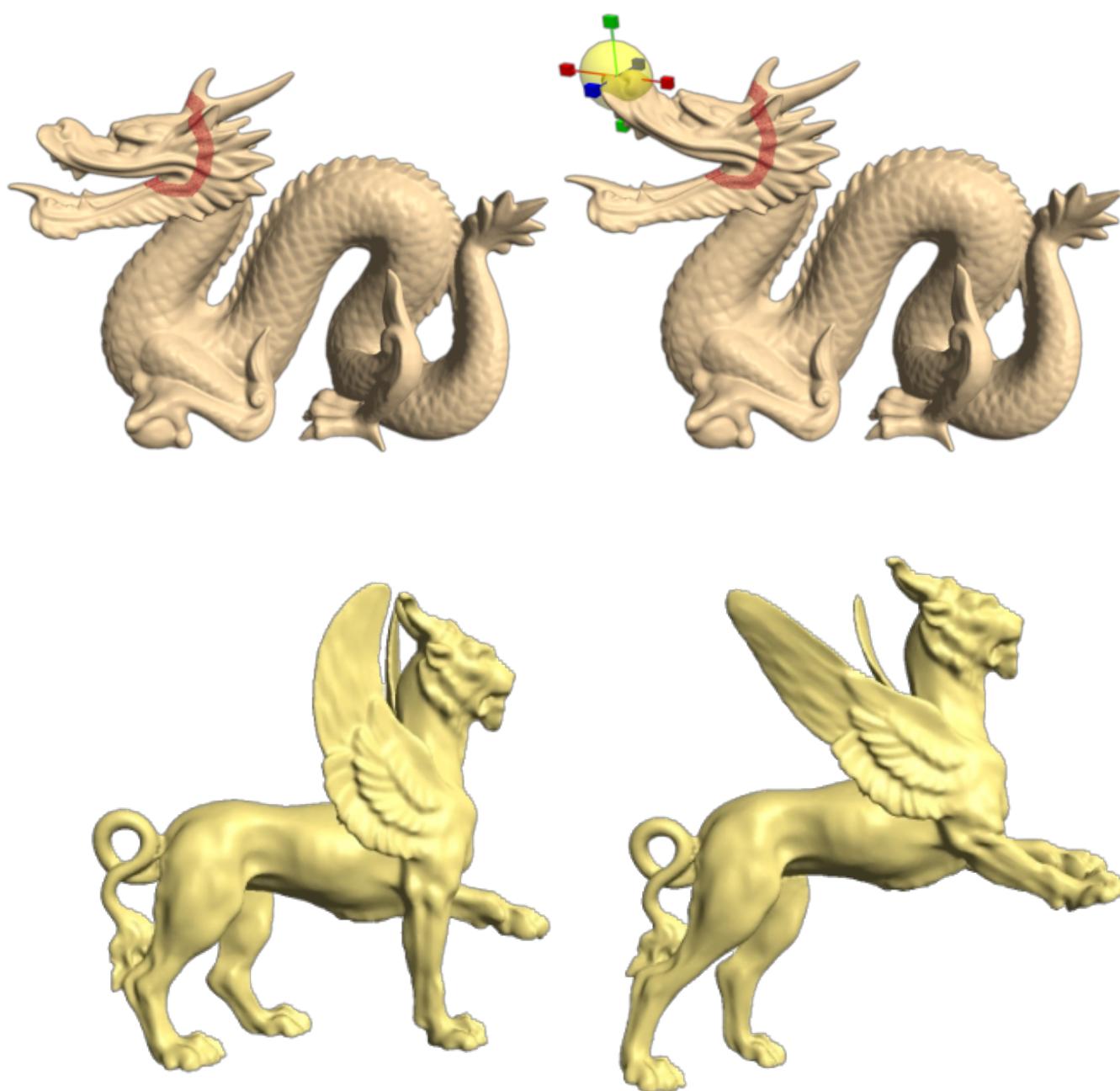
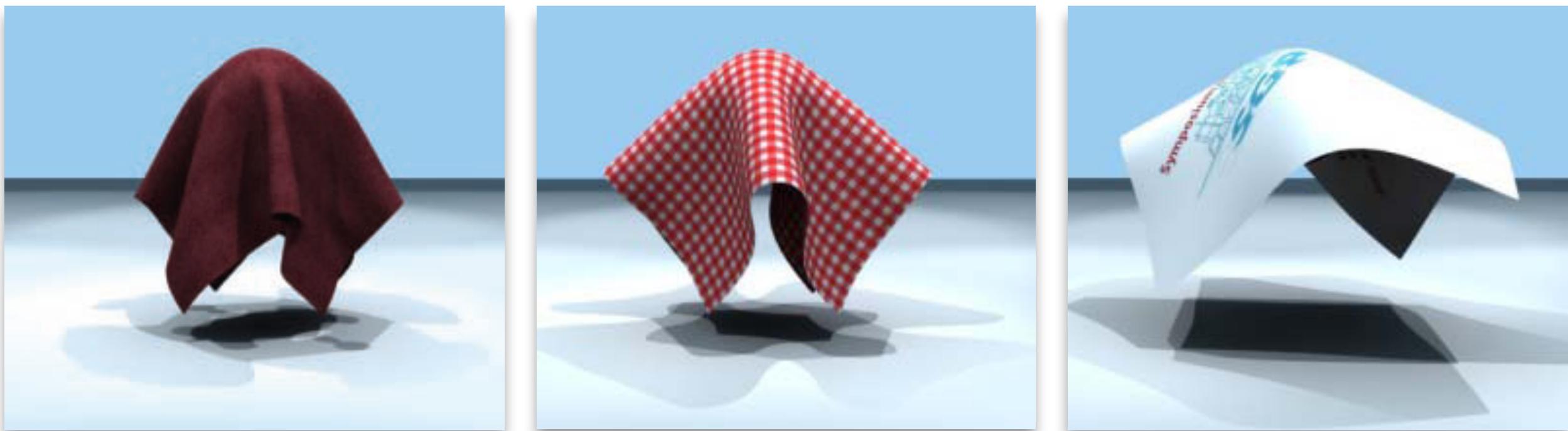
Differential Geometry – Motivation

- Describe and analyze geometric characteristics of shapes
 - e.g. how smooth?



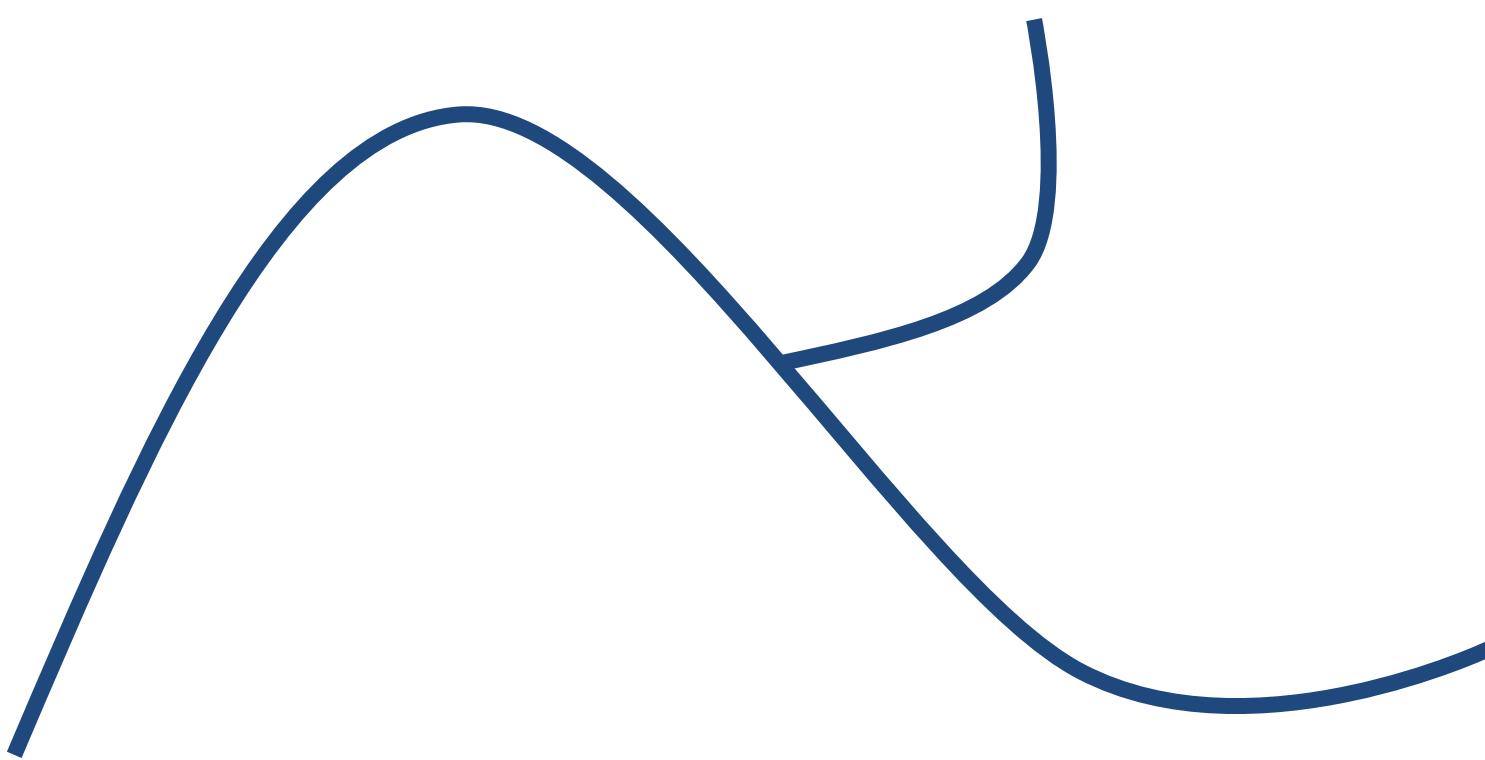
Differential Geometry – Motivation

- Describe and analyze geometric characteristics of shapes
 - e.g. how smooth?
 - how shapes deform



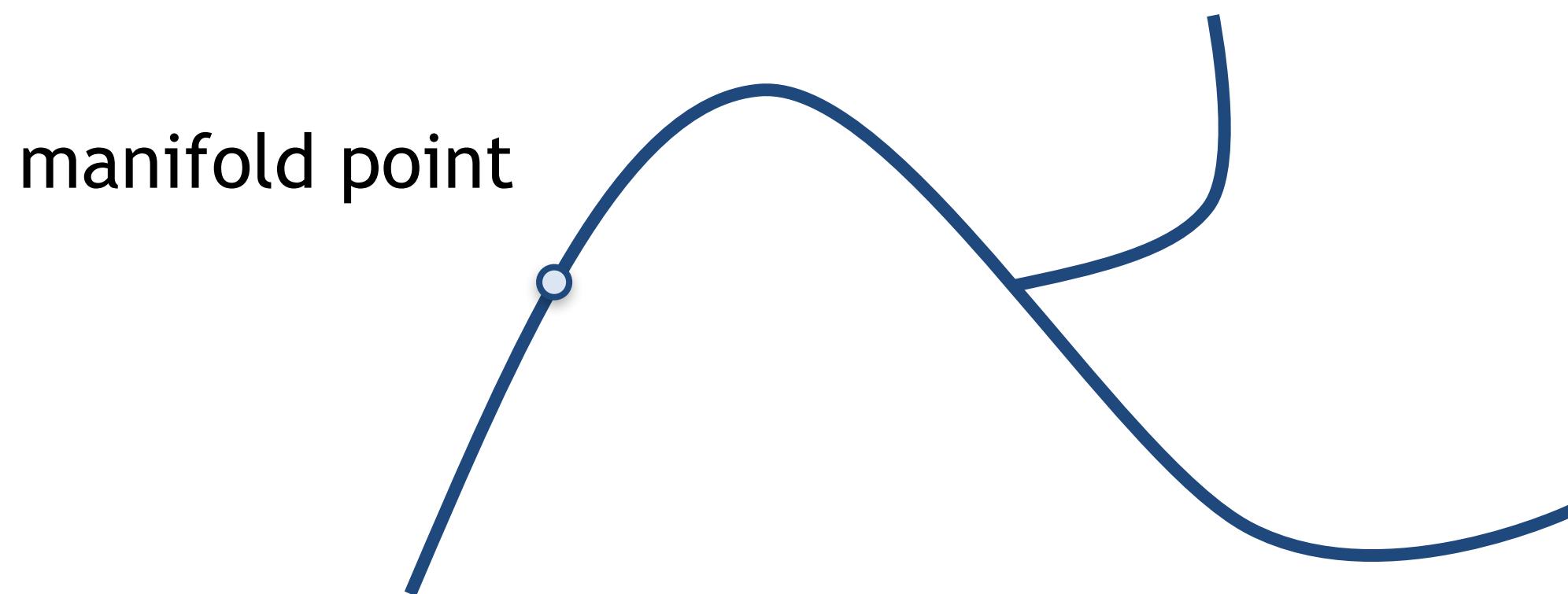
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



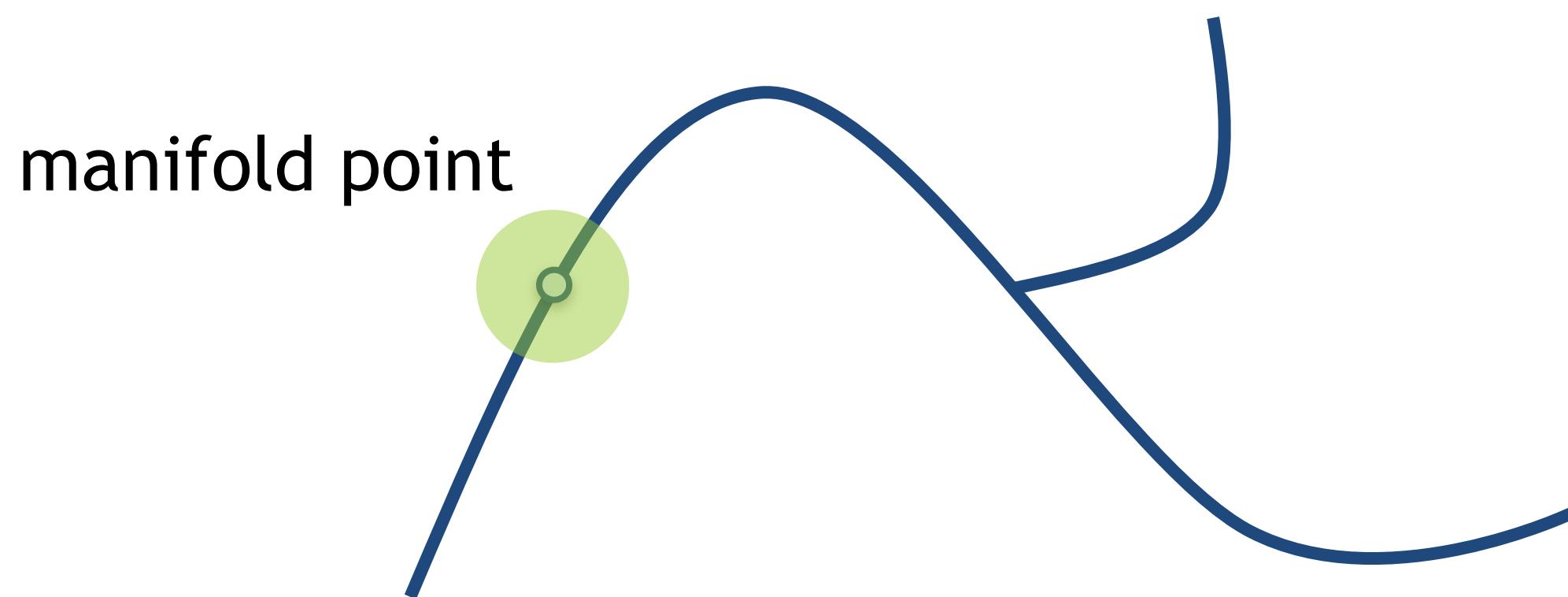
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



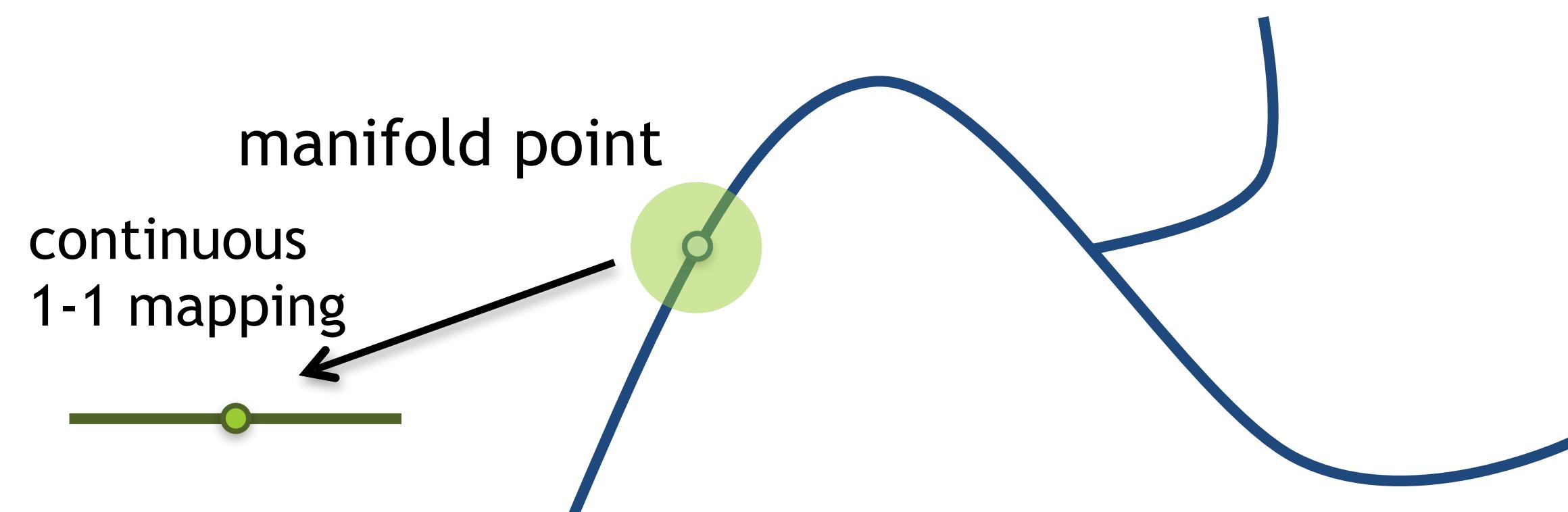
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



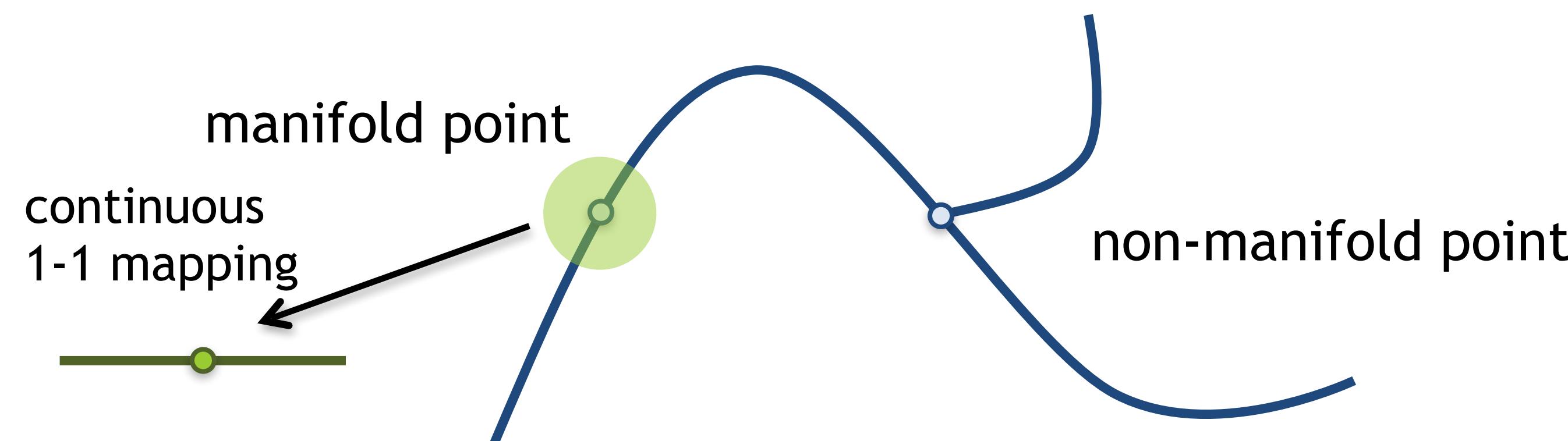
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



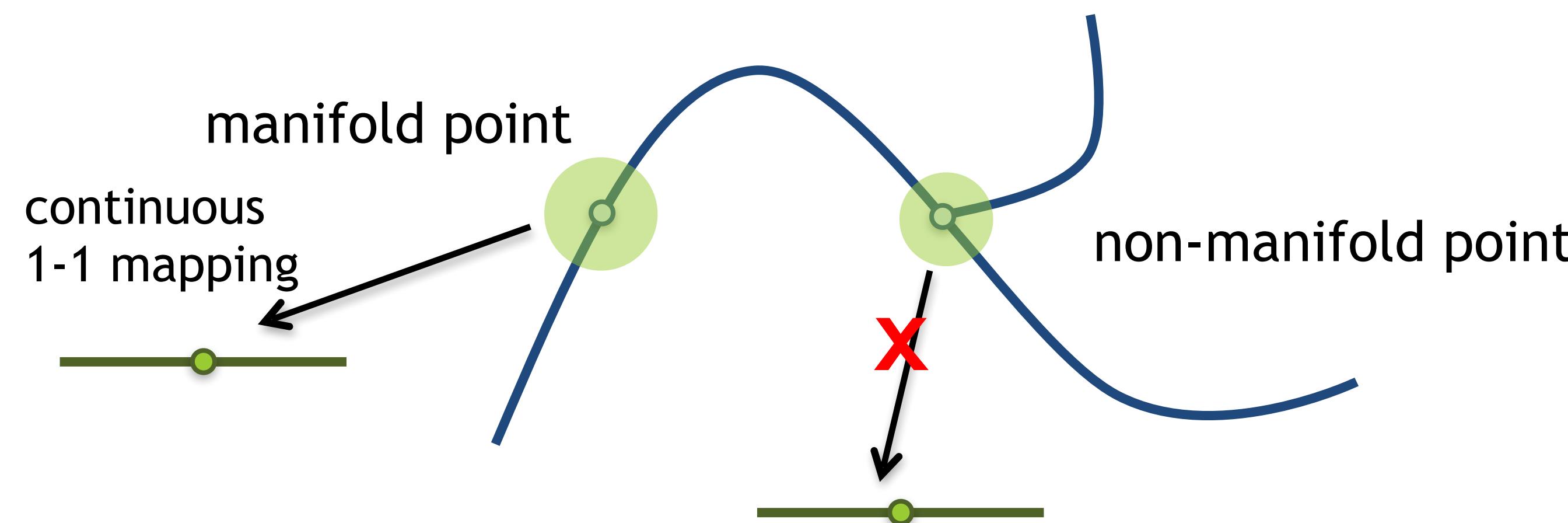
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



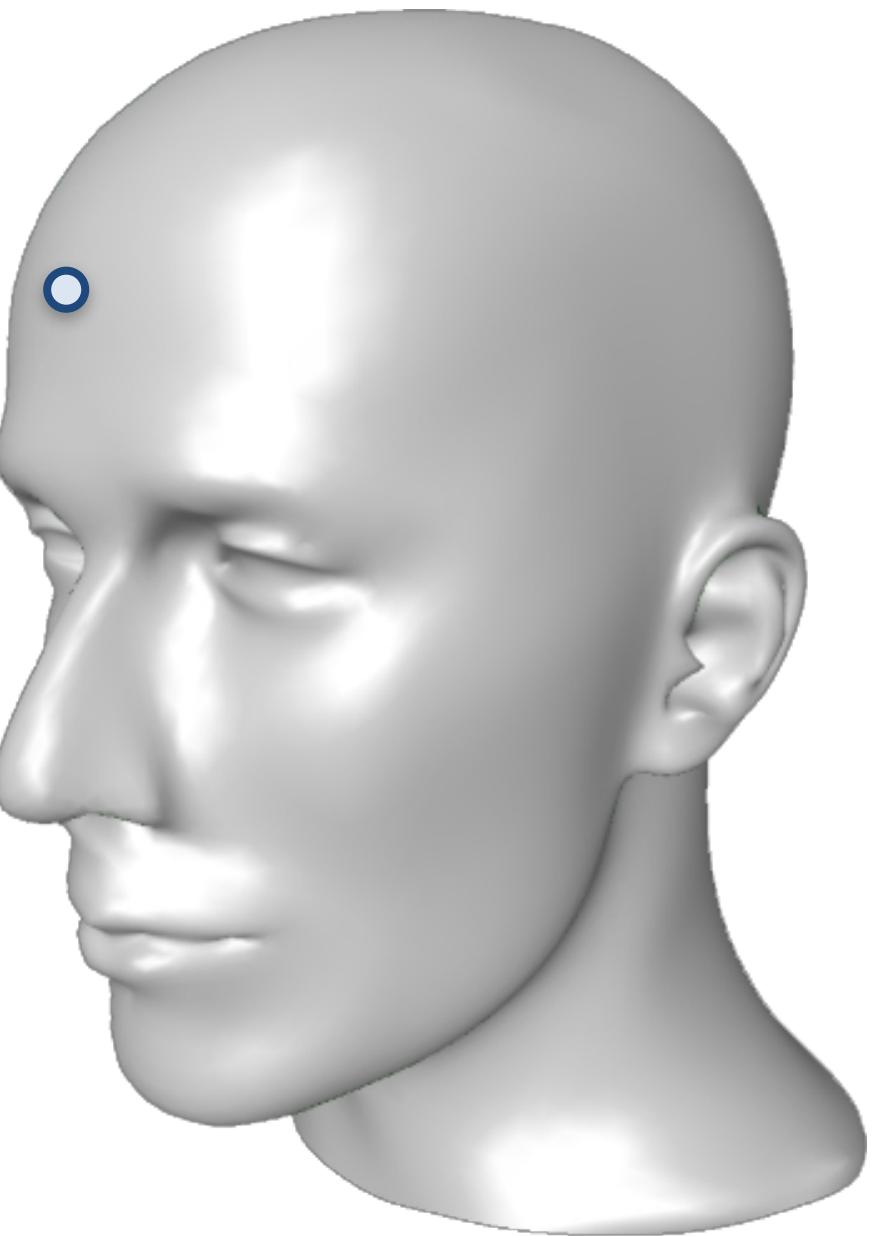
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



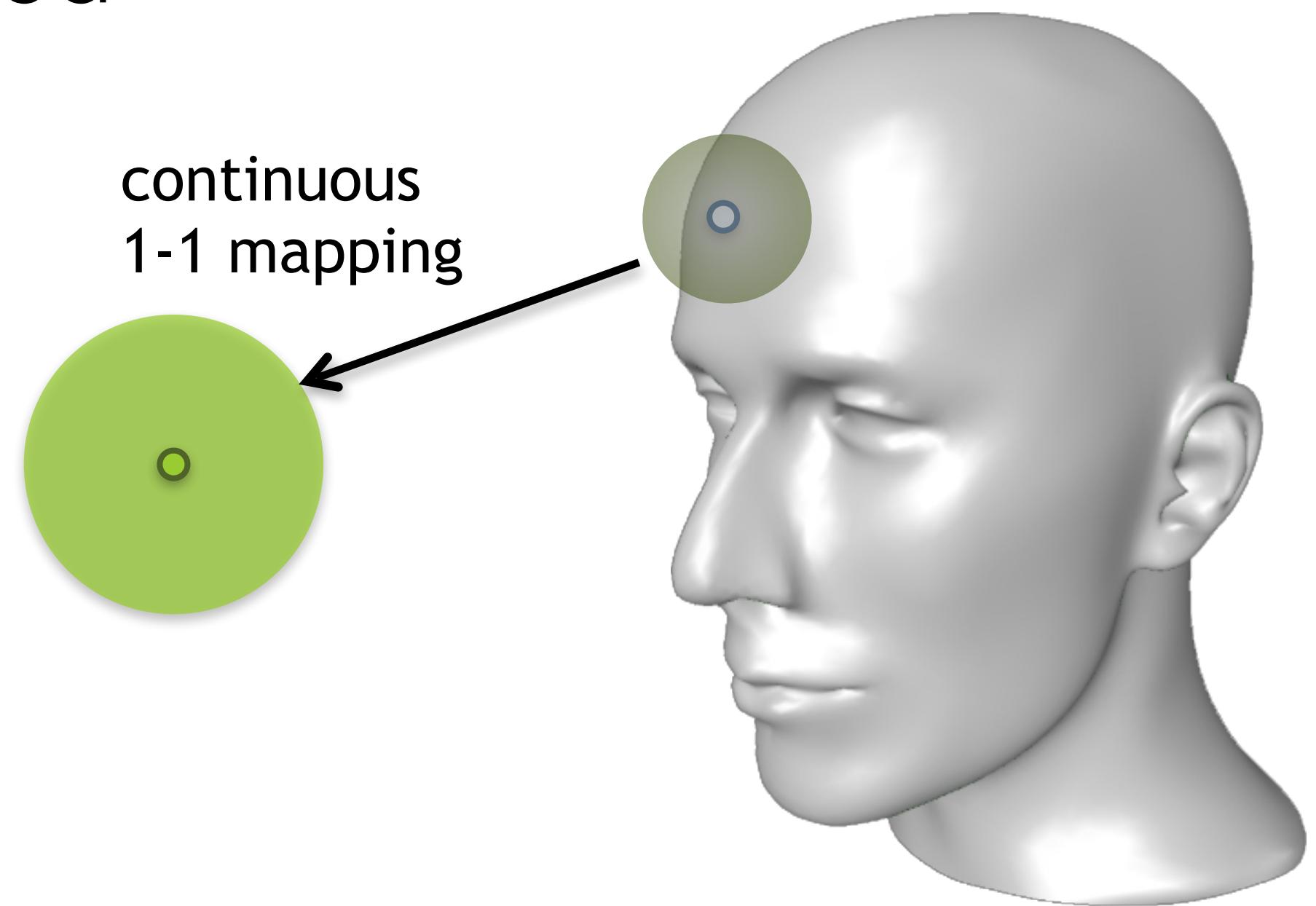
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



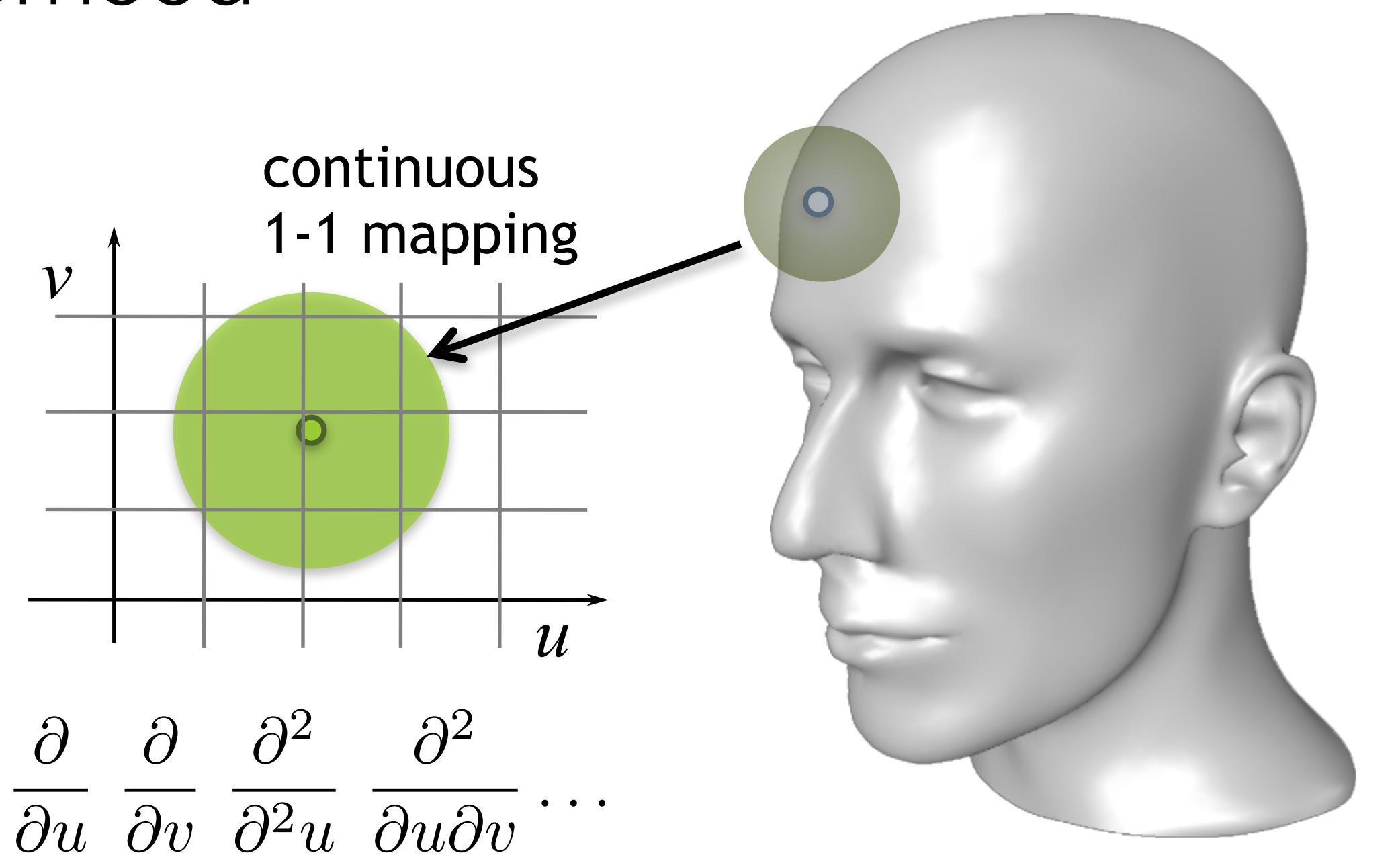
Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



Differential Geometry Basics

- Geometry of manifolds
- Things that can be discovered by local observation: point + neighborhood



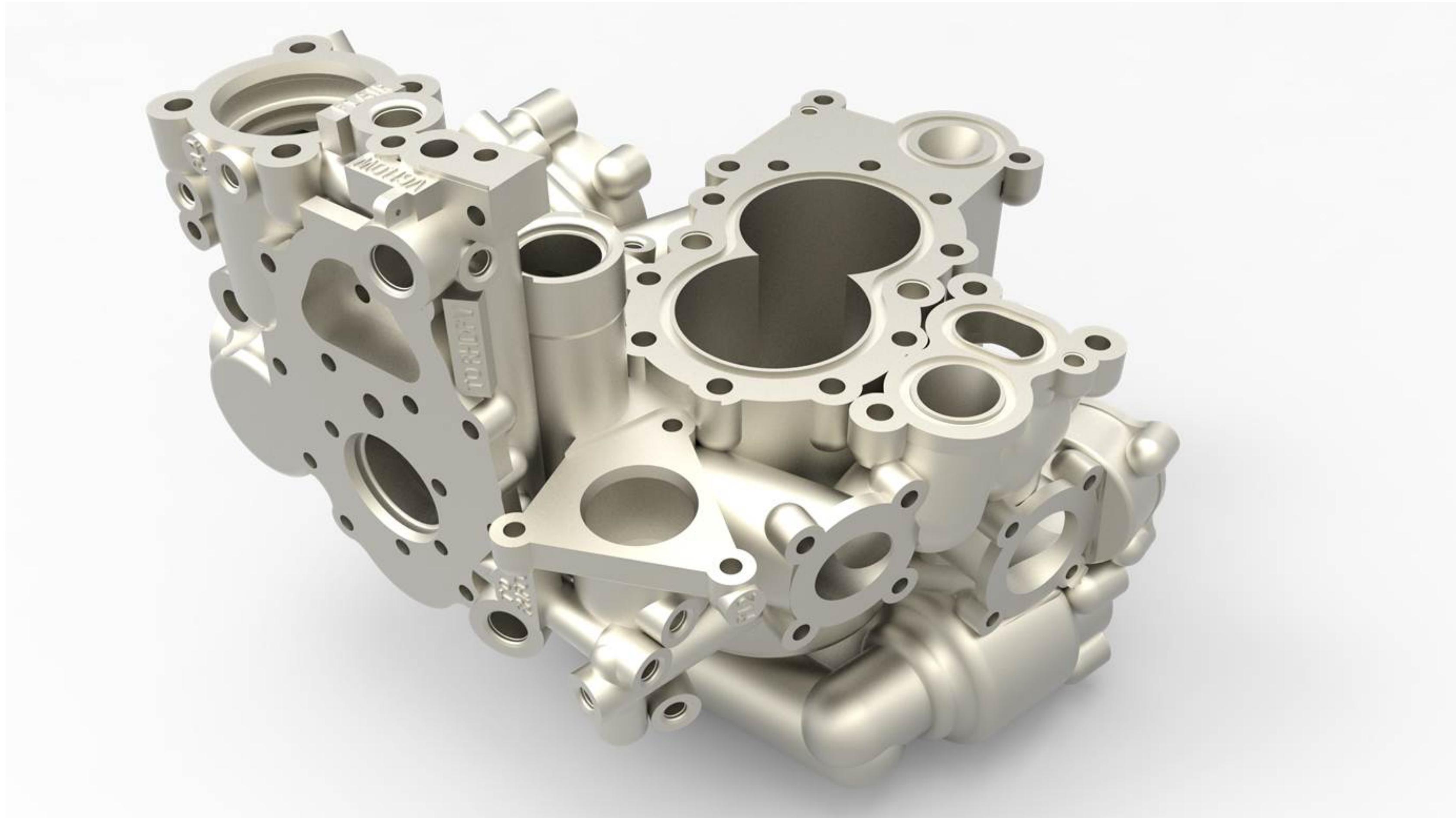
If a sufficiently smooth mapping can be constructed, we can look at its first and second derivatives

Tangents, normals, curvatures, curve angles

Distances, topology

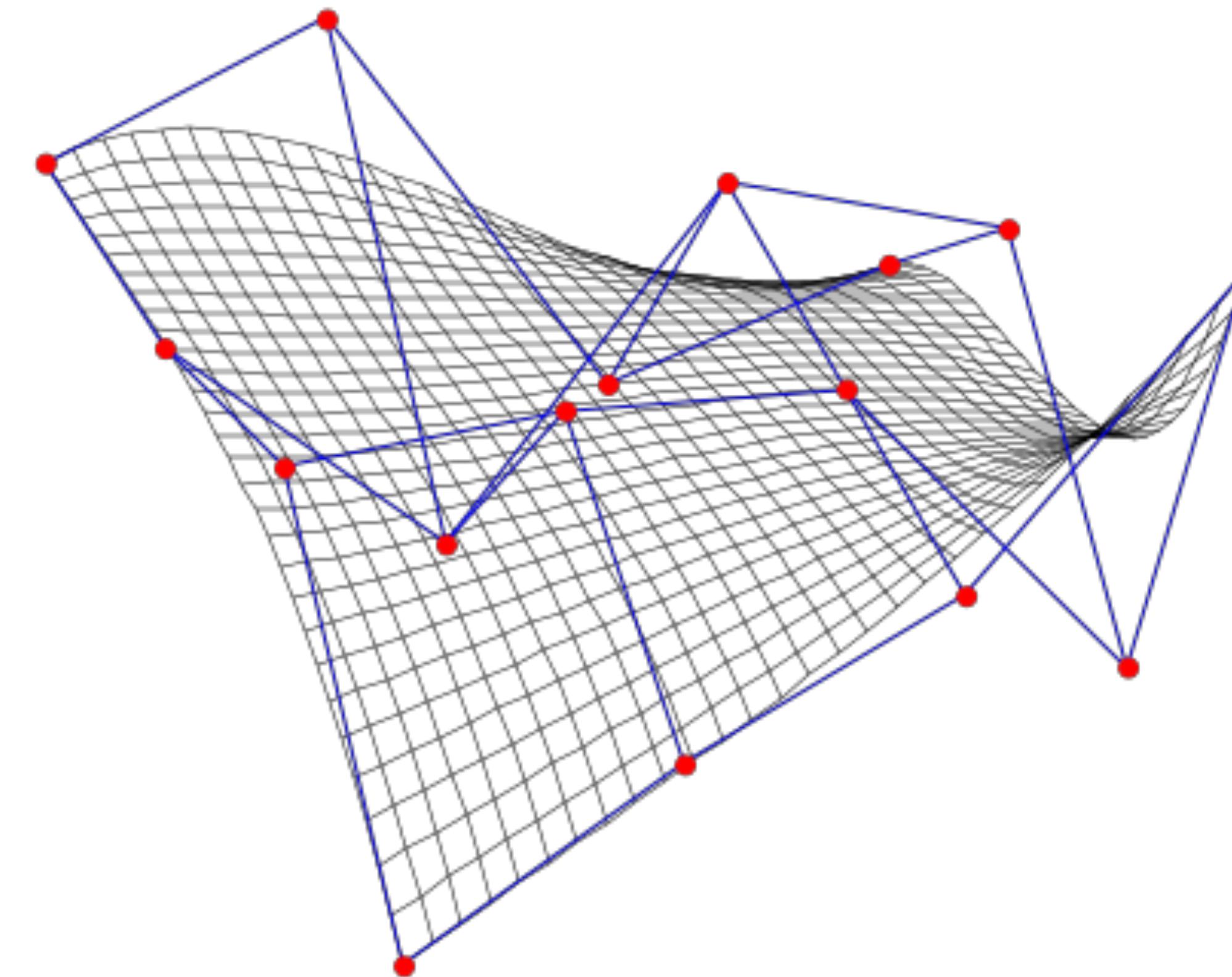
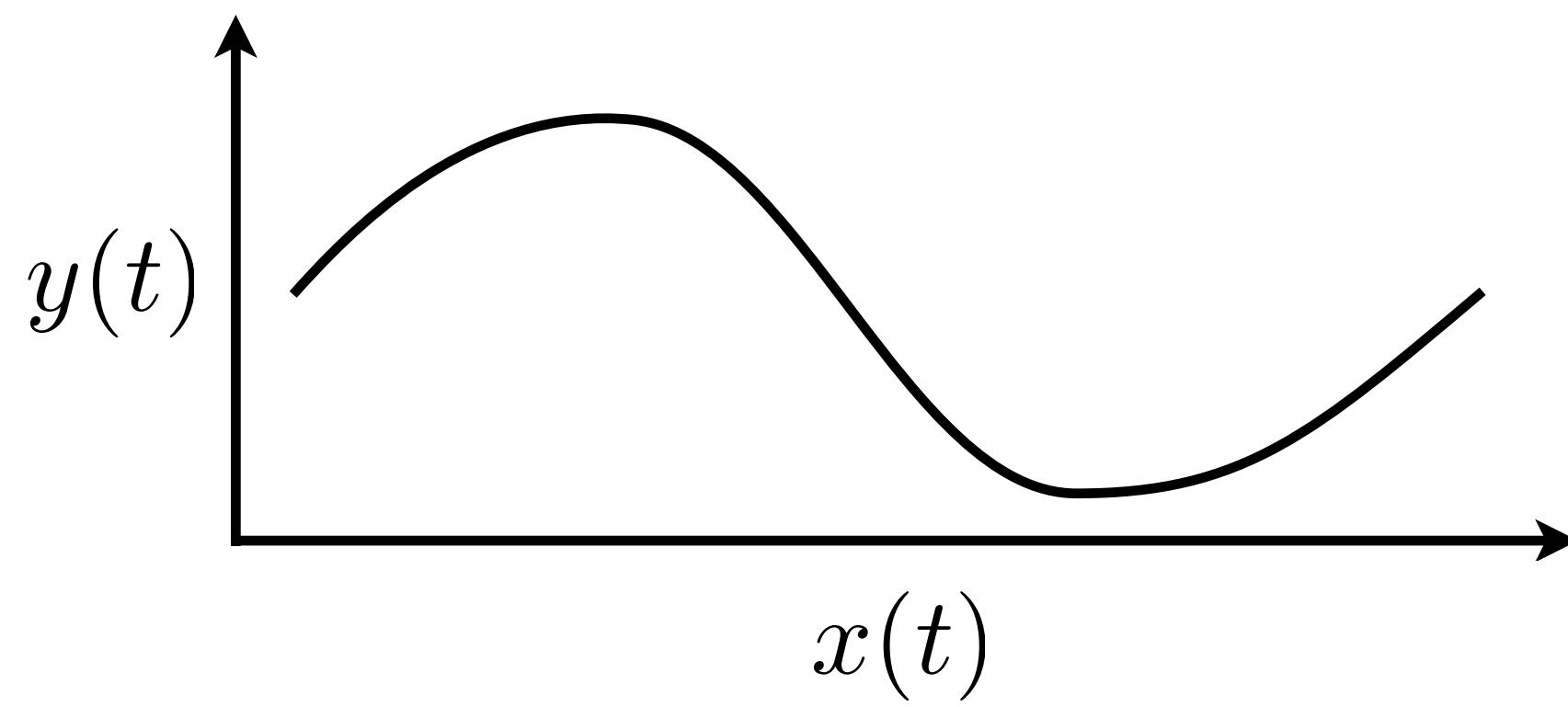
Curves

How do we model shapes?



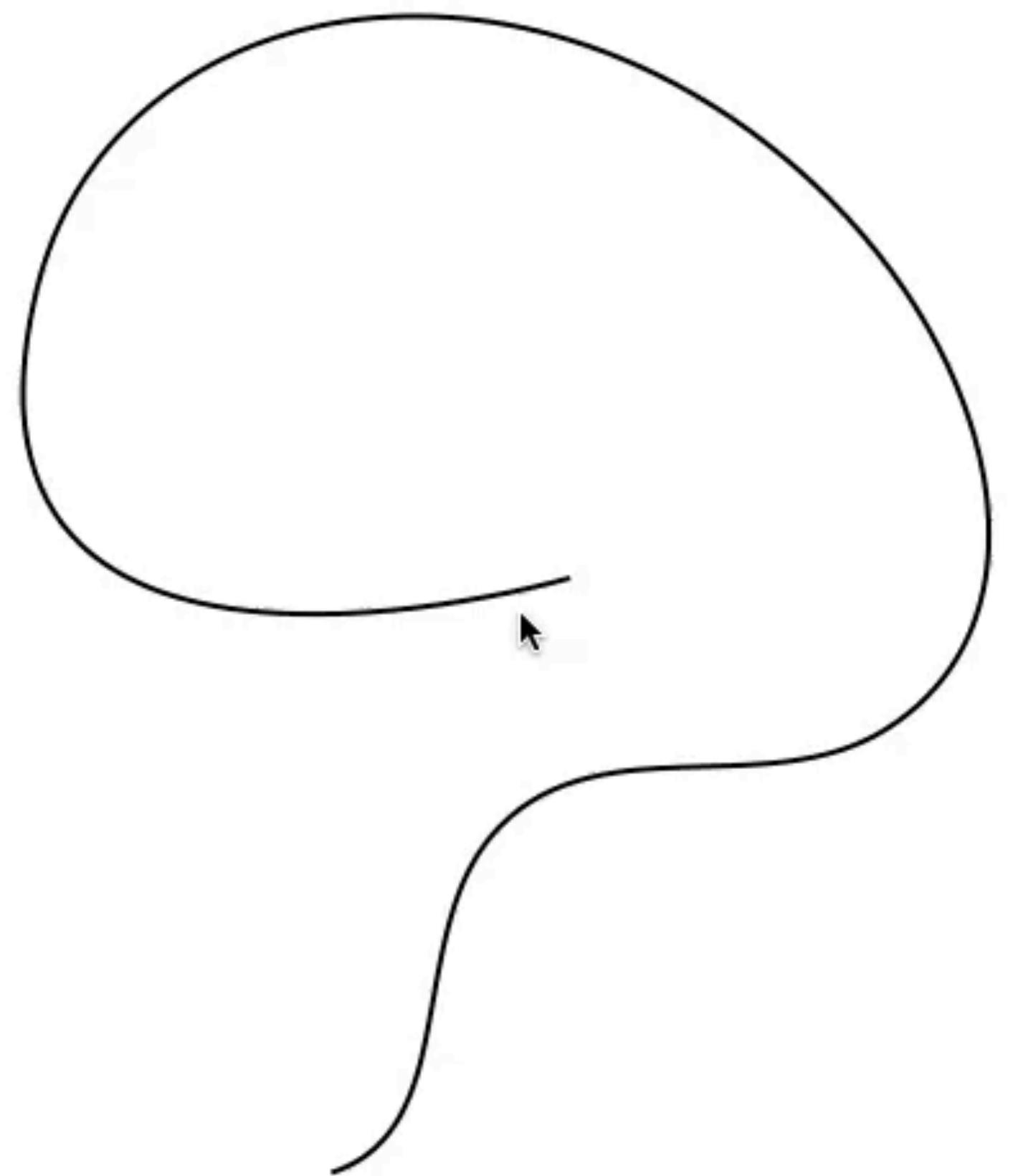
Delcam Plc. [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>) or GFDL (<http://www.gnu.org/copyleft/fdl.html>)], via Wikimedia Commons

Building blocks: curves and surfaces

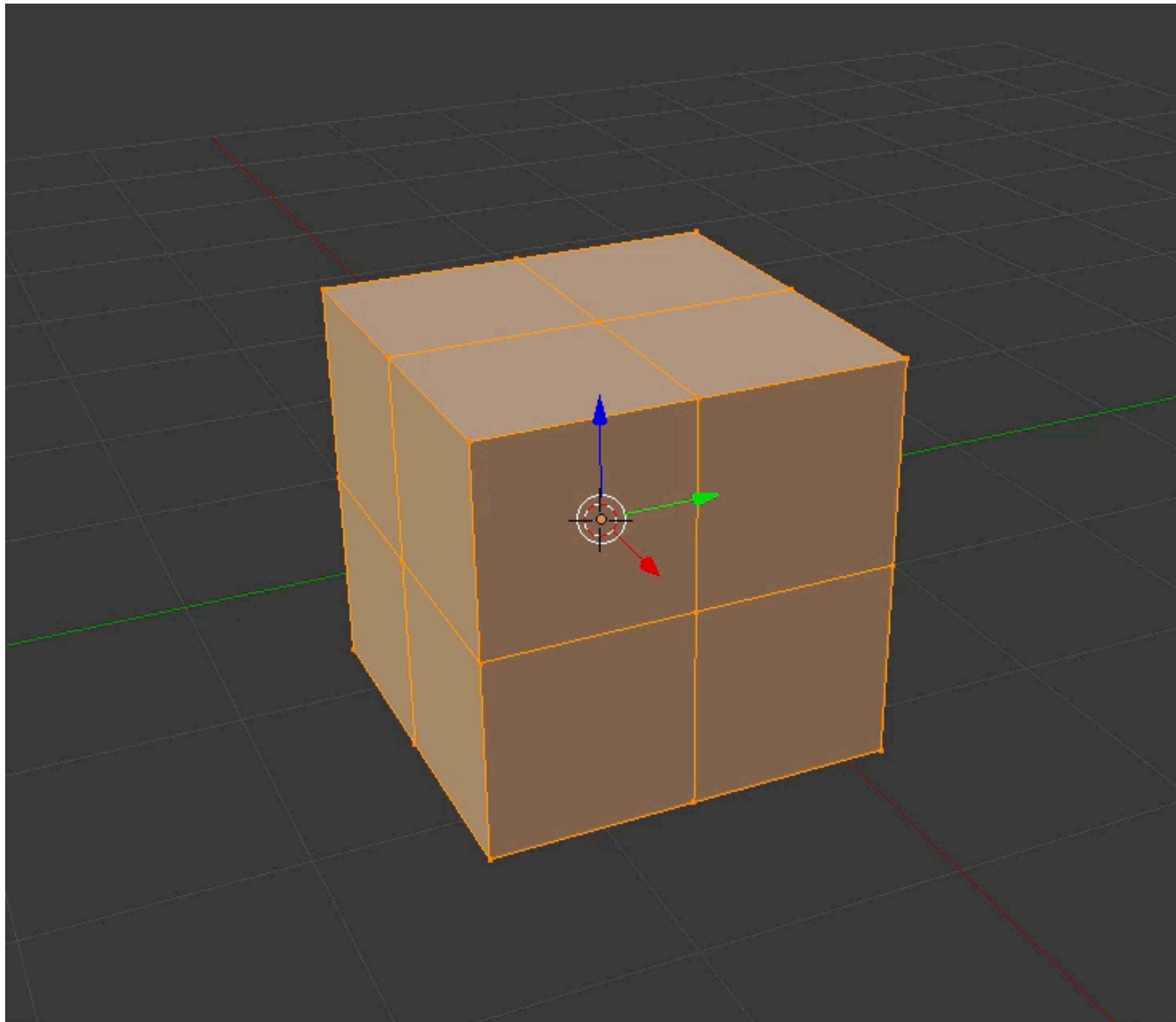


By Wojciech mula at Polish Wikipedia, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=9853555>

Keynote Demo



Blender Demo



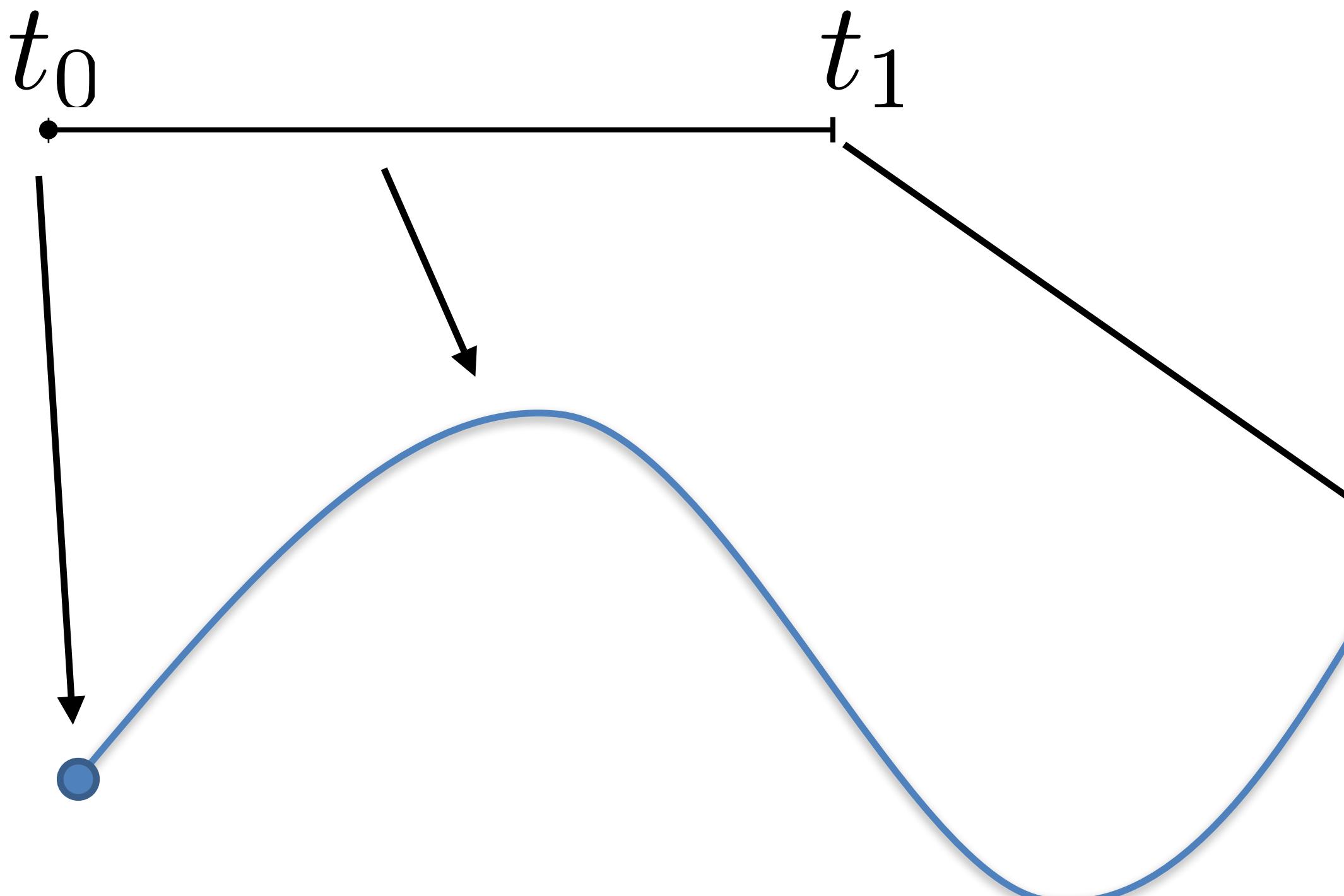
Modeling curves

- We need **mathematical concepts** to characterize the desired curve properties
- Notions from **curve geometry** help with designing user interfaces for curve creation and editing

2D parametric curve

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

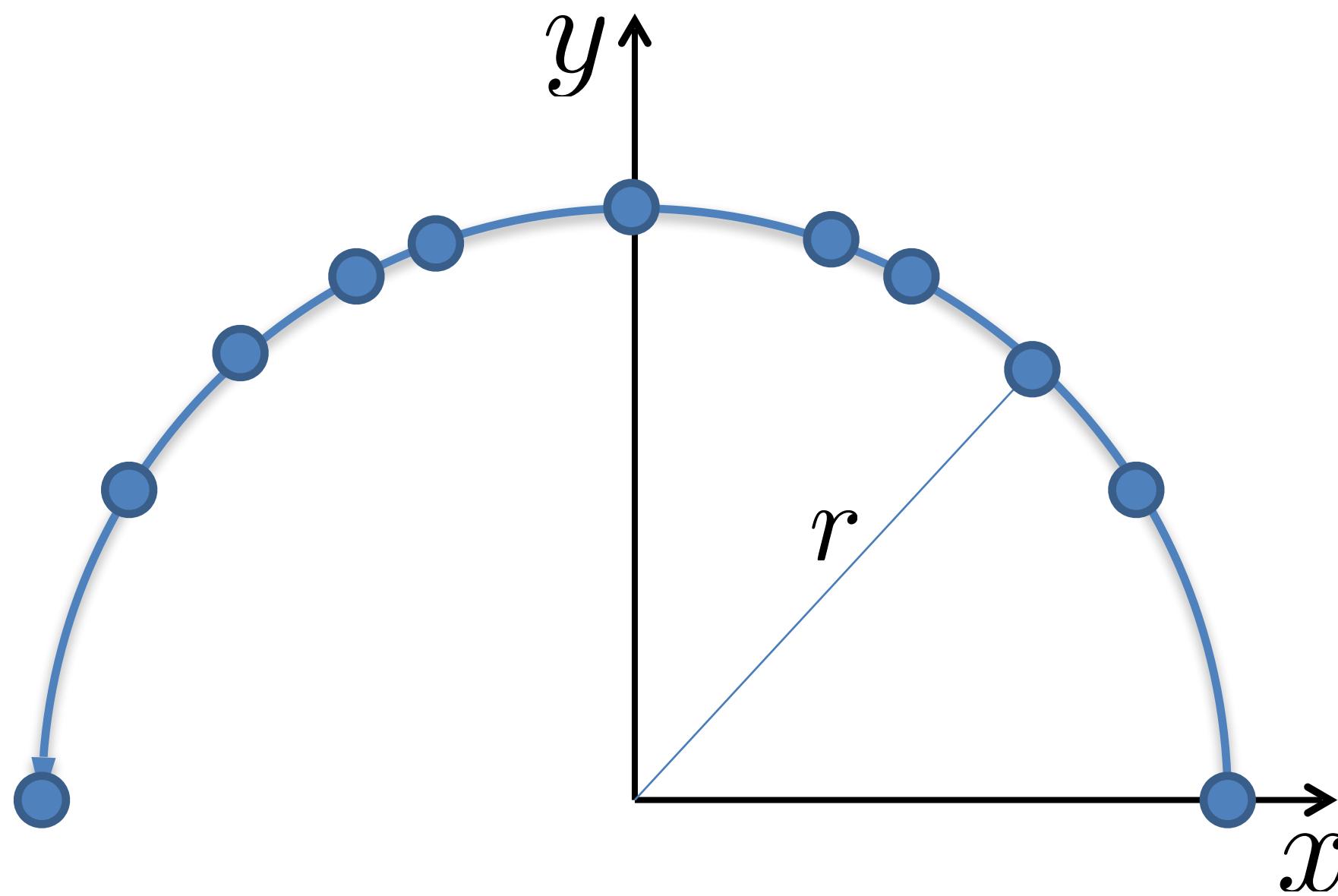
- $\mathbf{p}(t)$ must be continuous



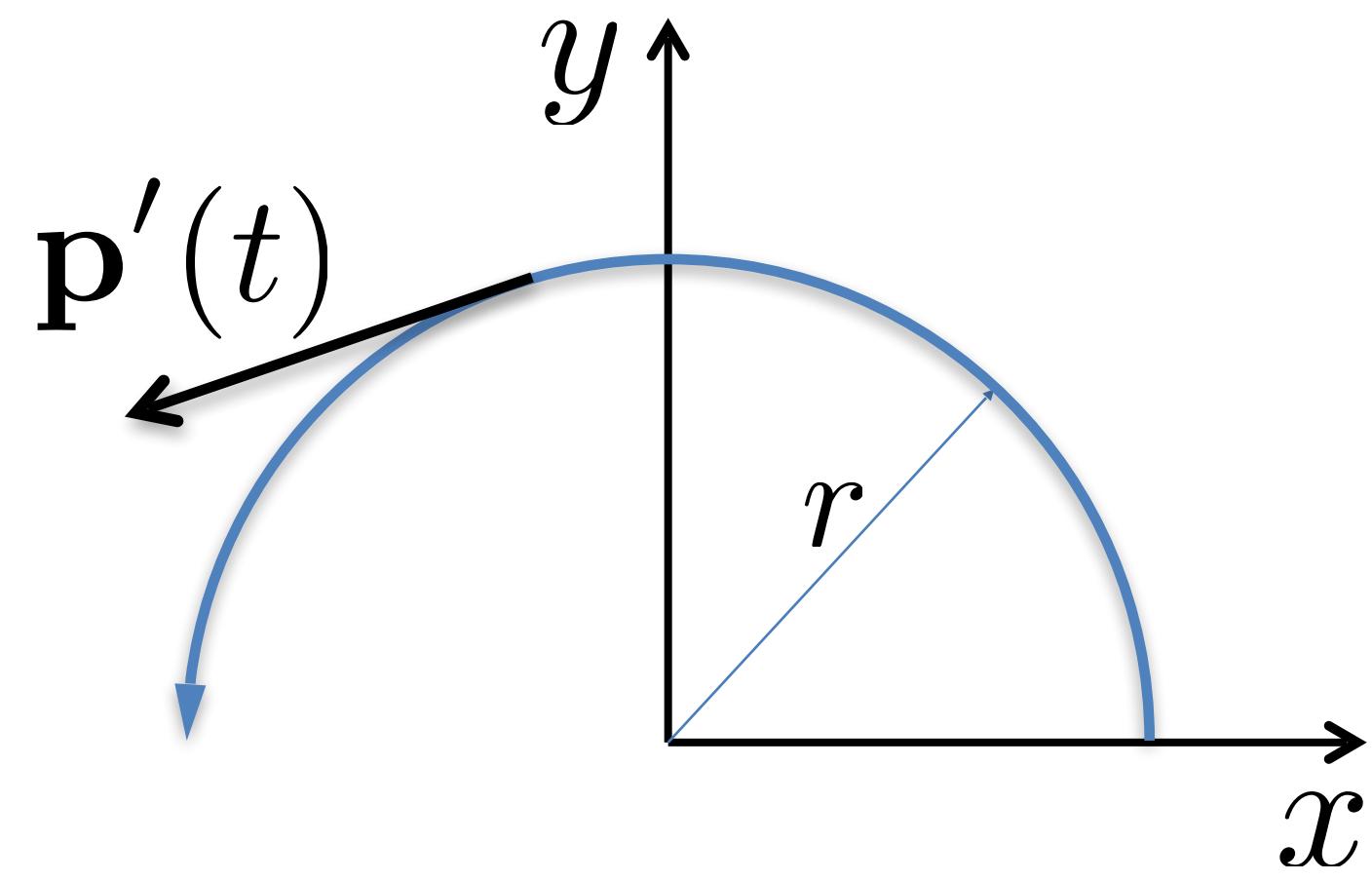
A curve can be parameterized in many different ways

$$\begin{pmatrix} r \cos t \\ r \sin t \end{pmatrix}, t \in [0, \pi]$$

$$\begin{pmatrix} -rt \\ r\sqrt{1-t^2} \end{pmatrix}, t \in [-1, 1]$$



Tangent vector



$$\mathbf{p}(t) = \begin{pmatrix} r \cos t \\ r \sin t \end{pmatrix}$$

$$\mathbf{p}'(t) = \begin{pmatrix} -r \sin t \\ r \cos t \end{pmatrix}$$

$$\|\mathbf{p}'(t)\| = \text{speed}$$

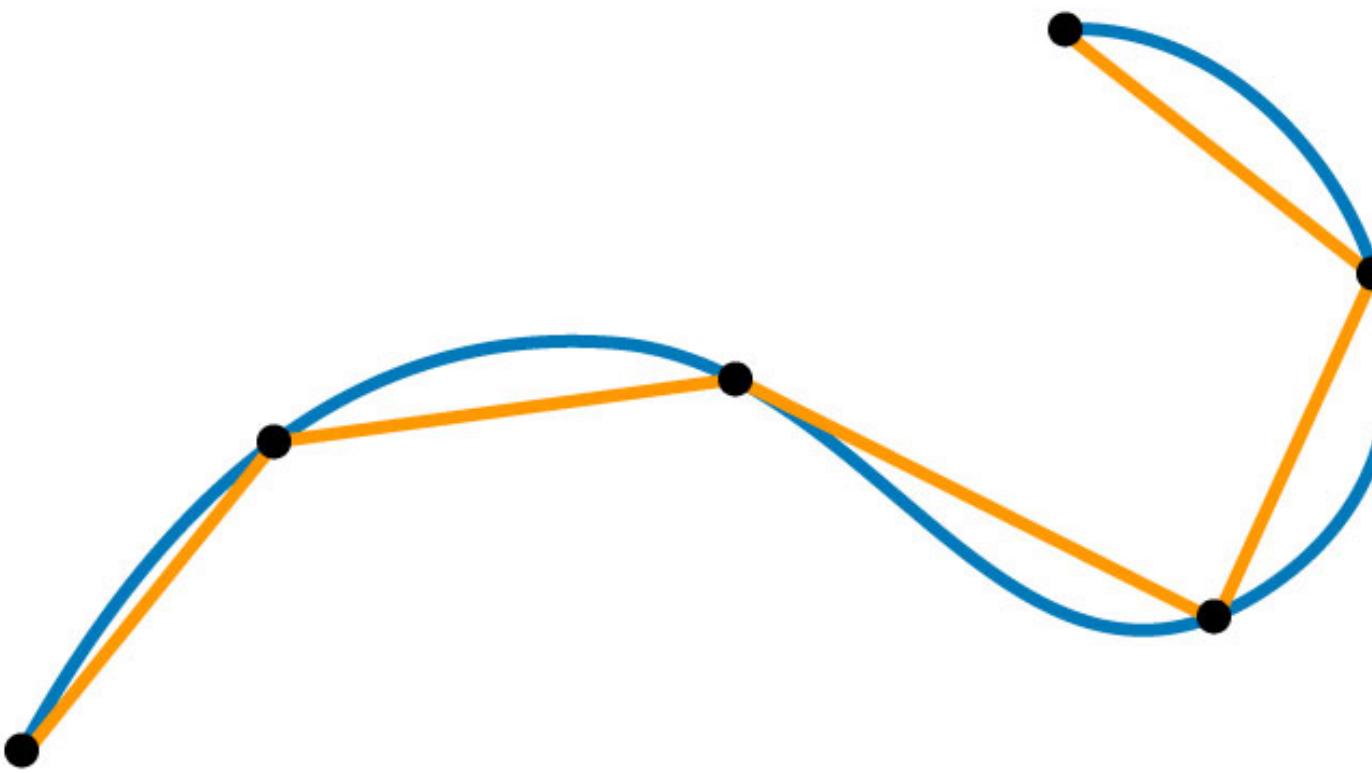
$$\frac{\mathbf{p}'(t)}{\|\mathbf{p}'(t)\|} = \mathbf{T}(t) = \text{unit tangent}$$

Parametrization-independent!

Arc length

- How long is the curve between t_0 and t ? How far does the particle travel?
- Speed is $\|\mathbf{p}'(t)\|$, so:

$$s(t) = \int_{t_0}^t \|\mathbf{p}'(t)\| dt$$

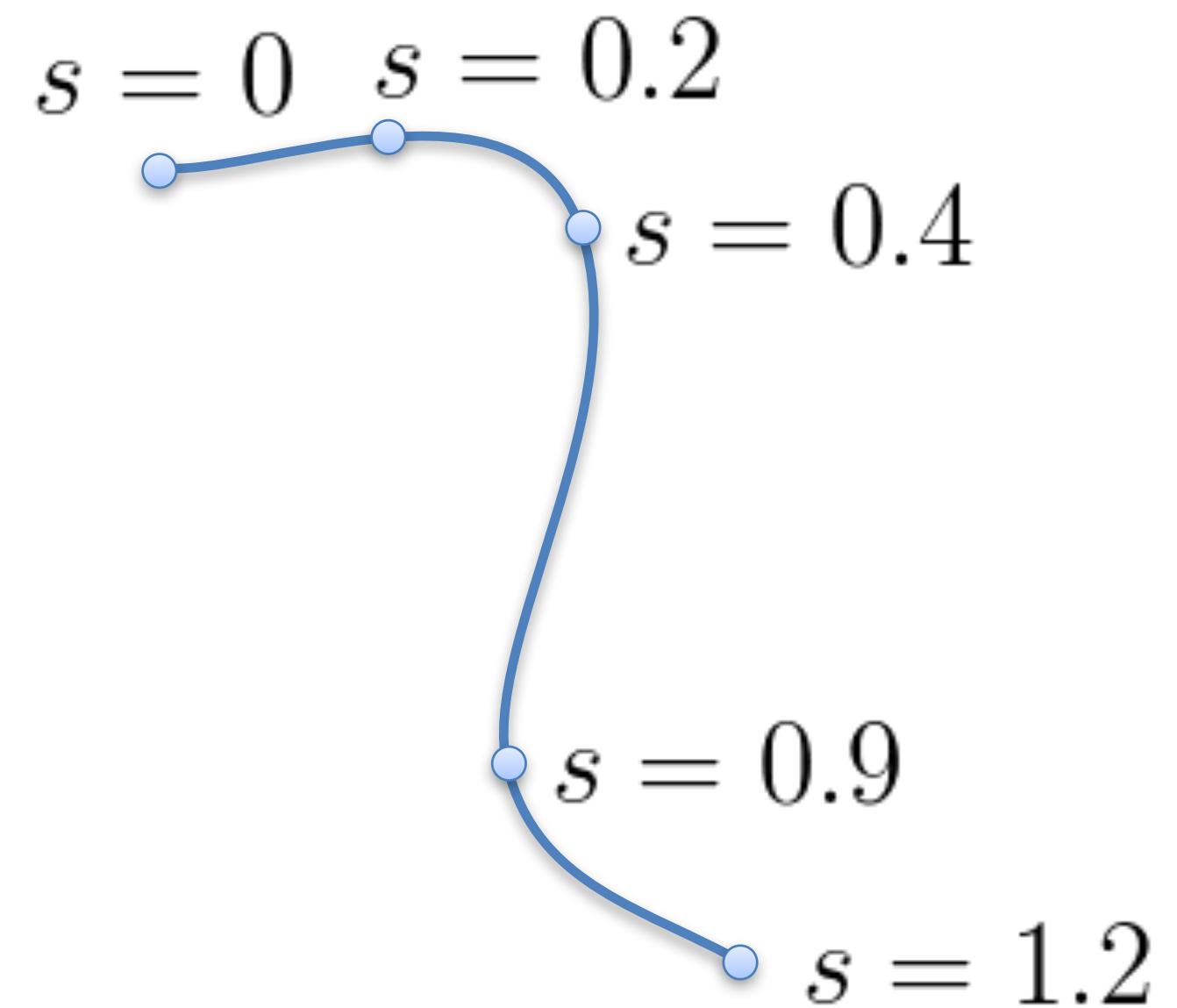


- Speed is nonnegative, so $s(t)$ is non-decreasing

Arc length parameterization

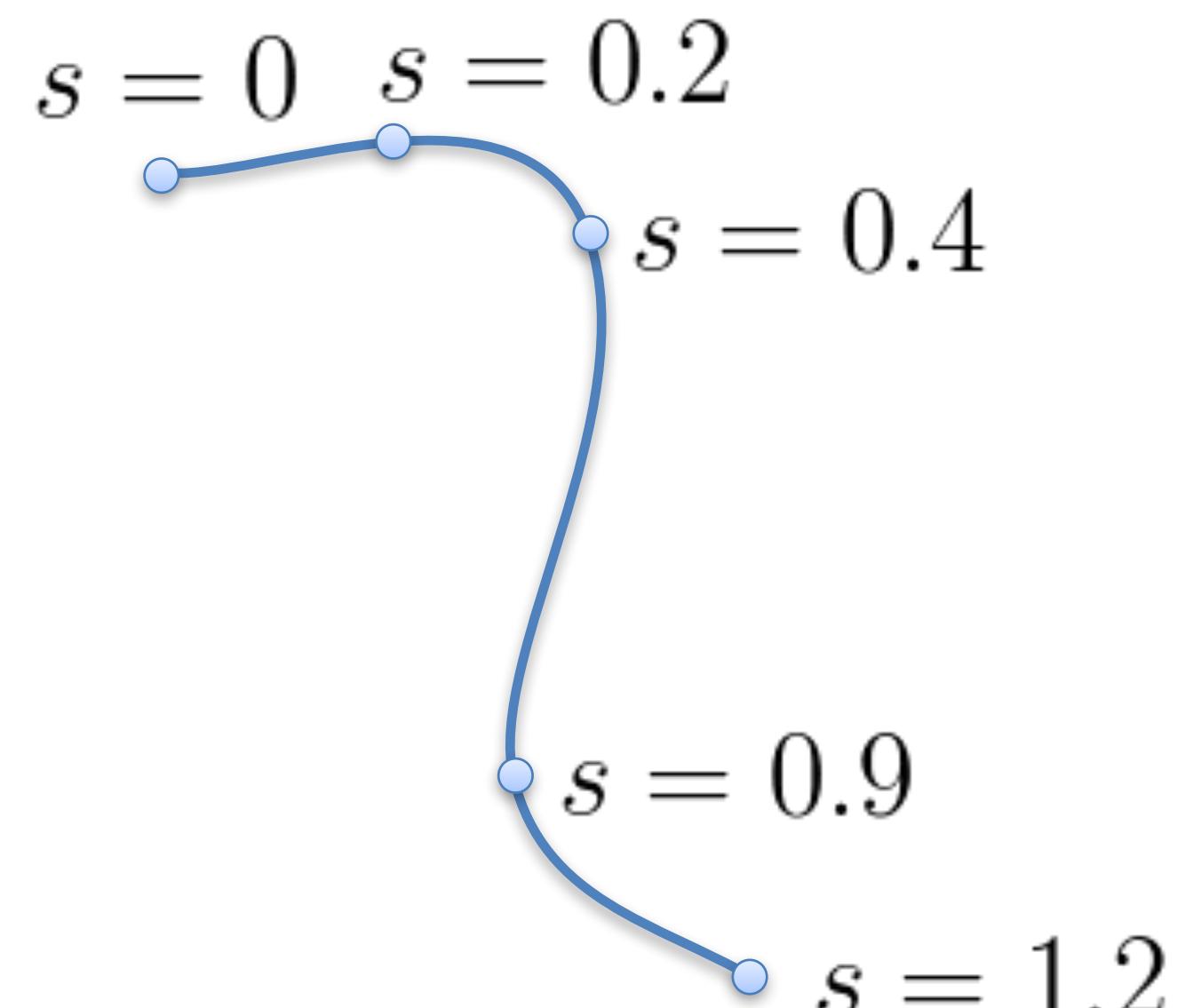
- Every curve has a natural parameterization:

$\mathbf{p}(s)$, such that $\|\mathbf{p}'(s)\| = 1$



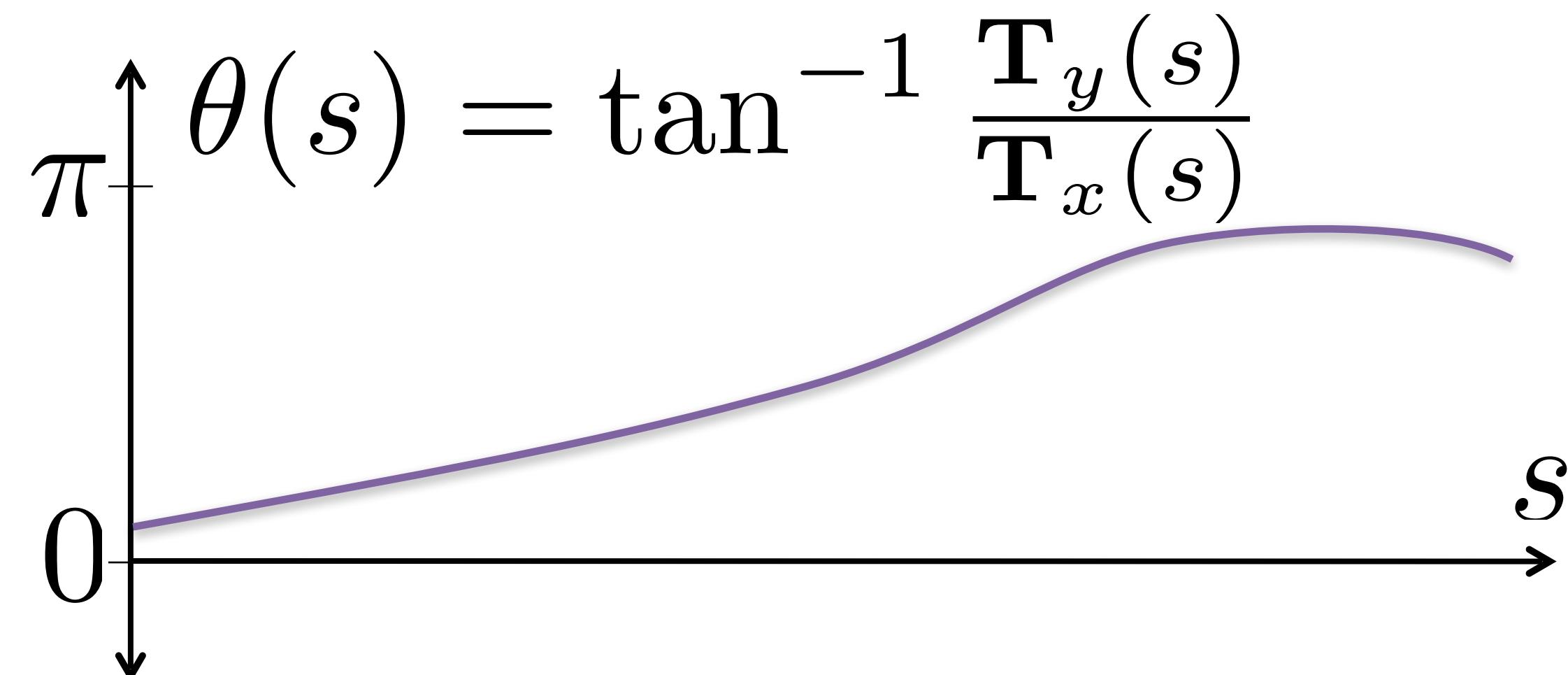
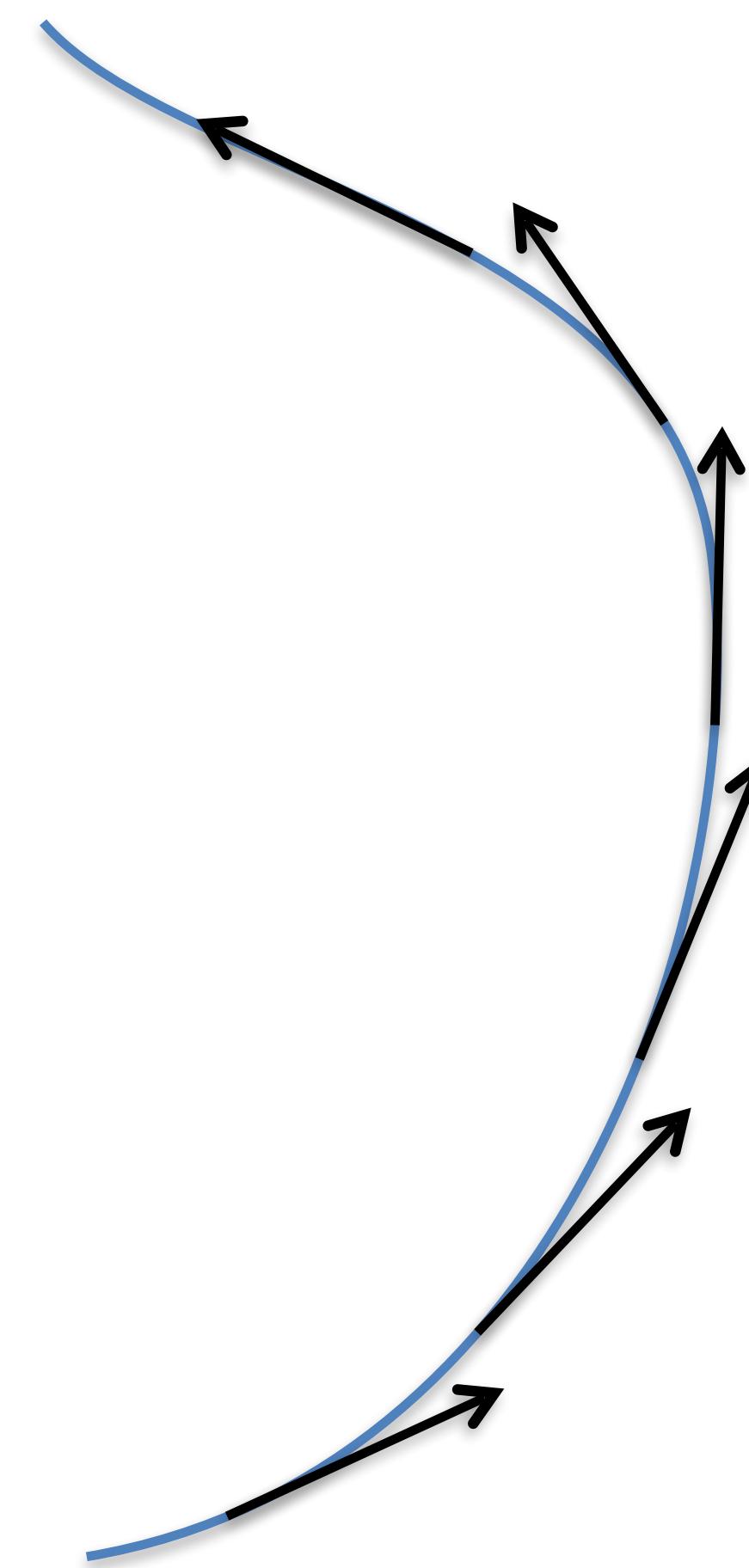
Arc length parameterization

- Every curve has a natural parameterization:
 $\mathbf{p}(s)$, such that $\|\mathbf{p}'(s)\| = 1$
- Isometry between parameter domain and curve
- Tangent vector is unit-length: $\mathbf{p}'(s) = \mathbf{T}(s)$



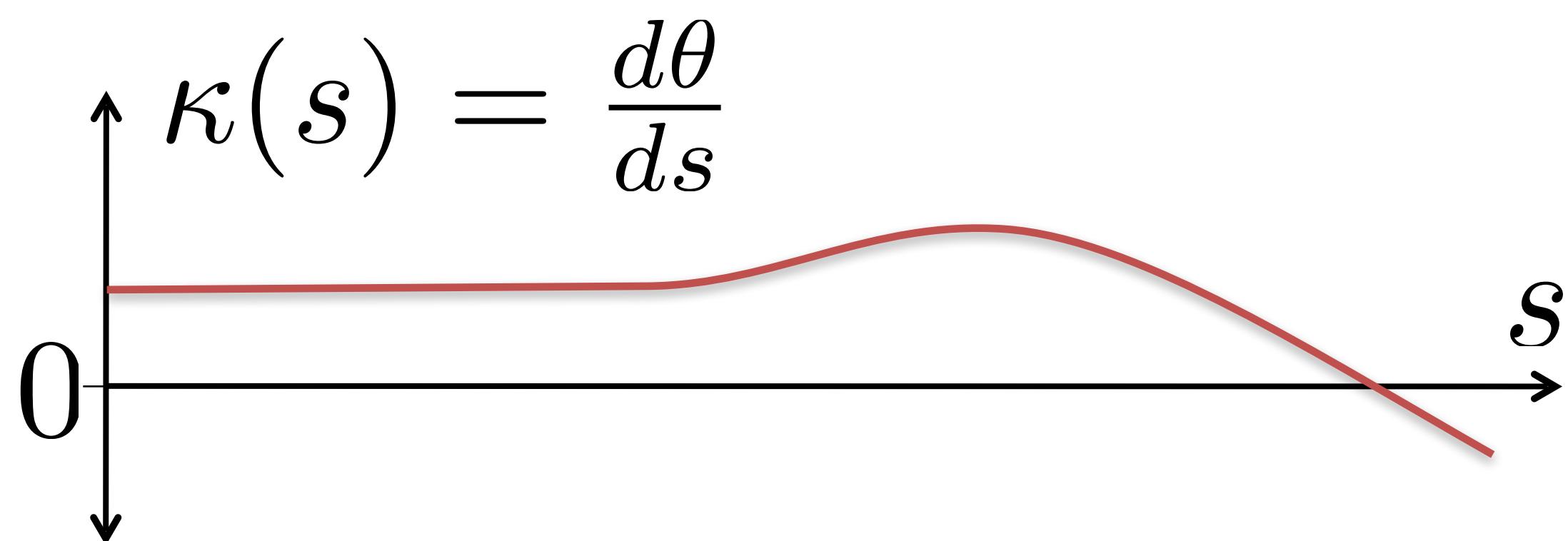
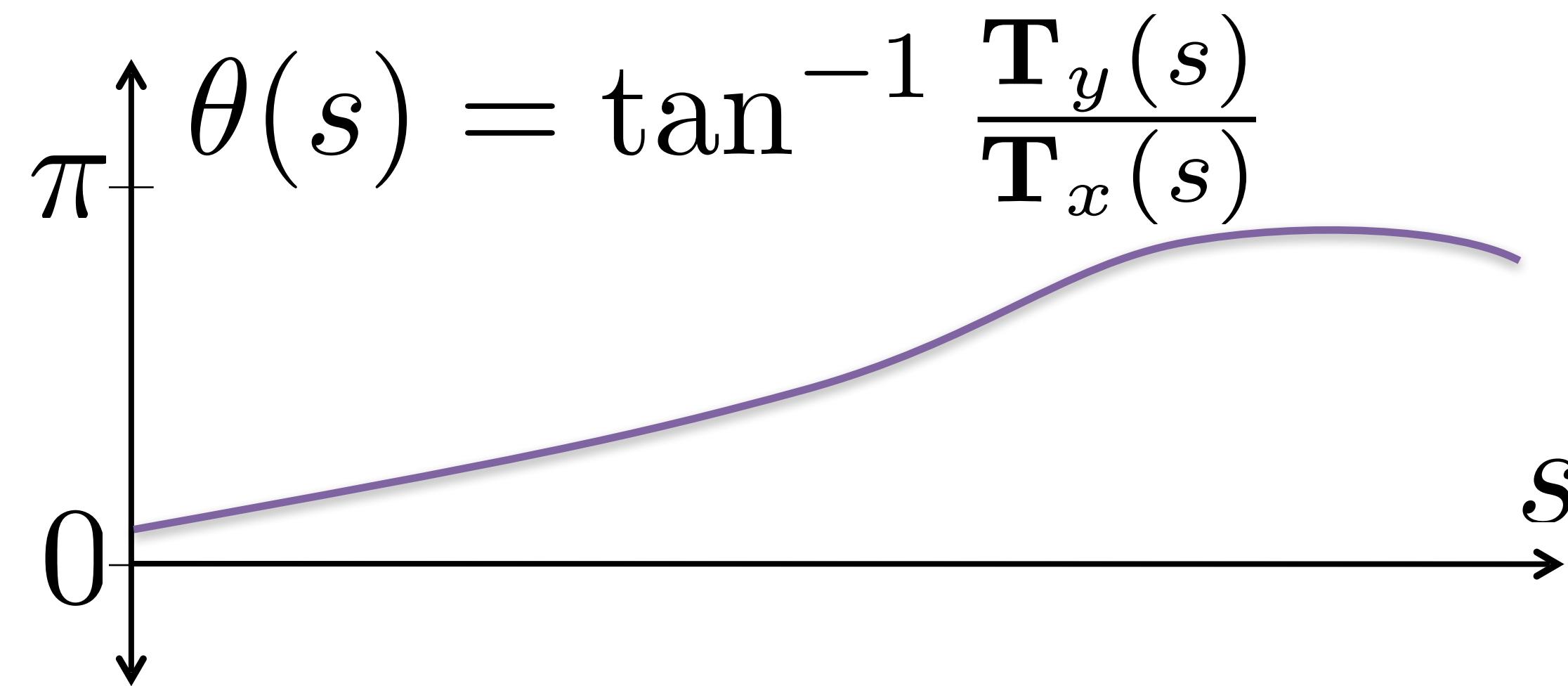
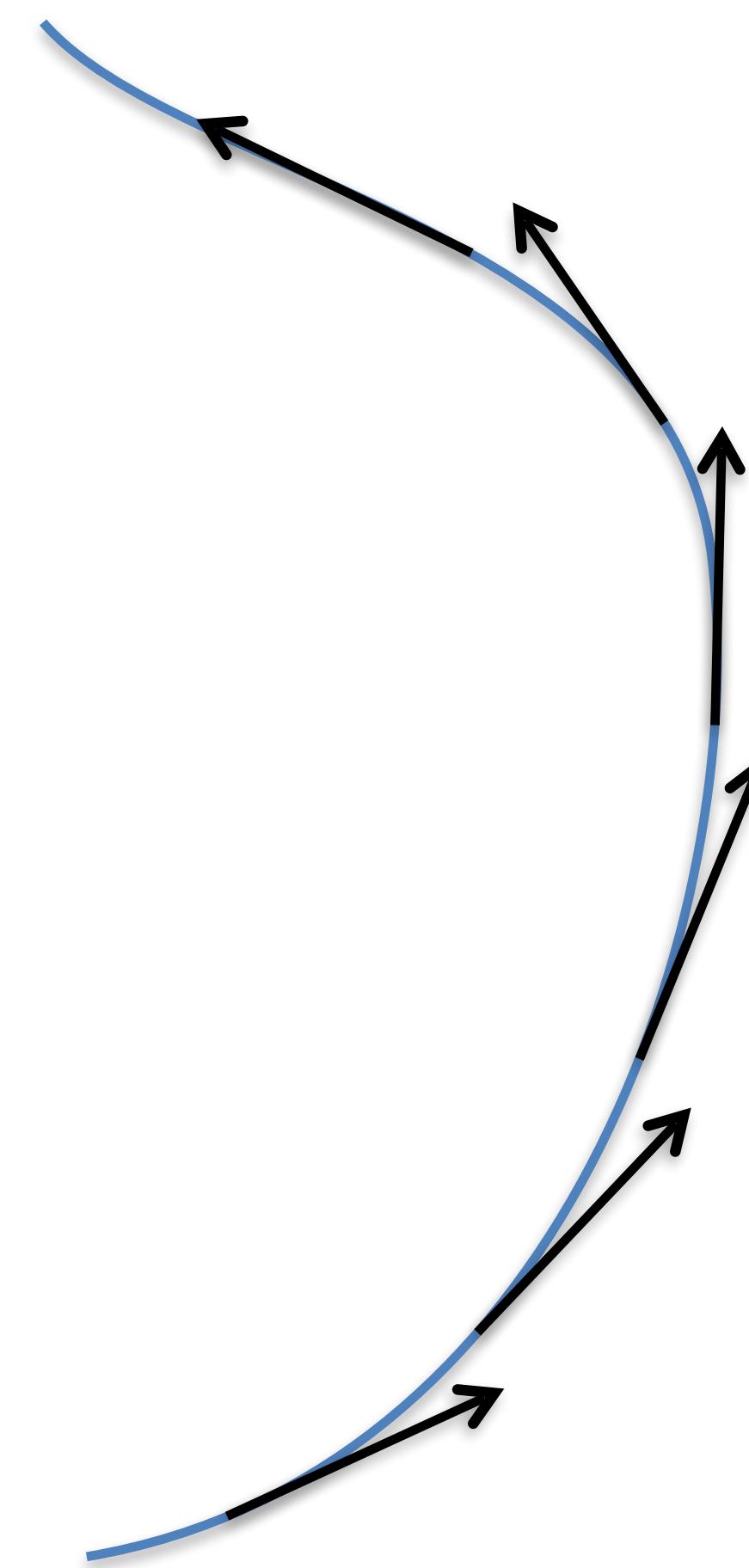
Curvature

- How much does the curve turn per unit s ?



Curvature

- How much does the curve turn per unit s ?



Curvature profile

- Given $\kappa(s)$, we can get $\theta(s)$ up to a constant by integration.
Integrating

$$\mathbf{p}(s) = \mathbf{p}_0 + \int_{s_0}^{s_1} \begin{pmatrix} \cos \theta(s) \\ \sin \theta(s) \end{pmatrix} ds$$

reconstructs the curve up to rigid motion.

Curvature of a circle

- Curvature of a circle:

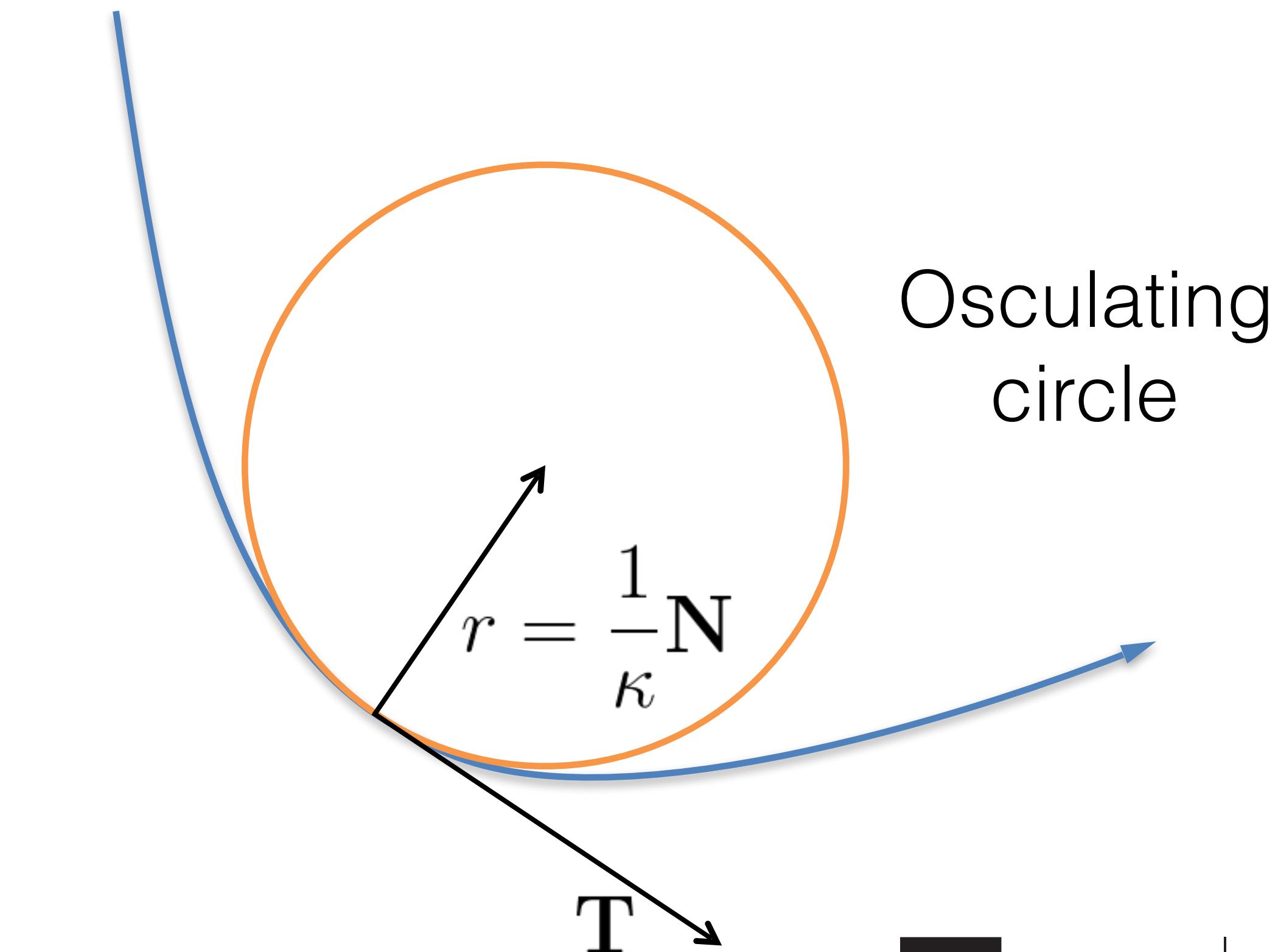
$$\mathbf{p}(s) = \begin{pmatrix} r \cos(s/r) \\ r \sin(s/r) \end{pmatrix}$$

$$\mathbf{p}'(s) = \begin{pmatrix} -\sin(s/r) \\ \cos(s/r) \end{pmatrix}$$

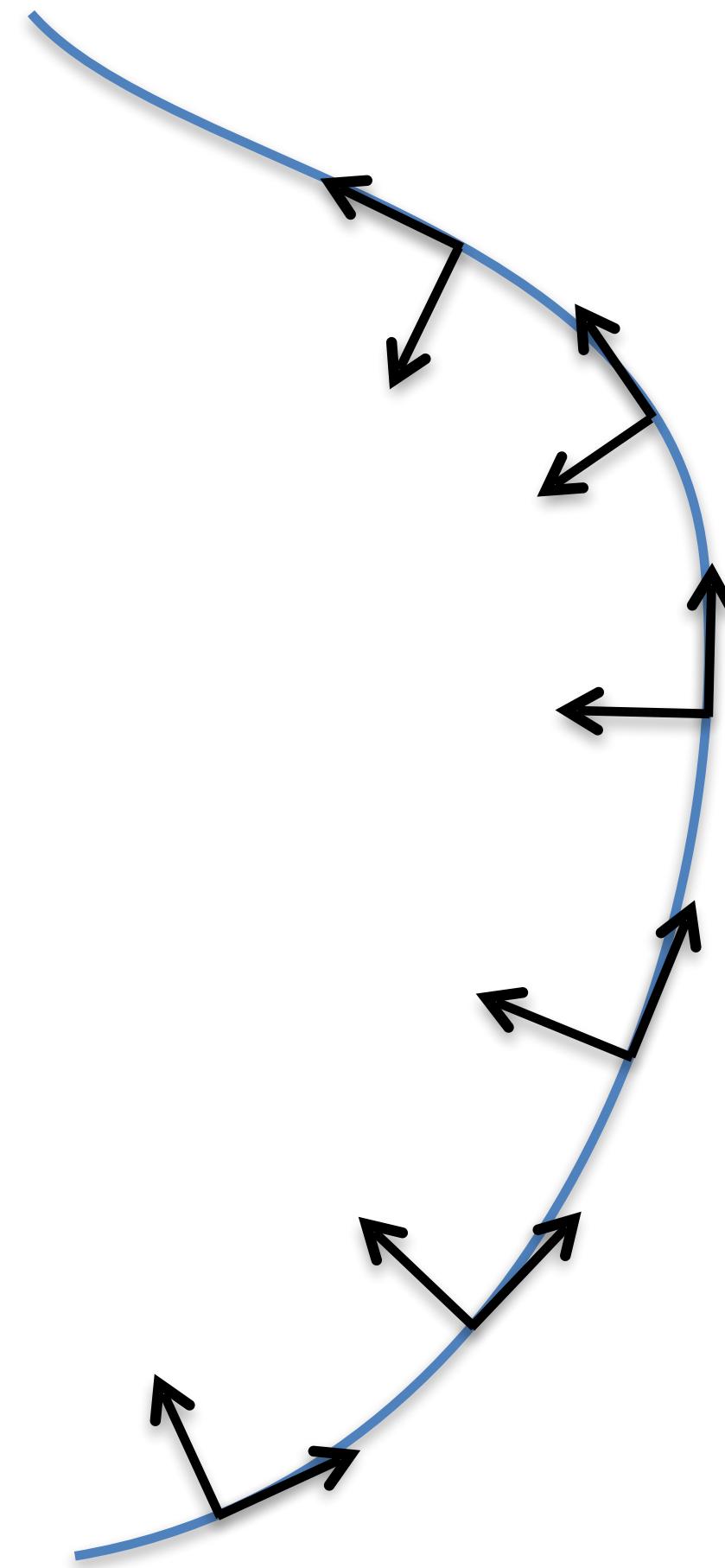
$$\begin{aligned}\theta(s) &= \tan^{-1} \frac{\cos(s/r)}{-\sin(s/r)} \\ &= s/r - \pi/2\end{aligned}$$

$$\kappa(s) = 1/r$$

$$\mathbf{N}(t) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{T}(t) = \text{Unit Normal}$$



Frenet Frame



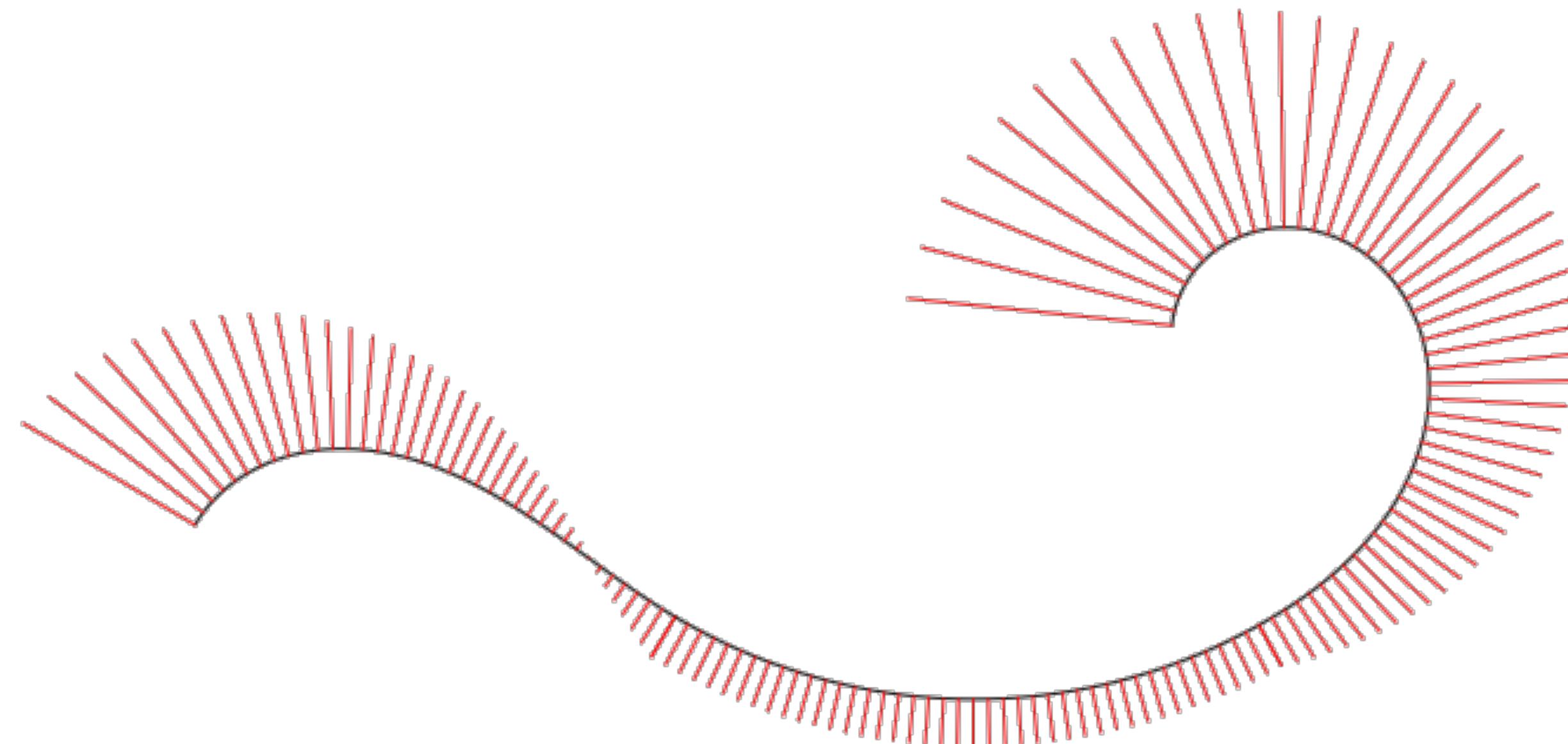
$$\mathbf{T}'(s) = \kappa(s)\mathbf{N}(s)$$

$$\mathbf{N}'(s) = -\kappa(s)\mathbf{T}(s)$$

$$\begin{pmatrix} \mathbf{T}' \\ \mathbf{N}' \end{pmatrix} = \begin{pmatrix} 0 & \kappa \\ -\kappa & 0 \end{pmatrix} \begin{pmatrix} \mathbf{T} \\ \mathbf{N} \end{pmatrix}$$

Curvature normal

- Points inward
- $-\kappa(s)\mathbf{N}(s)$ useful for evaluating curve quality



Smoothness

Two kinds, parametric and geometric:

C^1 : $\mathbf{p}(t)$ is continuously differentiable

G^1 : $\mathbf{p}(s)$ is continuously differentiable

Parametrization-Independent

$$C^1 \quad \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$

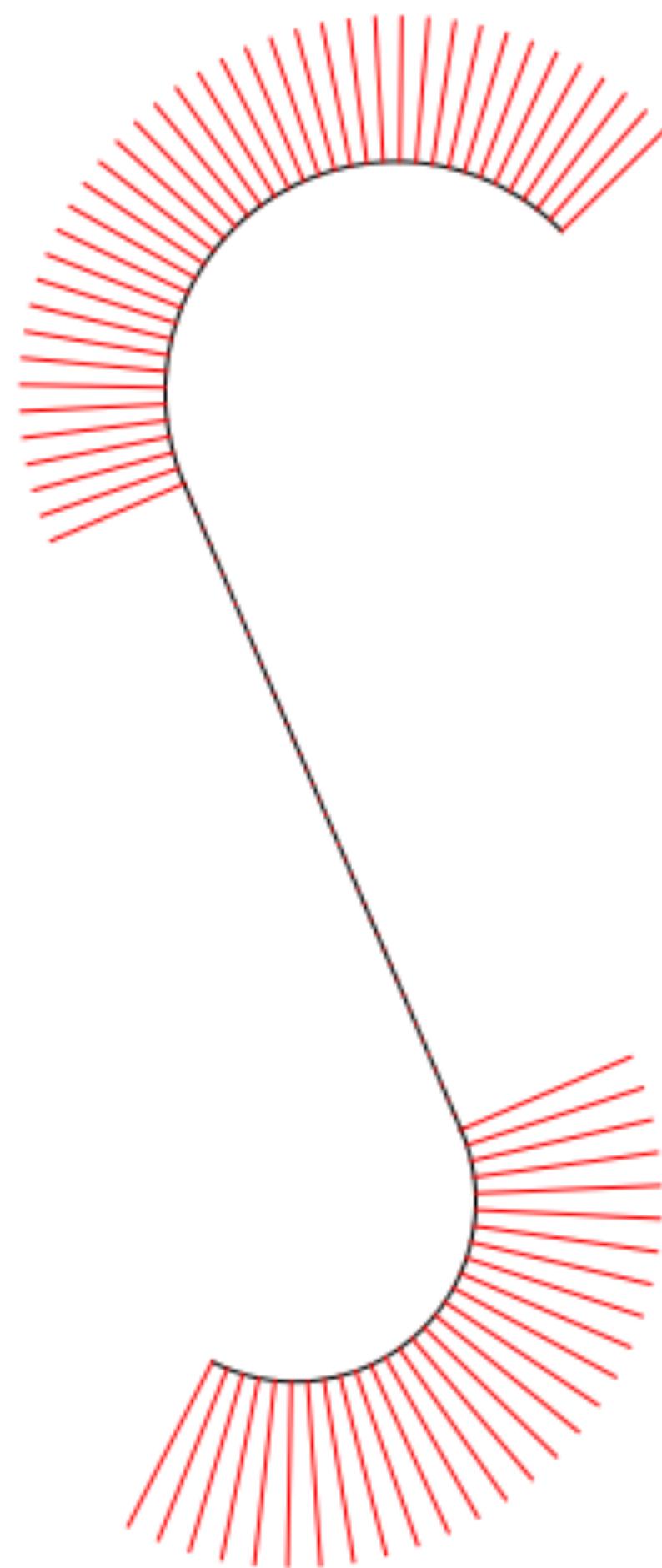
$$G^1 \quad \begin{pmatrix} \cos \hat{t} \\ \sin \hat{t} \end{pmatrix}, \quad \hat{t} = \begin{cases} t + 1 & \text{if } t < 1 \\ 2t & \text{if } t \geq 1 \end{cases}$$

Smoothness example

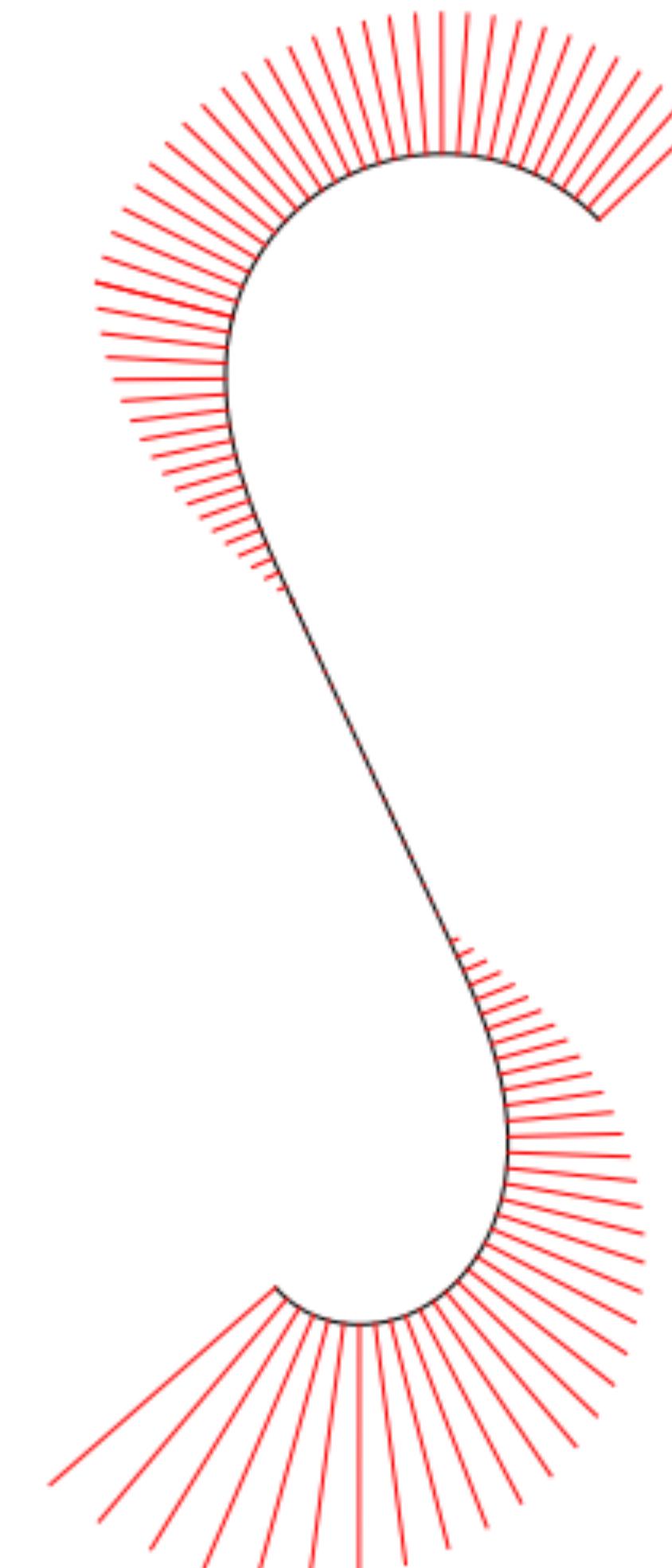
G^0



G^1

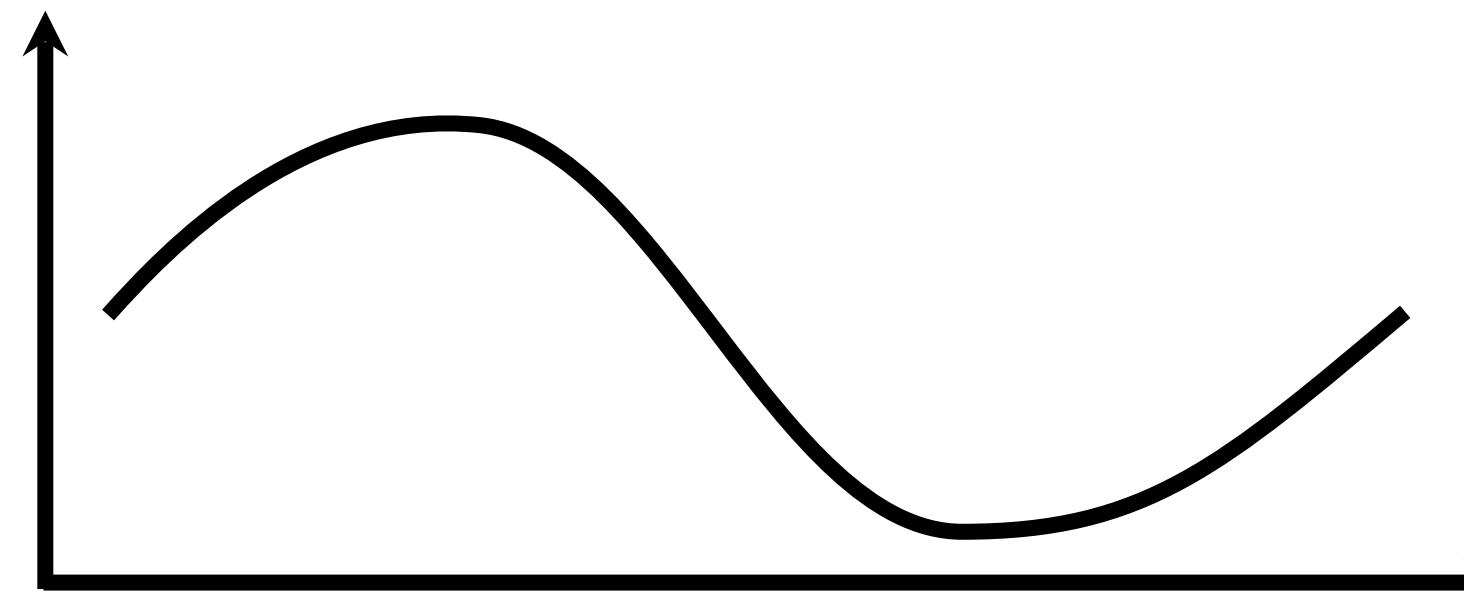


G^2



Recap on parametric curves

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



$$\|\mathbf{p}'(t)\| = \text{speed}$$

$$s(t) = \int_{t_0}^t \|\mathbf{p}'(t)\| dt$$

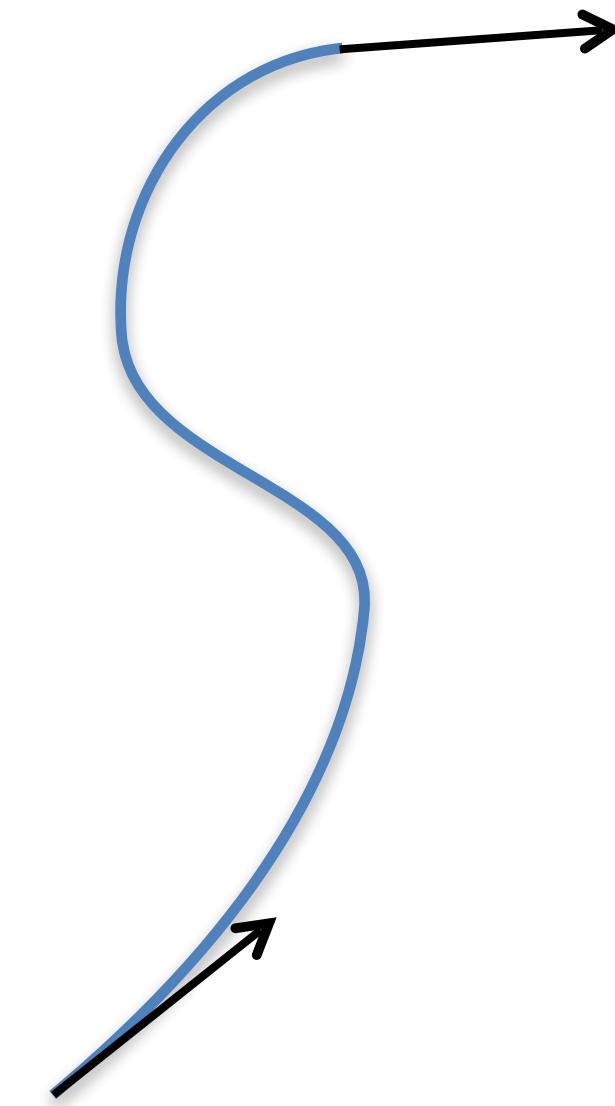
$$\kappa(s) = \frac{d\theta}{ds} = \mathbf{T}'(s) \cdot \mathbf{N}(s)$$

$$\frac{\mathbf{p}'(t)}{\|\mathbf{p}'(t)\|} = \mathbf{T}(t)$$

Turning

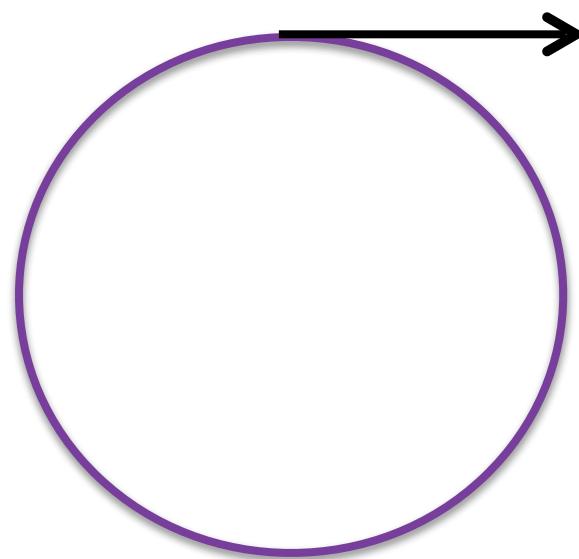
- Angle from start tangent to end tangent:

$$\int_{s_0}^{s_1} \kappa(s) ds = \int_{t_0}^{t_1} \kappa(t) \|\mathbf{p}'(t)\| dt$$



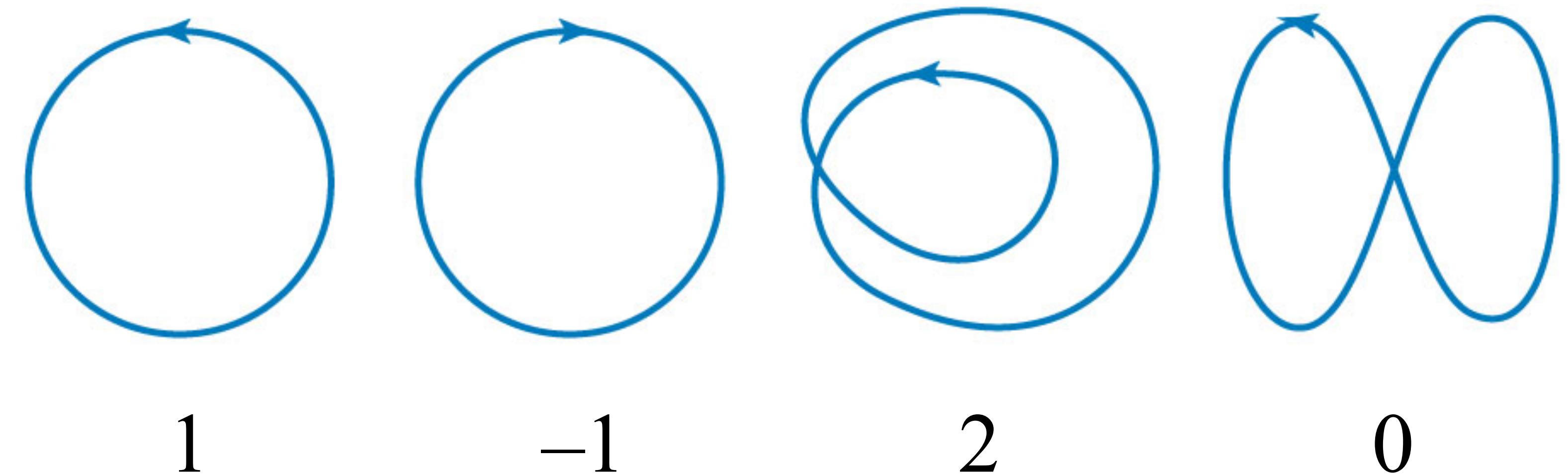
- If curve is closed, the tangent at the beginning is the same as the tangent at the end

$$\oint_{\mathbf{p}} \kappa(s) ds = 2\pi n$$



Turning numbers

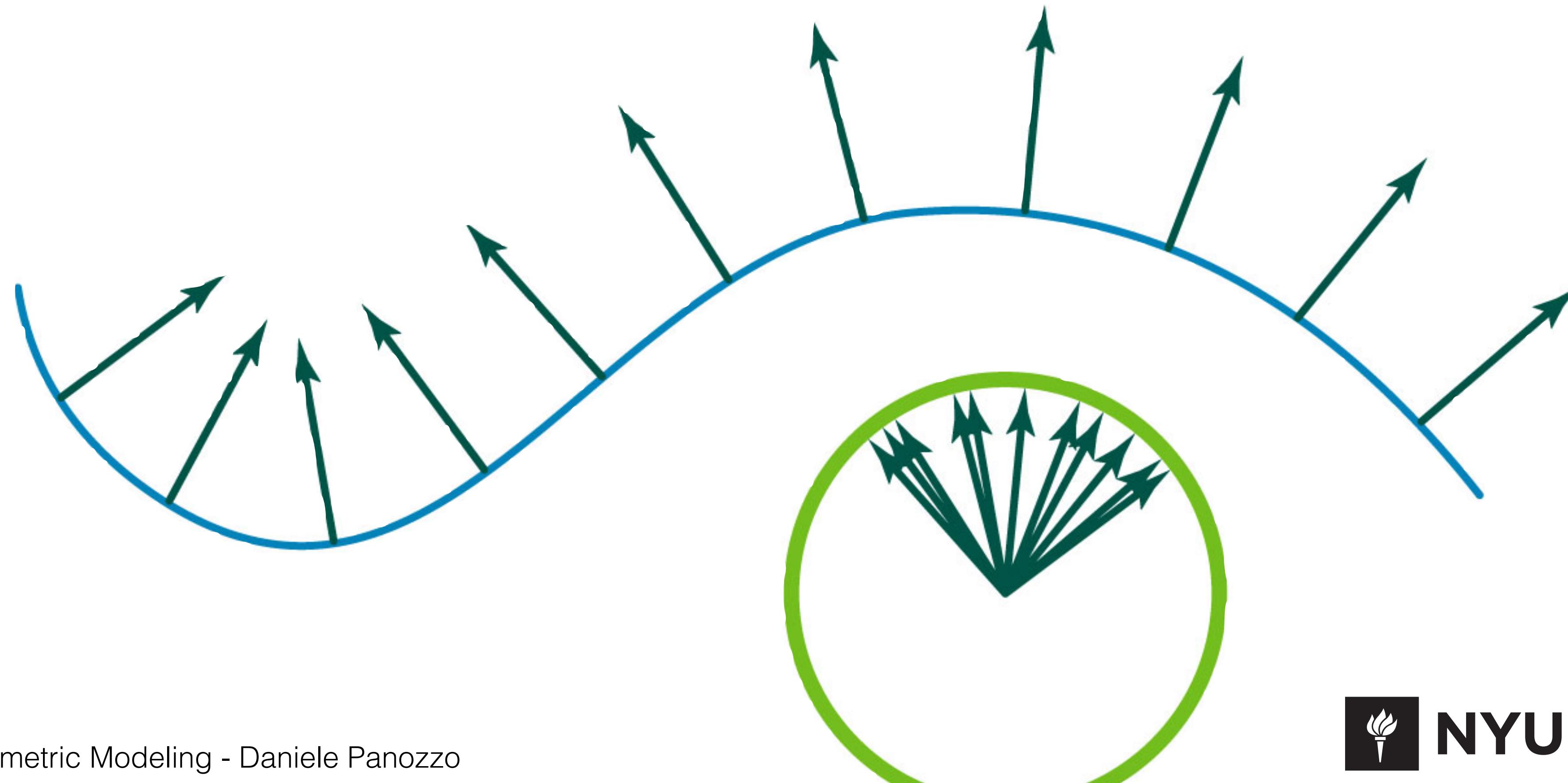
$$\oint_{\mathbf{P}} \kappa(s) ds = 2\pi n$$



- n measures how many full turns the tangent makes.

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

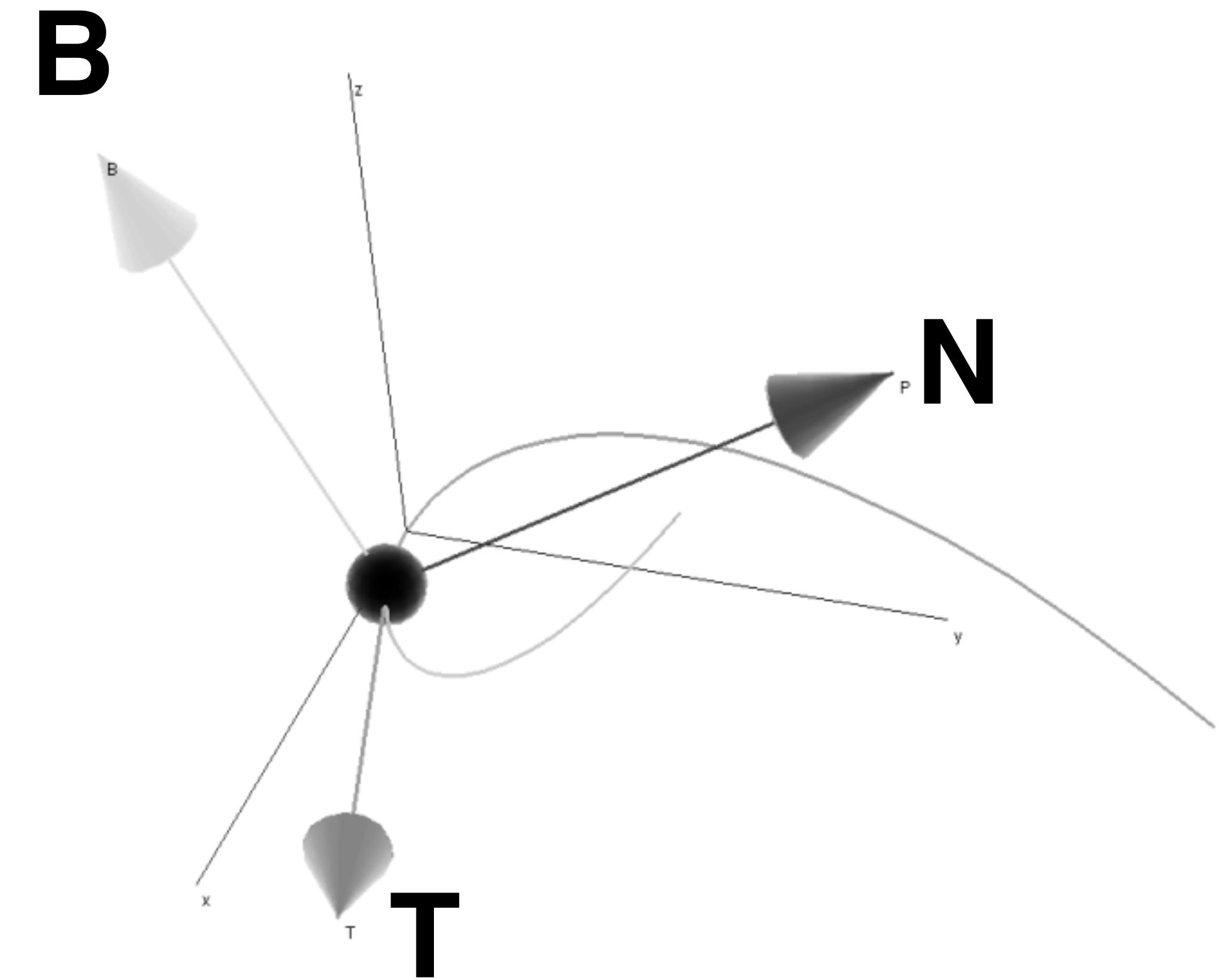
- Point on curve maps to point on unit circle.



Space curves (3D)

- In 3D, many vectors are orthogonal to \mathbf{T}
 $\mathbf{N}(s) := \mathbf{T}'(s)/\|\mathbf{T}'(s)\|$
 $\mathbf{B}(s) := \mathbf{T}(s) \times \mathbf{N}(s)$
- $\mathbf{T}, \mathbf{N}, \mathbf{B}$ are the “Frenet frame”
- τ is torsion: non-planarity

$$\begin{bmatrix} \mathbf{T}' \\ \mathbf{N}' \\ \mathbf{B}' \end{bmatrix} = \begin{bmatrix} \kappa & & \tau \\ -\kappa & & \\ & -\tau & \end{bmatrix} \begin{bmatrix} \mathbf{T} \\ \mathbf{N} \\ \mathbf{B} \end{bmatrix}$$



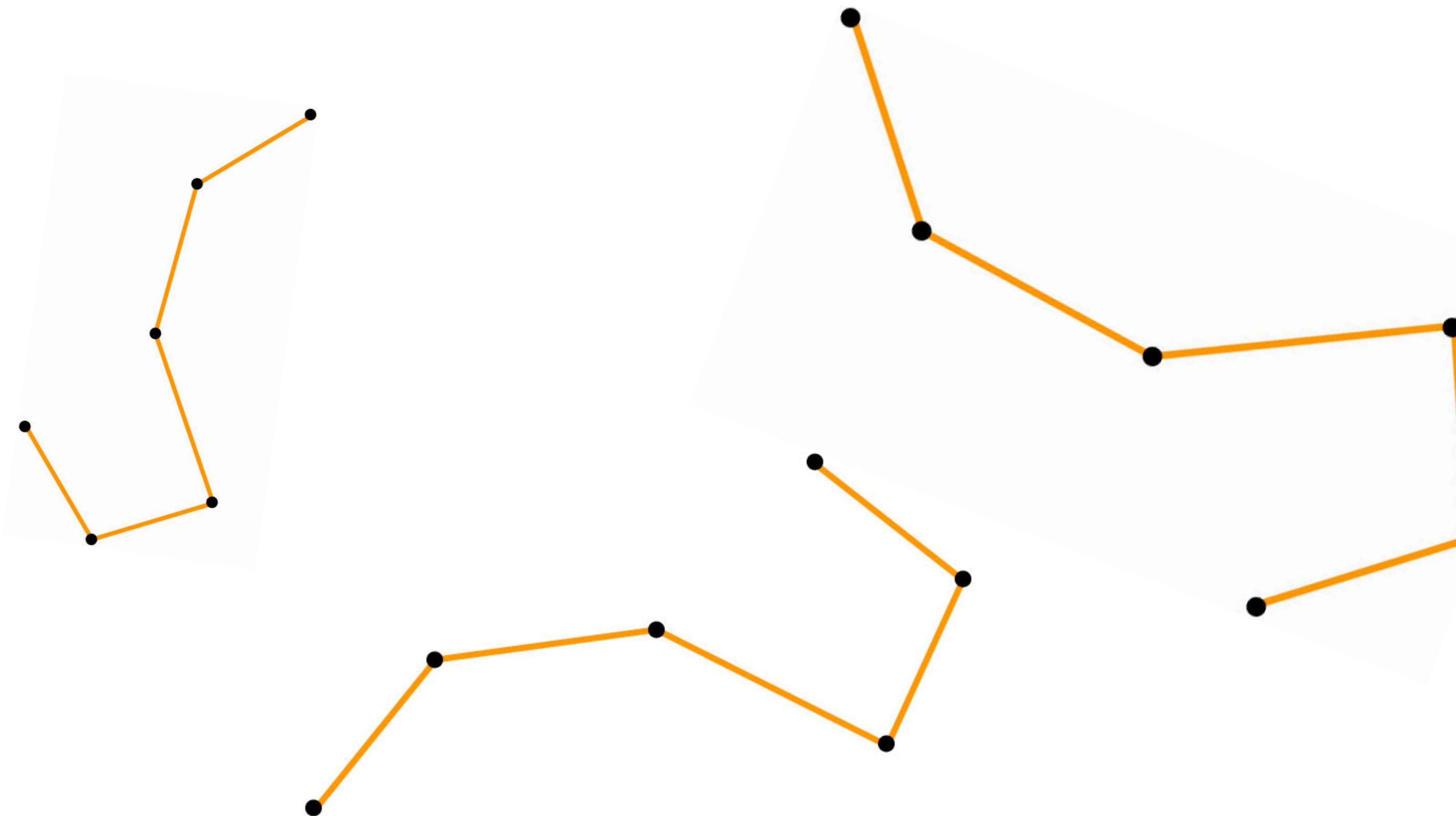
Discrete Differential Geometry of Curves

Some references: see
<http://ddg.cs.columbia.edu/>



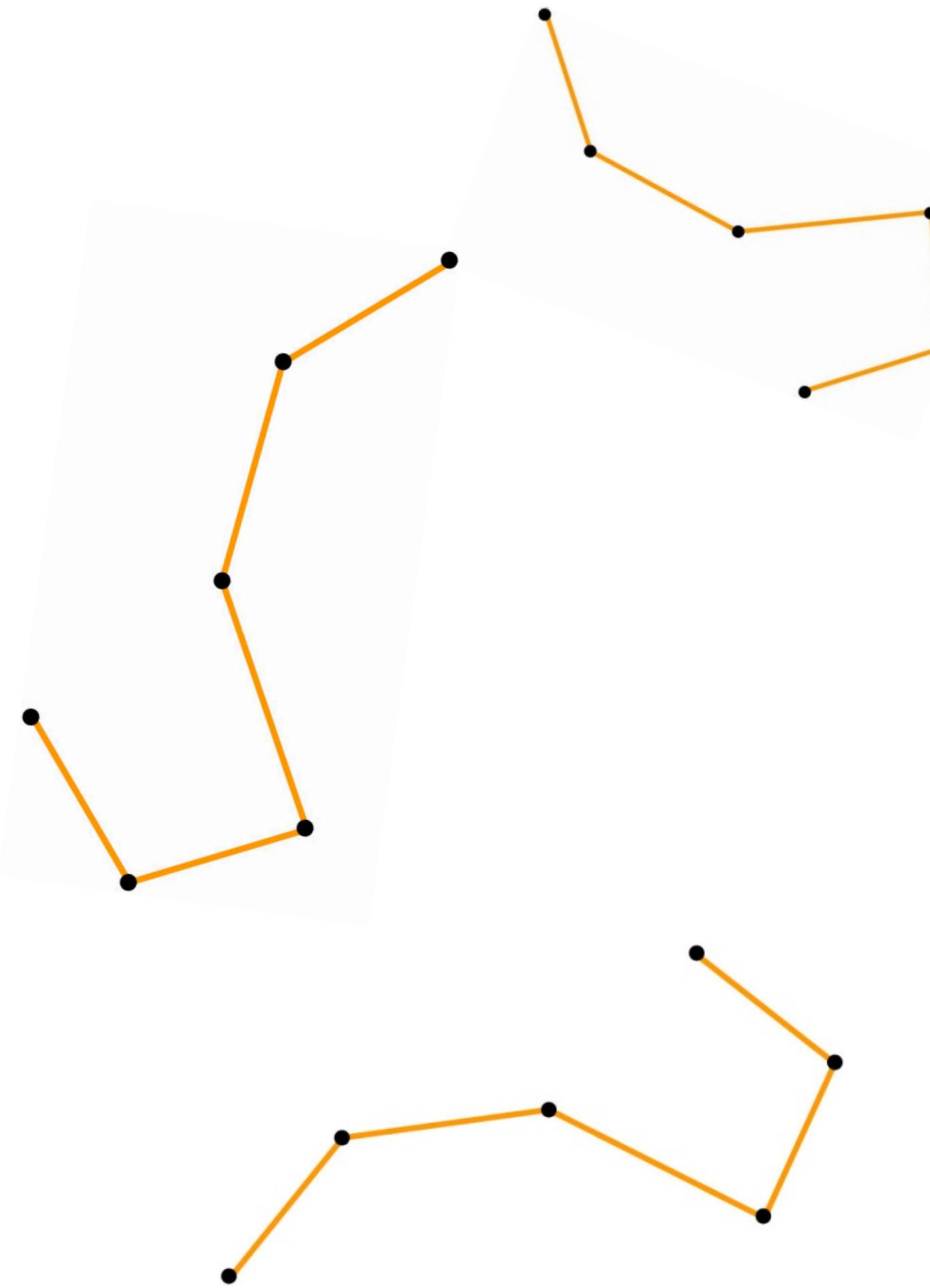
NYU | COURANT

Discrete Planar Curves



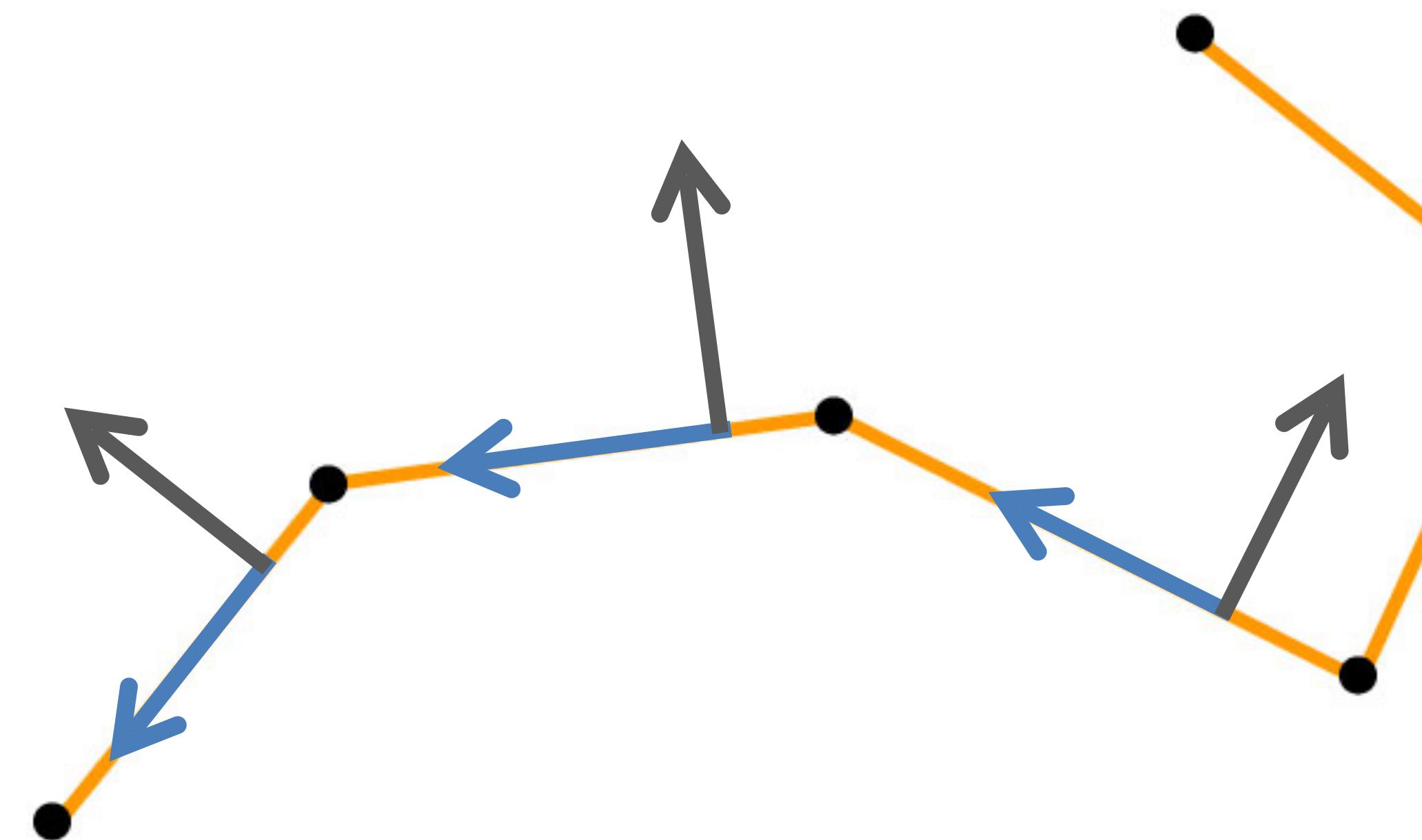
Discrete Planar Curves

- Piecewise linear curves
- Not smooth at vertices
- Can't take derivatives
- Generalize notions from the smooth world for the discrete case!
- **There is no one single way**



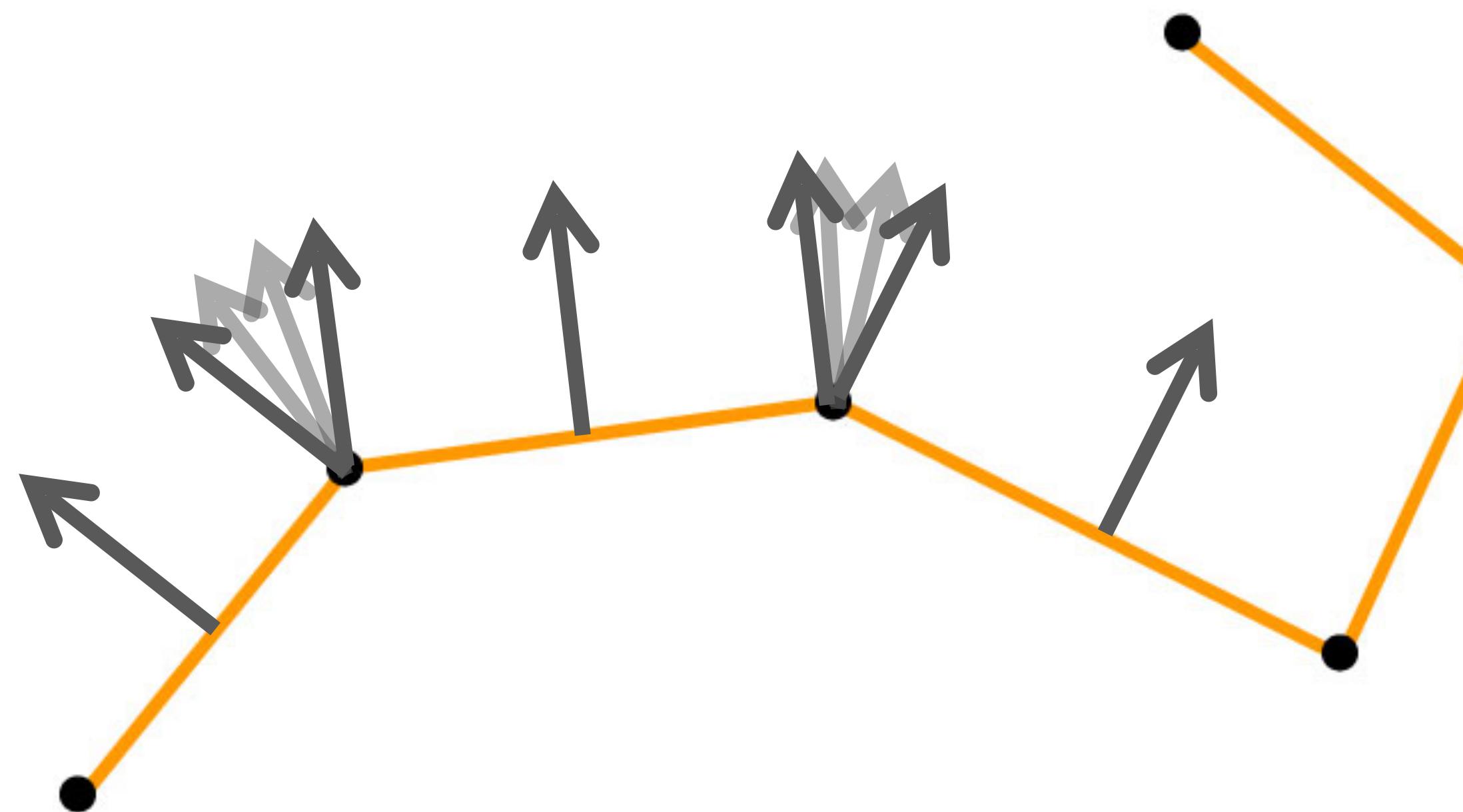
Tangents, Normals

- For any point on the edge, the tangent is simply the unit vector along the edge and the normal is the perpendicular vector



Tangents, Normals

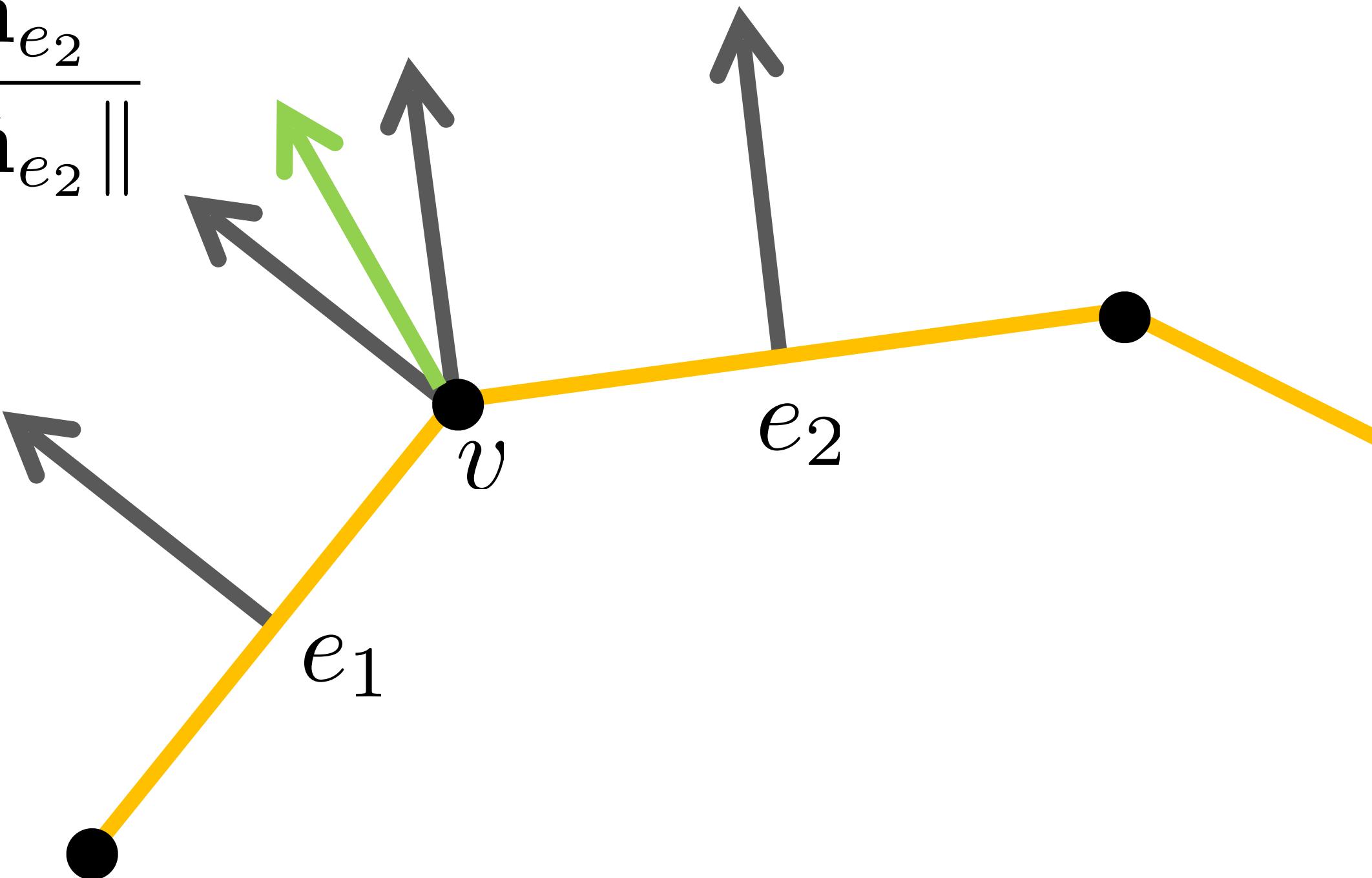
- For vertices, we have many options



Tangents, Normals

- Can choose to average the adjacent edge normals

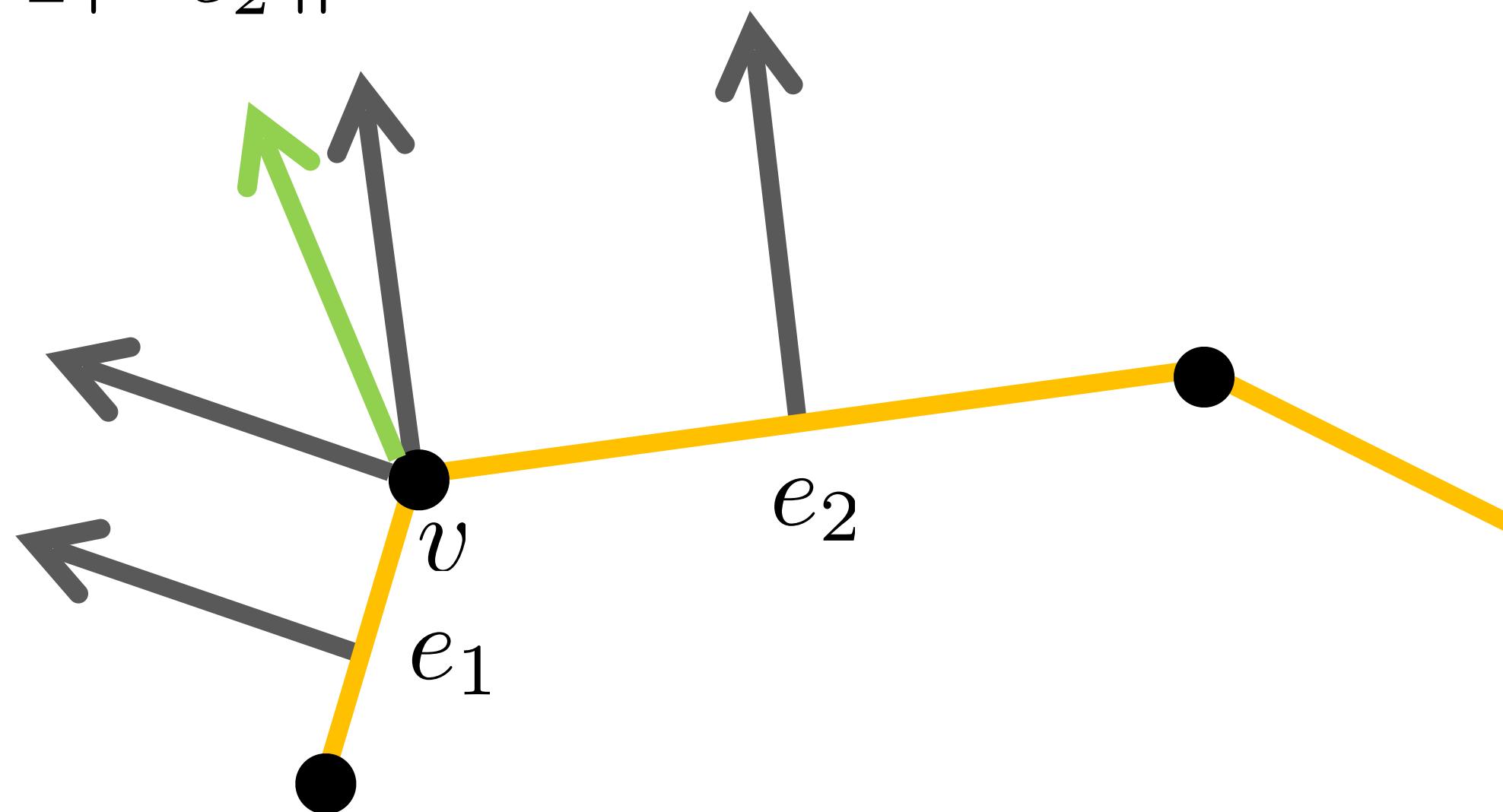
$$\hat{\mathbf{n}}_v = \frac{\hat{\mathbf{n}}_{e_1} + \hat{\mathbf{n}}_{e_2}}{\|\hat{\mathbf{n}}_{e_1} + \hat{\mathbf{n}}_{e_2}\|}$$



Tangents, Normals

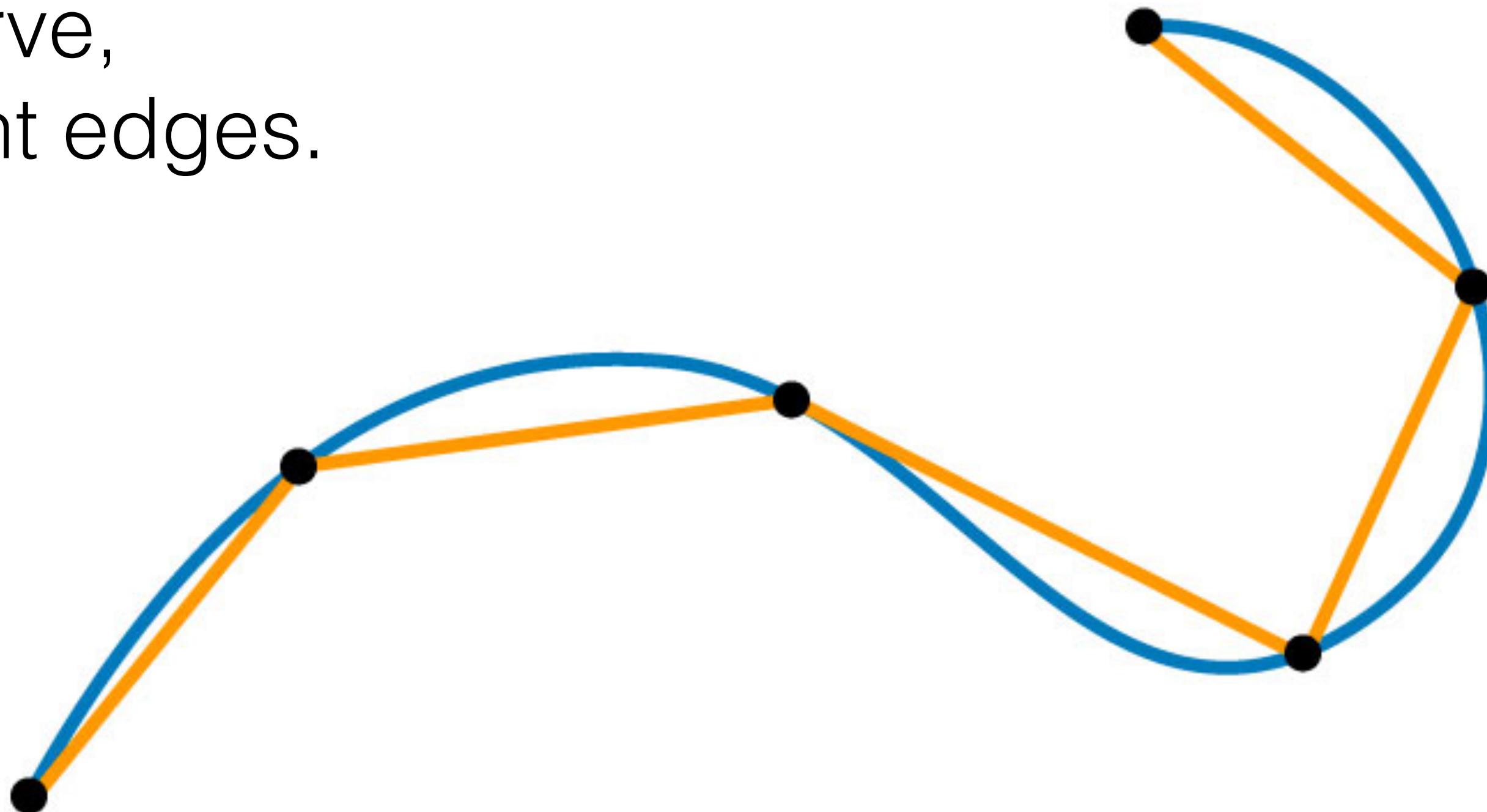
- Weight by edge lengths

$$\hat{\mathbf{n}}_v = \frac{|e_1| \hat{\mathbf{n}}_{e_1} + |e_2| \hat{\mathbf{n}}_{e_2}}{\| |e_1| \hat{\mathbf{n}}_{e_1} + |e_2| \hat{\mathbf{n}}_{e_2} \|}$$



Inscribed Polygon, p

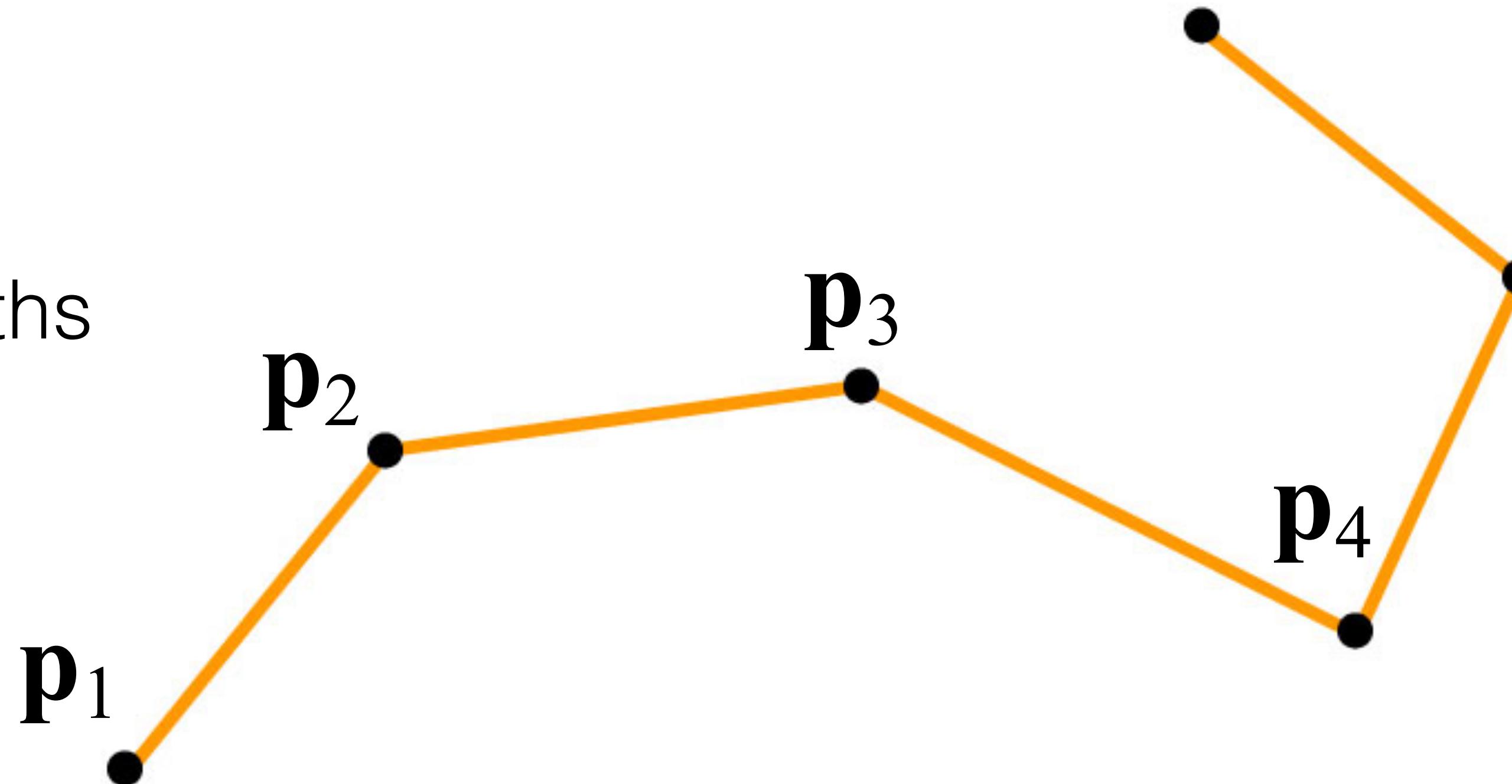
- Connection between discrete and smooth
- Finite number of vertices each lying on the curve, connected by straight edges.



The Length of a Discrete Curve

$$\text{len}(p) = \sum_{i=1}^{n-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|$$

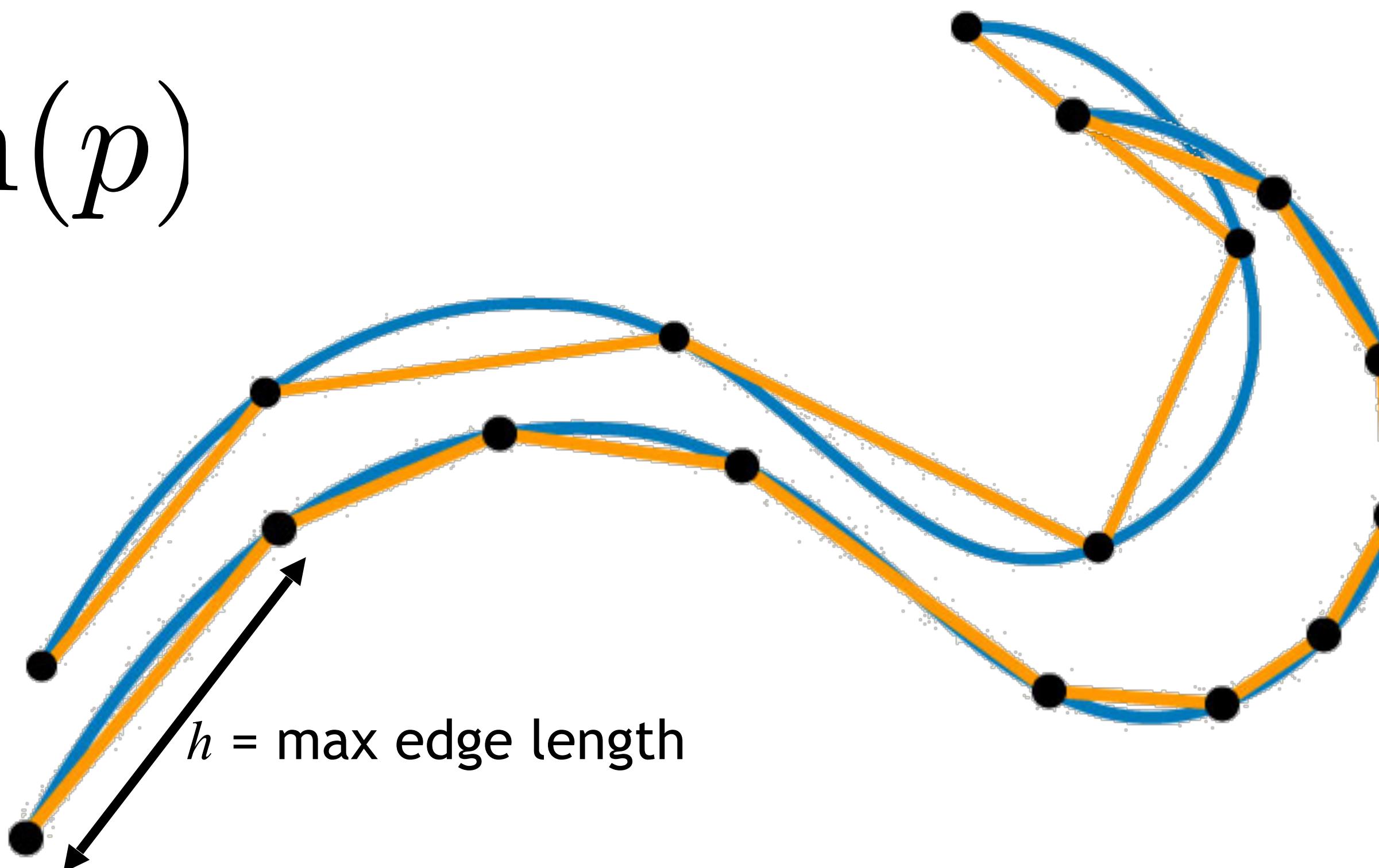
- Sum of edge lengths



The Length of a Continuous Curve

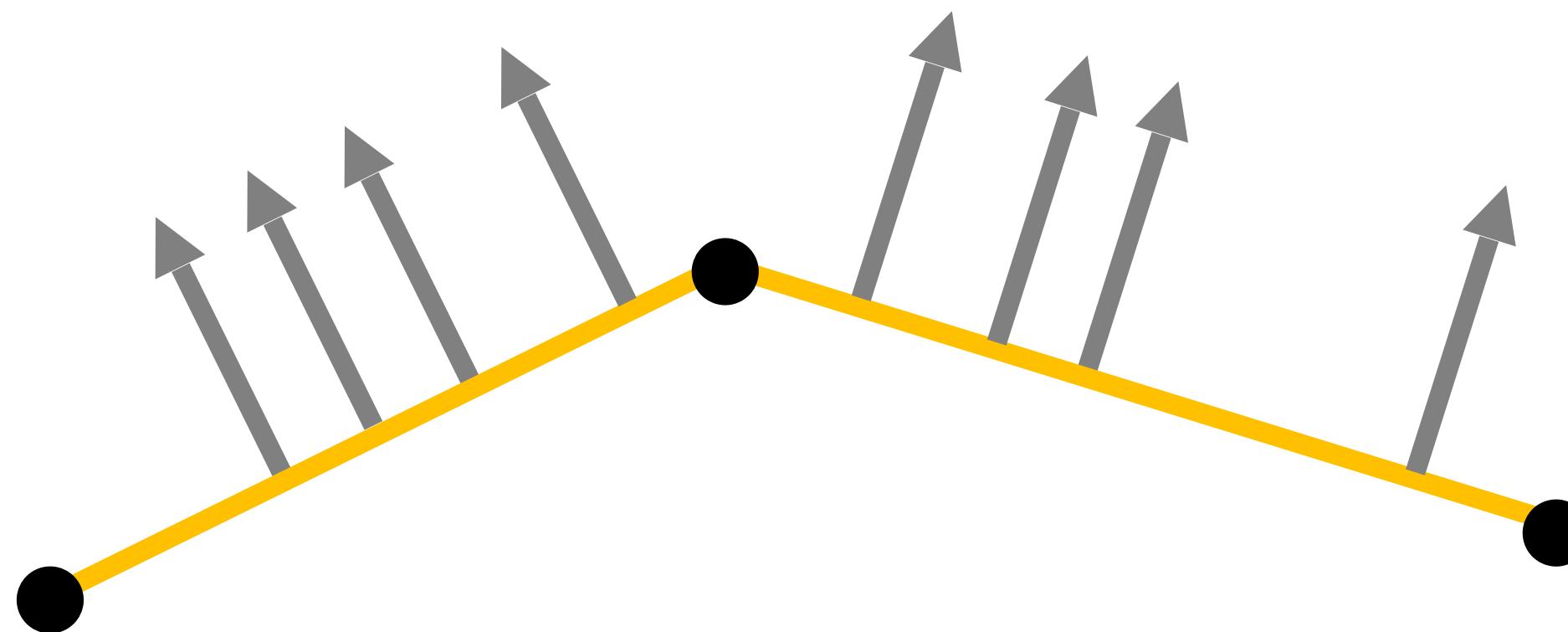
- Take limit over a refinement sequence

$$\lim_{h \rightarrow 0} \text{len}(p)$$



Curvature of a Discrete Curve

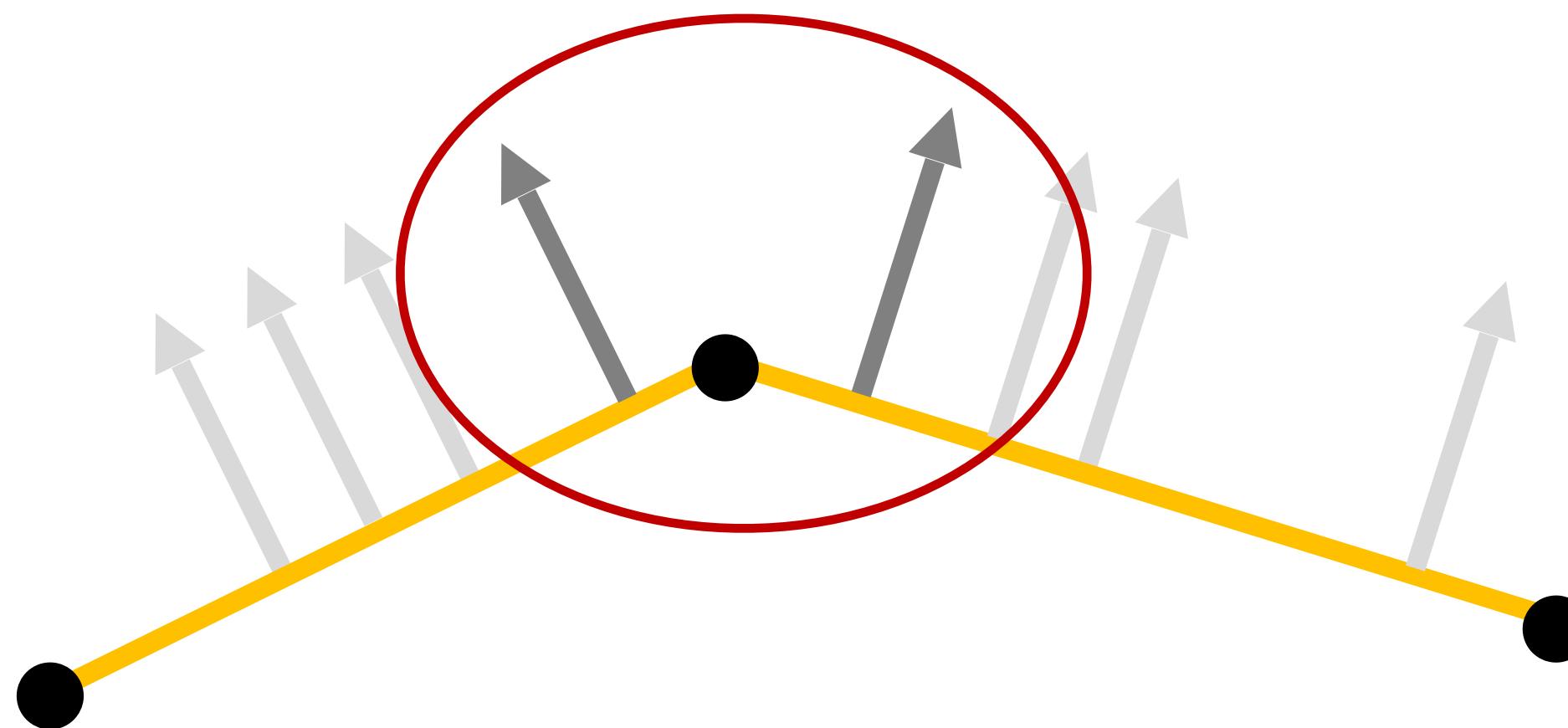
- Curvature is the change in normal direction as we travel along the curve



no change along each edge -
curvature is zero along edges

Curvature of a Discrete Curve

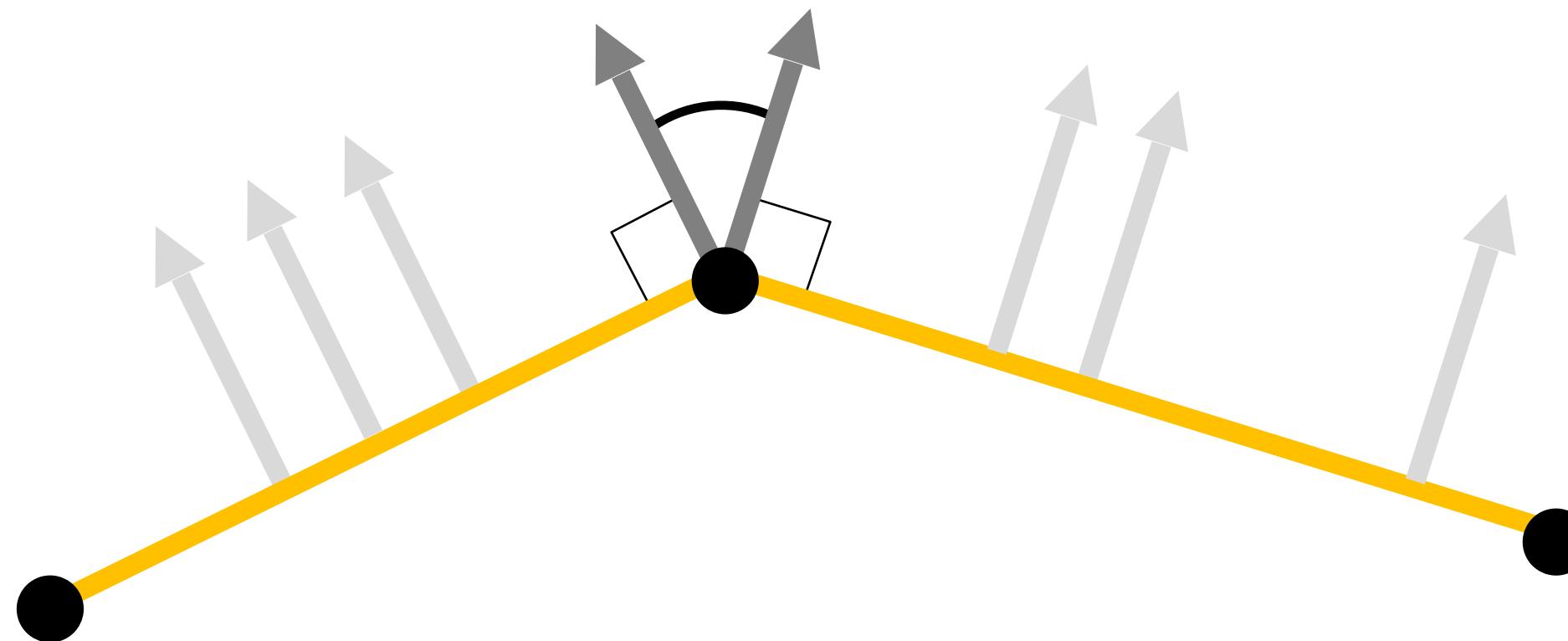
- Curvature is the change in normal direction as we travel along the curve



normal changes at vertices -
record the turning angle!

Curvature of a Discrete Curve

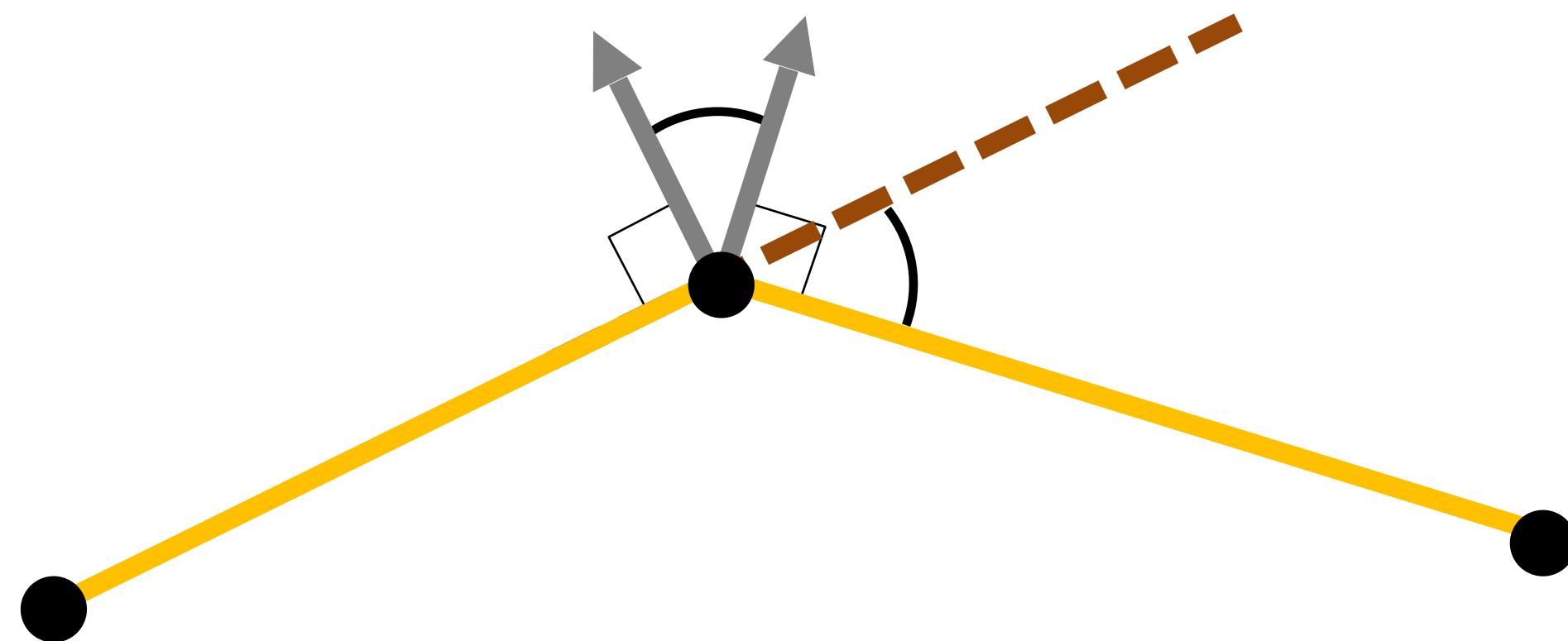
- Curvature is the change in normal direction as we travel along the curve



normal changes at vertices -
record the turning angle!

Curvature of a Discrete Curve

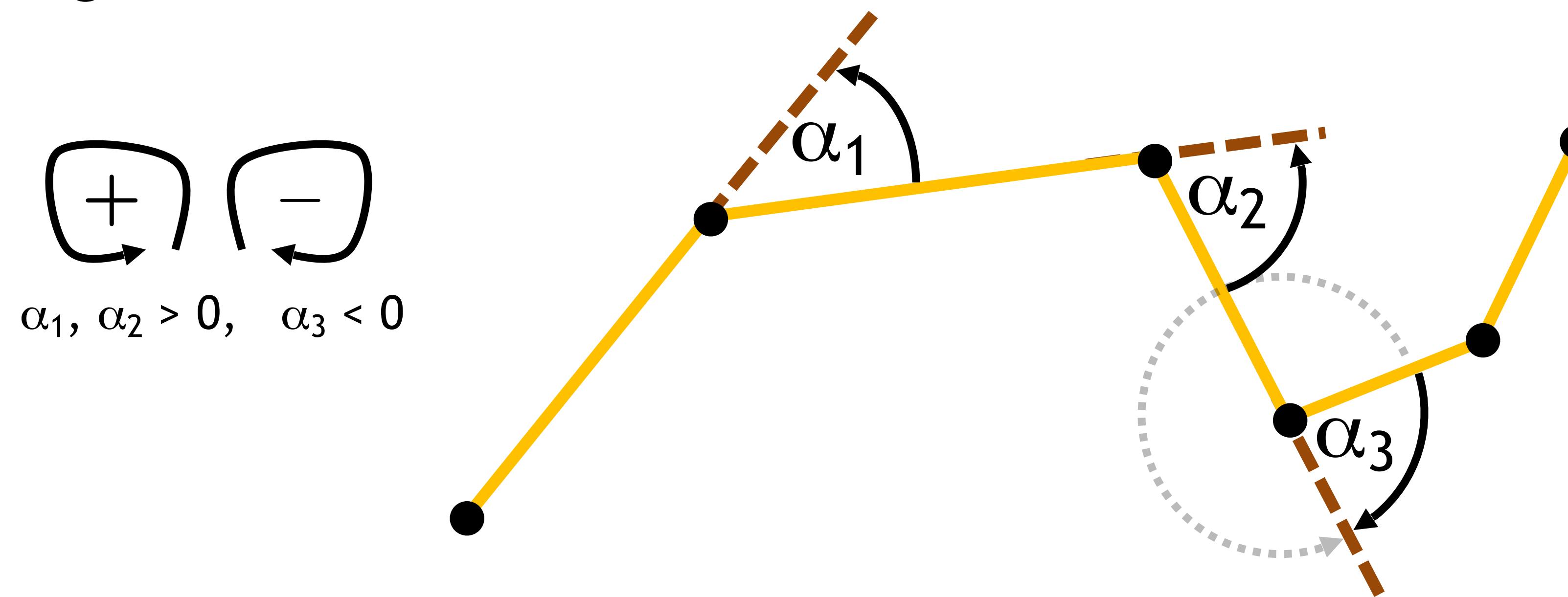
- Curvature is the change in normal direction as we travel along the curve



same as the turning angle
between the edges

Curvature of a Discrete Curve

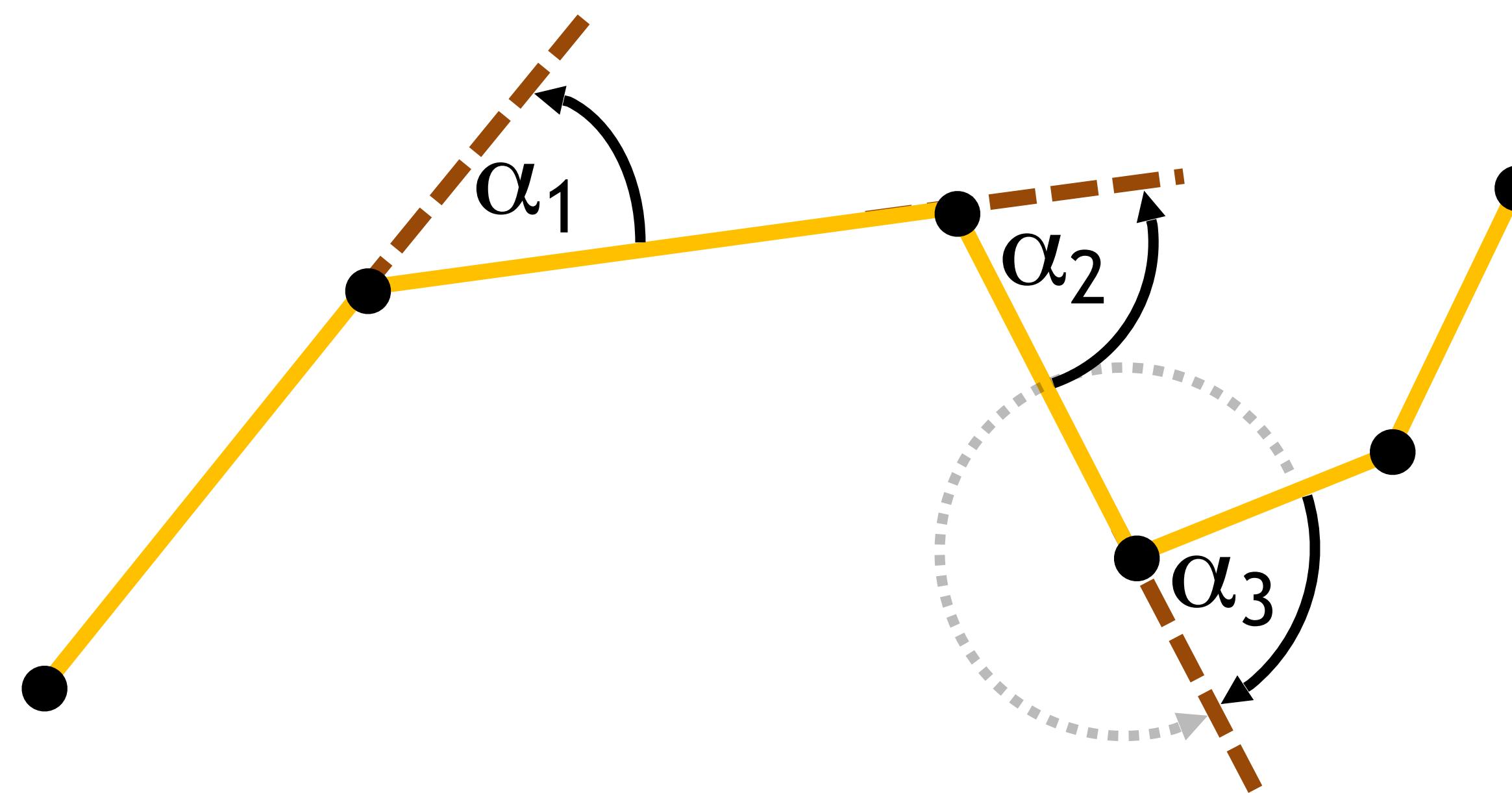
- Zero along the edges
- Turning angle at the vertices
= the change in normal direction



Total Signed Curvature

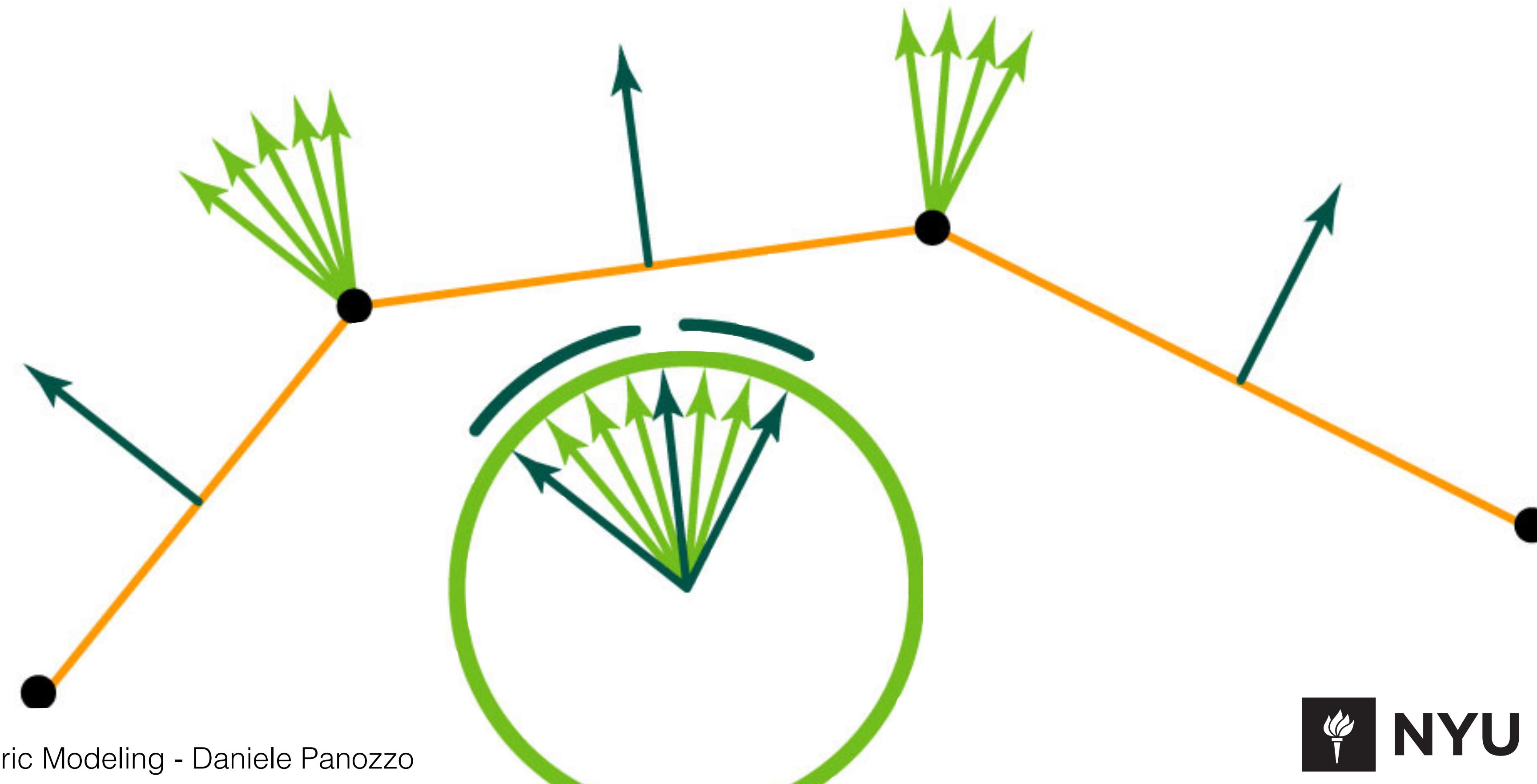
$$\text{tsc}(p) = \sum_{i=1}^n \alpha_i$$

- Sum of turning angles



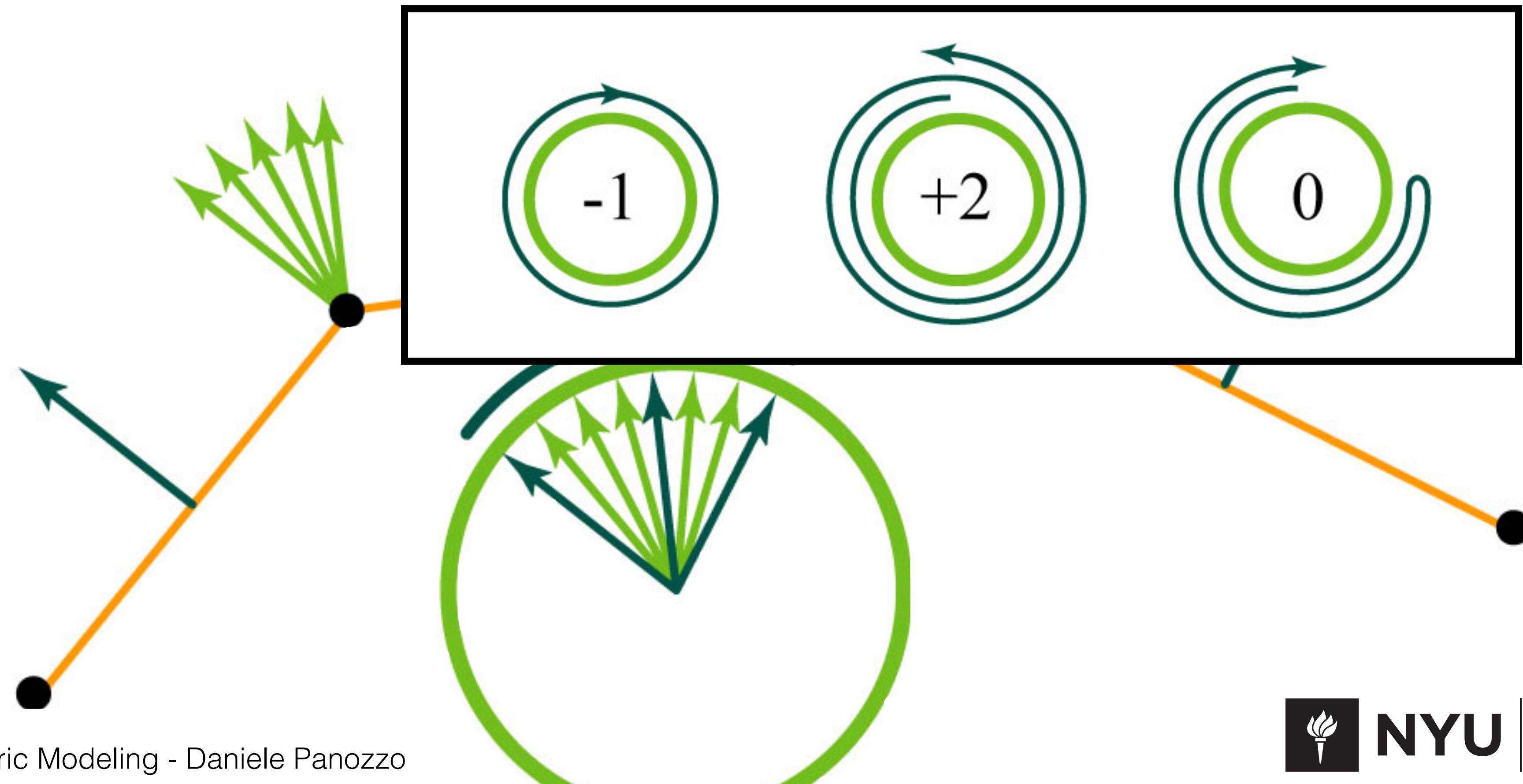
Discrete Gauss Map

- Edges map to points, vertices map to arcs.



Discrete Gauss Map

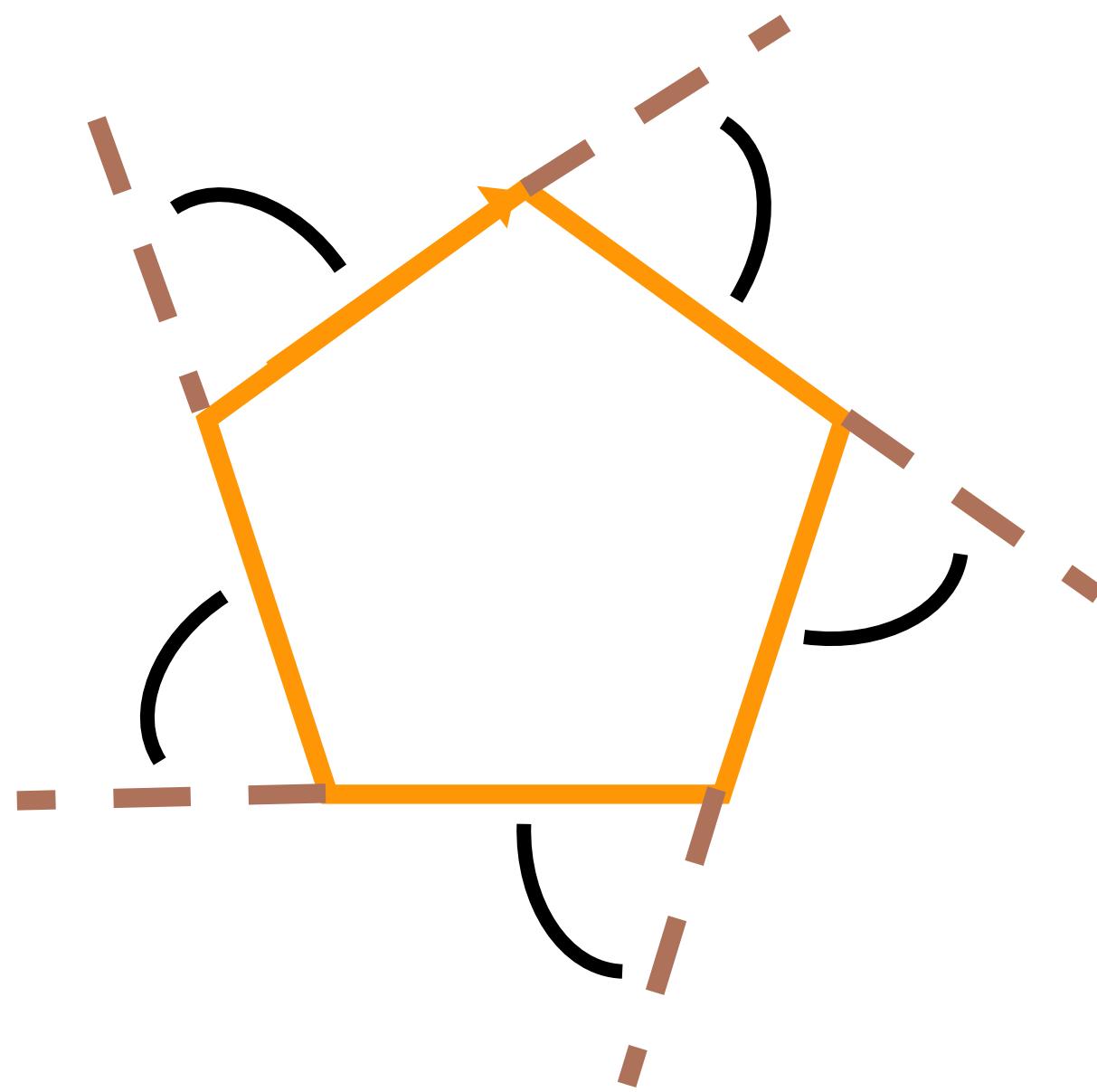
- Turning number well-defined for discrete curves.



Discrete Turning Number Theorem

$$\text{tsc}(p) = \sum_{i=1}^n \alpha_i = 2\pi k$$

- For a closed curve,
the total signed curvature is
an integer multiple of 2π .
 - proof: sum of exterior angles



Turning Number Theorem

Continuous world

$$\int_{\gamma} \kappa dt = 2\pi k$$

$k:$



Discrete world

$$\sum_{i=1}^n \alpha_i = 2\pi k$$



$$\kappa = \alpha_i \quad ??$$

Curvature is scale dependent

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

κ

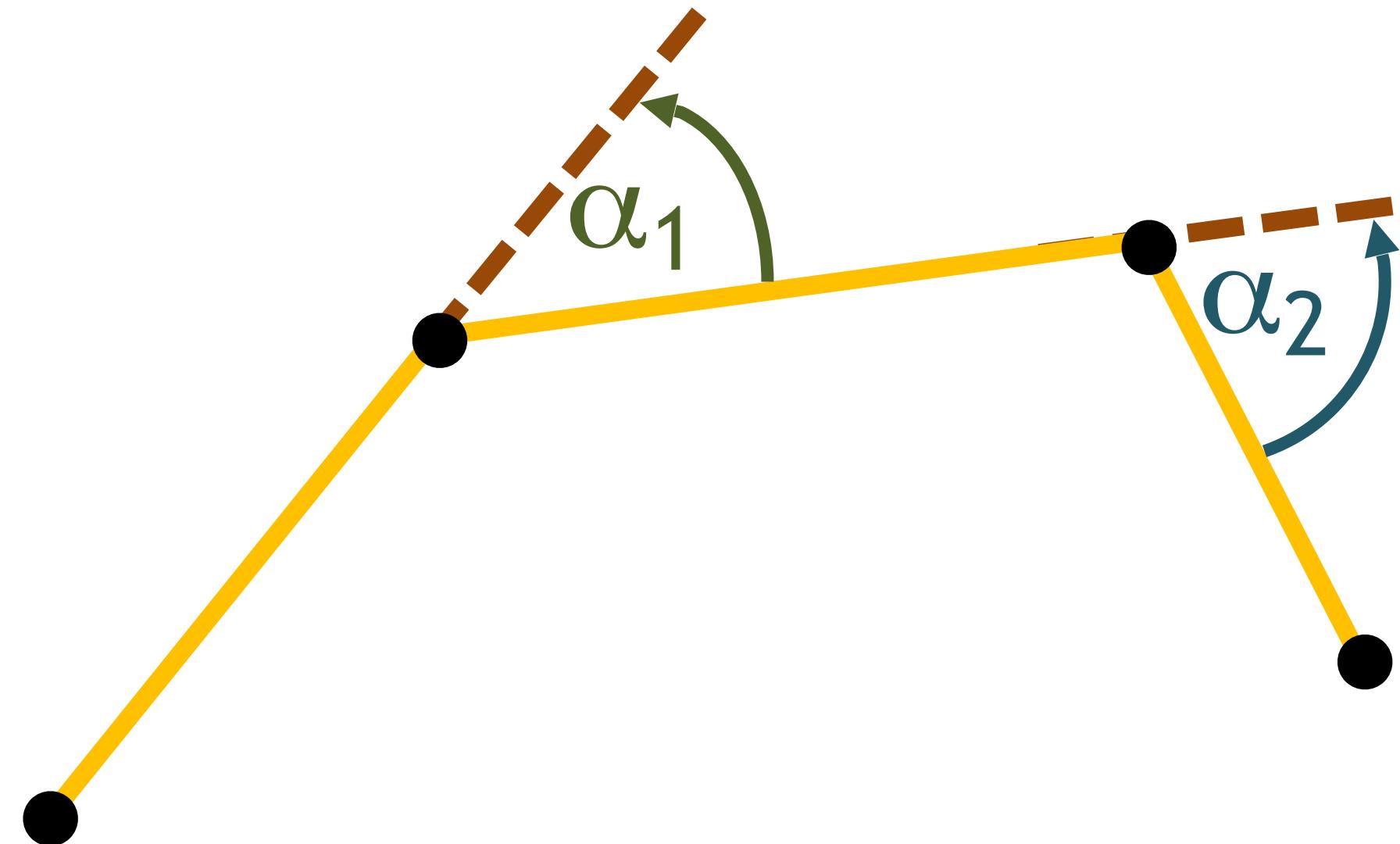
κ

κ

α_i is scale-independent

Discrete Curvature – Integrated Quantity!

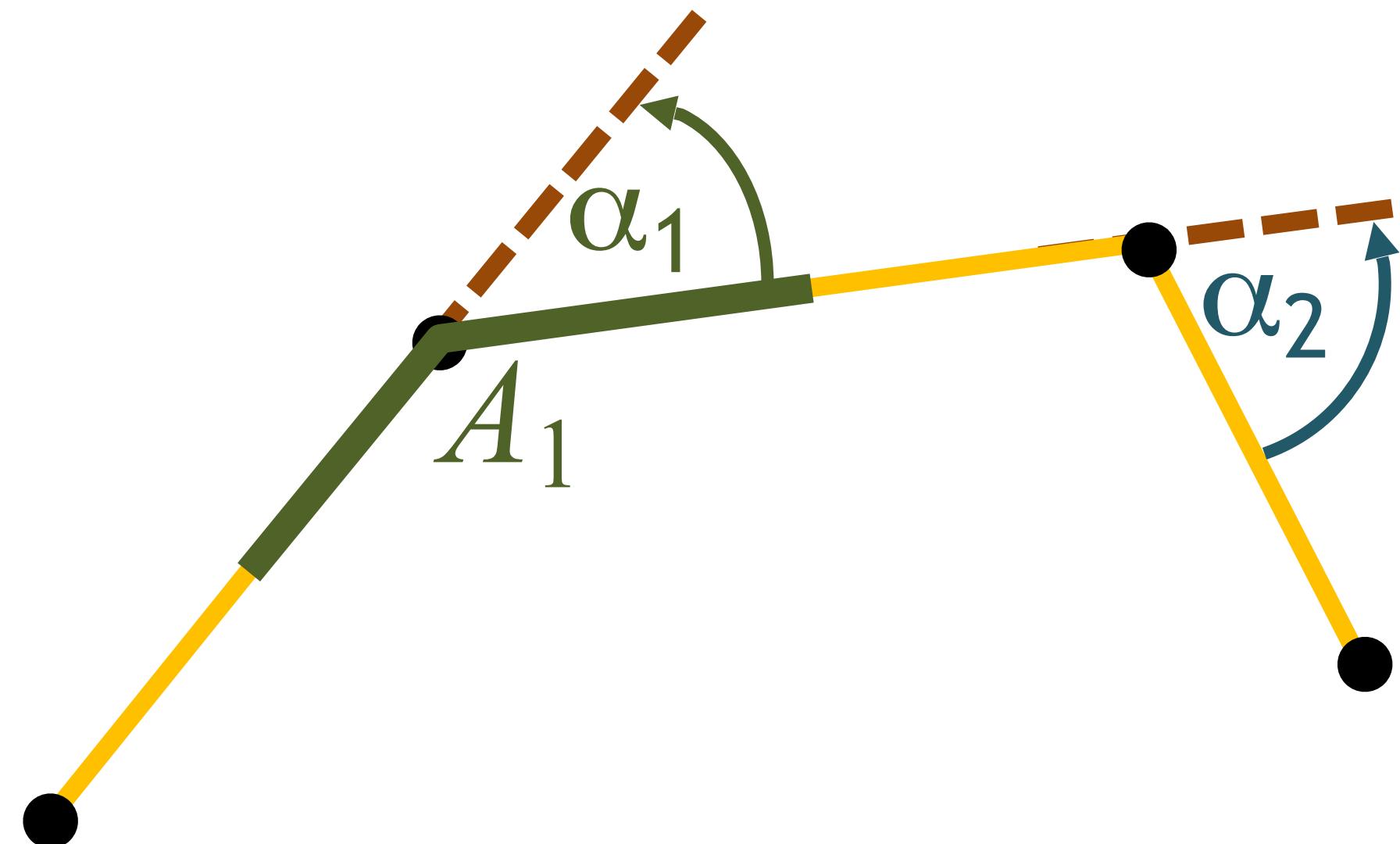
- Cannot view α_i as pointwise curvature
- It is *integrated curvature* over a local area associated with vertex i



Discrete Curvature – Integrated Quantity!

- Integrated over a local area associated with vertex i

$$\alpha_1 = A_1 \cdot \kappa_1$$

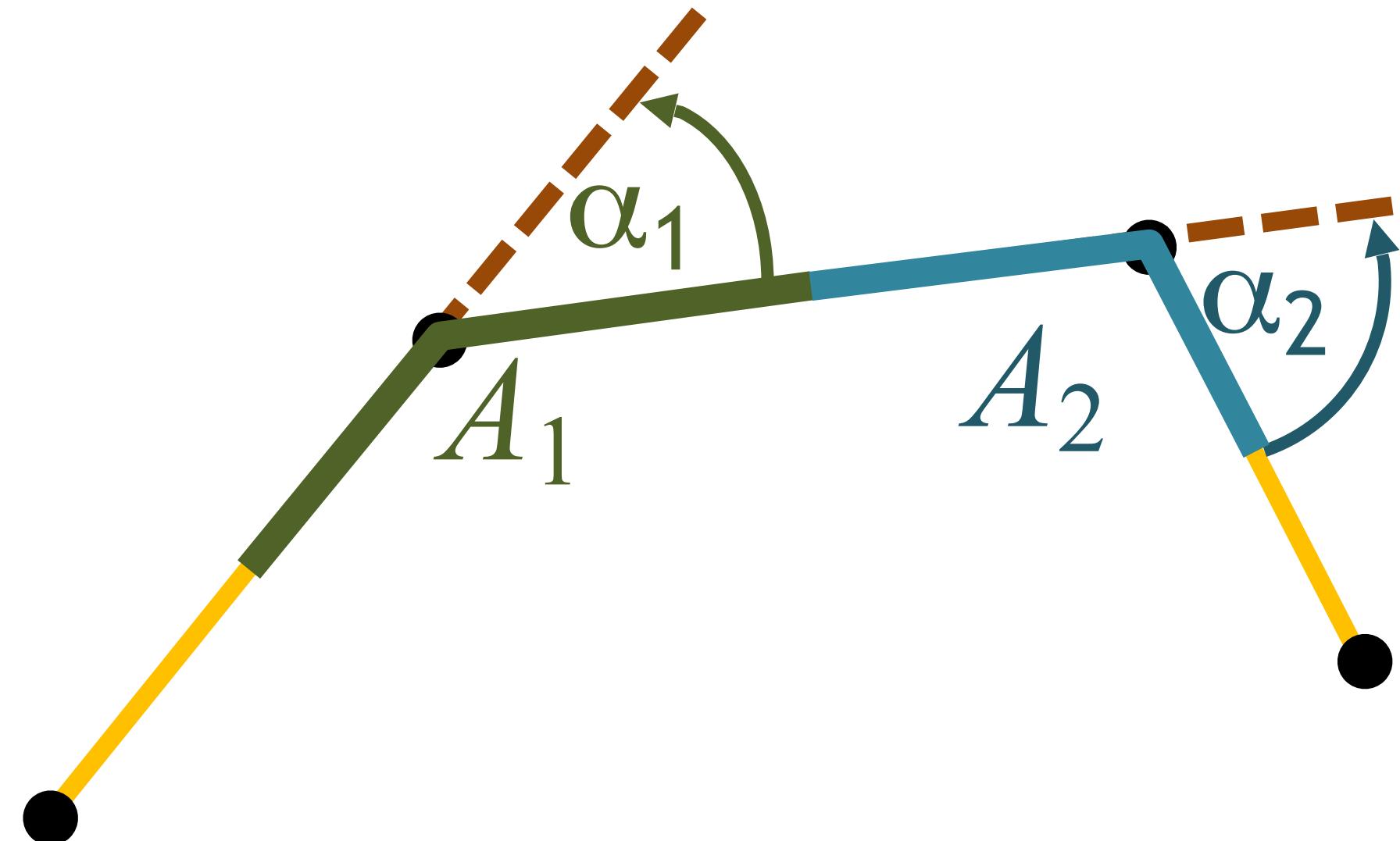


Discrete Curvature – Integrated Quantity!

- Integrated over a local area associated with vertex i

$$\alpha_1 = A_1 \cdot \kappa_1$$

$$\alpha_2 = A_2 \cdot \kappa_2$$



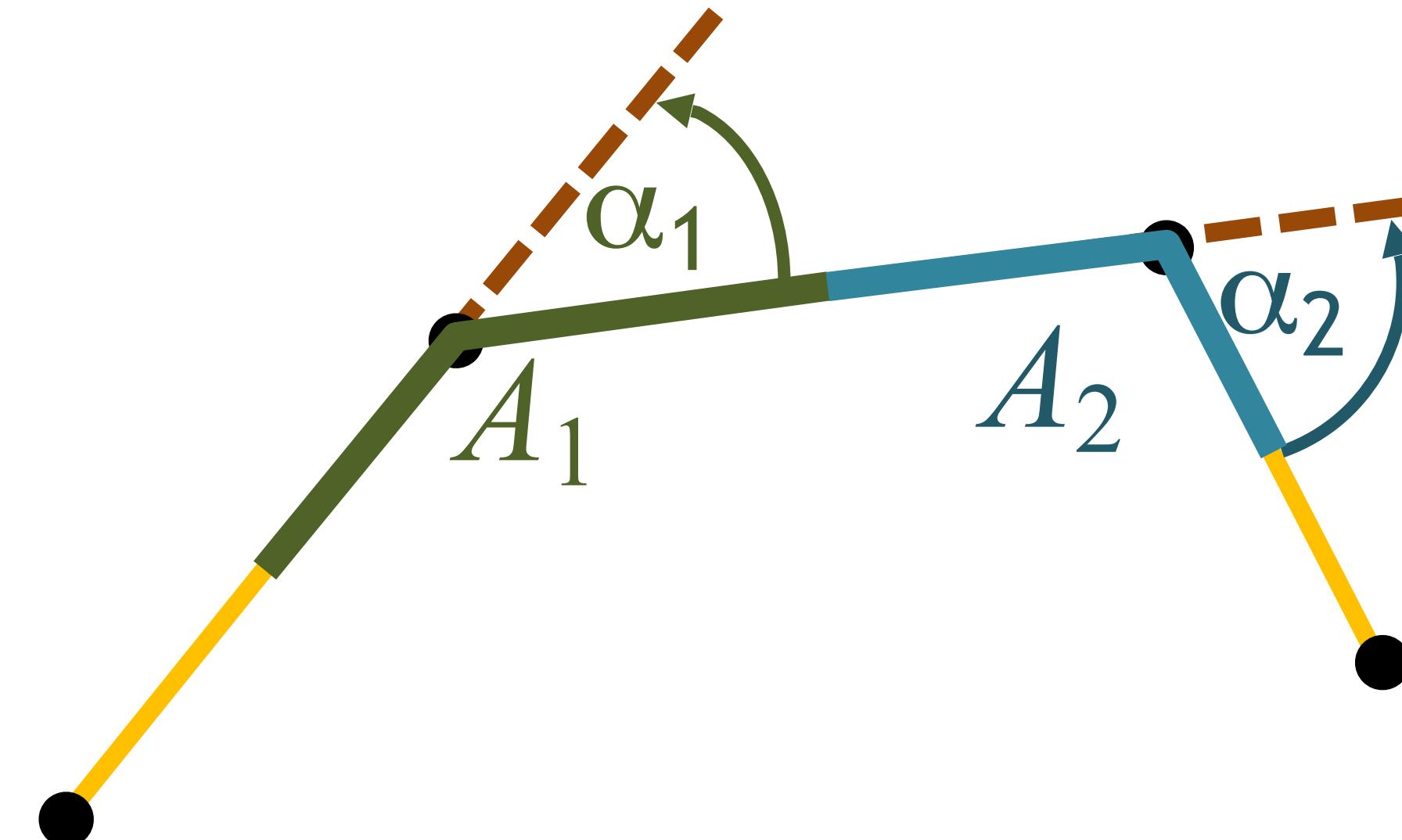
Discrete Curvature – Integrated Quantity!

- Integrated over a local area associated with vertex i

$$\alpha_1 = A_1 \cdot \kappa_1$$

$$\alpha_2 = A_2 \cdot \kappa_2$$

$$\sum A_i = \text{len}(p)$$



The vertex areas A_i form a covering of the curve.
They are pairwise disjoint (except endpoints).

Structure Preservation

- Arbitrary discrete curve
 - total signed curvature obeys discrete turning number theorem
 - even coarse mesh (curve)
 - which continuous theorems to preserve?
 - that depends on the application...

*discrete analogue
of continuous theorem*

Convergence

- Consider refinement sequence
 - length of inscribed polygon approaches length of smooth curve
 - in general, discrete measure approaches continuous analogue
 - which refinement sequence?
 - depends on discrete operator
 - pathological sequences may exist
 - in what sense does the operator converge?
(pointwise, L_2 ; linear, quadratic)

Recap

Structure-preservation

For an arbitrary (even coarse) discrete curve, the discrete measure of curvature **obeys** the discrete turning number theorem.

Convergence

In the limit of a refinement sequence, discrete measures of length and curvature **agree** with continuous measures.

Thank you