Sub-Topics

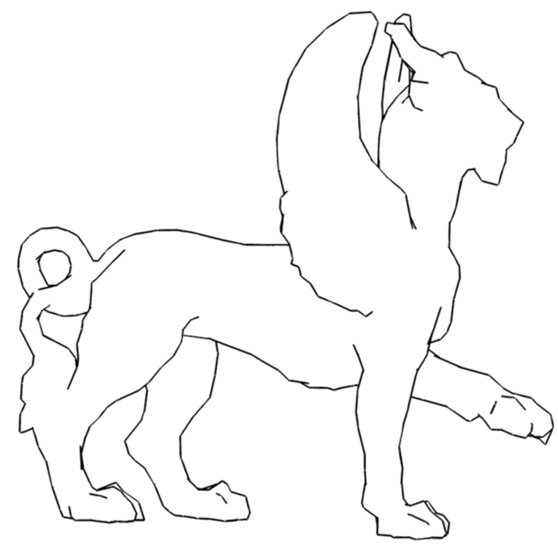
- Compute bounding box
- Compute Euler Characteristic
- Estimate surface curvature
- Line description for conveying surface shape
- Morse function and surface topology--Reeb graph
- Scalar field topology--Morse-Smale complex

Surface Curvature Estimation

By Prof. Eugene Zhang

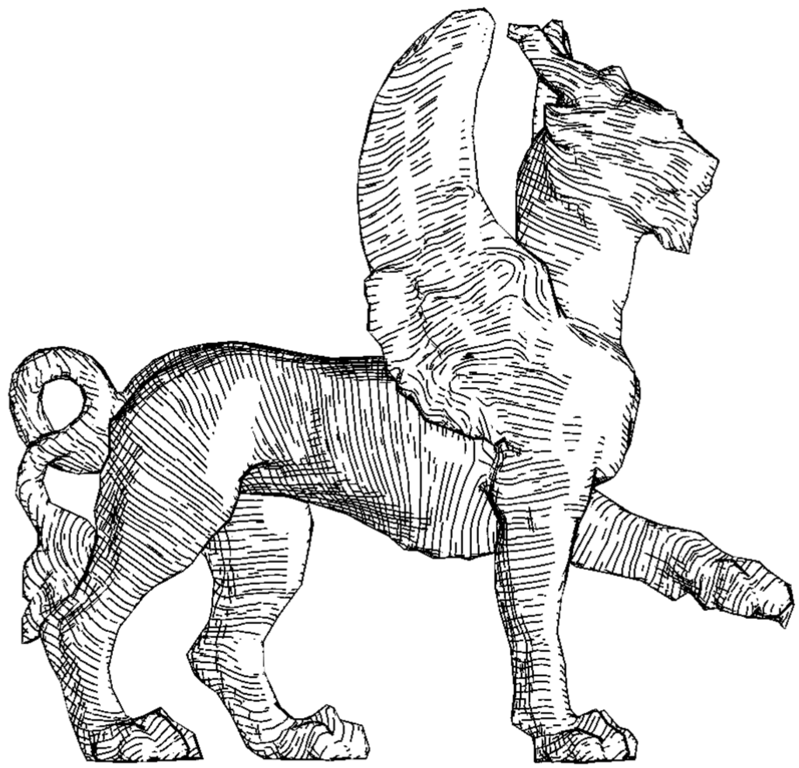
Shape Visualization

- How to convey shape with a few lines? What lines should be drawn?



Shape Visualization

- Placing lines along the principle curvature direction is best at illustrating the shape of an object



Curvature

- Curve
- Surface

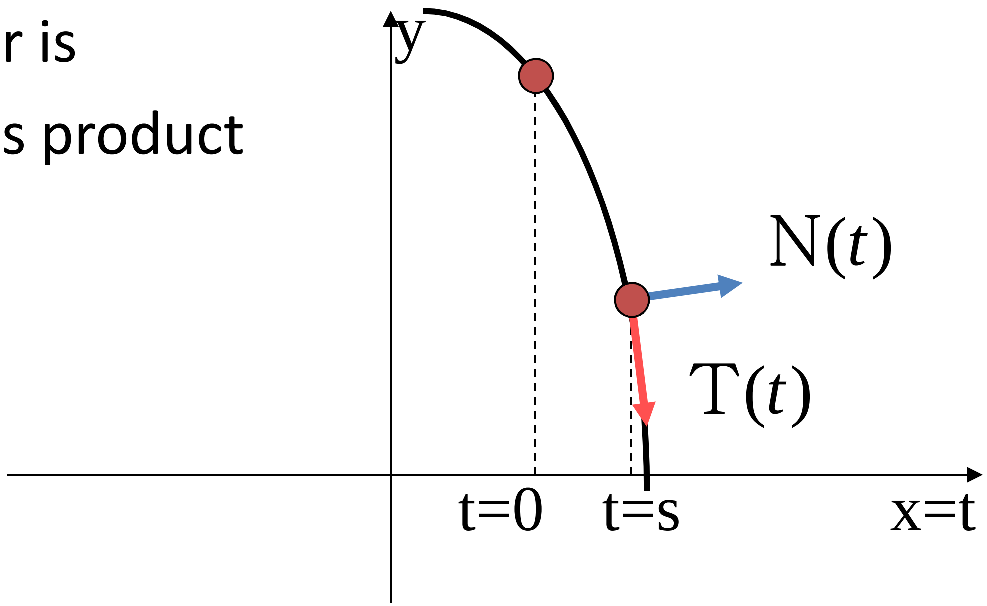
Curvature of a Planar Curve

A Planar Curve

- Given a 2D curve $\mathrm{r}(t) = (x(t), y(t))$
  - The unit tangent vector is defined as

$$\mathrm{T}(t) = \frac{\mathrm{r}'(t)}{|\mathrm{r}'(t)|} = \frac{(x'(t) \quad y'(t))}{\sqrt{(x'(t))^2 + (y'(t))^2}}$$

  - The unit normal vector is
  - counterclockwise cross product



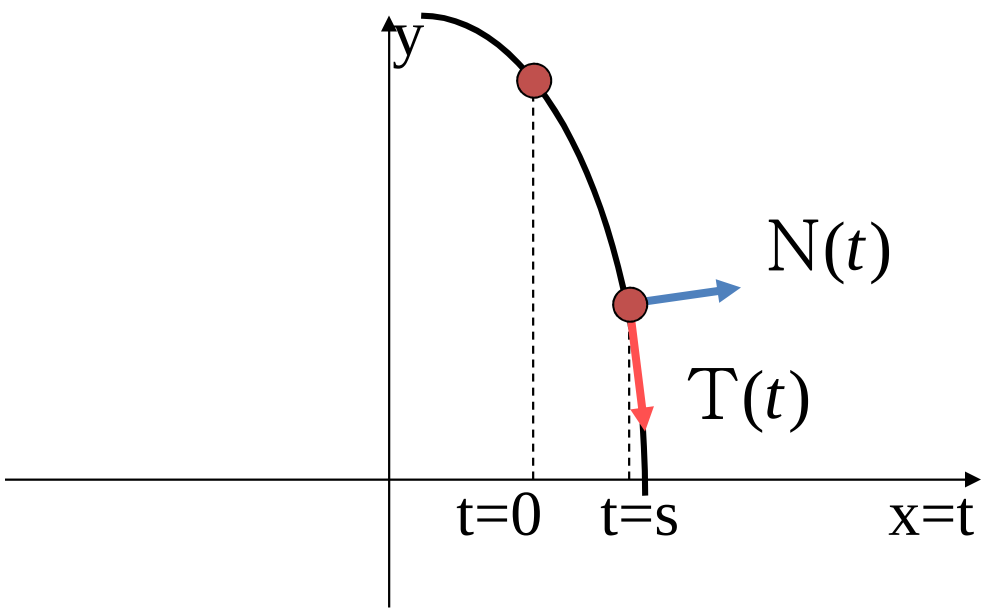
Arc Length

- The arc length of $r(t) = (x(t), y(t))$ is

$$s(u) = \int_{t=0}^{u} |r'(t)| dt = \int_{t=0}^{u} \sqrt{(x'(t))^2 + (y'(t))^2} dt$$

- and

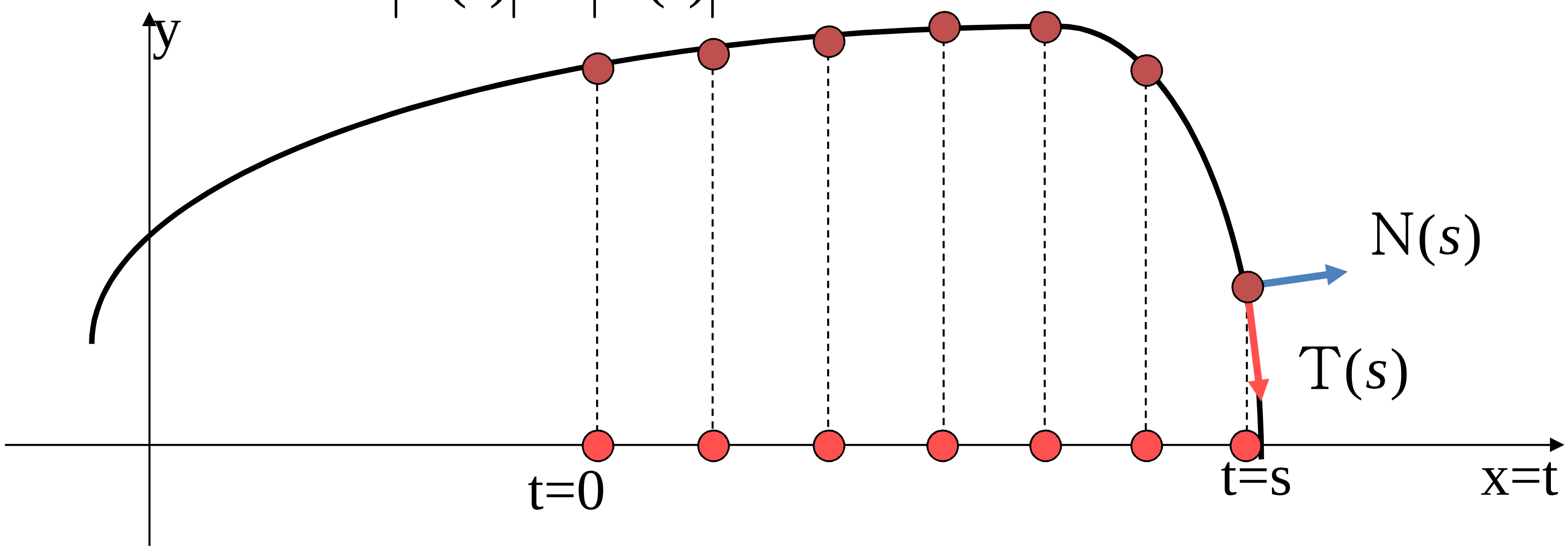
$$s'(t) = |r'(t)|$$



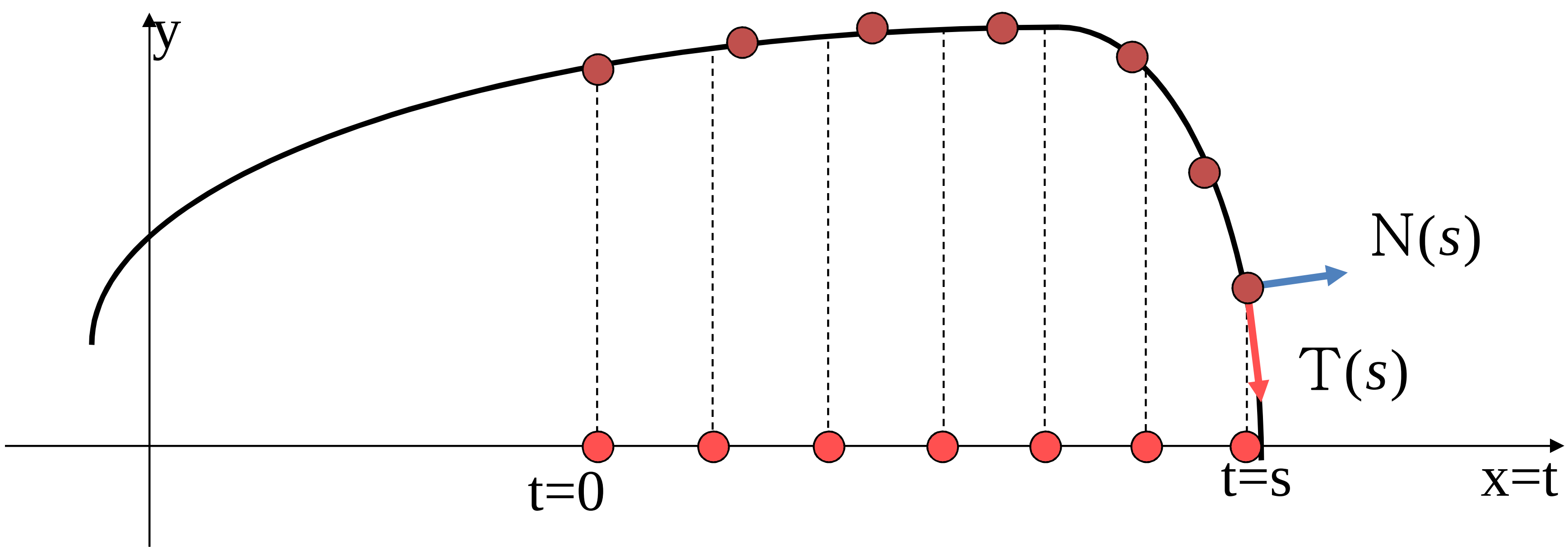
Arc Length Based Re-parameterization

- The curve $\mathrm{r}(t) = (x(t), y(t))$ becomes $\mathrm{r}(s) = (x(s), y(s))$
- and we have

$$|r'(s)| = \frac{|r'(t)|}{|s'(t)|} = \frac{|r'(t)|}{|r'(t)|} = 1$$



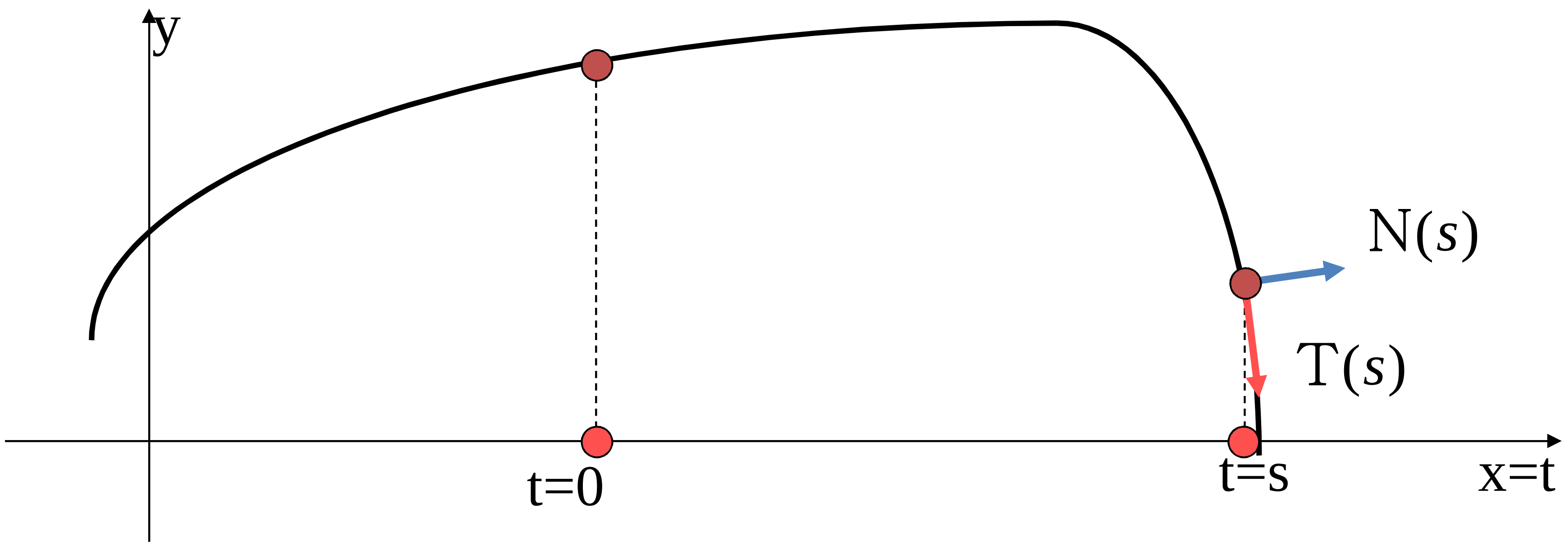
Arc Length Based Re-parameterization



Arc Length Based Re-parameterization

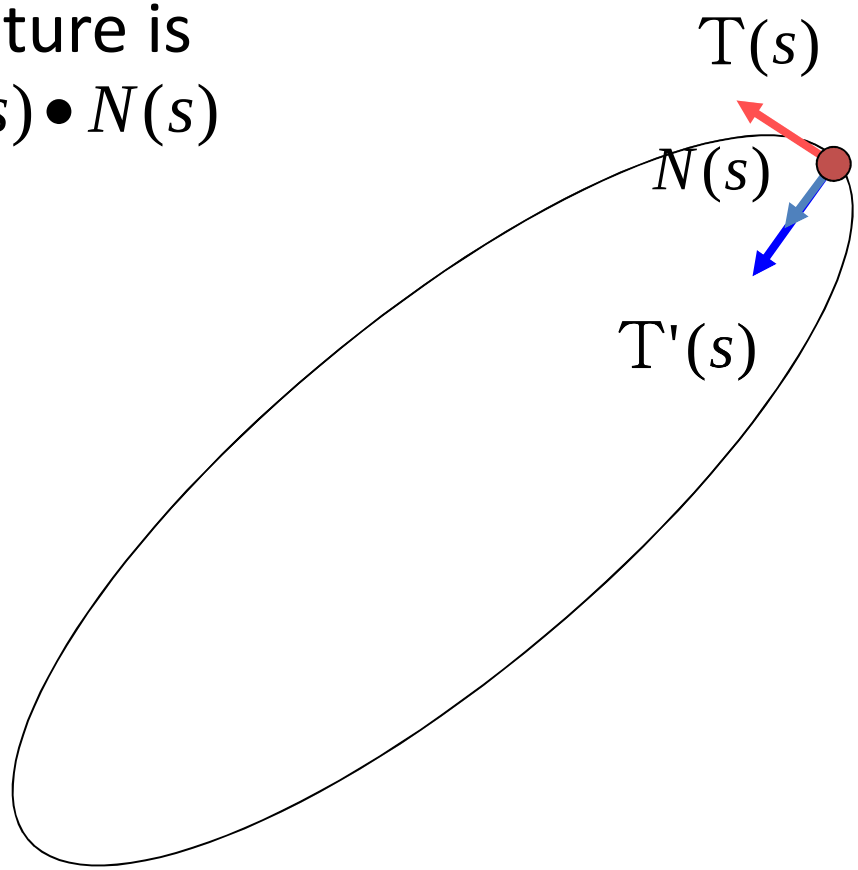
- The unit tangent vector is

$$T(s) = \frac{r'(s)}{\left|r'(s)\right|} = r'(s)$$

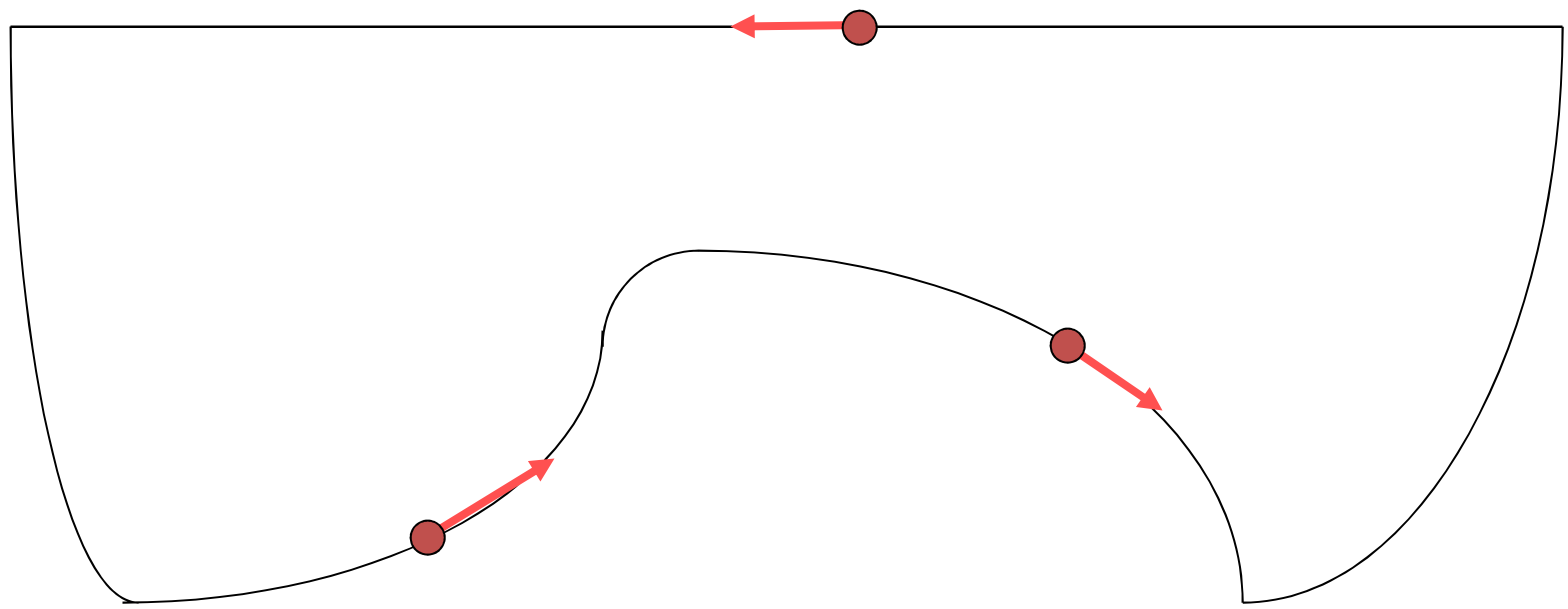


Curvature

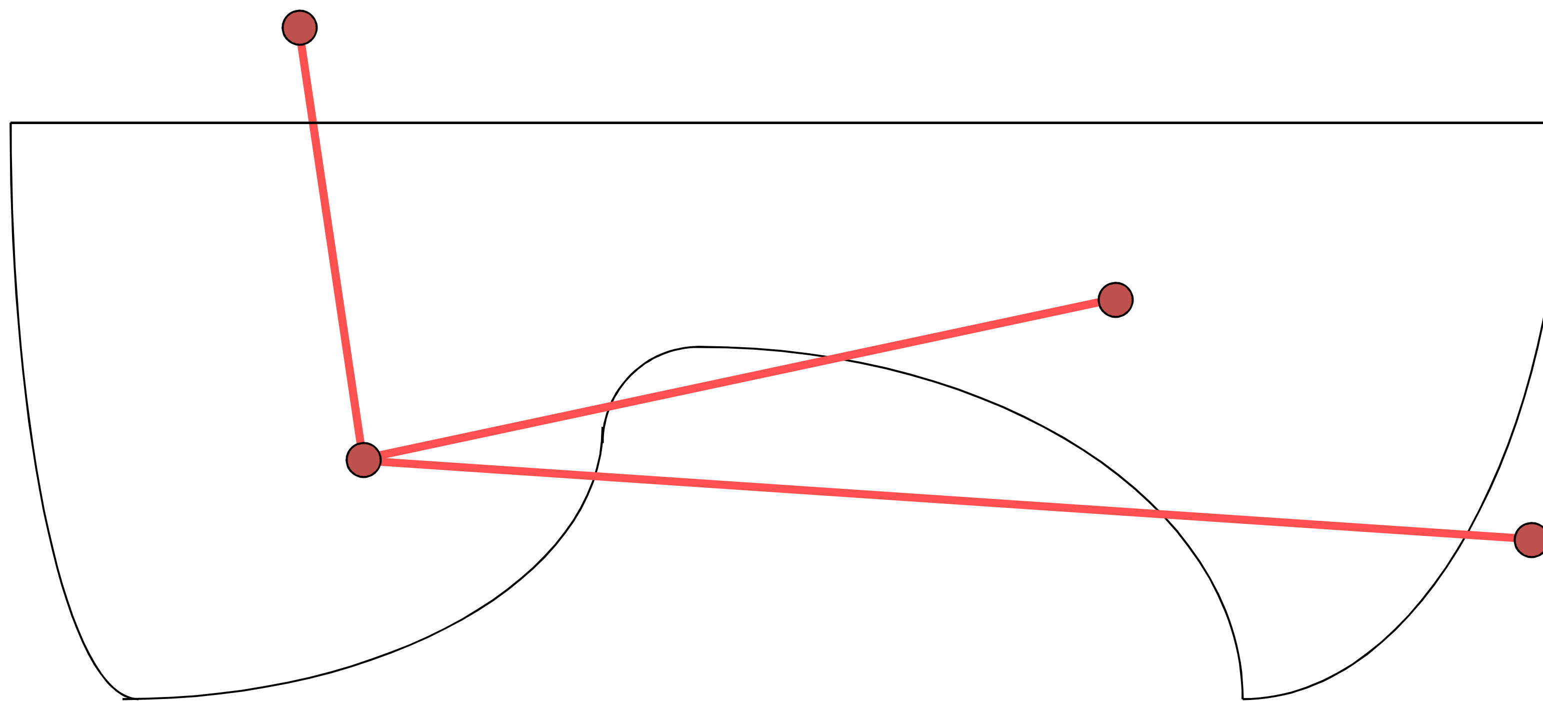
- The curvature is $\kappa(s) = T'(s) \bullet N(s)$



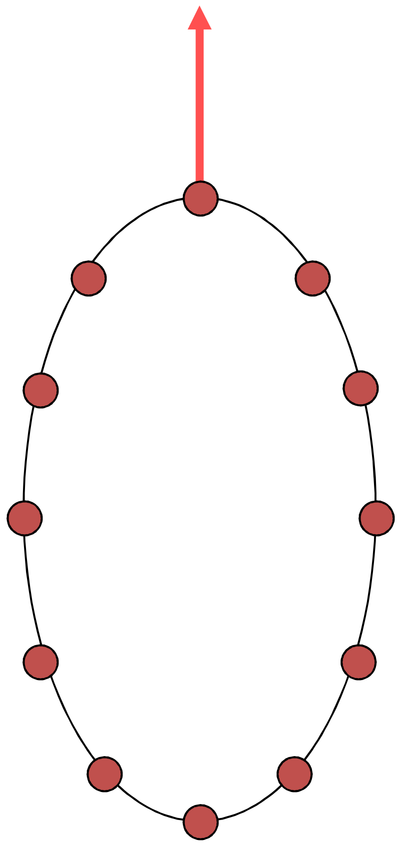
Signed Curvature

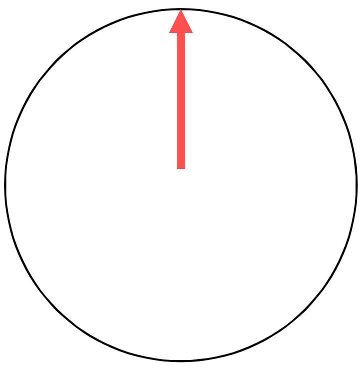


Topology of Curves



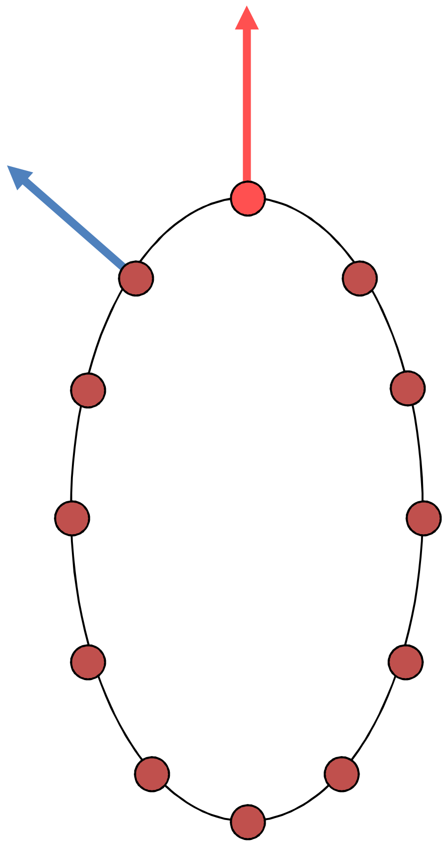
Topology of Curves

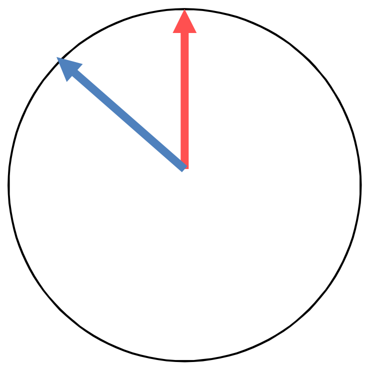




Gauss Circle

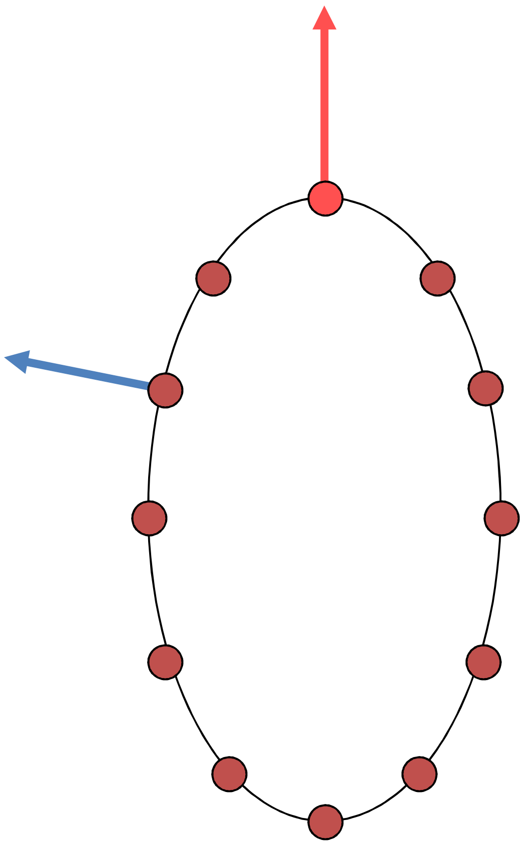
Topology of Curves

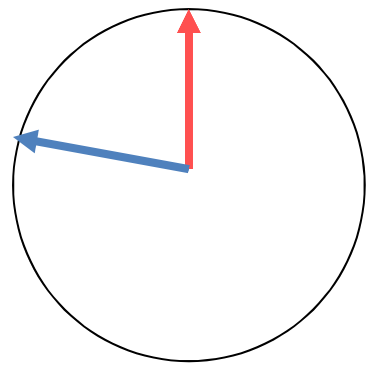




Gauss Circle

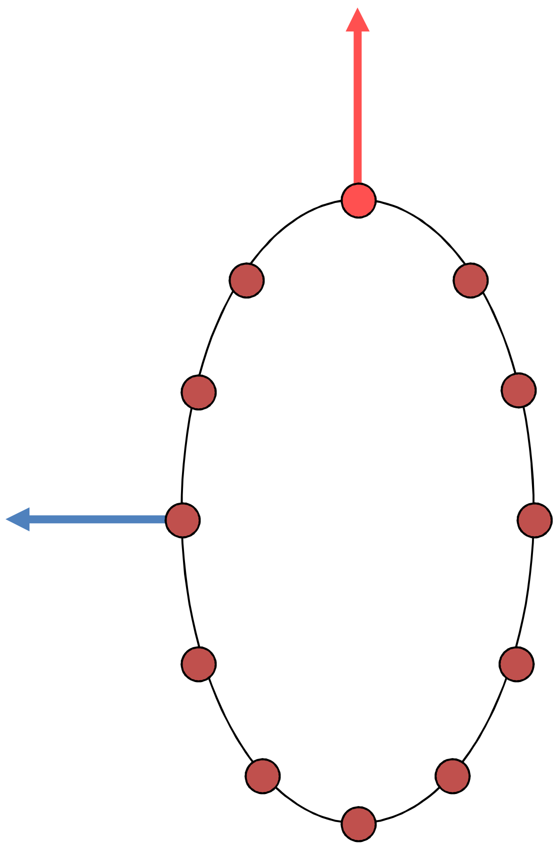
Topology of Curves

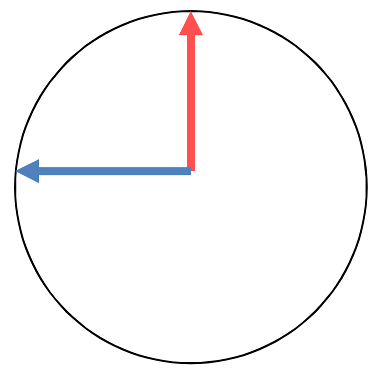




Gauss Circle

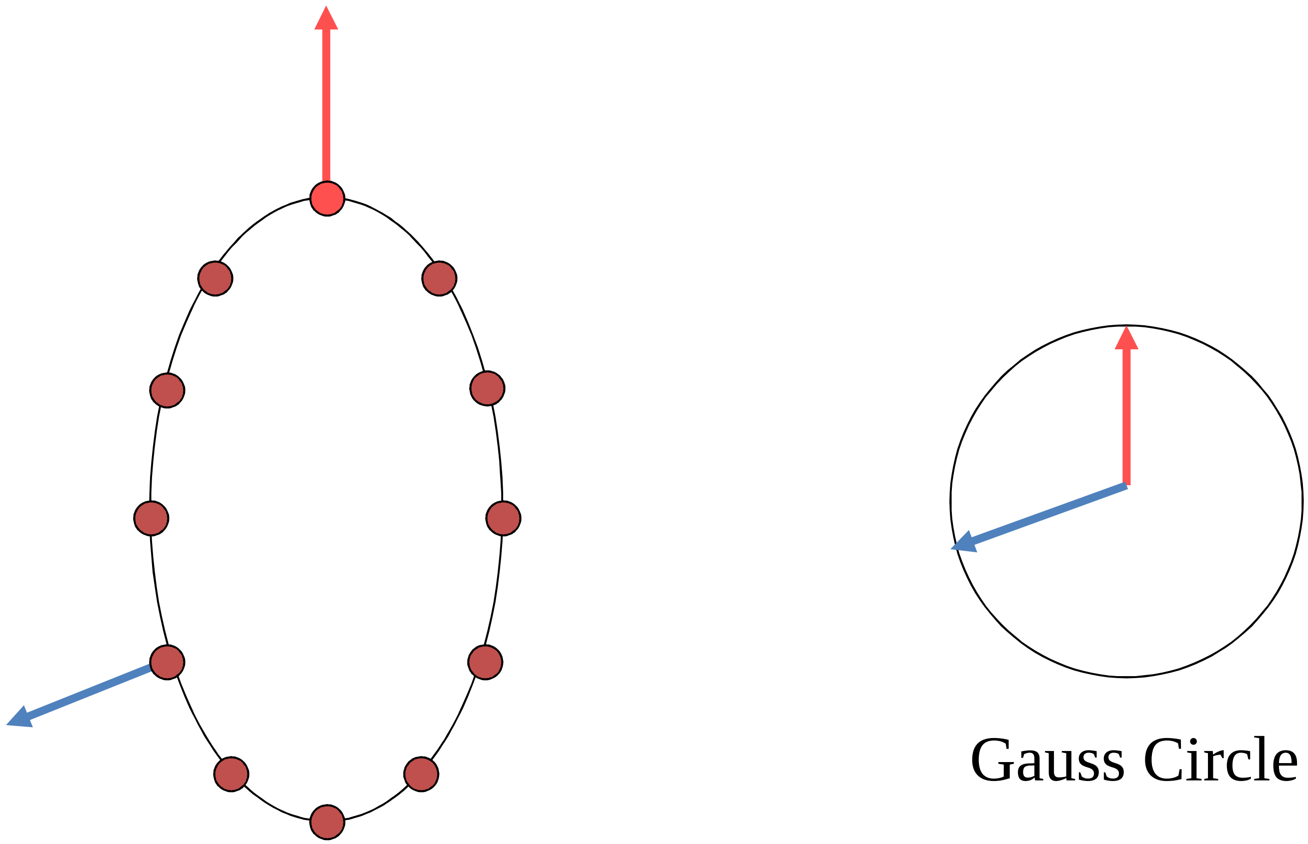
Topology of Curves





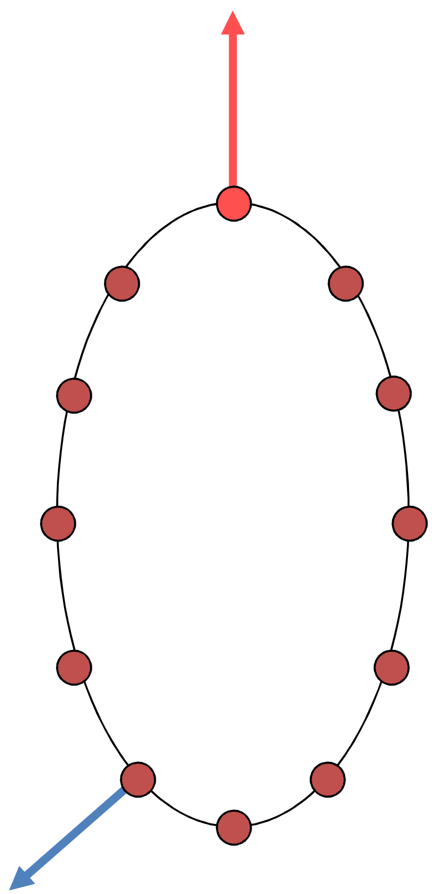
Gauss Circle

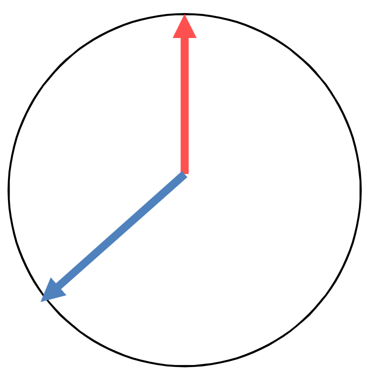
Topology of Curves



Gauss Circle

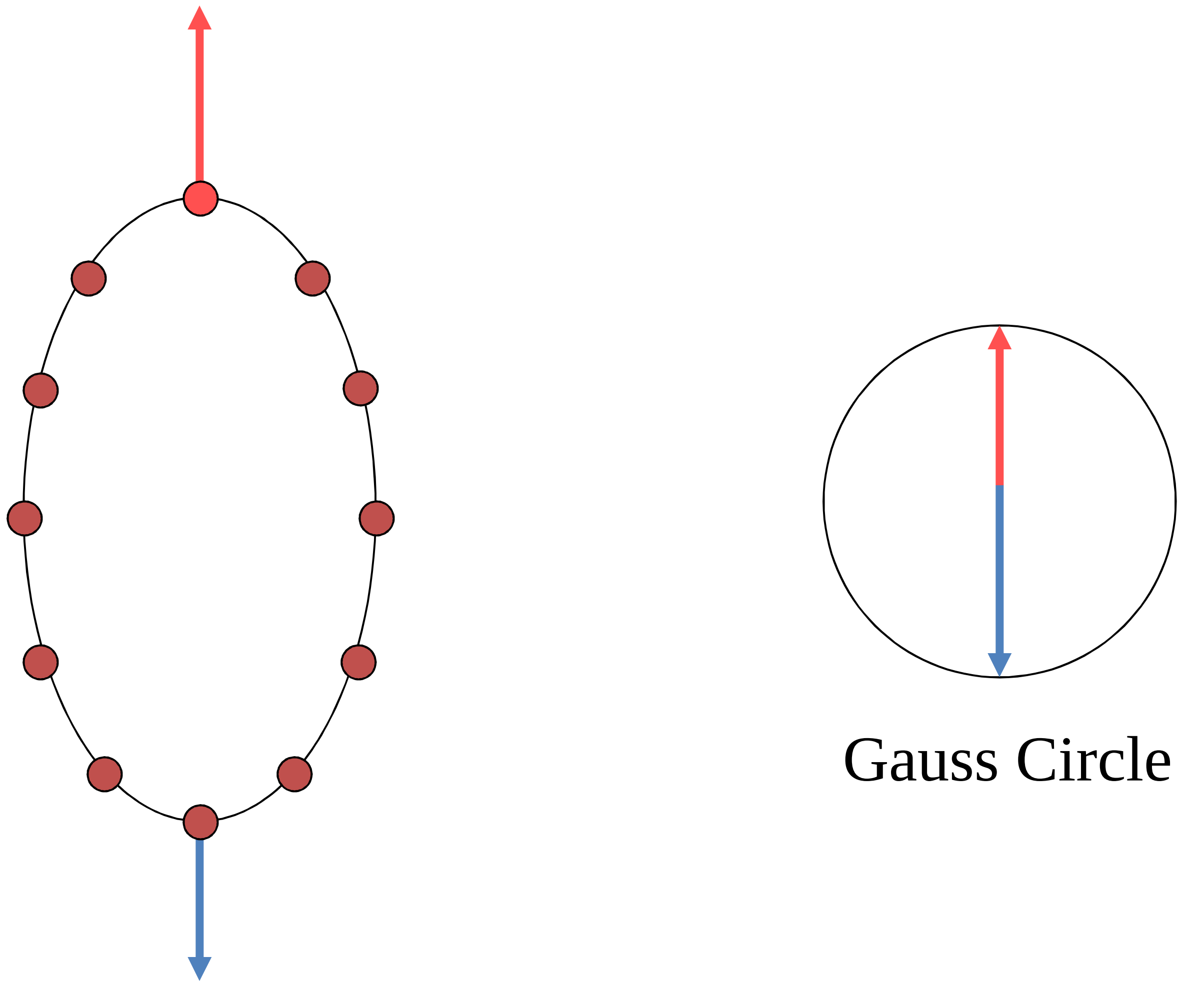
Topology of Curves



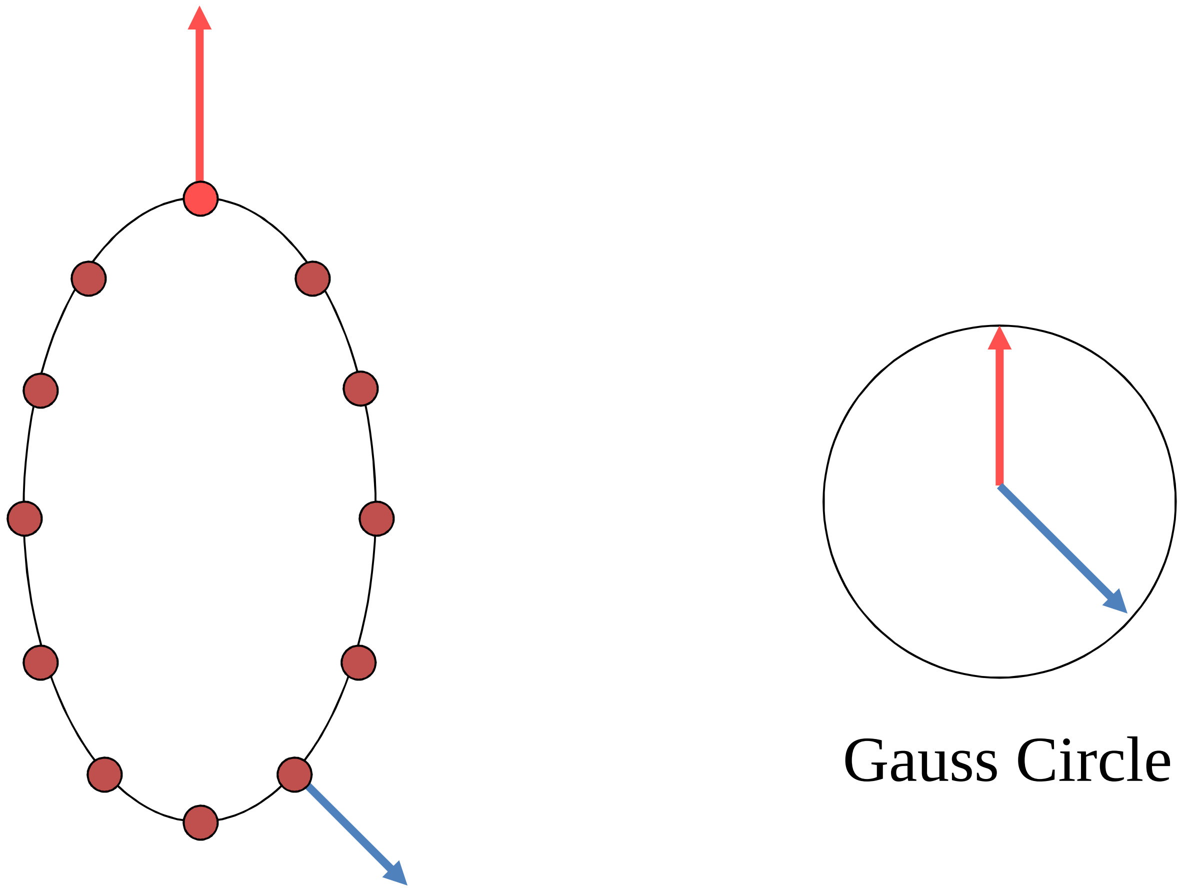


Gauss Circle

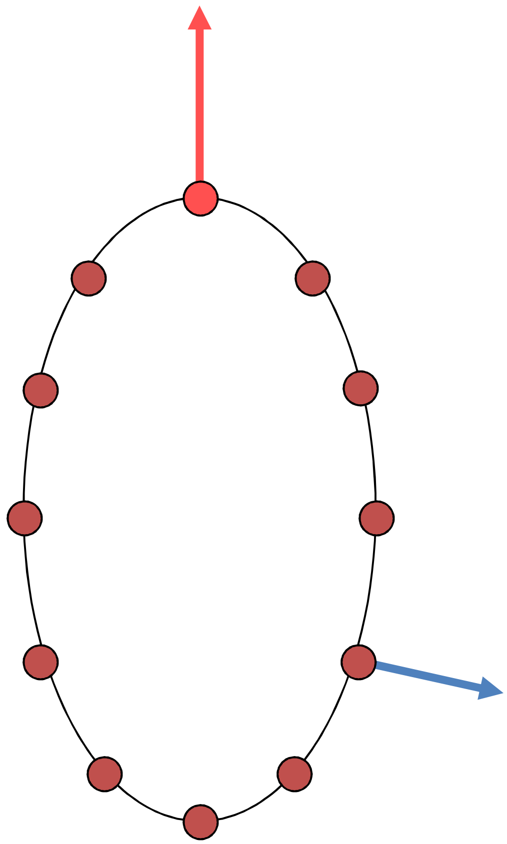
Topology of Curves

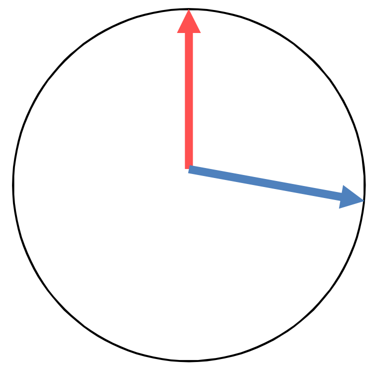


Topology of Curves



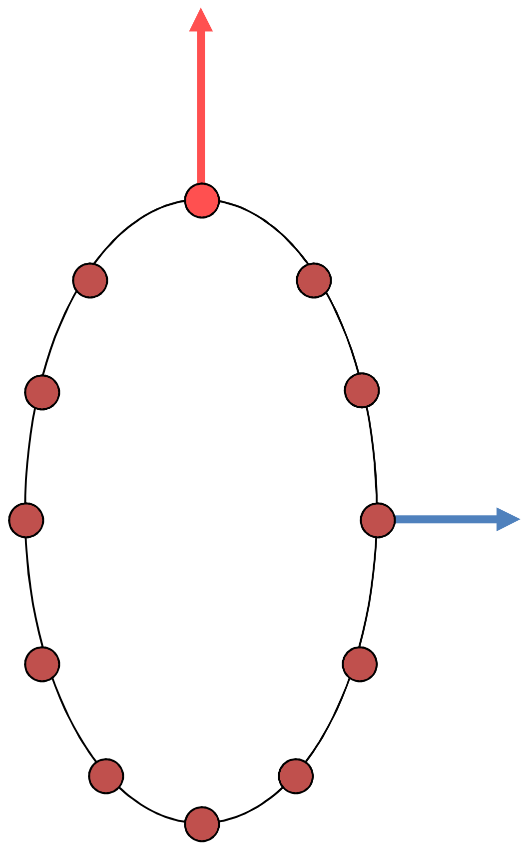
Topology of Curves

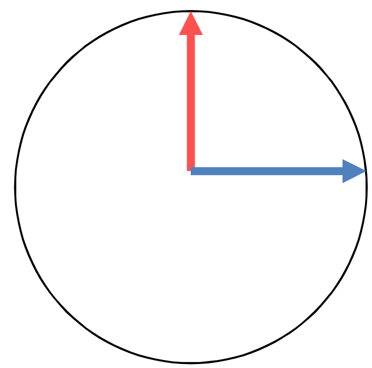




Gauss Circle

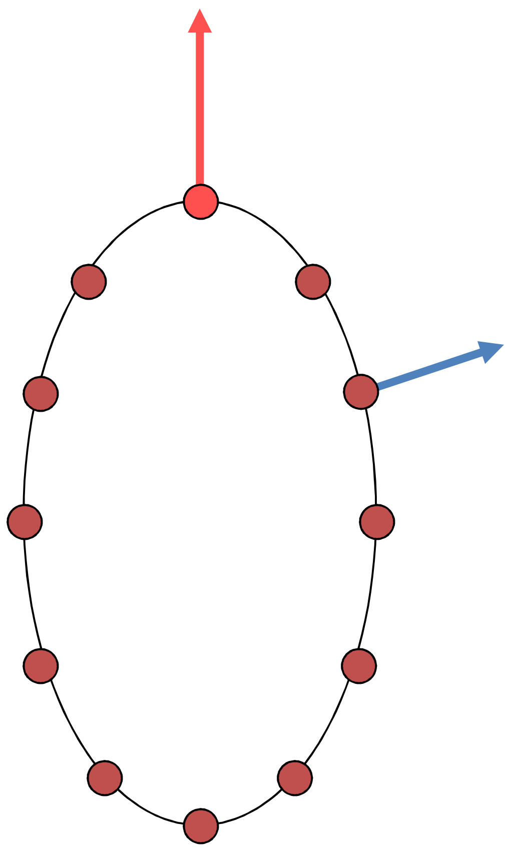
Topology of Curves

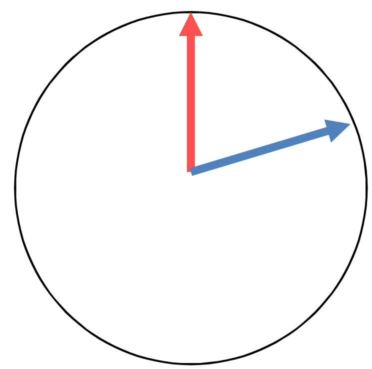




Gauss Circle

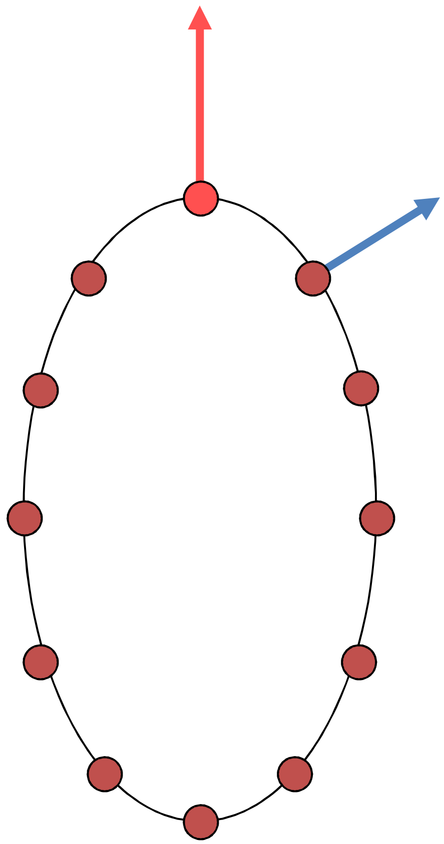
Topology of Curves

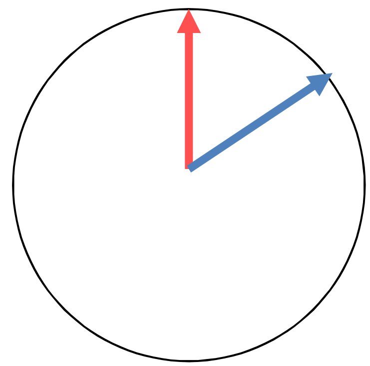




Gauss Circle

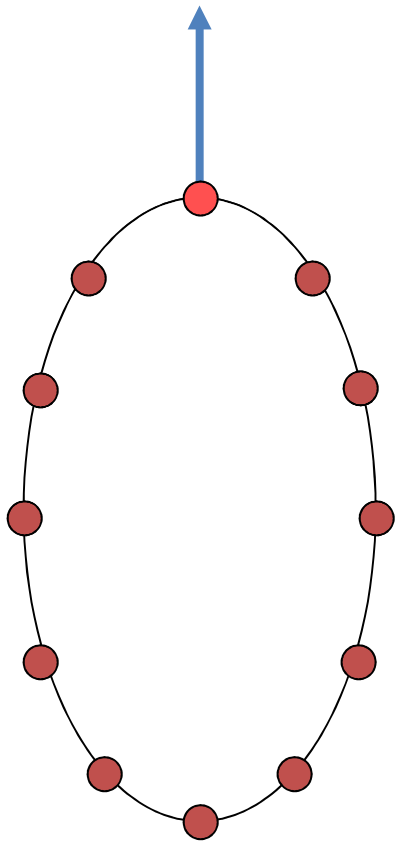
Topology of Curves

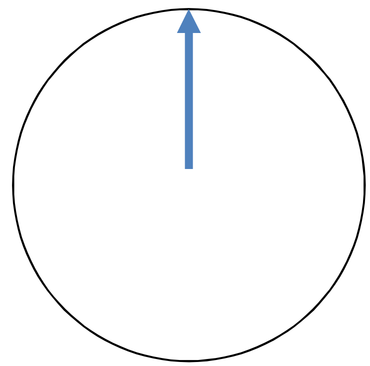




Gauss Circle

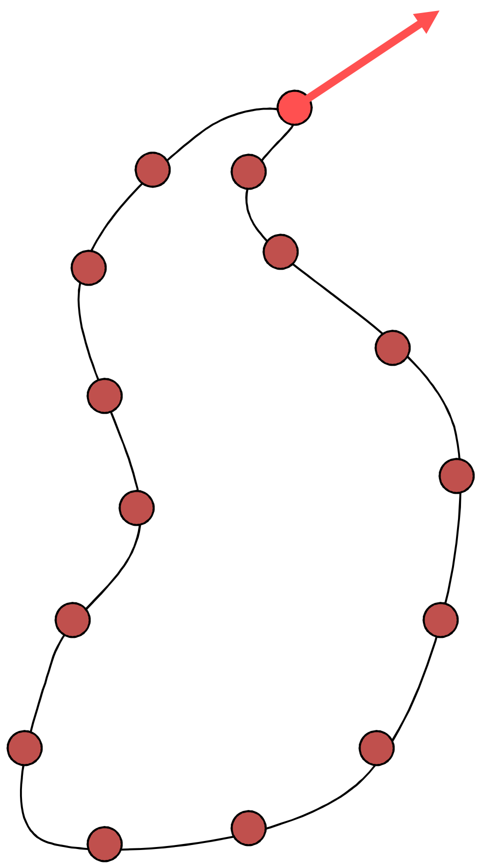
Topology of Curves

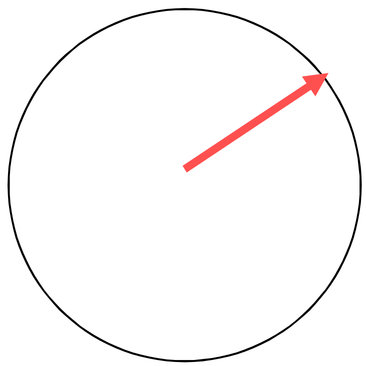




Gauss Circle

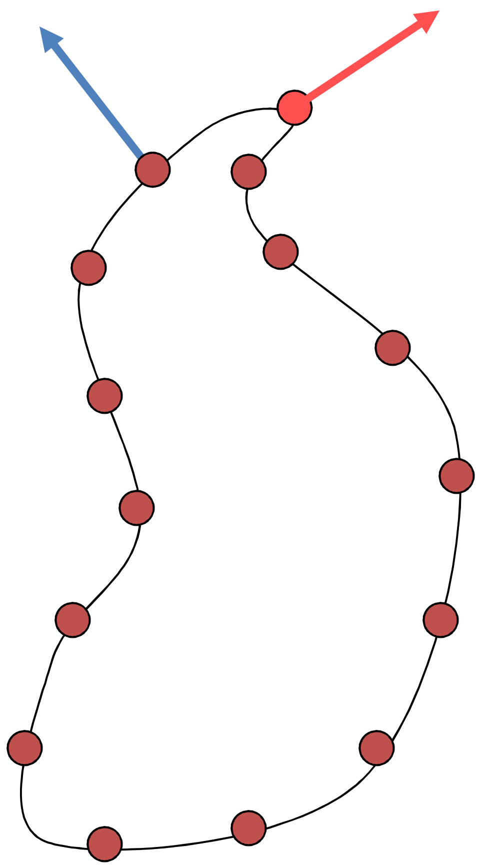
Topology of Curves

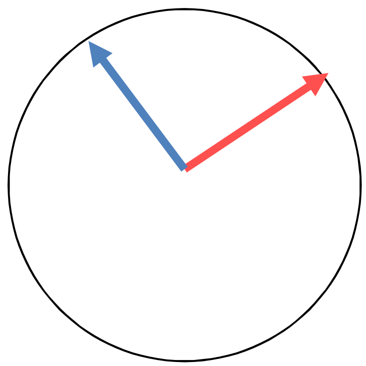




Gauss Circle

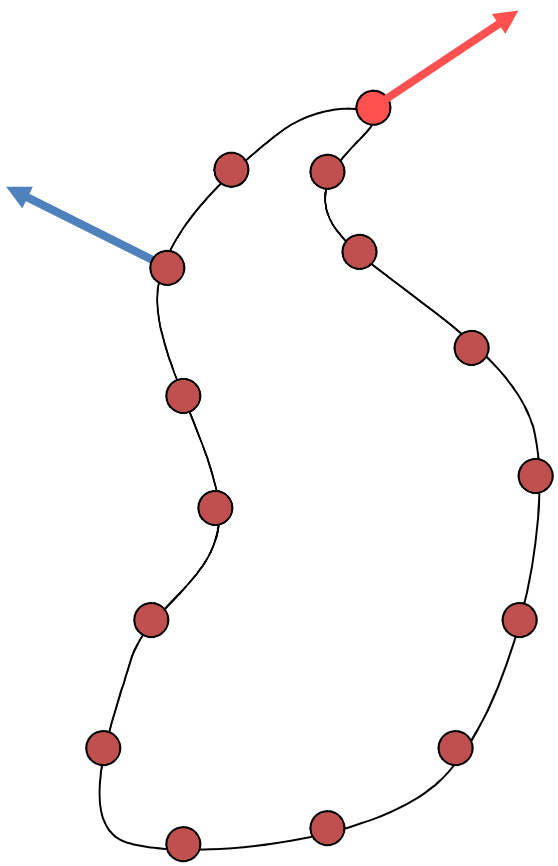
Topology of Curves

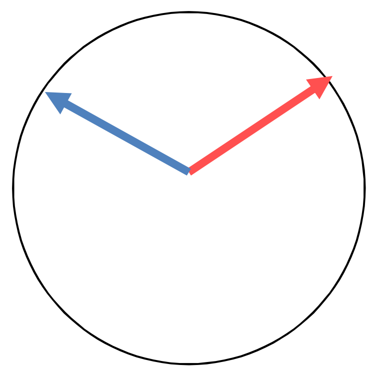




Gauss Circle

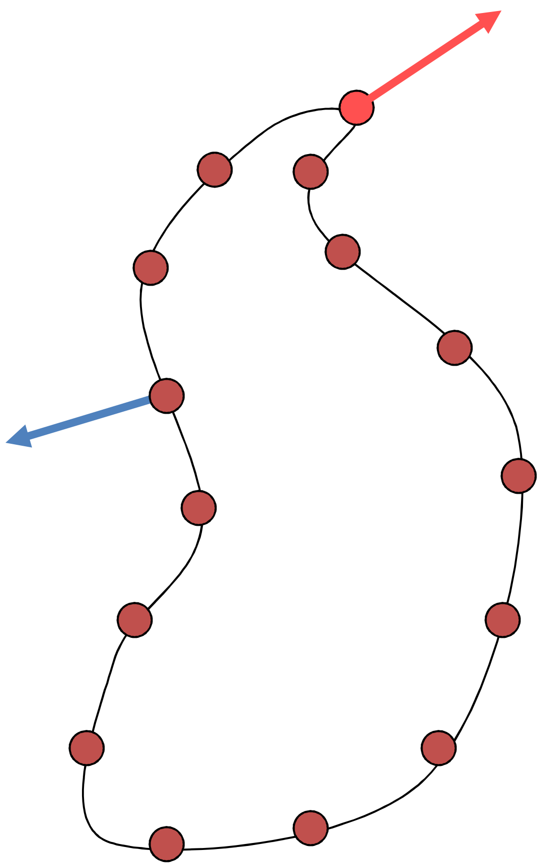
Topology of Curves

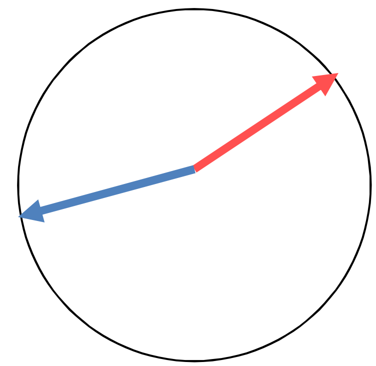




Gauss Circle

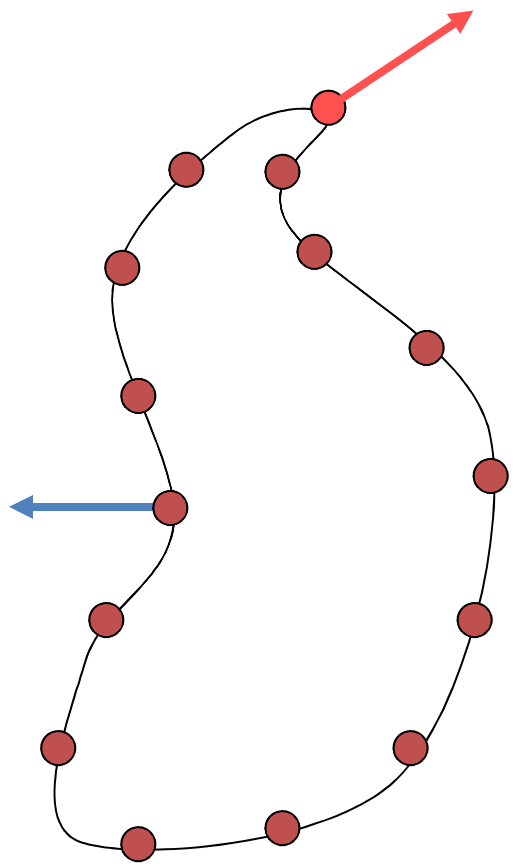
Topology of Curves

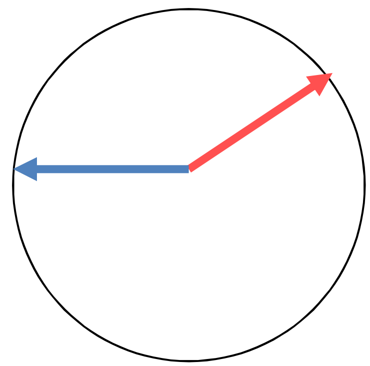




Gauss Circle

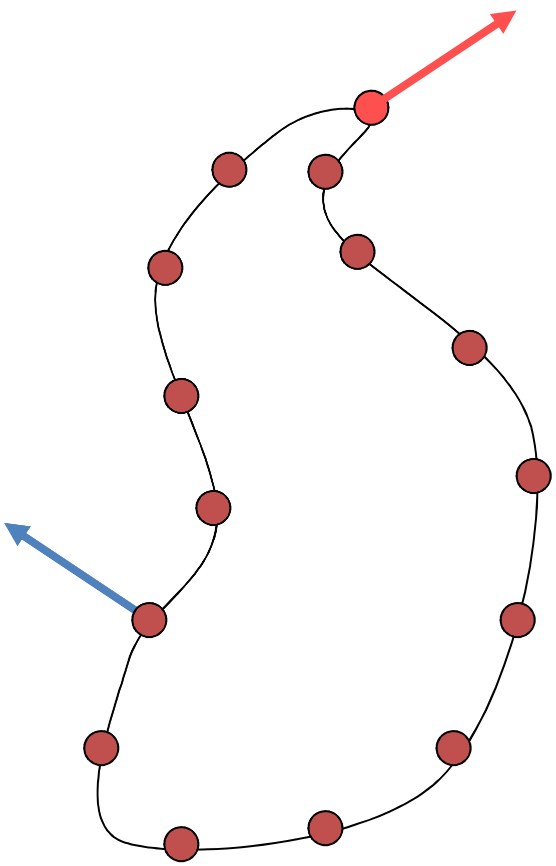
Topology of Curves

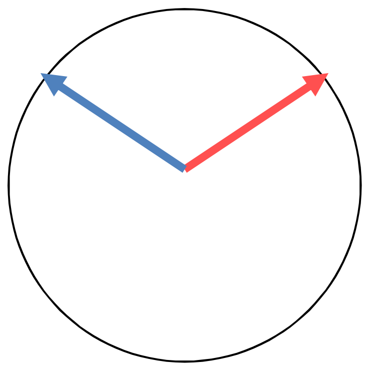




Gauss Circle

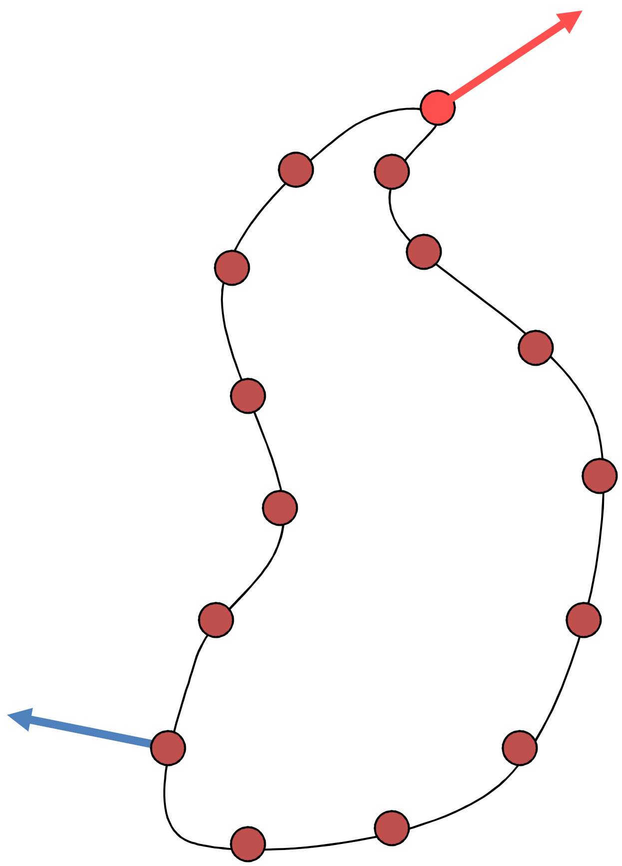
Topology of Curves

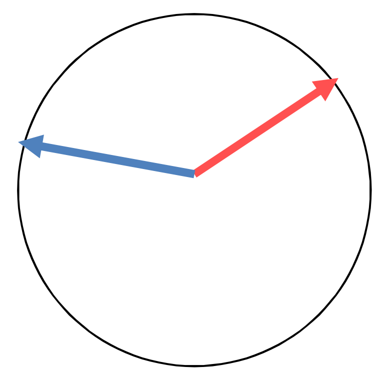




Gauss Circle

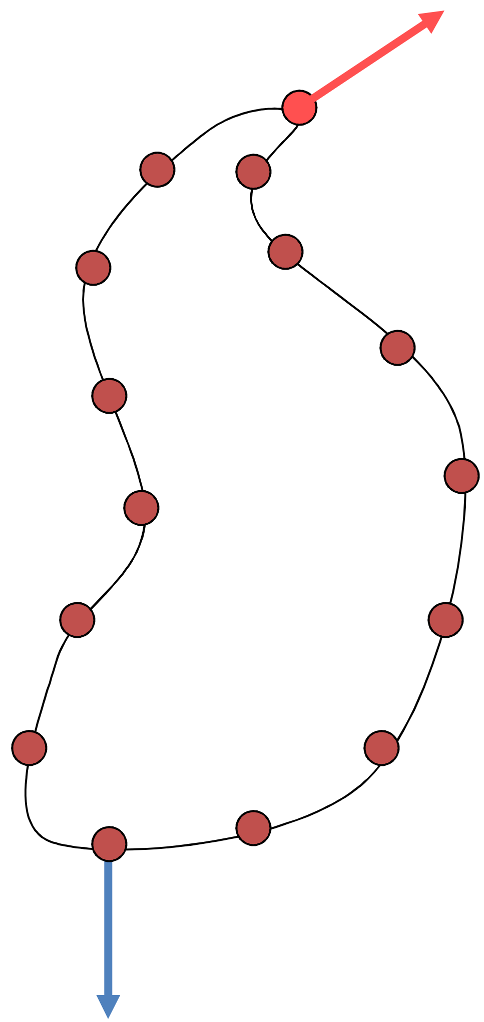
Topology of Curves

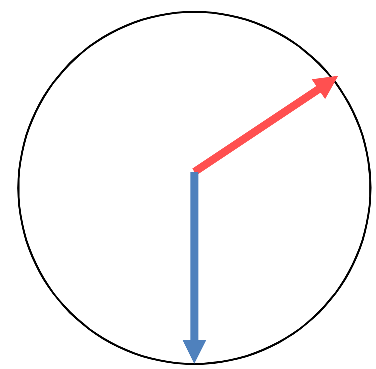




Gauss Circle

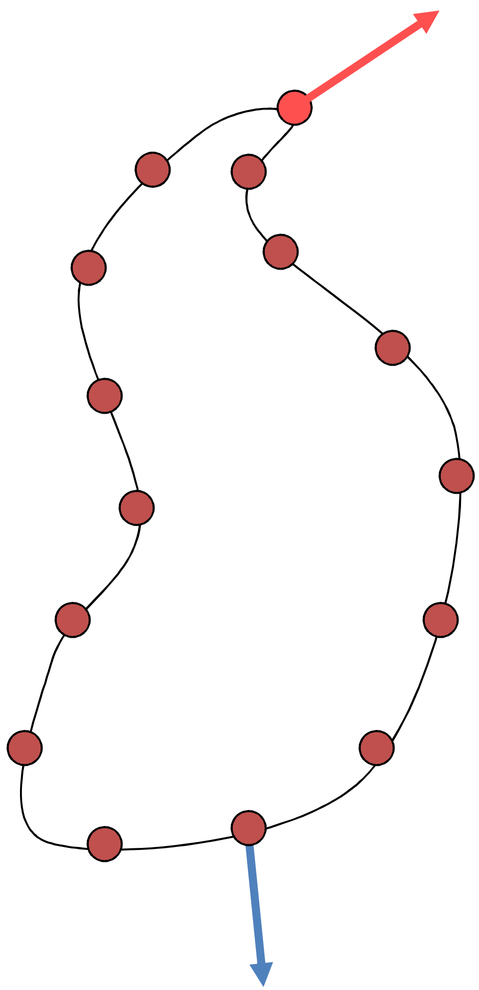
Topology of Curves

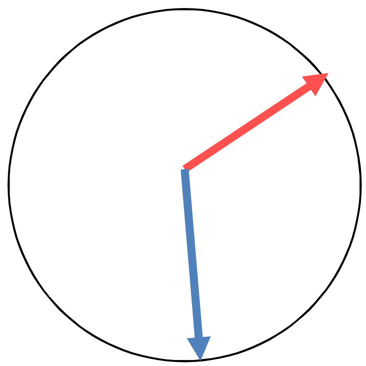




Gauss Circle

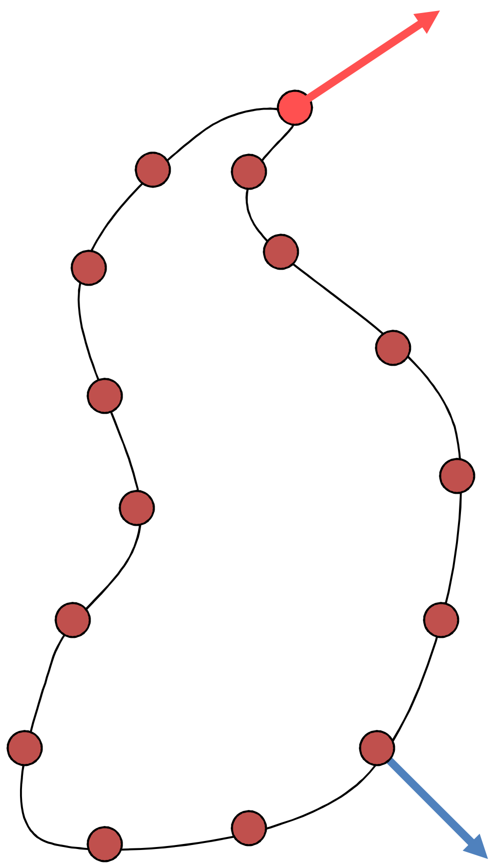
Topology of Curves

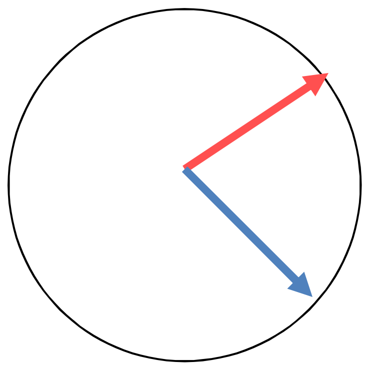




Gauss Circle

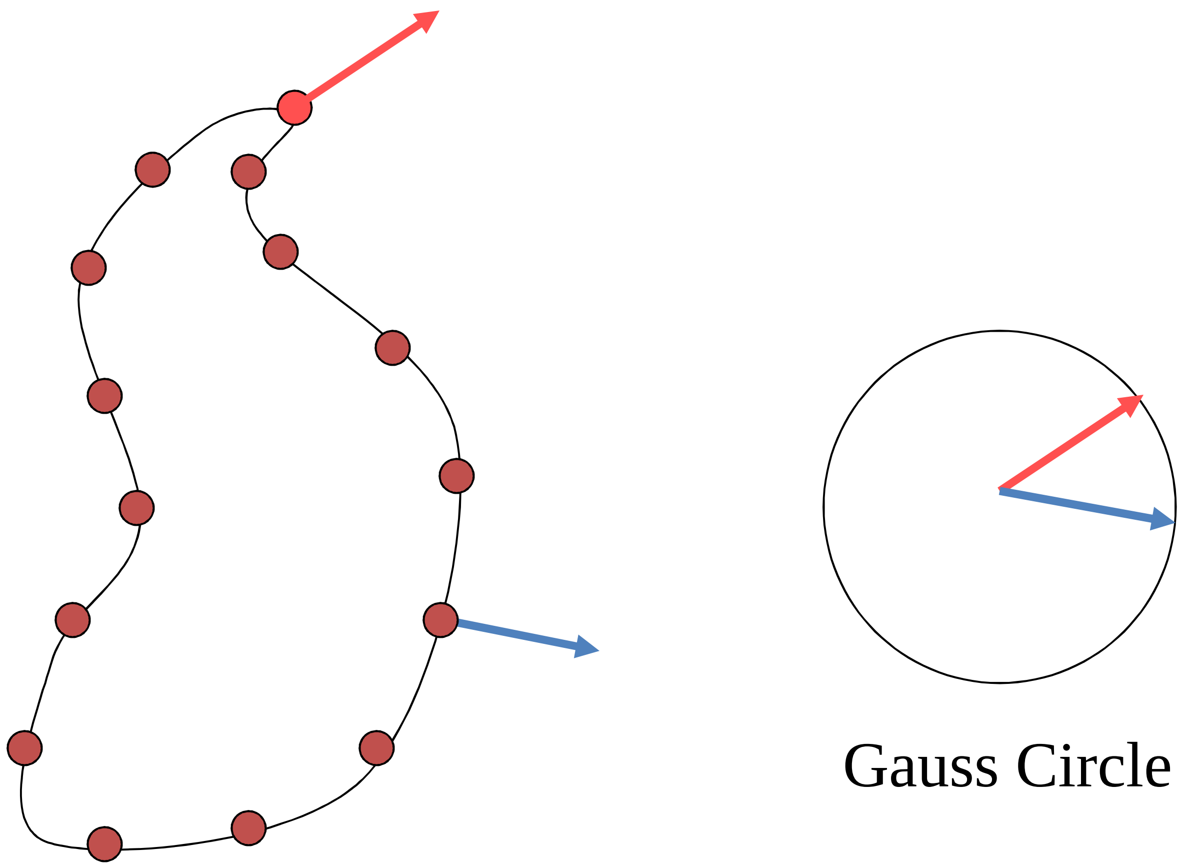
Topology of Curves



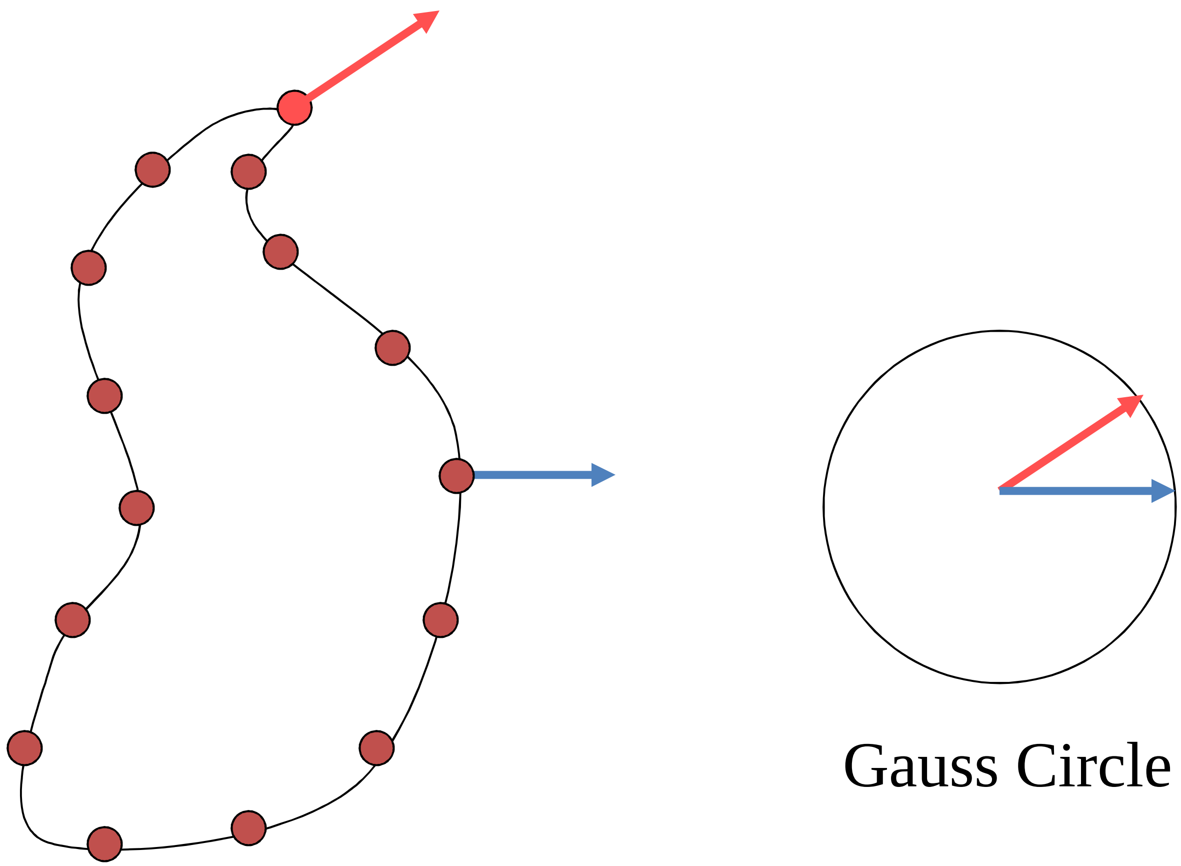


Gauss Circle

Topology of Curves

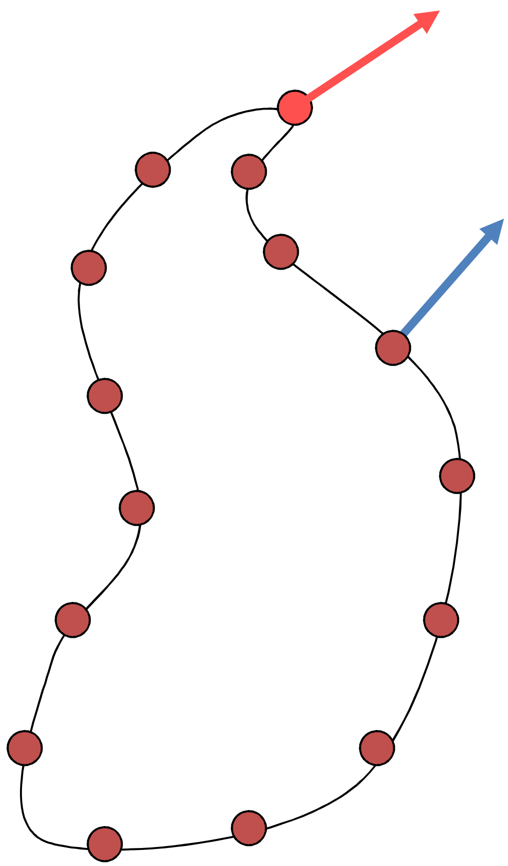


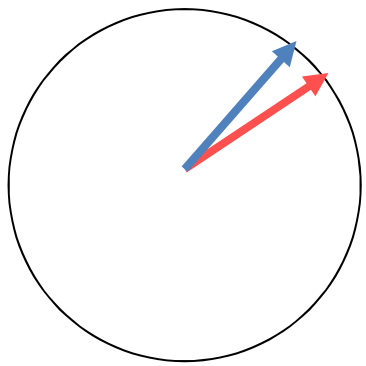
Topology of Curves



Gauss Circle

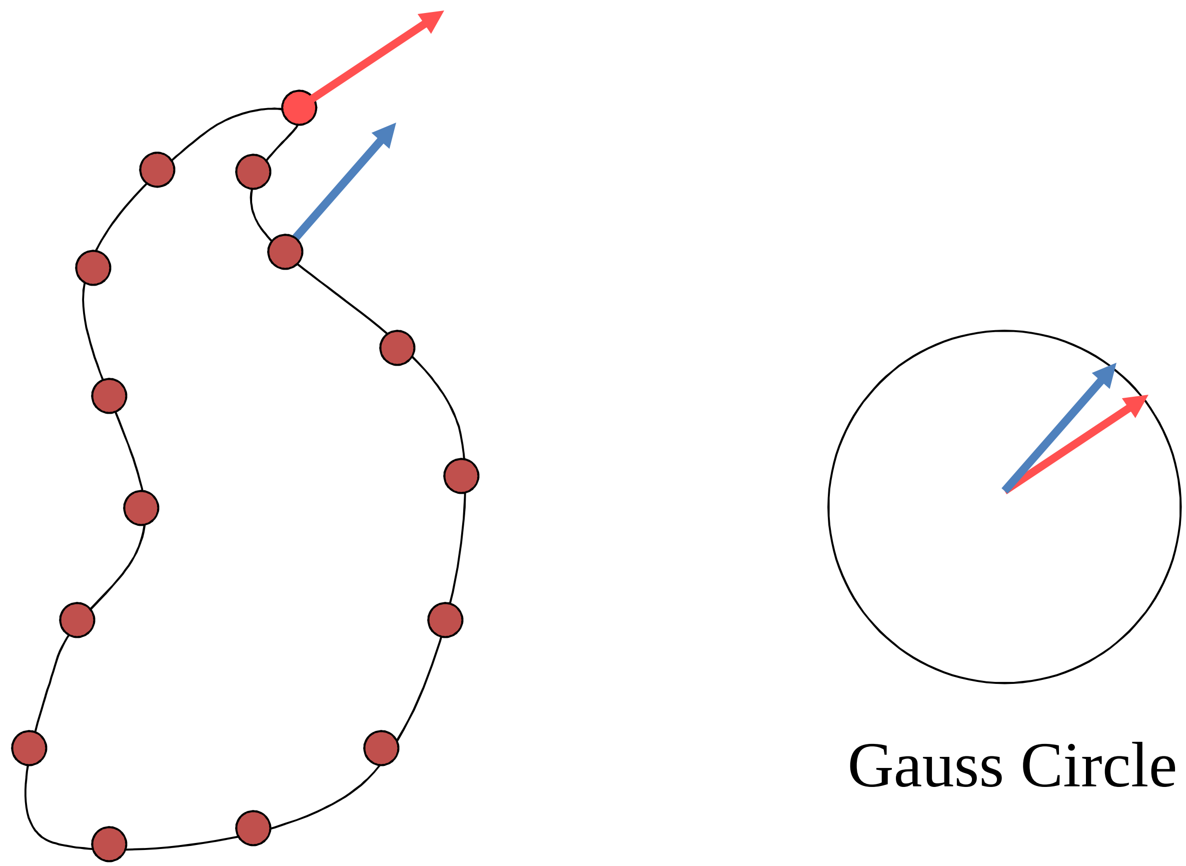
Topology of Curves



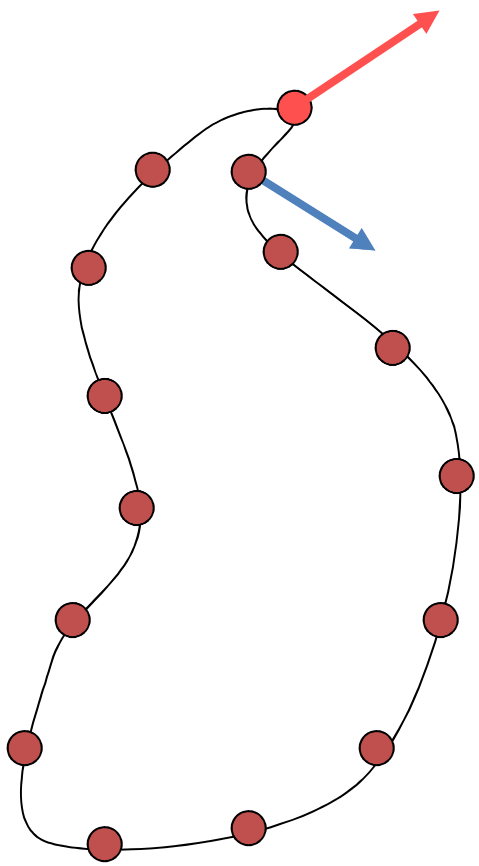


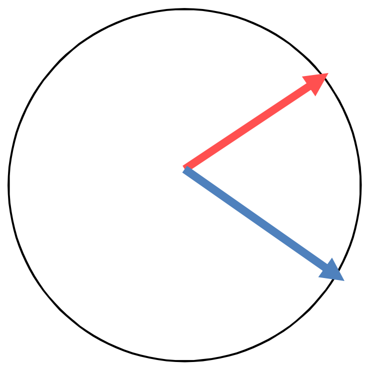
Gauss Circle

Topology of Curves



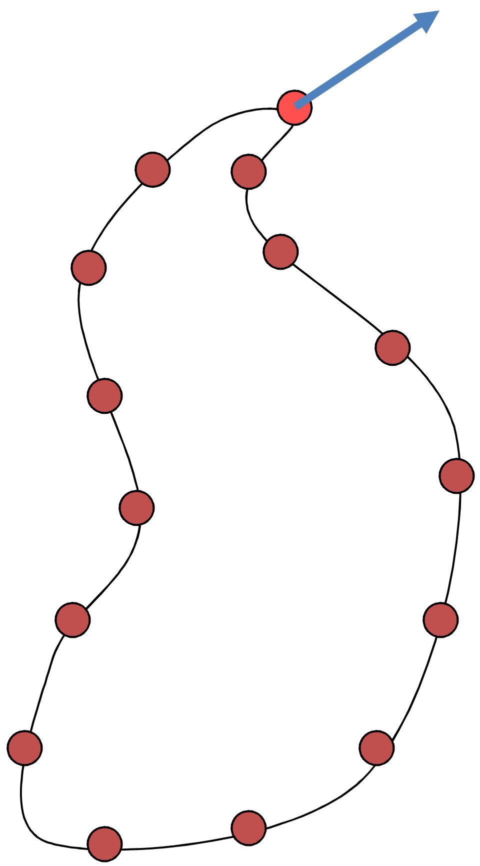
Topology of Curves

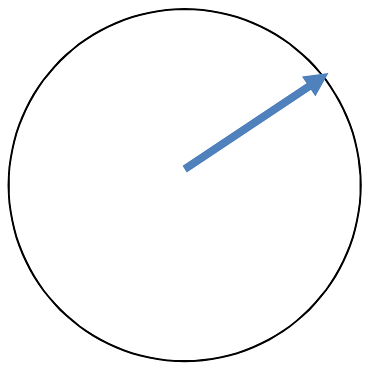




Gauss Circle

Topology of Curves





Gauss Circle

Conclusion

- Recall $\kappa(s)$ measures how fast unit tangent vectors change directions.
- Therefore, it also measures how fast unit normal vectors change directions, counting orientation
- The total curvature along a closed curve is 2π
- This shows that the total curvature of a curve is a topological quantity

Curvature of a Surface

Surface Curvature

- Use curvature of curves to define curvature of a surface
- For a point **P** on the surface
  - For every unit tangent vector V at **P**
    - Construct the plane that contains P and is parallel to V and N
    - Find the intersection of the surface and this plane
    - Compute the curvature of the intersection curve (called normal curvature)

Surface Curvature

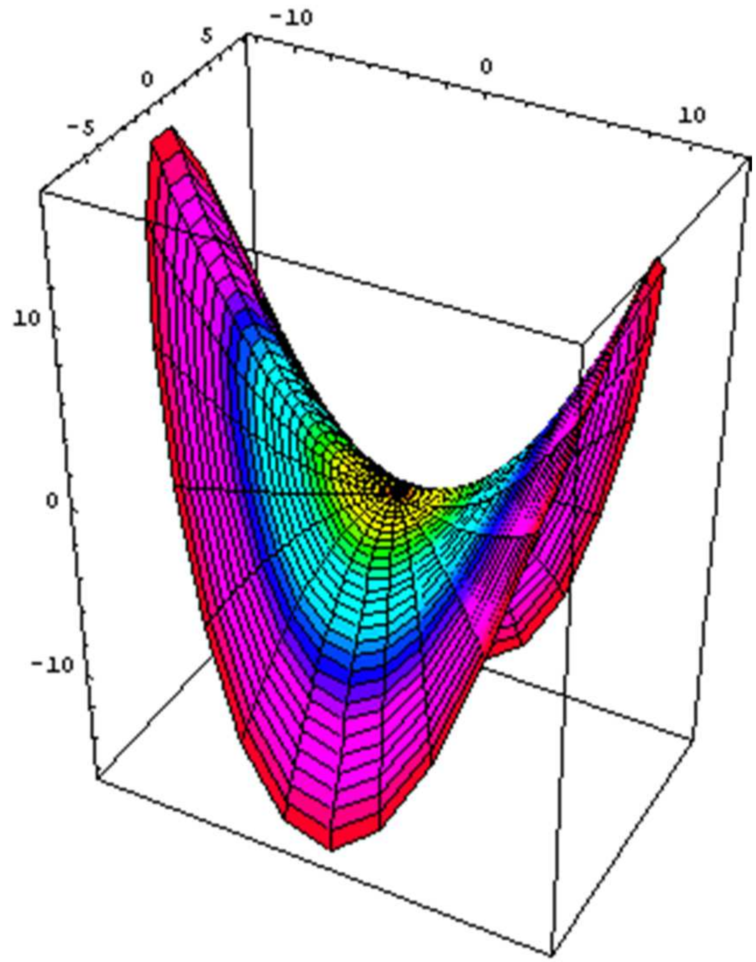


Image credit: http://www.lsus.edu/sc/math/rmabry/math223/quadricsurfaces/Images/index_gr_62.gif

Surface Curvature

- Let $t = aS_u + bS_v$ be a unit tangent vector
- The normal curvature in the direction *t* is:

$$S_{tt} \bullet N = \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} S_{uu} \bullet N & S_{uv} \bullet N \\ S_{vu} \bullet N & S_{vv} \bullet N \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} l & m \\ m & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Tangent Space

- Classifying curves of a surface through a point p

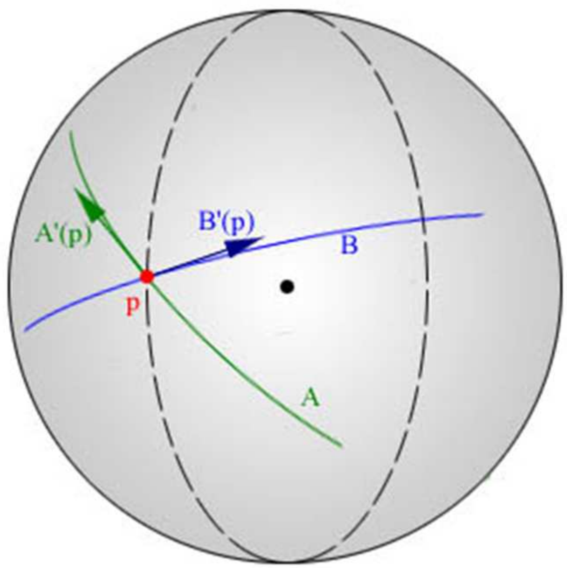


Image credit: www.peroxide.dk/.../tut10/pxdtut10.html

Tangent Space

- Two curves are equivalent if they are tangent to each other at p.

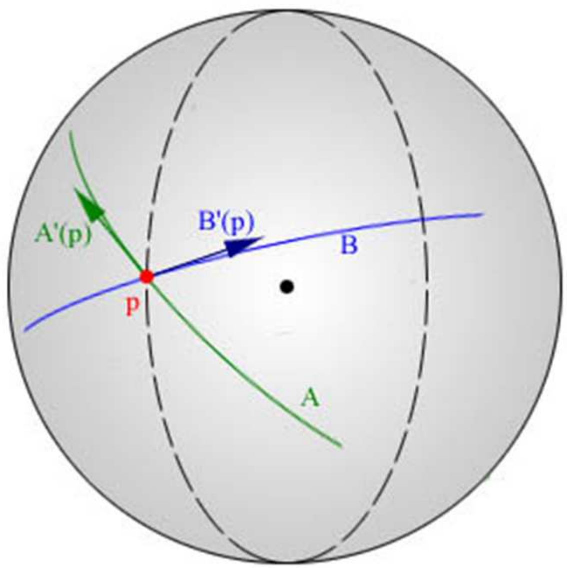


Image credit: www.peroxide.dk/.../tut10/pxdtut10.html

Tangent Space

- All different equivalent classes of curves form a line space: tangent space.

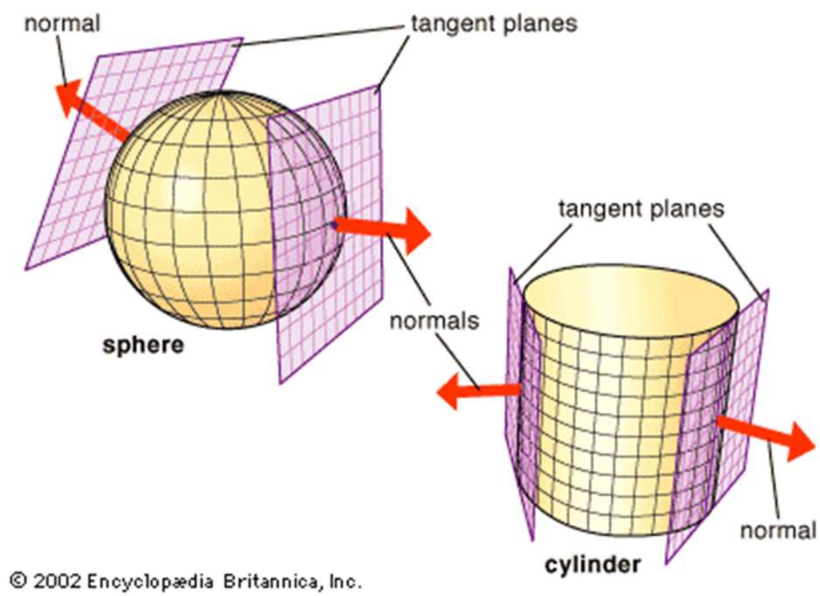


Image credit: **http://cache.eb.com/eb/image?id=70820&rendTypeId=4**

Tangent Space

- Each element in the tangent space, a vector, represents a class of equivalent curves.

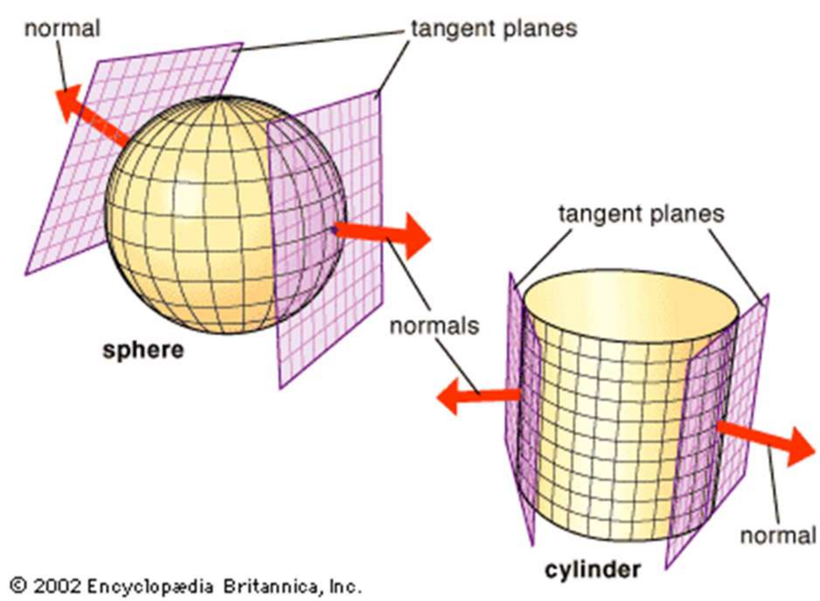


Image credit: **http://cache.eb.com/eb/image?id=70820&rendTypeId=4**

Tangent Space and Curvature

- All curves belonging to the same class have the same curvature at point p.

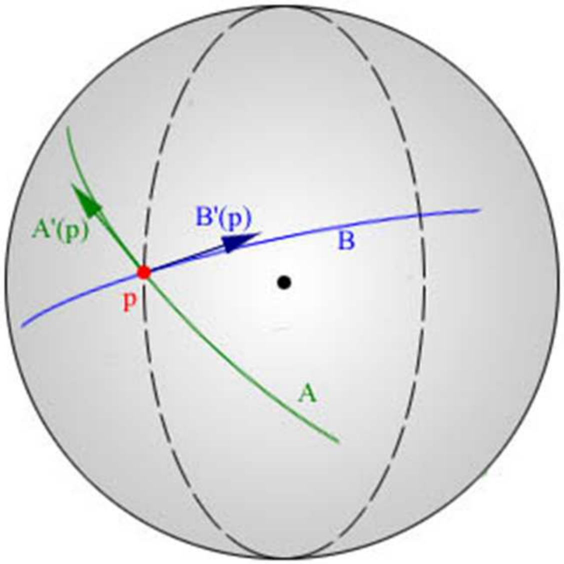


Image credit: www.peroxide.dk/.../tut10/pxdtut10.html

Tangent Space and Curvature

- Curvature depends only on the tangent vector.

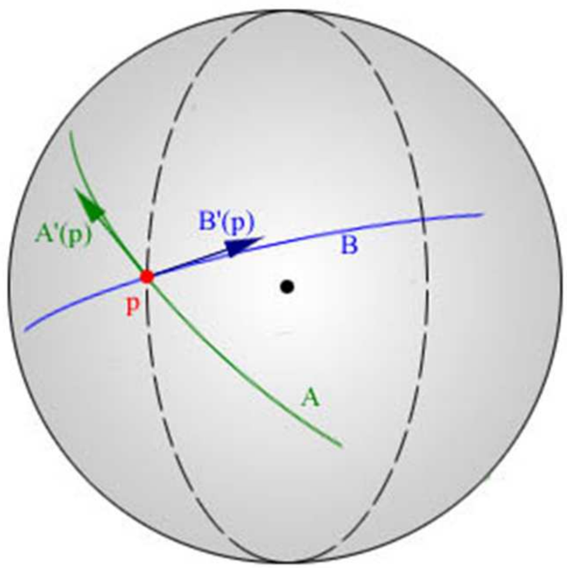


Image credit: www.peroxide.dk/.../tut10/pxdtut10.html

Tangent Space and Curvature

- A tangent vector can be represented by

  $t = aS_u + bS_v$ where

  $$S_u = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \end{pmatrix}$$

  $$S_v = \begin{pmatrix} \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{pmatrix}$$

- The curvature

  $\kappa(a,b)$

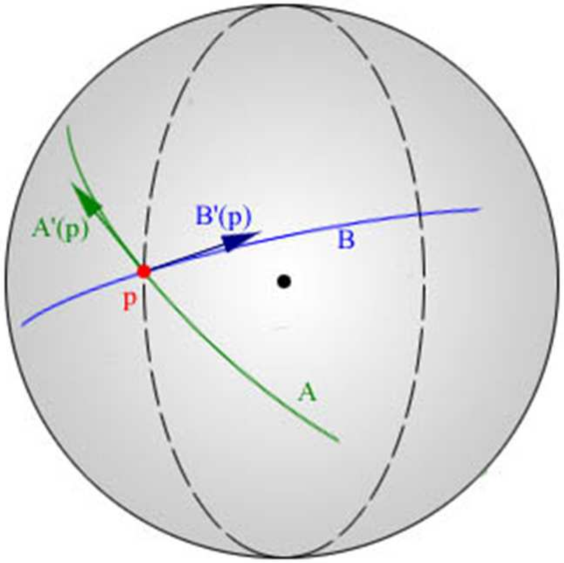


Image credit: www.peroxide.dk/.../tut10/pxdtut10.html

Curvature

- The curvature function is a quadratic function

$$\kappa(a,b) = la^2 + 2mab + nb^2$$
$$= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} l & m \\ m & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

- Where

$$l = S_{uu} \bullet N \quad m = S_{uv} \bullet N \quad n = S_{vv} \bullet N$$

Surface Curvature

$$\kappa(a,b) = la^2 + 2mab + nb^2$$

$$= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} l & m \\ m & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

- The quadric form has a maximum κ_1 and a minimum κ_2, which are the eigenvalues of the matrix.
- The corresponding eigenvectors are principle curvature directions.

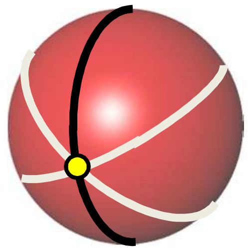
Discrete Principle Curvatures

- κ_1 and κ_2 satisfy $x^2 - 2Hx + K = 0$
- So $\kappa_{1,2} = H \pm \sqrt{H^2 - K}$

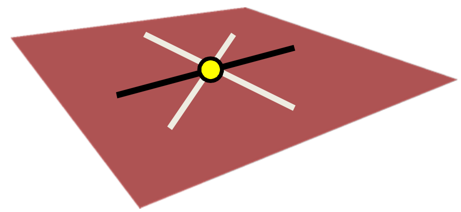
- Principle directions are found by finding the eigenvectors of the curvature tensor

Curvature Tensor

Isotropic



spherical

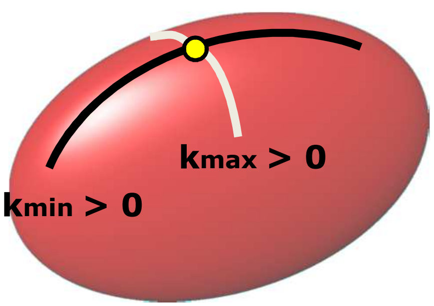


planar

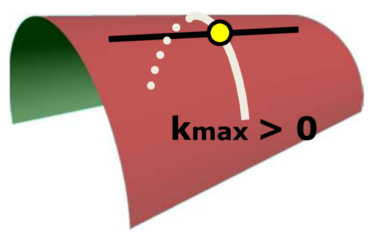
$\mathcal{T}(\mathbf{v})$

Anisotropic

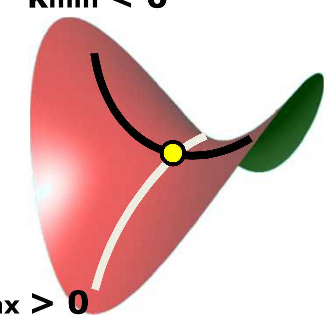
2 principal directions



elliptic



parabolic



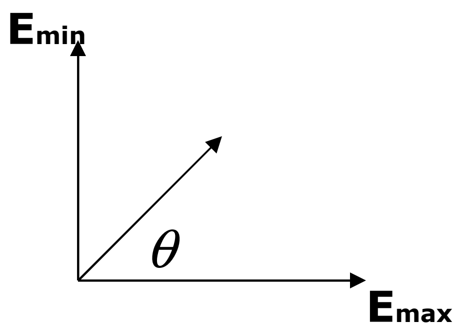
hyperbolic

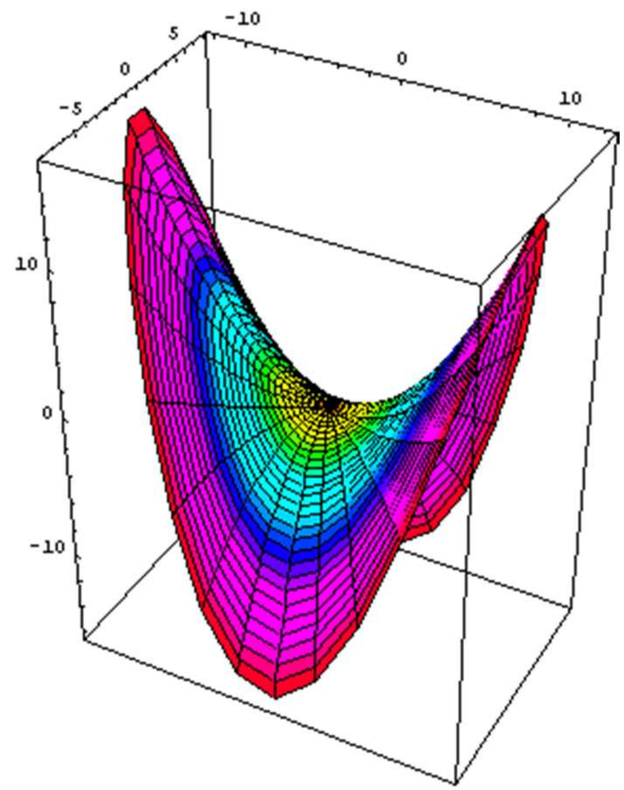
Image credit: Alliez et al.

Surface Curvature

- Some special numbers about the curvature tensor:
  - Normal curvature

$$\kappa_\theta = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta$$





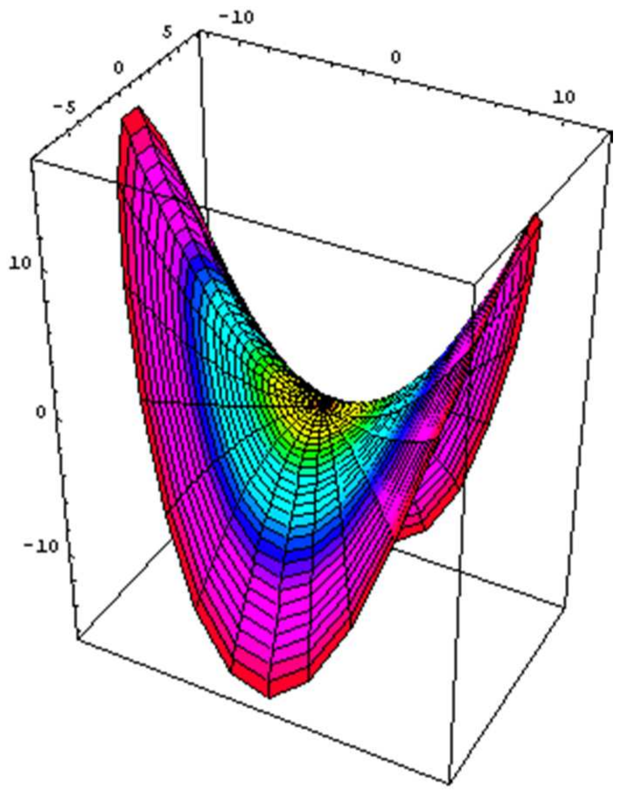
Surface Curvature

- Some special numbers about the curvature tensor:
  - Mean curvature

$$H = \frac{\kappa_1 + \kappa_2}{2}$$

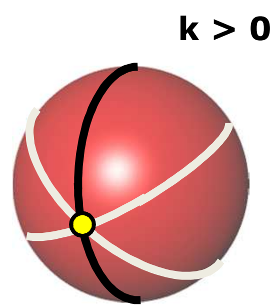
  - Gaussian curvature

$$K = \kappa_1 \kappa_2$$

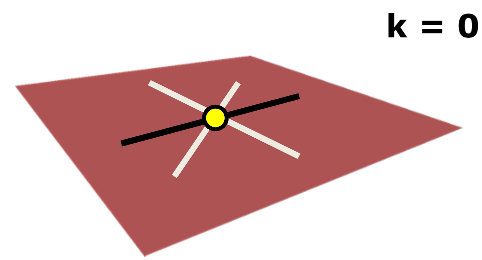


Curvature Tensor

Isotropic



spherical

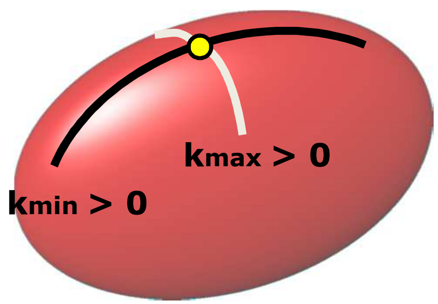


planar

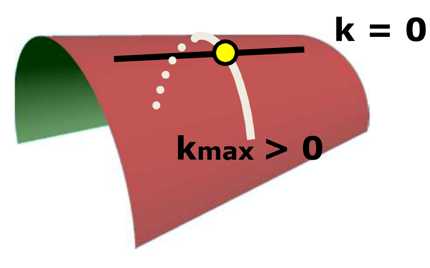
$\mathscr{T}(\mathbf{v})$

Anisotropic

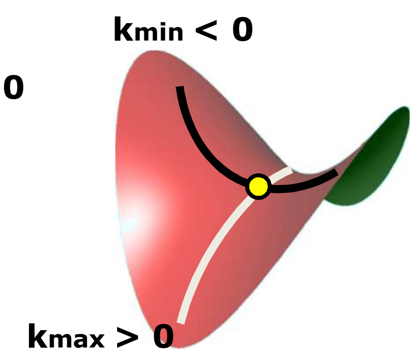
2 principal directions



elliptic



parabolic



hyperbolic

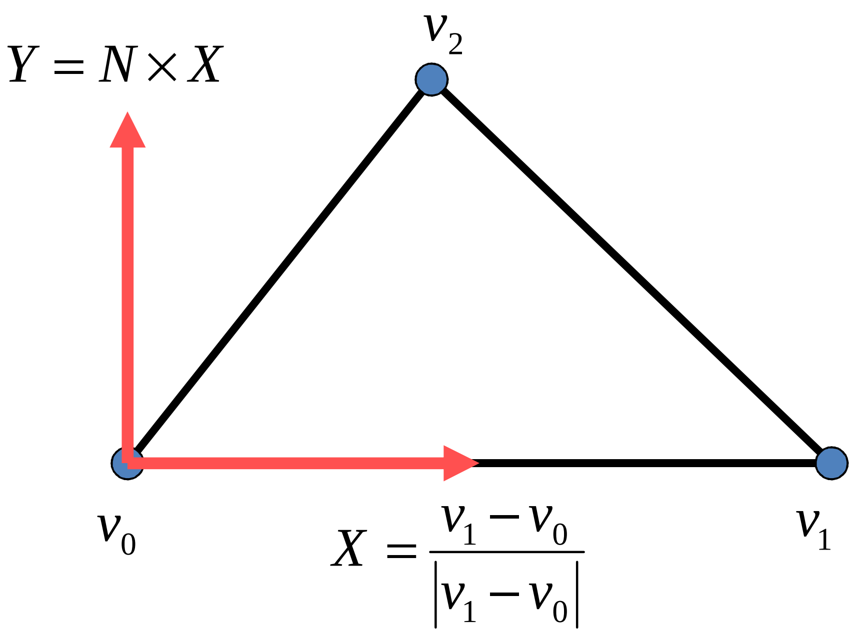
Image credit: Alliez et al.

How to Compute Curvature Tensor on a Mesh?

- Triangle:
  - Normal (well-defined, not continuous)
  - Curvature (zero inside a triangle)
- Vertex:
  - Normal (the average of the normal of the incident triangles)
  - Curvature

Local Frame

- Triangle



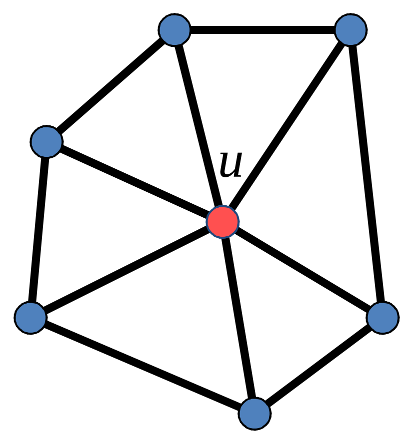
Local Frame

- Vertex
  - Find a 3D vector w

    $$X = N \times w$$

    $$Y = N \times X$$

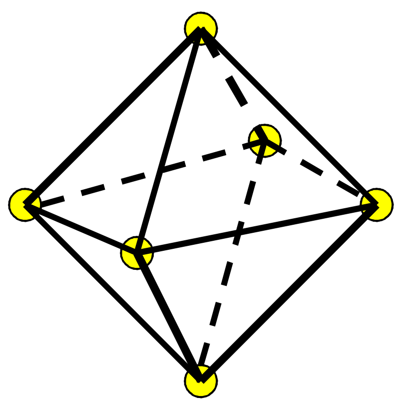
  - How do you find w?
    - Anyway is fine so long w is not co-linear with N

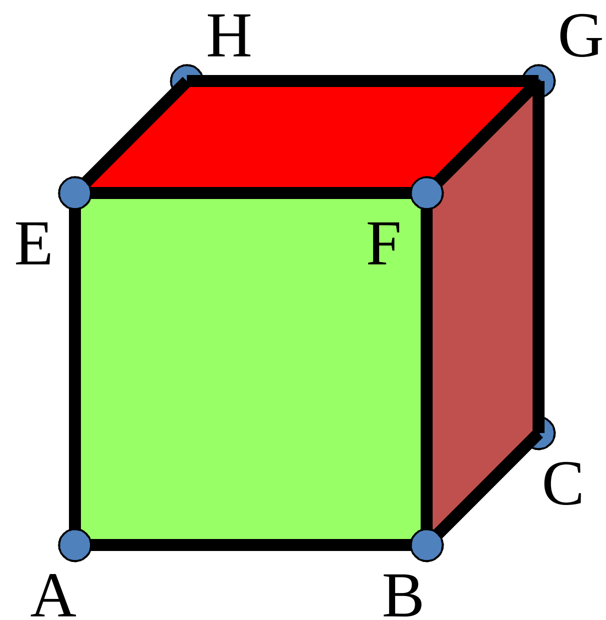


Discrete Gaussian Curvature

- Discrete Gaussian curvature for a vertex:

$$K(v) = 2\pi - \sum_{t \in \sigma(v)} \alpha(t_v)$$

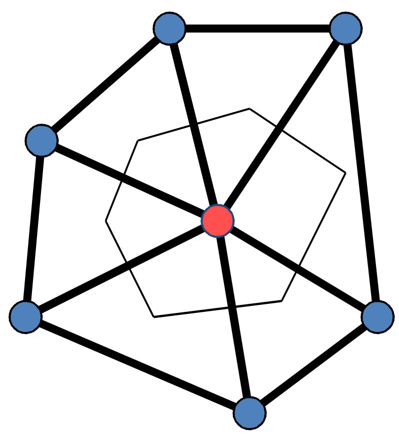




Discrete Gaussian Curvature

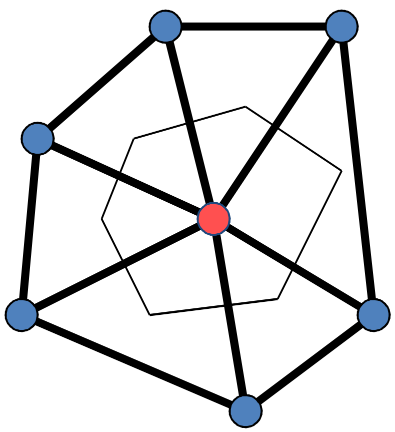
- But now the Gaussian curvature is not smooth.
- Treat the curvature at a vertex as a spatial average of its surrounding space

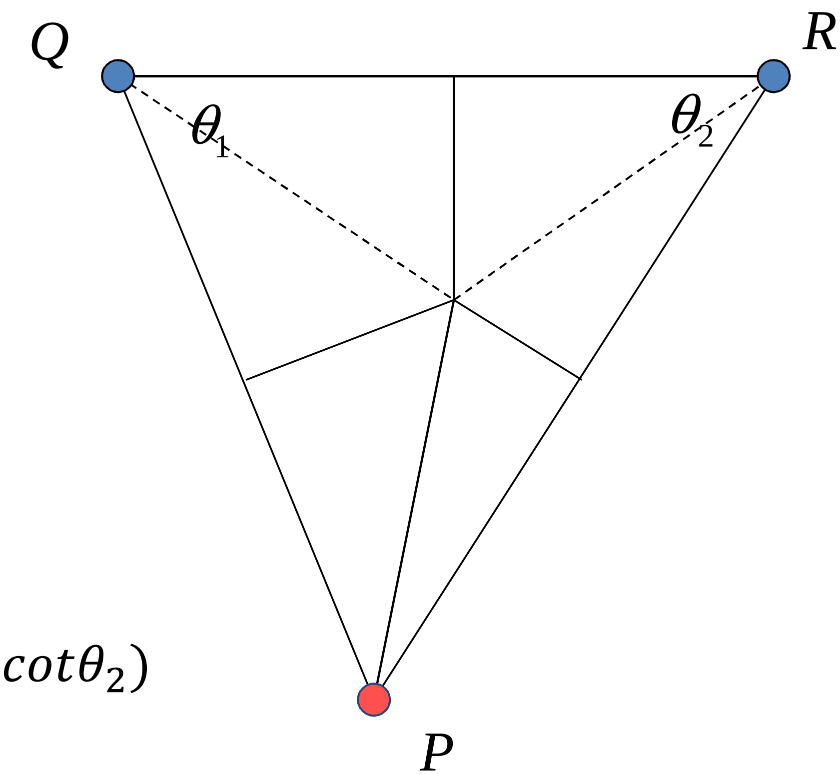
$$K(v) = \frac{2\pi - \sum_{t \in \sigma(v)} \alpha(t_v)}{Area(v)}$$



Voronoi Area Computation

- Non-obtuse (<π/2)

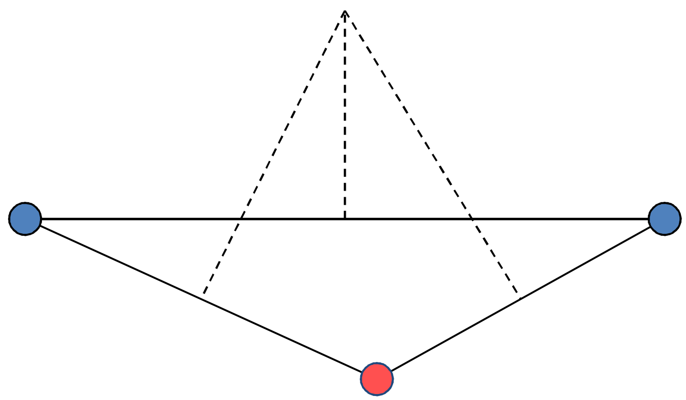




$$A_{Voronoi} = \frac{1}{8}(|PR|^2 cot\theta_1 + |PQ|^2 cot\theta_2)$$

Voronoi Area Computation

- Obtuse ($>\pi/2$)
  - Voronoi region is outside of the triangle



The algorithm for Voronoi Area Computation

```
𝒜_Mixed = 0
For each triangle T from the 1-ring neighborhood of x
    If T is non-obtuse,          // Voronoi safe
        // Add Voronoi formula (see Section 3.3)
        𝒜_Mixed += Voronoi region of x in T
    Else                         // Voronoi inappropriate
        // Add either area(T)/4 or area(T)/2
        If the angle of T at x is obtuse
            𝒜_Mixed += area(T)/2
        Else
            𝒜_Mixed += area(T)/4
```

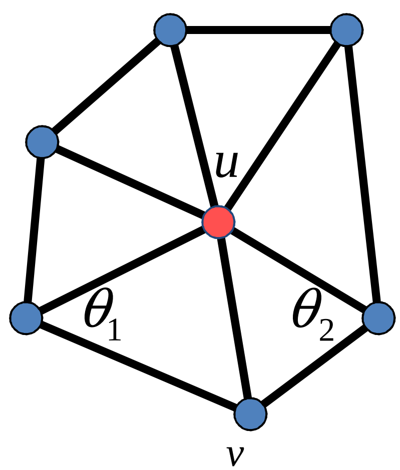
Discrete Mean Curvature

- Discrete mean curvature:

$$2H(u)N_u = \frac{\sum_{v \in \sigma(u)} \beta(u_v)}{Area(u)}$$

- where

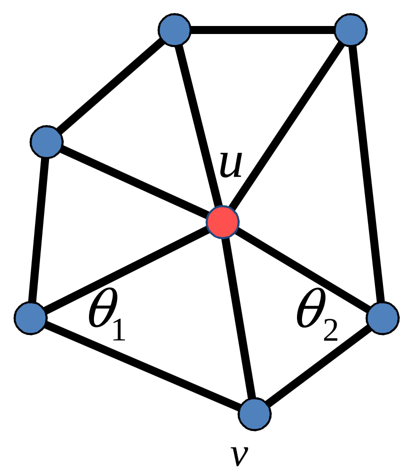
$$\beta(u_v) = \frac{(\cot\theta_1 + \cot\theta_2)}{2}(u - v)$$



Normal Curvature Along an Edge

- Discrete normal curvature for a vertex u along an edge (u, v) is:

$$\kappa_{u,v}^{\ \ N} = \frac{2(u-v) \bullet N_u}{\|u-v\|^2}$$



Mean Curvature and Normal Curvatures

- Discrete mean curvature at a vertex v is the weighted sum of the normal curvatures for edges incident to v:

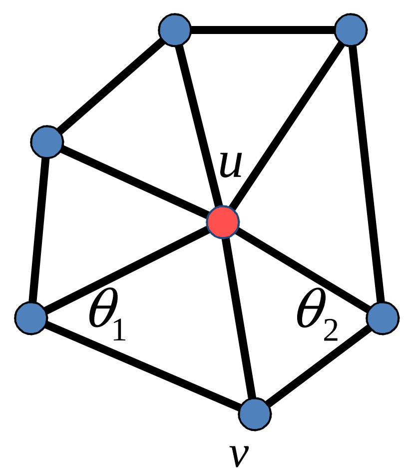
$$H(u) = \frac{1}{2}((2H(u)N_u) \bullet N_u) = \frac{\sum_{v \in \sigma(u)} \beta(u_v) \bullet N_u}{2Area(u)}$$

$$\beta(u_v) \bullet N_u = \frac{(\cot\theta_1 + \cot\theta_2)}{2}(u - v) \bullet N_u$$

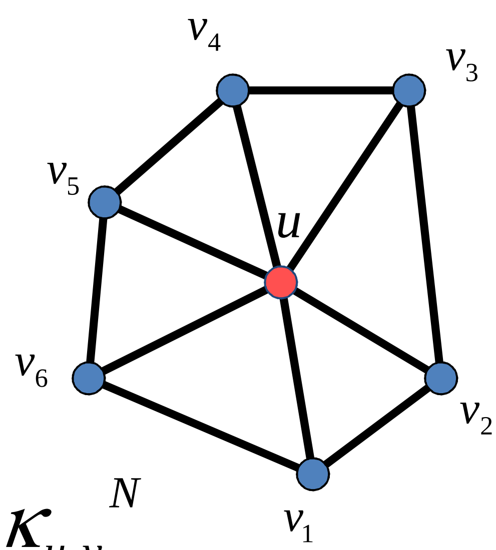
$$= \frac{(\cot\theta_1 + \cot\theta_2)}{2} \frac{\|u - v\|^2}{\|u - v\|^2}(u - v) \bullet N_u$$

$$= \frac{(\cot\theta_1 + \cot\theta_2)\|u - v\|^2}{4} \frac{2(u - v) \bullet N_u}{\|u - v\|^2}$$

$$= \frac{(\cot\theta_1 + \cot\theta_2)\|u - v\|^2}{4} \kappa_{u,v}{}^N$$



Discrete Curvature Tensor



- We know that:

$$\kappa(a,b) = la^2 + 2mab + nb^2$$

- Need to compute *l*, *m*, and *n*.
- For each edge (*u*, *v*), we also know $\kappa_{u,v}$
- Now we have a set of equations based on each edge:

$$\kappa_i = \kappa_{(a_i,b_i)} = la_i^2 + 2ma_ib_i + nb_i^2$$

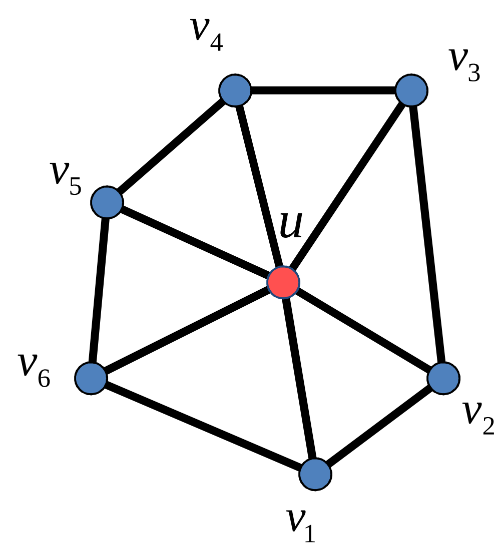
Discrete Curvature Tensor

- Solve the following system of linear equations:

$$\kappa_i = \kappa_{(a_i, b_i)} = l a_i^{\,2} + 2 m a_i b_i + n b_i^{\,2}$$

- or equivalently

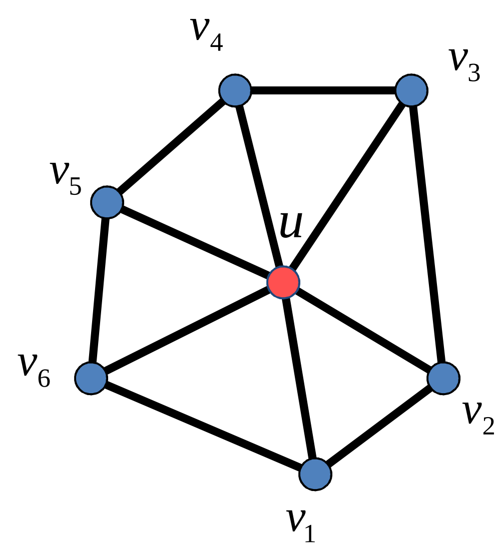
$$\begin{pmatrix} a_1^{\,2} & 2a_1b_1 & b_1^{\,2} \\ a_2^{\,2} & 2a_1b_1 & b_2^{\,2} \\ \vdots & \vdots & \vdots \\ a_n^{\,2} & 2a_nb_n & b_n^{\,2} \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$



Discrete Curvature Tensor

- Solve the following system of linear equations:

$$\kappa_i = \kappa_{(a_i, b_i)} = l a_i^{2} + 2m a_i b_i + n b_i^{2}$$



- or equivalently

$$\begin{pmatrix} a_1^{2} & 2a_1 b_1 & b_1^{2} \\ a_2^{2} & 2a_1 b_1 & b_2^{2} \\ \vdots & \vdots & \vdots \\ a_n^{2} & 2a_n b_n & b_n^{2} \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$

- The a’s and b’s are the 2D coordinates of tangent vectors.

Discrete Curvature Tensor

- How to solve it efficiently?

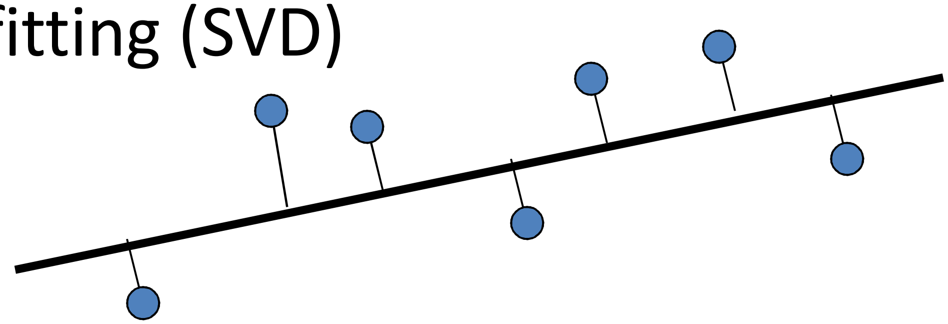
$$\begin{pmatrix} {a_1}^2 & 2a_1b_1 & {b_1}^2 \\ {a_2}^2 & 2a_1b_1 & {b_2}^2 \\ \vdots & \vdots & \vdots \\ {a_n}^2 & 2a_nb_n & {b_n}^2 \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$

Discrete Curvature Tensor

- How to solve it efficiently?

$$\begin{pmatrix} a_1^{\ 2} & 2a_1b_1 & b_1^{\ 2} \\ a_2^{\ 2} & 2a_1b_1 & b_2^{\ 2} \\ \vdots & \vdots & \vdots \\ a_n^{\ 2} & 2a_nb_n & b_n^{\ 2} \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$

- Least-square fitting (SVD)



Discrete Curvature Tensor

- How to solve it efficiently?

$$\begin{pmatrix} a_1^{\ 2} & 2a_1b_1 & b_1^{\ 2} \\ a_2^{\ 2} & 2a_1b_1 & b_2^{\ 2} \\ \vdots & \vdots & \vdots \\ a_n^{\ 2} & 2a_nb_n & b_n^{\ 2} \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$

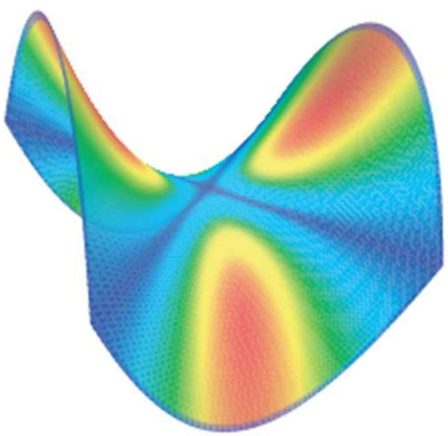
- Another way:

$$\begin{pmatrix} a_1^{\ 2} & 2a_1b_1 & b_1^{\ 2} \\ a_2^{\ 2} & 2a_1b_1 & b_2^{\ 2} \\ \vdots & \vdots & \vdots \\ a_n^{\ 2} & 2a_nb_n & b_n^{\ 2} \end{pmatrix}^T \begin{pmatrix} a_1^{\ 2} & 2a_1b_1 & b_1^{\ 2} \\ a_2^{\ 2} & 2a_1b_1 & b_2^{\ 2} \\ \vdots & \vdots & \vdots \\ a_n^{\ 2} & 2a_nb_n & b_n^{\ 2} \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} a_1^{\ 2} & 2a_1b_1 & b_1^{\ 2} \\ a_2^{\ 2} & 2a_1b_1 & b_2^{\ 2} \\ \vdots & \vdots & \vdots \\ a_n^{\ 2} & 2a_nb_n & b_n^{\ 2} \end{pmatrix}^T \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix}$$

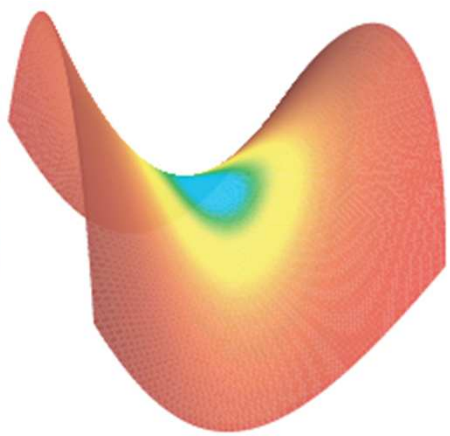
What Next?

- Put l,m,n into a 2x2 matrix and solve for eigenvalues (principle curvatures) and eigenvectors (principle directions)

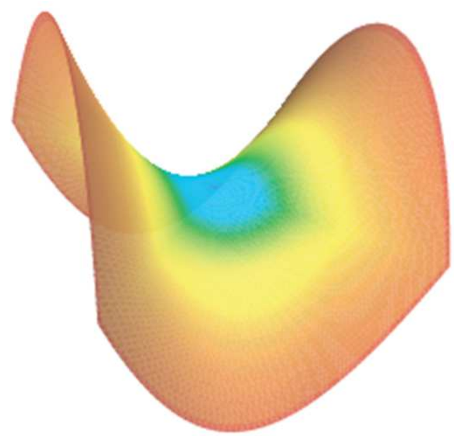
Examples



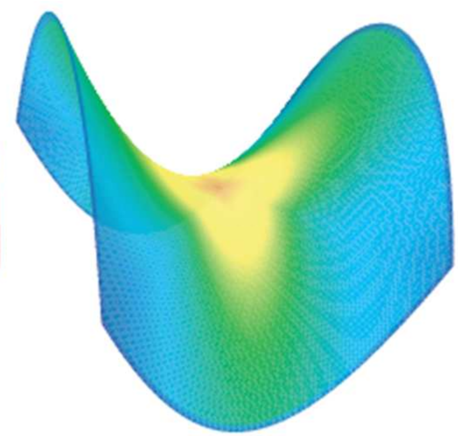
mean



Gaussian



minimum



maximum

Image credit: Mark Meyer et al.

Examples

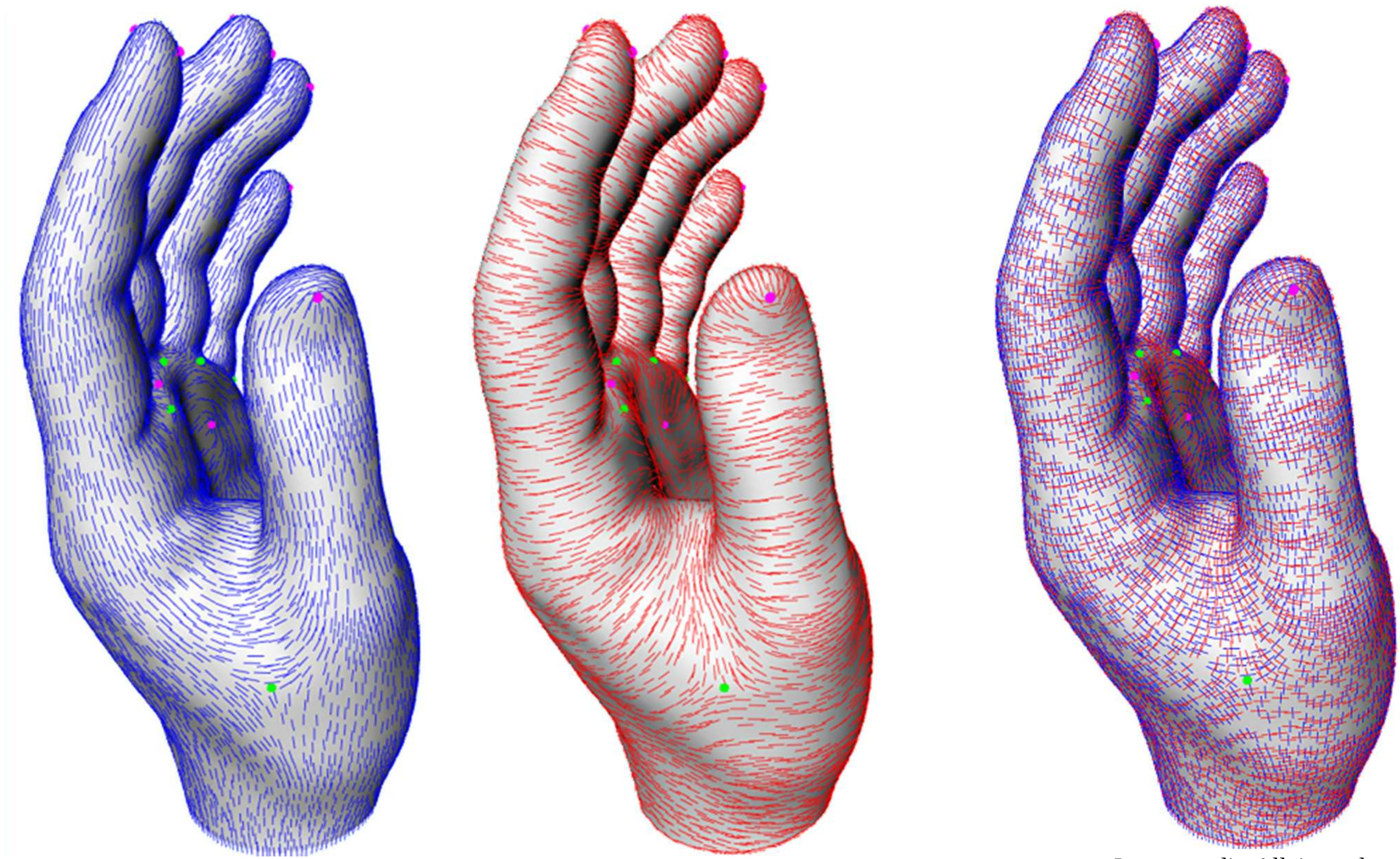
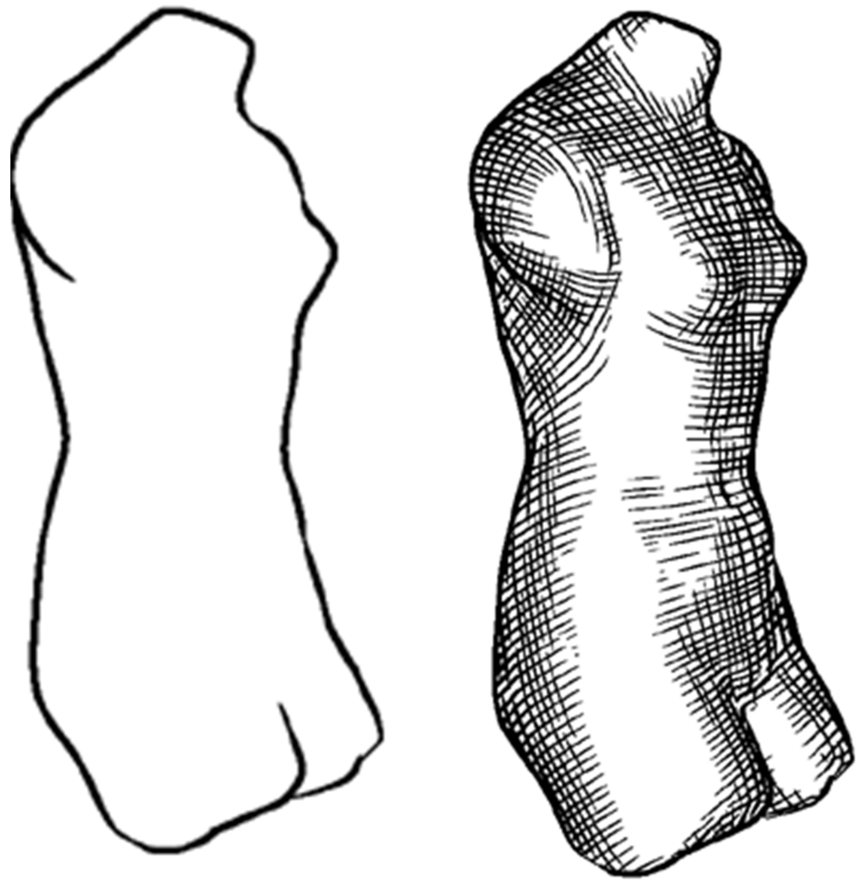


Image credit: Alleiz et al.

Hatch Drawing

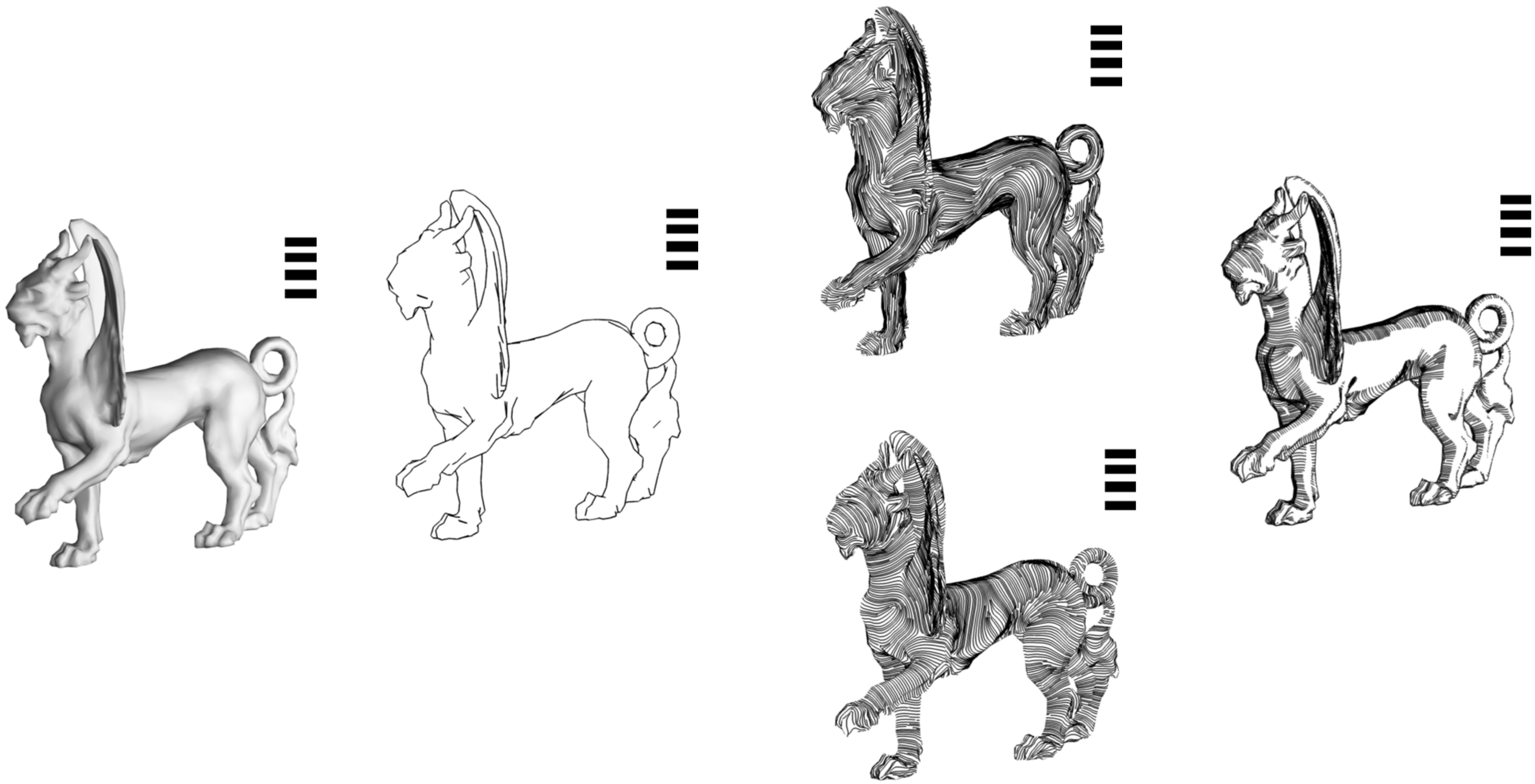
Praun et al.

A Comparison

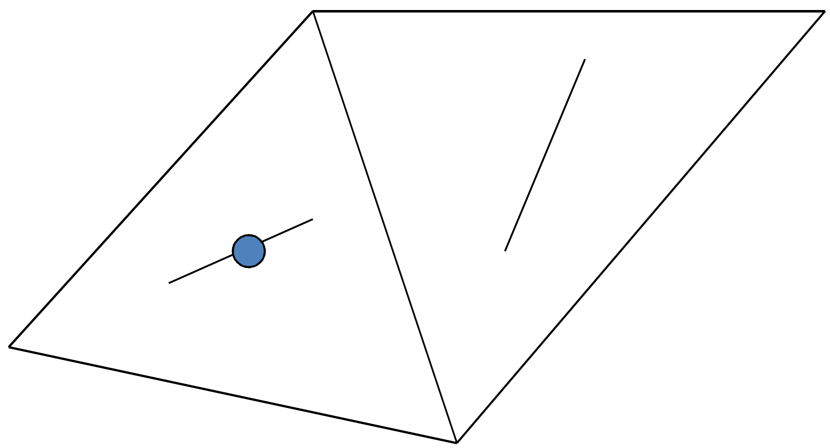


Hertzmann and Zorin

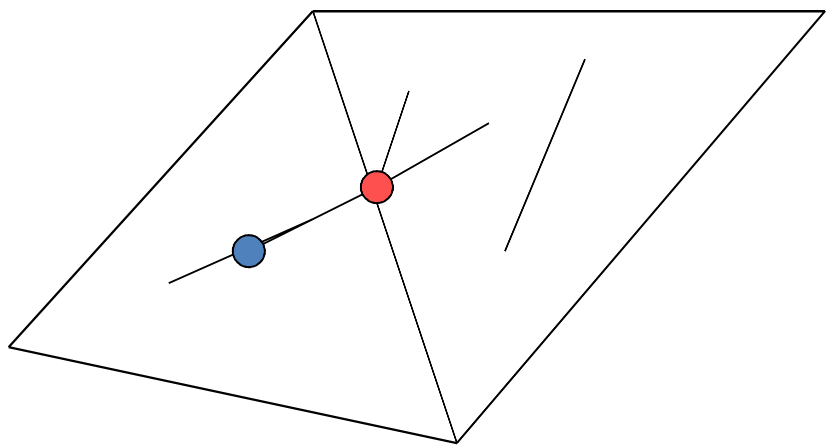
Rendering



Tracing Streamlines

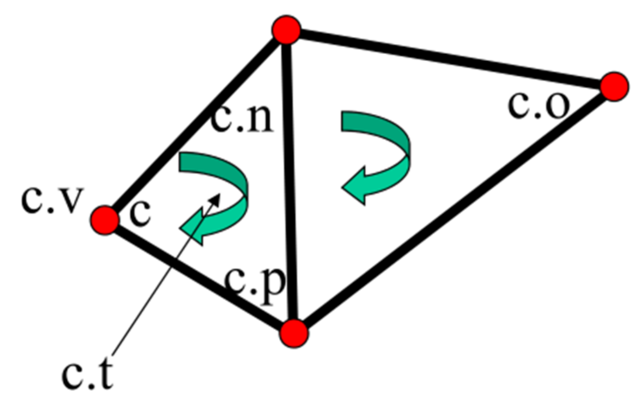


Tracing Streamlines



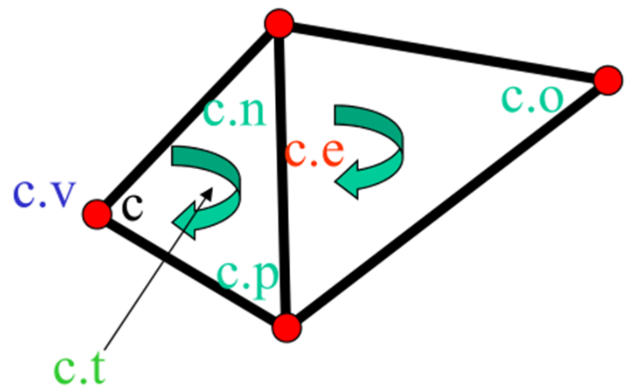
Corner Tables

- What is a corner?
- Why is it useful?
  - Angles
  - Other corner-related properties



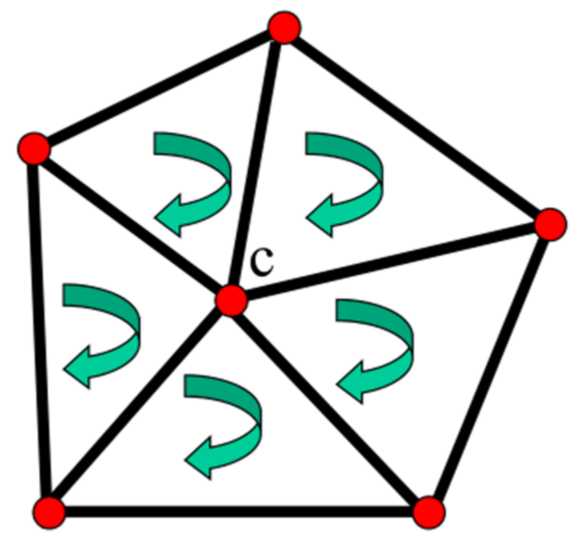
Corner Tables

- Operations
  - .p, .n, .o
  - .v
  - .e
  - .t
  - Can cascade:
  - What is c.o.t, c.o.p, c.o.n?
  - c.o.p=c.p.o?
  - c.o=NULL?



Corner Tables

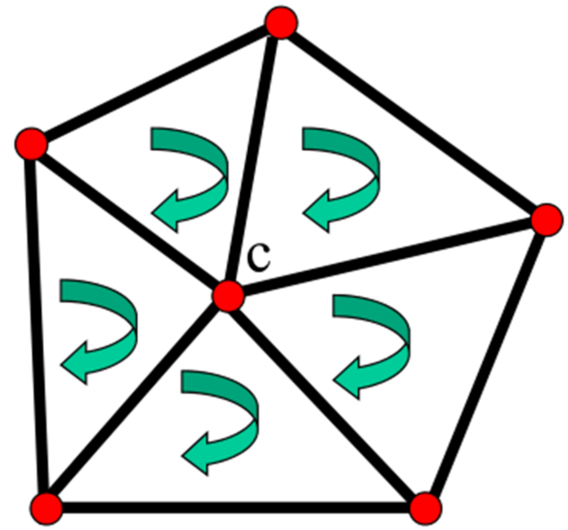
- Go around a vertex
- Where is c.p.o.n?
- Where is c.p.o.p?
- Where is (c.p.o.p).p.o.p?



Constructing Corner Tables

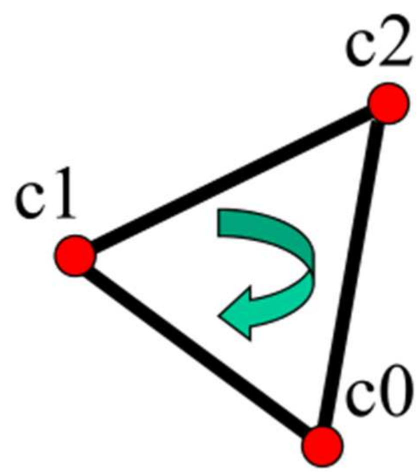
- Index
- .o, .p, .n, .v, .e, .t

```
class Corner{
public:
        unsigned char Edge_count;   //special variable for edges search 1/21/05
        int index;
        int v;              //the ID of the vertex of the corner
        int n;              //the next corner according to the orientation
        int p;              //the previous corner according to the orientation
        int t;              //the triangle the corner belongs to
        int ot;             //the index of its opposite triangle for traversal
        int o;              //the index of its opposite corner
        Edge *e;            //the opposite edge of the corner
        float angle;         //the angle of the corner
        float BeginAng, EndAng;  //for correct angle allocation of the corner around vertex v
        float r;
        bool orient;

        ///////////////////////////////////////////////////
        /* Optional variables */
        Edge *edge[2];      //two edges associated with this corner

        Corner()
        {
                e = NULL;
                edge[0] = edge[1] = NULL;
        }

        double get_Angle();
};
```



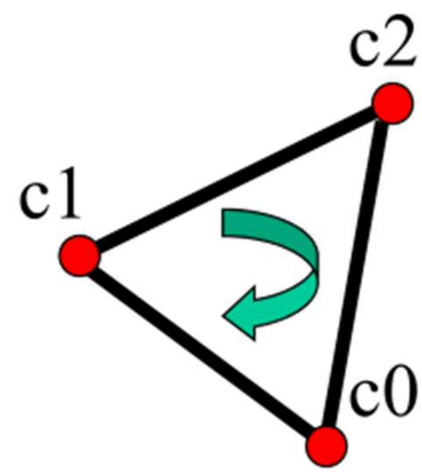
Constructing Corner Tables

- Read in all vertices and triangles
- Set num_corners = 3*num_tris
- For i=0 to num_tris
  - T=tlist[i]
  - T has three corners
    - c0=clist[3*i]
    - c1=clist[3*i+1]
    - c2=clist[3*i+2]
  - Such that ci.v=T.verts[i]



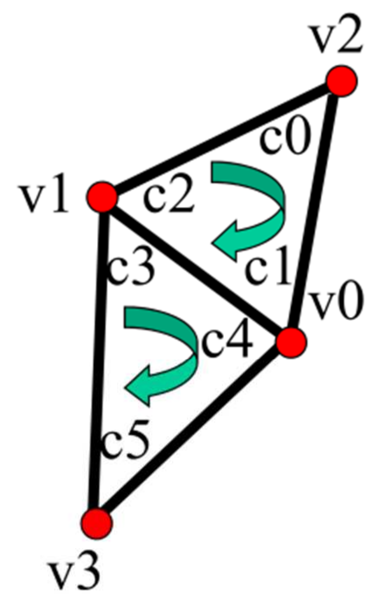
Constructing Corner Tables

- Construct the following table:
  - corner index, min(c.p.v.index, c.n.v.index), max(c.p.v.index, c.n.v.index)



Constructing This Table

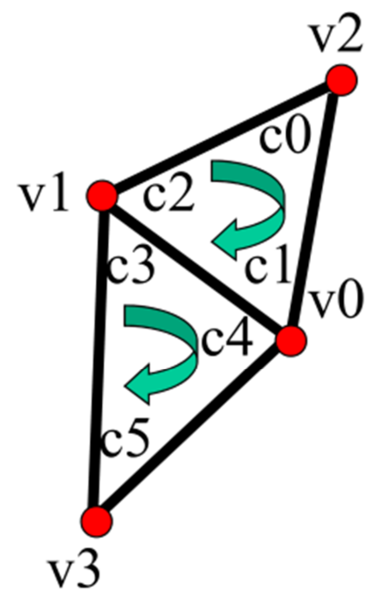
c.index	Min(c.p.v.index, c.n.v.index)	Max(c.p.v.index, c.n.v.index)	c.o.index
0	0	1	
1	1	2	
2	0	2	
3	0	3	
4	1	3	
5	0	1	



t0=(v2, v0, v1)
t1=(v1, v0, v3)

Sort According to Min/Max

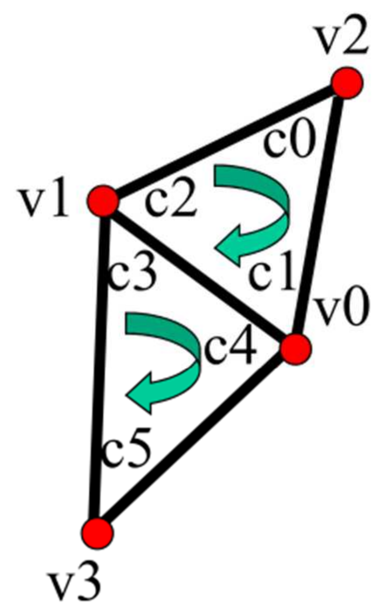
c.index	Min(c.p.v.index, c.n.v.index)	Max(c.p.v.index, c.n.v.index)	c.o.index
0	0	1	
5	0	1	
2	0	2	
3	0	3	
1	1	2	
4	1	3	



t0=(v2, v0, v1)
t1=(v1, v0, v3)

Look for Pairs and Set Up Links

c.index	Min(c.p.v.index, c.n.v.index)	Max(c.p.v.index, c.n.v.index)	c.o.index
0	0	1	5
5	0	1	0
2	0	2	NULL
3	0	3	NULL
1	1	2	NULL
4	1	3	NULL



t0=(v2, v0, v1)

t1=(v1, v0, v3)