04 - Normal Estimation, Curves
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 $x$ in the cloud, pick $k$ nearest neighbors or all points in $r$-ball: $\{x_i \mid \|x_i - x\| < r\}$

$$X_1, X_2, \ldots, X_n$$

• Find a plane $\Pi$ that minimizes the sum of square distances:

$$\min \sum_{i=1}^{n} \text{dist}(x_i, \Pi)^2$$
Local Plane Fitting

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

\[ \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n \]

• Find a plane \( \Pi \) 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 $\sum \text{distances}^2$)
Notations

• Input points:

\[ x_1, x_2, \ldots, x_n \in \mathbb{R}^d \]

• Looking for a (hyper) plane passing through \( c \) with normal \( n \) s.t.

\[
\min_{c, n, \|n\| = 1} \sum_{i=1}^{n} \left( (x_i - c)^T n \right)^2
\]
Notations

• Input points: \( x_1, x_2, \ldots, x_n \in \mathbb{R}^d \)

• Centroid:

\[
m = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

• Vectors from the centroid:

\( y_i = x_i - m \)
Centroid: 0-dim Approximation

• It can be shown that:
  \[ m = \arg\min_c \sum_{i=1}^{n} \left( (x_i - c)^T n \right)^2 \]
  \[ m = \arg\min_c \sum_{i=1}^{n} ||x_i - c||^2 \]

• \( m \) minimizes SSD
• \( m \) will be the origin of the (hyper)-plane
• Our problem becomes:
  \[ \min_{||n||=1} \sum_{i=1}^{n} \left( y_i^T n \right)^2 \]
Hyperplane Normal

- Minimize!

\[
\min_{n^T n = 1} \sum_{i=1}^{n} (y_i^T n)^2 = \min_{n^T n = 1} \sum_{i=1}^{n} n^T y_i y_i^T n = \\
\min_{n^T n = 1} n^T \left( \sum_{i=1}^{n} y_i y_i^T \right) n = \min_{n^T n = 1} n^T (Y Y^T) n
\]

\[
Y = \begin{pmatrix}
y_1 & y_2 & \cdots & y_n
\end{pmatrix}
\]
Hyperplane Normal

• Minimize!

\[
\min_{n^T n = 1} \sum_{i=1}^{n} (y_i^T n)^2 = \min_{n^T n = 1} \sum_{i=1}^{n} n^T y_i y_i^T n = \\
\min_{n^T n = 1} n^T \left( \sum_{i=1}^{n} y_i y_i^T \right) n = \min_{n^T n = 1} n^T (Y Y^T) n
\]

\[
Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}
\]

\[
f(n) = n^T S n \quad (S = Y Y^T) \\
\min f(n) \quad s.t. \quad n^T n = 1
\]
Hyperplane Normal

- Constrained minimization – Lagrange multipliers

\[ f(n) = n^T S n \quad (S = YY^T) \]
\[ \min f(n) \quad s.t. \quad n^T n = 1 \]

\[ \mathcal{L}(n, \lambda) = f(n) - \lambda(n^T n - 1) \]
\[ \nabla \mathcal{L} = 0 \]

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

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

Matrix Cookbook!

https://archive.org/details/imm3274
Hyperplane Normal

• Constrained minimization – Lagrange multipliers

\[ f(n) = n^T S n \quad (S = Y Y^T) \]
\[ \min f(n) \quad s.t. \quad n^T n = 1 \]

\[ \mathcal{L}(n, \lambda) = f(n) - \lambda (n^T n - 1) \]
\[ \nabla \mathcal{L} = 0 \]

\[ \frac{\partial \mathcal{L}}{\partial n} = 0 \iff S n = \lambda n \]
\[ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff n^T n = 1 \]
Hyperplane Normal

- Constrained minimization – Lagrange multipliers

\[ f(n) = n^T S n \quad (S = Y Y^T) \]

\[ \min f(n) \quad s.t. \quad n^T n = 1 \]

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

\[ \nabla \mathcal{L} = 0 \]

\[ \frac{\partial \mathcal{L}}{\partial n} = 0 \iff S n = \lambda n \]

\[ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff n^T n = 1 \]

What can be said about \( n \)??
Hyperplane Normal

- Constrained minimization – Lagrange multipliers

\[ f(n) = n^T S n \quad (S = YY^T) \]
\[ \min f(n) \quad s.t. \quad n^T n = 1 \]

\[ \mathcal{L}(n, \lambda) = f(n) - \lambda(n^T n - 1) \]
\[ \nabla \mathcal{L} = 0 \]

\[ \frac{\partial \mathcal{L}}{\partial n} = 0 \iff S n = \lambda n \]
\[ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \iff n^T n = 1 \]

\textbf{n} is the eigenvector of \( S \) with the smallest eigenvalue
Summary – Best Fitting Plane Recipe

- **Input:** \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d \)
- Compute centroid = plane origin
- Compute scatter matrix

\[
\mathbf{S} = \mathbf{Y} \mathbf{Y}^T \\
\mathbf{Y} = (y_1 \ y_2 \ \ldots \ y_n) \\
y_i = \mathbf{x}_i - \mathbf{m}
\]

- The plane normal \( \mathbf{n} \) is the eigenvector of \( \mathbf{S} \) with the smallest eigenvalue

\[
\mathbf{S} = \mathbf{V} \begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_d 
\end{pmatrix} \mathbf{V}^T
\]
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:

\[
\text{var}(\mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{v}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}'_i - \mathbf{m}\|^2 =
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{m} + \mathbf{v}^T \mathbf{y}_i - \mathbf{m}\|^2 = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{y}_i^T \mathbf{v})^2 = \frac{1}{n} \mathbf{v}^T \mathbf{S} \mathbf{v}
\]
What does the Scatter Matrix do?

- The scatter matrix measures the variance of our data points along the direction $v$.

\[
\text{var}(x_1, \ldots, x_n; v) = \frac{1}{n} \sum_{i=1}^{n} \| x'_i - m \|^2 = \\
= \frac{1}{n} \sum_{i=1}^{n} \| (m + v^T y_i) - m \|^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i^T v)^2 = \frac{1}{n} v^T S v
\]
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:

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

• Any vector is an eigenvector

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

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

• $\mu$ is close to zero, much smaller than $\lambda$
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 \(x_i \in k\text{NN}(x_j)\) or \(x_j \in k\text{NN}(x_i)\)

- Propagate normal orientation through graph
  - For neighbors \(x_i, x_j\): Flip \(n_j\) if \(n_i^T n_j < 0\)
  - Fails at sharp edges/corners

- Propagate along “safe” paths (parallel tangent planes)
  - Minimum spanning tree with angle-based edge weights

\[ w_{ij} = 1 - |n_i^T n_j| \]

“Surface reconstruction from unorganized points”, Hoppe et al., SIGGRAPH 1992
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

manifold point

continuous
1-1 mapping
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

Diagram:
- Manifold point
- Non-manifold point
- Continuous 1-1 mapping
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?
Building blocks: curves and surfaces

$y(t)$

$x(t)$

A modeling session

• Demo with Keynote for 2D
• Demo with Blender for 3D
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
A 2D parametric curve $p(t)$ is defined as $p(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$, for $t \in [t_0, t_1]$. One key property of a parametric curve is that it must be continuous.
A curve can be parameterized in many different ways

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

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

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

\[ \|p'(t)\| = \text{speed} \]
\[ \frac{p'(t)}{\|p'(t)\|} = T(t) = \text{unit tangent} \]

Parametrization-independent!
Arc length

• How long is the curve between $t_0$ and $t$? How far does the particle travel?

• Speed is $\|p'(t)\|$, so:

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

• Speed is nonnegative, so $s(t)$ is non-decreasing
Arc length parameterization

• Every curve has a natural parameterization:

\[ p(s), \text{ such that } \|p'(s)\| = 1 \]
Arc length parameterization

- Every curve has a natural parameterization: \( p(s) \), such that \( \| p'(s) \| = 1 \)

- Isometry between parameter domain and curve

- Tangent vector is unit-length: \( p'(s) = T(s) \)
Curvature

- How much does the curve turn per unit $s$?

$$\theta(s) = \tan^{-1} \frac{T_y(s)}{T_x(s)}$$
Curvature

- How much does the curve turn per unit $s$?

$$\theta(s) = \tan^{-1} \frac{T_y(s)}{T_x(s)}$$

$$\kappa(s) = \frac{d\theta}{ds}$$
Curvature profile

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

$$p(s) = p_0 + \int_{s_0}^{s_1} \begin{pmatrix} \cos \theta(s) \\ \sin \theta(s) \end{pmatrix} ds$$

reconstructs the curve up to rigid motion.
Curvature of a circle

- Curvature of a circle:

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

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

\[ \theta(s) = \tan^{-1} \frac{\cos(s/r)}{-\sin(s/r)} = \frac{s}{r} - \frac{\pi}{2} \]

\[ \kappa(s) = \frac{1}{r} \]

\[ \mathbf{N}(t) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \]

\[ \mathbf{T}(t) = \text{Unit Normal} \]

Osculating circle

\[ r = \frac{1}{\kappa} \mathbf{N} \]
Frenet Frame

\[ T'(s) = \kappa(s)N(s) \]
\[ N'(s) = -\kappa(s)T(s) \]

\[
\begin{pmatrix}
T' \\
N'
\end{pmatrix}
= \begin{pmatrix}
0 & \kappa \\
-\kappa & 0
\end{pmatrix}
\begin{pmatrix}
T \\
N
\end{pmatrix}
\]
Curvature normal

- Points inward
  \[ T'(s) = \kappa(s)N(s) \]
- \(-\kappa(s)N(s)\) useful for evaluating curve quality
Smoothness

Two kinds, parametric and geometric:

\[ C^1: \mathbf{p}(t) \text{ is continuously differentiable} \]
\[ G^1: \mathbf{p}(s) \text{ is continuously differentiable} \]

Parametrization-Independent

\[ C^1 \begin{pmatrix} \cos t \\ \sin t \end{pmatrix} \]
\[ G^1 \begin{pmatrix} \cos \hat{t} \\ \sin \hat{t} \end{pmatrix} \text{, } \hat{t} = \begin{cases} t + 1 & \text{if } t < 1 \\ 2t & \text{if } t \geq 1 \end{cases} \]
Smoothness example

\[ G^0 \]

\[ G^1 \]

\[ G^2 \]
Recap on parametric curves

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

\[ \|p'(t)\| = \text{speed} \]

\[ s(t) = \int_{t_0}^{t} \|p'(t)\| dt \]

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

\[ \frac{p'(t)}{\|p'(t)\|} = T(t) \]
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

\[ \int_{p} \kappa(s) \, ds = 2\pi n \]

- \( n \) 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 $\mathbf{T}$
  
  $$
  \mathbf{N}(s) := \frac{\mathbf{T}'(s)}{\|\mathbf{T}'(s)\|}
  $$

- $\mathbf{B}(s) := \mathbf{T}(s) \times \mathbf{N}(s)$

- $\mathbf{T}, \mathbf{N}, \mathbf{B}$ are the “Frenet frame”

- $\tau$ 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/
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

\[ \hat{n}_v = \frac{\hat{n}_{e_1} + \hat{n}_{e_2}}{\|\hat{n}_{e_1} + \hat{n}_{e_2}\|} \]
Tangents, Normals

• Weight by edge lengths

\[ \hat{n}_v = \frac{|e_1|\hat{n}_{e_1} + |e_2|\hat{n}_{e_2}}{\|e_1\hat{n}_{e_1} + e_2\hat{n}_{e_2}\|} \]
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

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

$h = \text{max edge length}$
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.
Curvature of a Discrete Curve

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

\[ \alpha_1, \alpha_2 > 0, \quad \alpha_3 < 0 \]
Total Signed Curvature

\[ tsc(p) = \sum_{i=1}^{n} \alpha_i \]

- 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 $2\pi$.
- proof: sum of exterior angles
Turning Number Theorem

\[ \int_\gamma \kappa \, dt = 2\pi k \]

\[ k: \begin{array}{ccc}
-1 & +2 & 0 \\
\end{array} \]

\[ \kappa = \alpha_i \]

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

\[ \begin{array}{ccc}
-1 & +2 & 0 \\
\end{array} \]
Curvature is scale dependent

\[ \kappa = \frac{1}{r} \]

\( \kappa \) is scale-dependent

\( \alpha_i \) is scale-independent

\( \kappa \)
Discrete Curvature – Integrated Quantity!

- Cannot view $\alpha_i$ 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