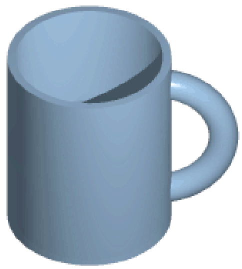
Basic Geometric Computation on Meshes

Xin (Shane) Li

Basic Geometry of Curves and Surfaces



- What does it mean by:
  - two objects have the same geometric shape
    - They have the same vertex table?
    - These two objects "overlap" with each other in the 3D space?
    - equivalent under some transformation (rotation, translation, scaling…)?
  - two objects have the same topology?
    - Equivalent if one can deform to the other under continuous stretching and bending, without tearing or gluing (not a rigorous definition but gives you the intuition)
    - If there is a one-to-one map between the two shapes that does not change each point's neighboring information
  - two objects have similar geometry?
    - Need to be able to measure some properties quantitatively

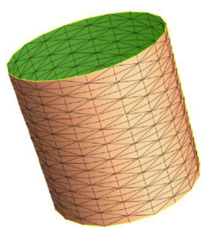
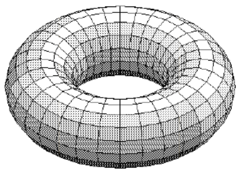
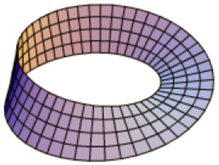
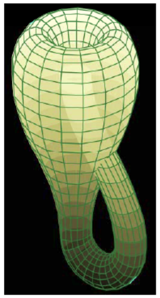
Basic Geometry Properties

- Computing basic geometry and topology properties of surfaces on triangle meshes

- Using half-edge data structure to compute the approximated:
  - length of a curve
  - area of a surface patch
  - volume of a solid object

Basic Topology Properties

- Topological Classification of Surfaces
  - Topological equivalence-relationship can be characterized by:
    - # of connected components → c
    - # of boundaries → b
    - # of genus → g
    - (orientability) → o (true/false)

Orientable Surface Examples	Non-orientable Surface Examples
 	 

  - How to compute c, b, and g of a given surface using half-edge data structure?
    - c → BFS $(O(N_F))$
    - b → boundary detection + boundary loop tracing $(O(N_E+N_{BE}))$
    - g → (for each component) Euler Formula $(O(n))$ $2-2g=N_F-N_E+N_V$

Normal Vectors

- Many operations in computer graphics require normal vectors (per face or per vertex), e.g. phone shading
- Face Normal vector: the normalized cross-product of two triangle edges:

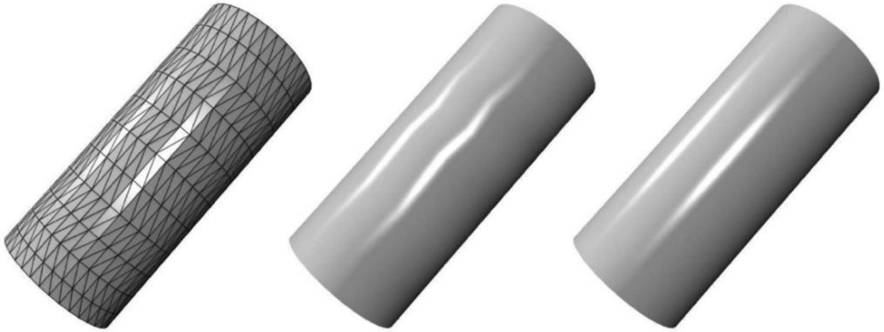
$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}$$

- Vertex Normal: (spatial averages of normal vectors sampled in a local neighboring region)

$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \, \mathbf{n}(T) \right\|}$$

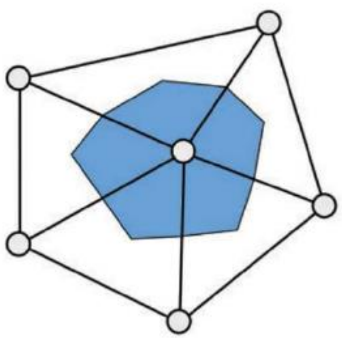
  - Different weights used:
    - Constant weights: $\alpha_T = 1$
    - Triangle area: $\alpha_T = |T|$
    - Incident triangle angles: $\alpha_T = \theta_T$

Why more complicated weights?
→ Uniformity of the sampling on a small disk region surrounding vertex v…

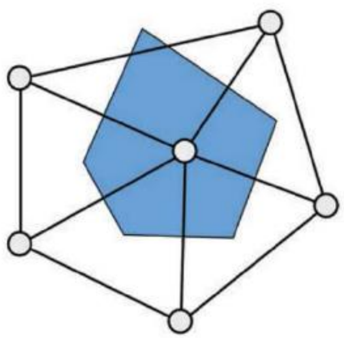


Local Averaging Region

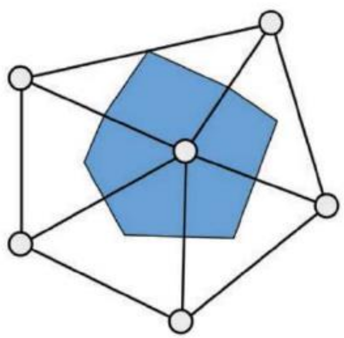
- A straightforward approximation:
  - x → mesh vertex v_i
  - $N(x)$ → one-ring (n-ring) neighborhoods $N_n(v_i)$
- Size of local neighborhoods → stability and accuracy of evaluation
  - Bigger: more smooth, more stable against noise
  - Smaller: more accurately capture fine-scale variations; preferable for clean data
- More accurate approximation than 1-ring/n-ring
  - Barycentric cell: connect triangle barycenters + edge midpoints
  - Voronoi cell: triangle circumcenters + perpendicular bisector
  - Mixed-voronoi cell: midpoint of edge opposing obtuse angle on center vertex + …



Barycentric cell



Voronoi cell



Mixed Voronoi cell

More other differential operators

- In general: to compute discrete differential properties as spatial averages over a local neighborhood $N(x)$ of the point x on the mesh
- More differential operators
  - Gaussian curvature k_G
  - Mean curvature k_m
  - Laplace operator (later)

Example codes using MeshLib

- Computing the area of a triangle

```
double ComputeAreaFace(Face * f)
{
        Vertex * v[3];
        int i=0;
        for (MeshVertexIterator fvit(f); !fvit.end(); ++fvit,++i)
                v[i]=*fvit;
        double fArea = (v[1]->point()-v[0]->point())^(v[2]->point()-v[0]->point()).norm()/2.0;
        return fArea;
}
```

Note: in the MeshLib implementation codes I provided, the "^" operator between two points is the cross product. Namely, $p1 \wedge p2$ returns a vector whose direction is perpendicular to $p1$ and $p2$, and magnitude is 2 times the area of the triangle formed by the origin and these two points.

Example codes using MeshLib

- Computing the corner angles inside a triangle

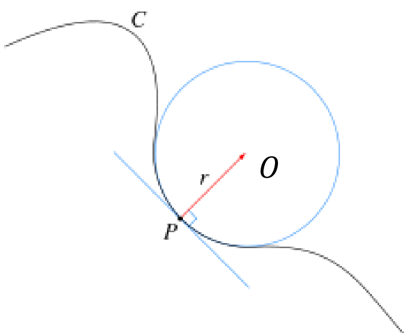
```
void ComputeCornerAngles(Face * f, double cAngles[3])
{

}
```

HW2

1) Integrate the halfedge mesh lib into your GUI
2) Compute vertex normal, apply it to produce better shading effects, using glNormal()
3) Compute the topological properties b, c, g of the mesh, print them on the screen
4) Compute the Gaussian curvature k_G on every vertex, color the vertex accordingly

Curvature of a Smooth Curve

A definition by Cauchy (by Osculating Circle):



1. Center of curvature O: intersection of two infinitely close normal near P
2. Radius of curvature: distance from O to P
3. Curvature κ: the inverse of the radius of curvature

Intuition: flat region vs curved region on a curve

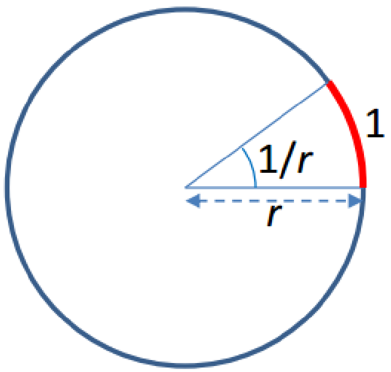
Definition in Differential Geometry:
For a C^2 continuous curve $\gamma(t)$, parameterized using its arc-length (C^2 and arc length will be defined officially in 2 weeks)
Tangent vector (velocity vector): $\mathbf{T}(\mathrm{t}) = \gamma'(t)$
Normal vector: $\boldsymbol{T}'(t) = \kappa(t)\boldsymbol{N}(t)$
Intuition: how quick the direction changes

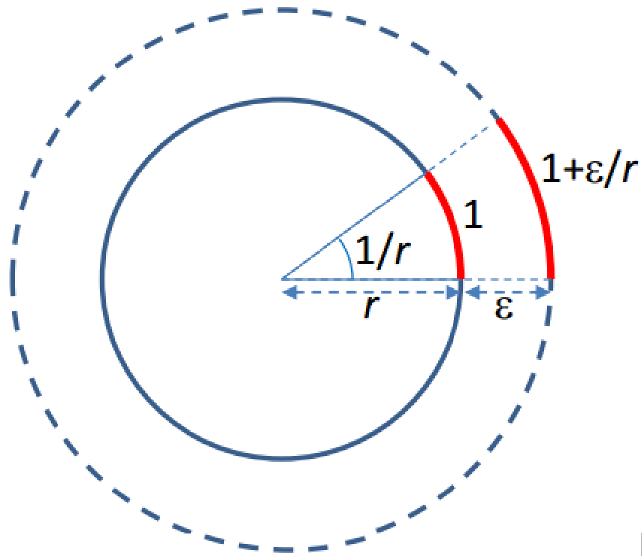
Curvature of a Smooth Curve

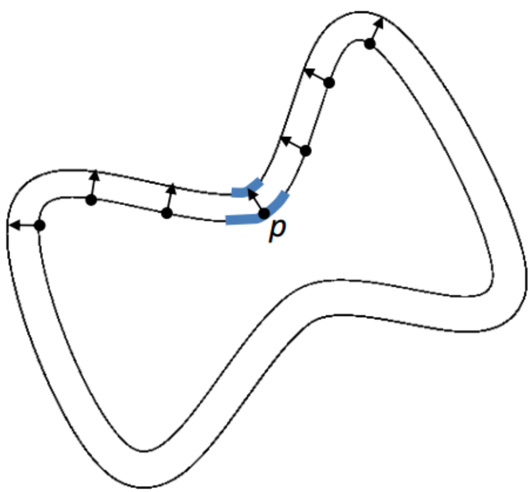
Another definition:

- Curvature = the rate of change in length as a function of offset distance ϵ

$$= \frac{\epsilon}{r} \,/\, \epsilon = 1 \,/\, r$$

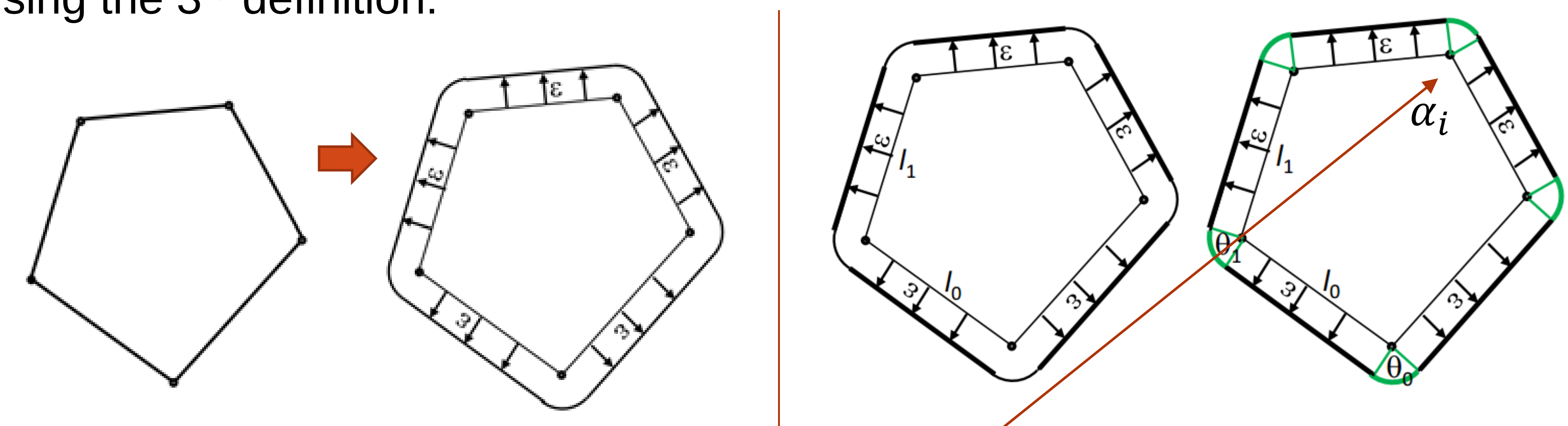






Curvature of a Discrete Curve

Using the 3rd definition:



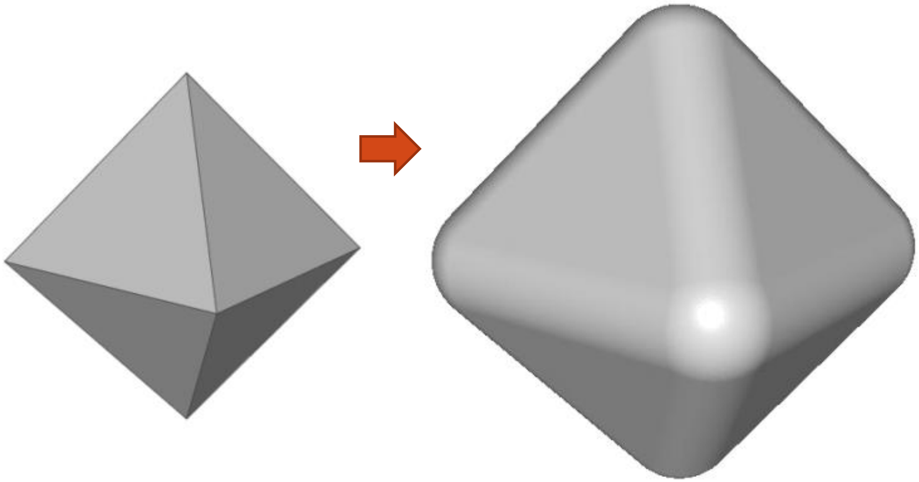
Total length of the offset curve = the length of the old curve + the lengths of the arcs

$$l_{new} = \sum_{i=0}^{N} (l_i + \epsilon\theta_i)$$

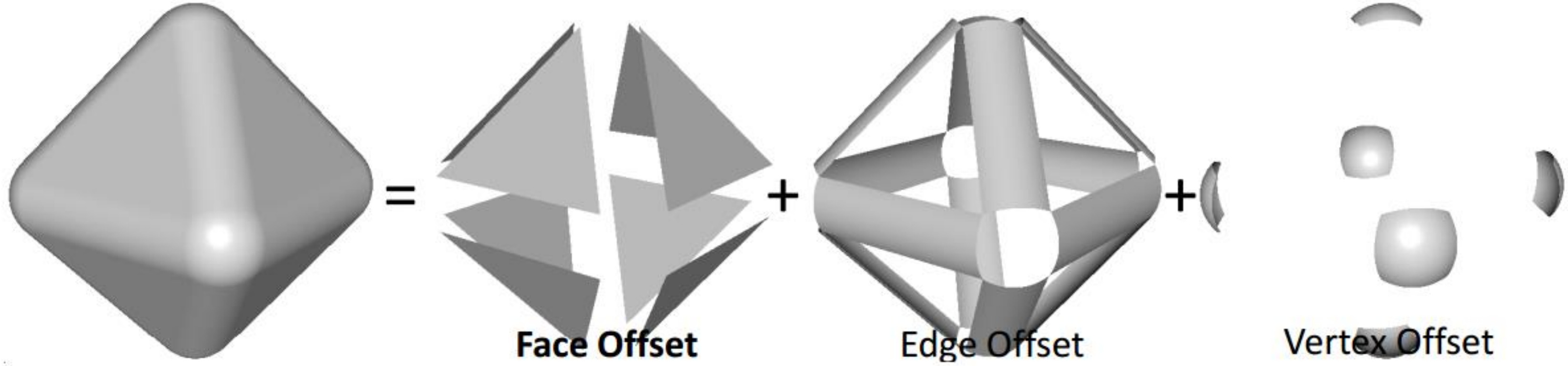
θ_i is the deficit angle, $\theta_i = \pi - \alpha_i$

Therefore, discrete curvature of a curve = angular defect of a vertex

Curvature of a Discrete Surface



The area of the offset surface = $A_f + A_e + A_v$



Curvature of a Discrete Surface

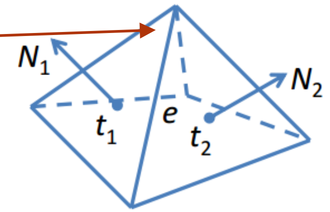
The area of the offset surface :

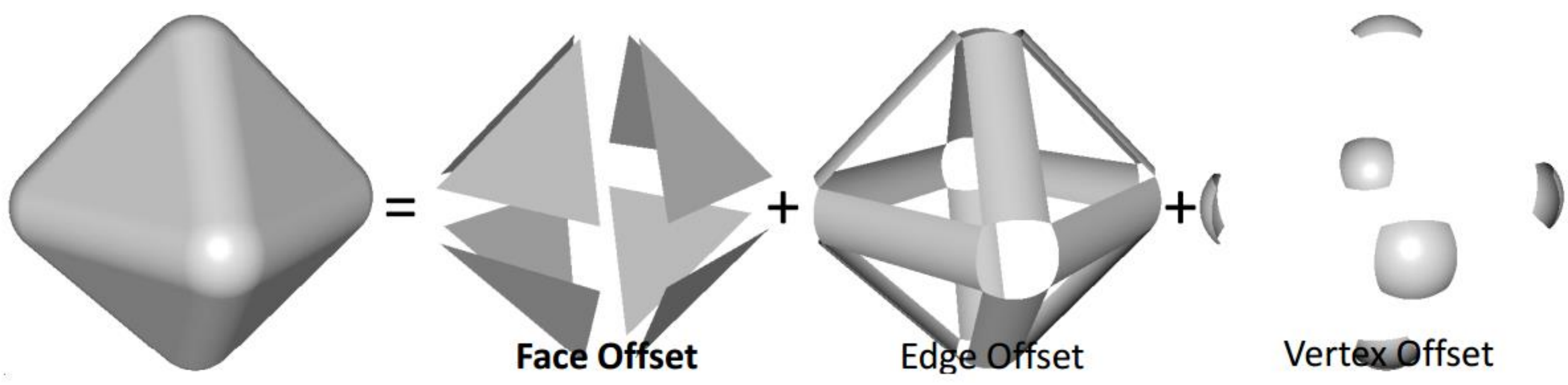
$$A_\epsilon = A_f + A_e + A_v$$

$$A_\epsilon = \sum_{f \in F} A(f) + \epsilon \sum_{e \in E} |e| \theta_e + \epsilon^2 \sum_{v \in V} \theta_v$$

Where:

- ❑ θ_e is the dihedral angle at edge e, $\cos\theta_e = < N_1, N_2 >$
- ❑ θ_v is the solid angle at vertex v, $\theta_v = 2\pi - \sum_i \alpha_i$





Discrete Gaussian Curvature

The Discrete Gaussian curvature at v is the angle of deficit:

$$\kappa_G = 2\pi - \sum_{i=0}^{n} \alpha_i$$

where α_i is the angle between e_i and e_{i+1} at v , α_n is the angle between e_n and e_0 , n is the total number of edges incident to v

The Discrete mean curvature at e is the dihedral angle :

$$\kappa_H = \theta_e$$

Note: The discrete mean curvature at v will be explained later.