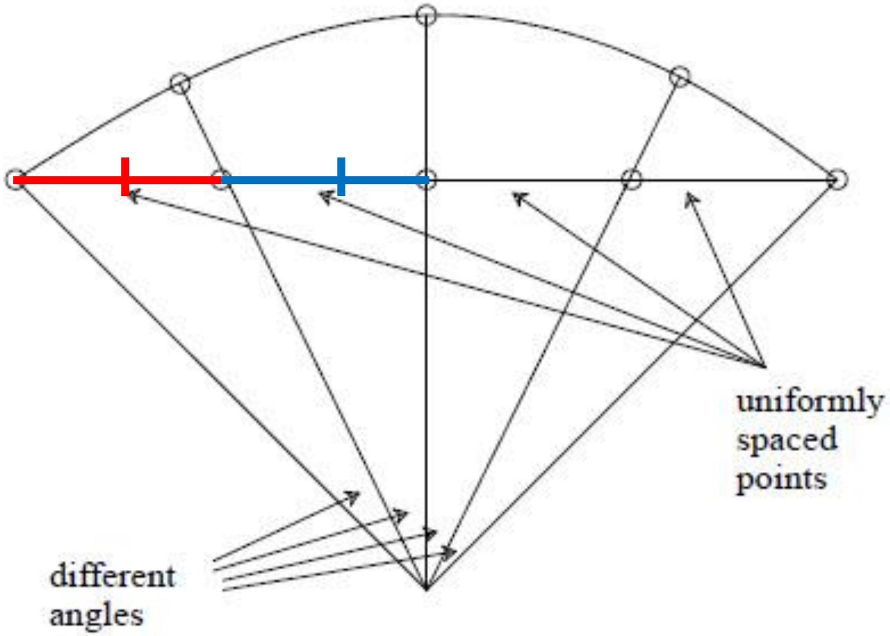
3.3 Interpolation of rotations represented by quaternions

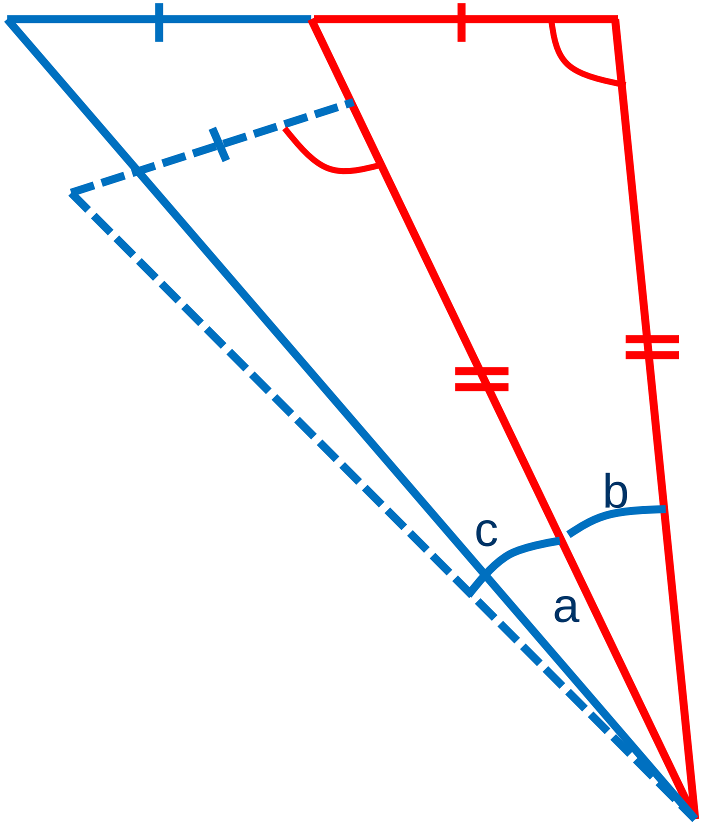
S^3 : set of unit quaternions

S^2 :set of unit 3D vectors

Interpolation will be performed on S^3



(direct linear interpolation produces nonlinear motion)



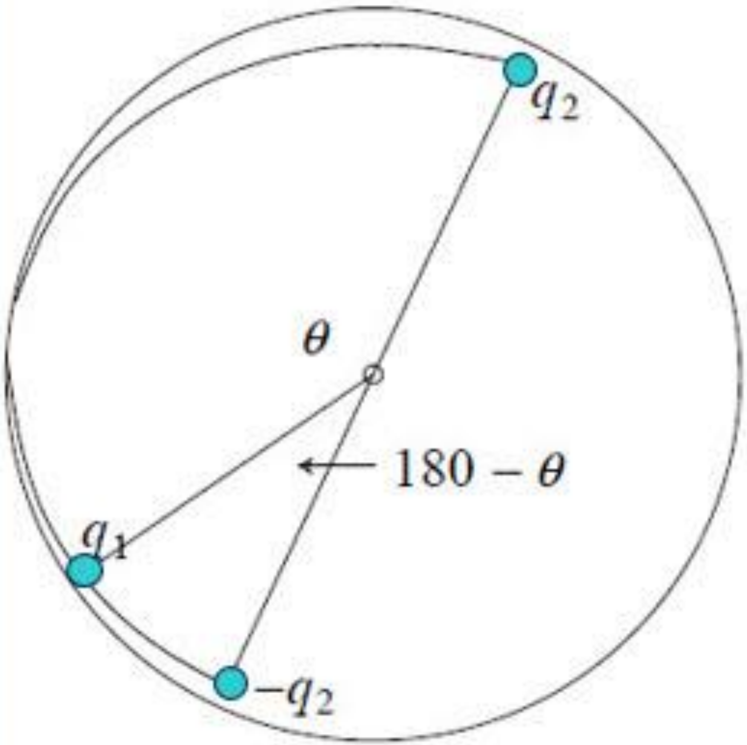
b = c

But c > a

So, b > a

Interpolation of rotations represented by quaternions

How? use **spherical linear interpolation (slerp)**



Since q_2 *and* $-q_2$ represent the same rotation, we use **great circular arc** with shorter path.

This can be determined by testing the inner product:

$$\cos\theta = q_1 \cdot q_2 = s_1 s_2 + v_1 \cdot v_2$$

Slerp is desirable because …

- slerp produces the shortest path between the two orientations on that unit sphere in 4D; (this is equivalent to finding the "minimum torque" rotation in 3D space, which you can think of as the smoothest transition between two orientations)

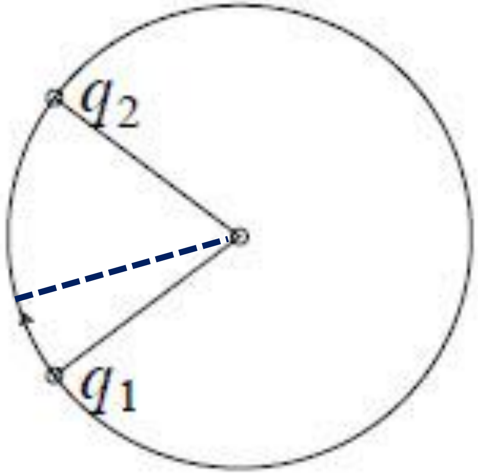
- it travels this path at a constant speed, which basically means you have full control over the nature of the transition

Interpolation of rotations represented by quaternions

Two approaches:

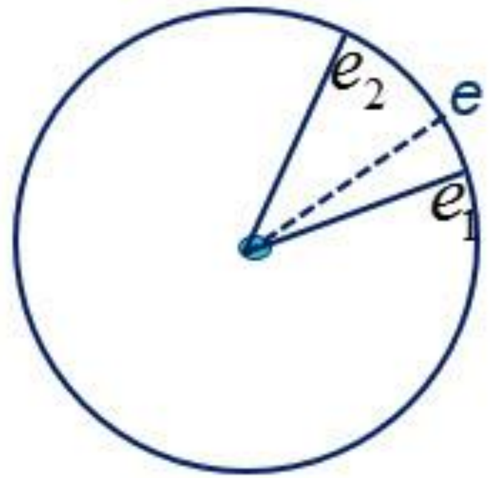
(1) View S^3 *as a group*

$$\boxed{slerp(q_1, q_2; u) = q_1\left(q_1^{-1} q_2\right)^u, \quad 0 \le u \le 1}$$



This is the extension of unit complex number interpolation

$$\boxed{e = e_1\left(e_1^{-1} e_2\right)^u = \mathrm{e}^{\mathrm{i}(\theta_1 + u(\theta_2 - \theta_1))}}$$



How to define q^{α}

Given a vector $v = \theta \hat{v} \in R^3$ with $\hat{v} \in S^2$, the exponential can be defined as

$$\boxed{\exp(v) = \sum_{i=0}^{\infty} \frac{v^i}{i!} = (\cos\theta, \hat{v} \sin\theta) \in S^3}$$

$$\boxed{-\pi < \theta < \pi}$$

exp is one-to-one when $|\theta| < \pi$. Hence, can define *log* as the inverse of *exp*. Consequently, can define

$$\boxed{q^{\alpha} = \exp(\alpha \log q)}$$

How to compute q^{α}

$$\text{If} \quad q = (w, x, y, z) \in S^3,$$

$$\text{we have } w^2 + x^2 + y^2 + z^2 = 1.$$

$$\text{Define } \cos\theta = w, \ \sin\theta = (x^2 + y^2 + z^2)^{1/2}$$

$$\text{and } \hat{v} = \left(x/\sin\theta, y/\sin\theta, z/\sin\theta\right)$$

$$\text{then since } \exp(\theta\hat{v}) = \underline{q = (\cos\theta, \hat{v}\sin\theta)}$$

$$\text{so } \log(q) = \theta\hat{v}.$$

$$\text{Hence, } \boxed{\begin{aligned} q^{\alpha} &= \exp(\alpha \log q) = \exp(\alpha \cdot \theta\hat{v}) \\ &= \underline{(\cos(\alpha\theta), \hat{v}\sin(\alpha\theta))} \end{aligned}}$$

Interpolation of rotations represented by quaternions

(ii) From 4D geometry

$$\boxed{slerp(q_0, q_1, ; t) = \frac{\sin((1-t)\theta)}{\sin\theta} q_0 + \frac{\sin(t\theta)}{\sin\theta} q_1 \qquad (*)}$$

where $q_0 \cdot q_1 = \cos\theta$

Show that *slerp(q_0, q_1; t)* is a *unit quaternion*

Before we show that *slerp*(q_0, q_1; t) is a unit quaternion, let's see the geometric meaning of this definition first.

First, the formula is symmetric. The symmetry can be seen in the fact that Slerp(q_0, q_1; t) = Slerp(q_1, q_0; $1 - t$).

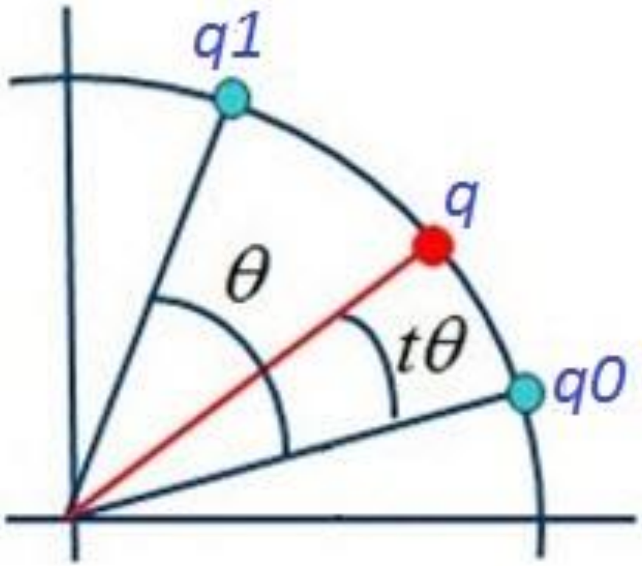
In the limit as $\theta \to 0$, this formula reduces to the corresponding symmetric formula for linear interpolation,

$$slerp(q_0, q_1, ; t) = (1 - t)q_0 + tq_1$$

A *Slerp* path is, in fact, the spherical geometry equivalent of a path along a line segment in the plane; a great circle is a spherical geodesic.

Next, we show the $\boxed{\text{correctness (or, derivation) of (*).}}$

The interpolation between q_0 and q_1 in 4D space can be written as



$$q(t) = c_0(t)q_0 + c_1(t)q_1$$

where $c_0(t)$ and $c_1(t)$ are real-valued functions for 0<= t <= 1 with $c_0(0) = 1,\ c_1(0) = 0,\ c_0(1) = 0,$ and $c_1(1) = 1.$

As t uniformly varies between 0 and 1, the values $q(t)$ are required to uniformly vary along the circular arc from q_0 to q_1. That is, the angle between $q(t)$ and q_0 is

$t\theta$ and the angle between *q(t)* and q_1 is $(1-t)\theta$.

Taking the inner product of *q(t)* with q_0, we get

$$\cos(t\theta) = c_0(t) + \cos(\theta)c_1(t) \qquad (1)$$

Taking the inner product of *q(t)* with q_1, we get

$$\cos((1-t)\theta) = \cos(\theta)c_0(t) + c_1(t) \qquad (2)$$

Solving (1) and (2), we get

$$c_0(t) = \frac{\sin((1-t)\theta)}{\sin\theta}$$

$$c_1(t) = \frac{\sin(t\theta)}{\sin\theta}$$

Thus, the correctness of (*) has been proven.

Interpolation of rotations represented by quaternions

Show that *slerp(q_1, q_2; u)* is indeed a unit quaternion

$$
\begin{aligned}
&Slerp(q_1, q_2; u) \\
&= [\frac{\sin((1-u)\theta)}{\sin\theta} w_1 + \frac{\sin(u\theta)}{\sin\theta} w_2, \\
&\frac{\sin((1-u)\theta)}{\sin\theta} x_1 + \frac{\sin(u\theta)}{\sin\theta} x_2, \\
&\frac{\sin((1-u)\theta)}{\sin\theta} y_1 + \frac{\sin(u\theta)}{\sin\theta} y_2, \\
&\frac{\sin((1-u)\theta)}{\sin\theta} z_1 + \frac{\sin(u\theta)}{\sin\theta} z_2\,] = [w, x, y, z]
\end{aligned}
$$

Interpolation of rotations represented by quaternions

Since q_1 *and* q_2 are unit quaternions, we have

$$|Slerp(q_1, q_2; u)|^2 = w^2 + x^2 + y^2 + z^2$$

$$= \frac{\sin^2((1-u)\theta)}{\sin^2\theta}\underbrace{\left(w_1^2 + x_1^2 + y_1^2 + z_1^2\right)}_{=1}$$

$$+ \frac{2\sin((1-u)\theta)\sin(u\theta)}{\sin^2\theta}\underbrace{\left(w_1w_2 + x_1x_2 + y_1y_2 + z_1z_2\right)}_{=\cos\theta}$$

$$+ \frac{\sin^2(u\theta)}{\sin^2\theta}\underbrace{\left(w_2^2 + x_2^2 + y_2^2 + z_2^2\right)}_{=1}$$

Interpolation of rotations represented by quaternions

So,

$$|\,Slerp(q_1,q_2;u)\,|^2 = \frac{\sin^2((1-u)\theta)}{\sin^2\theta} + \frac{\sin^2(u\theta)}{\sin^2\theta} + \frac{2\cos\theta\sin(u\theta)\sin((1-u)\theta)}{\sin^2\theta}$$

$$= \frac{\sin^2\theta\cos^2(u\theta) - \cos^2\theta\sin^2(u\theta) + \sin^2(u\theta)}{\sin^2\theta}$$

$$= \frac{\sin^2\theta}{\sin^2\theta} = 1$$

$$\boxed{\begin{aligned} &= \sin^2(u\theta)\left(1-\cos^2\theta\right) \\ &= \sin^2(u\theta)\sin^2\theta \end{aligned}}$$

Interpolation of rotations represented by quaternions

Show that

$$Slerp(q_1, q_2; 1/2) = \frac{q_1 + q_2}{|q_1 + q_2|} = \frac{q_1 + q_2}{2\cos(\theta/2)}$$

Question: Does the 2nd approach generate the same curve as the 1st approach?

Interpolation of rotations represented by quaternions

Show that

$$\boxed{Slerp(q_1, q_2; 1/2) = \frac{q_1 + q_2}{|q_1 + q_2|} = \frac{q_1 + q_2}{2\cos(\theta/2)}}$$

Proof:

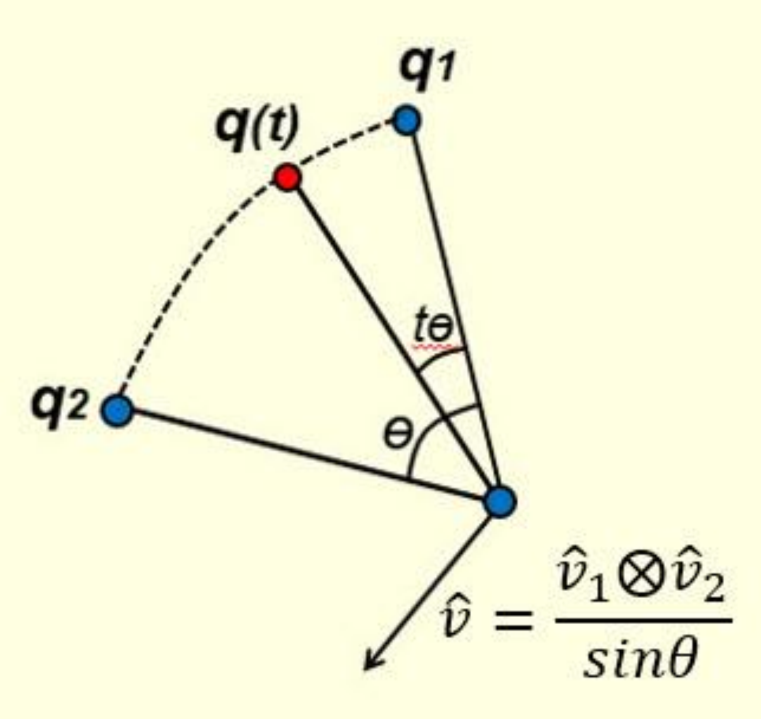
$$|\,Slerp(q_1, q_2; 1/2)\,| = \left| \frac{q_1 + q_2}{2\cos(\theta/2)} \right| = 1$$

$$Hence, |q_1 + q_2| = 2\cos(\theta/2)$$

Interpolation of rotations represented by quaternions

(III) Third approach

$$q_1 = (\cos\theta_1, \sin\theta_1\,\hat{v}_1)$$
$$q_2 = (\cos\theta_2, \sin\theta_2\,\hat{v}_2)$$



Define $\hat{v} = \dfrac{\hat{v}_1 \otimes \hat{v}_2}{\sin\theta}$

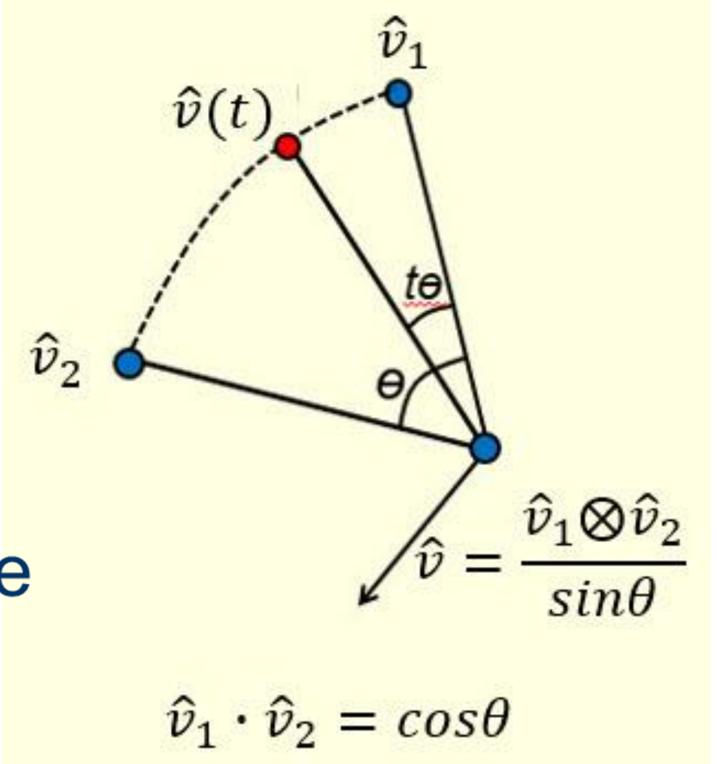
Define

$$q(t) = \left[\cos\left(\frac{t\theta}{2}\right), \sin\left(\frac{t\theta}{2}\right)\hat{v}\right] * [0, \sin\theta_1\hat{v}_1] * \left[\cos\left(\frac{t\theta}{2}\right), -\sin\left(\frac{t\theta}{2}\right)\hat{v}\right], \quad 0 \le t \le 1$$

Interpolation of rotations represented by quaternions

(III) Third approach

Actually instead of considering q_1 and q_2 in S^3 we can just consider $\hat{v}_1$ and $\hat{v}_2$ in S^2



Define $\hat{v} = \frac{\hat{v}_1 \otimes \hat{v}_2}{sin\theta}$ and then define

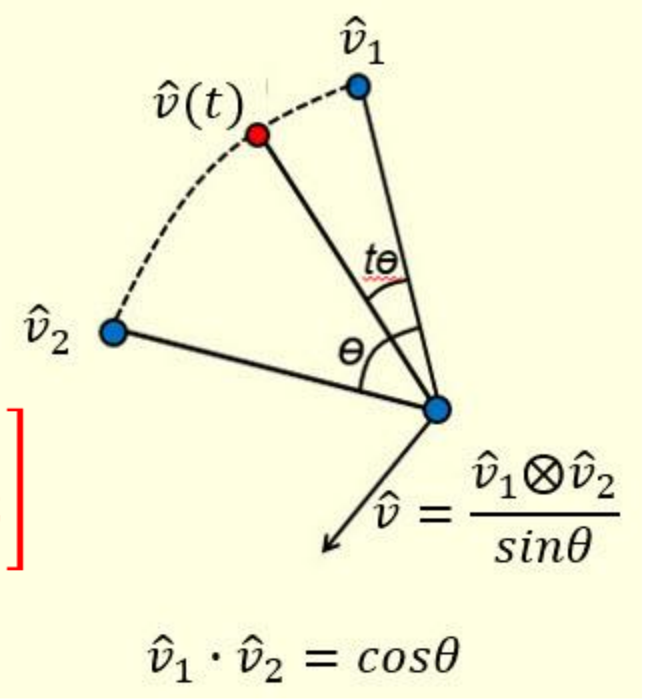
$$\hat{v}(t) = [cos\left(\frac{t\theta}{2}\right), sin\left(\frac{t\theta}{2}\right)\hat{v}] * [0, \hat{v}_1] * [cos\left(\frac{t\theta}{2}\right), -sin\left(\frac{t\theta}{2}\right)\hat{v}]$$

Interpolation of rotations represented by quaternions

(III) Third approach

It can be shown that

$$\hat{v}(t) = \left[0, \frac{\sin((1-t)\theta)}{sin\theta}\hat{v}_1 + \frac{\sin(t\theta)}{sin\theta}\hat{v}_2\right]$$

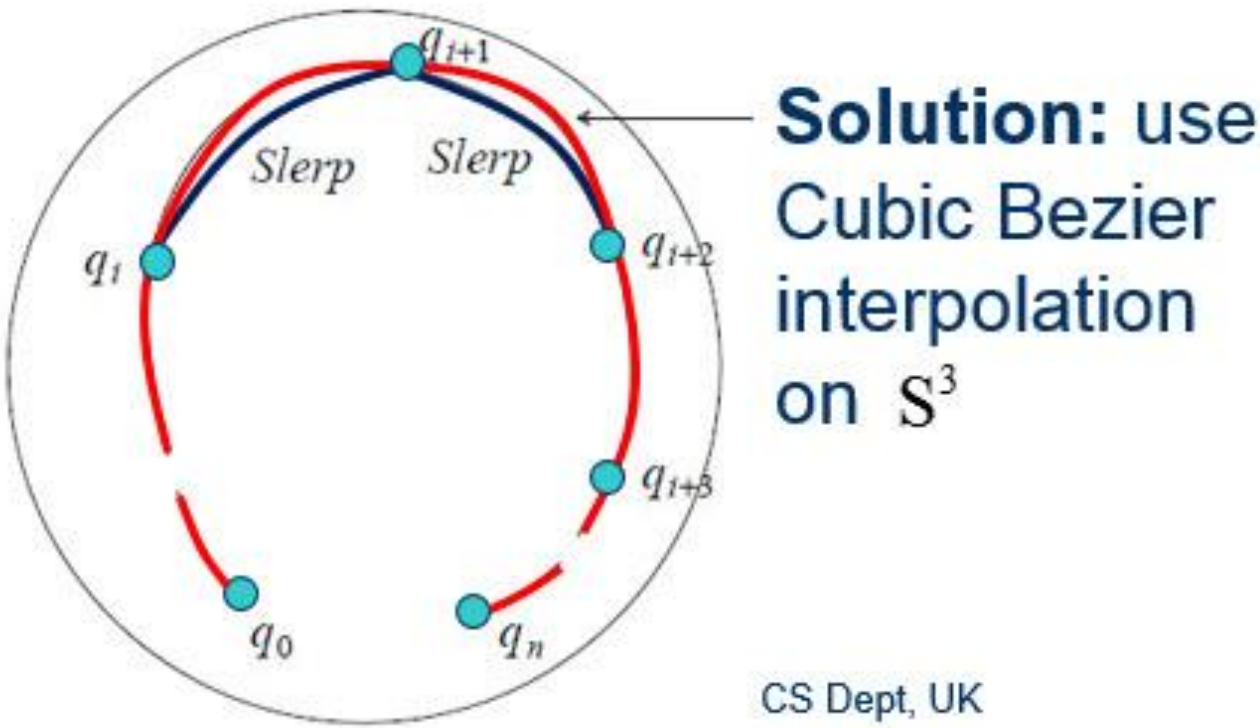


Now compare it with

$$Slerp(q_1, q_2; t) = \frac{\sin((1-t)\theta)}{sin\theta} q_1 + \frac{\sin(t\theta)}{sin\theta} q_2$$

How to interpolate between a series of orientations?

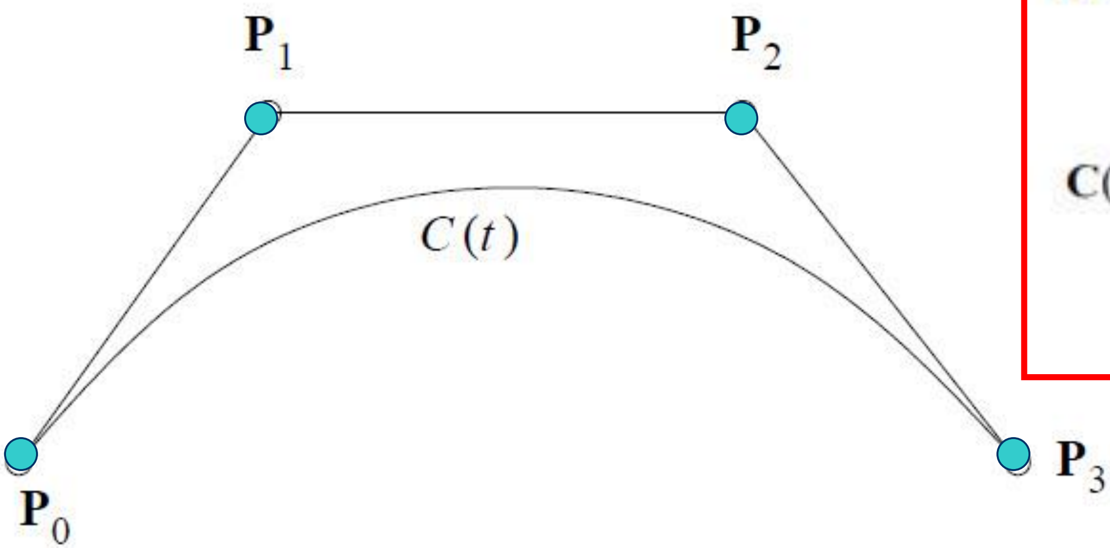
Problem with slerping between points: *first order discontinuity*



Review: Bezier Curve Segment of Degree 3

$$\mathbf{C}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3$$

$$0 \le t \le 1$$

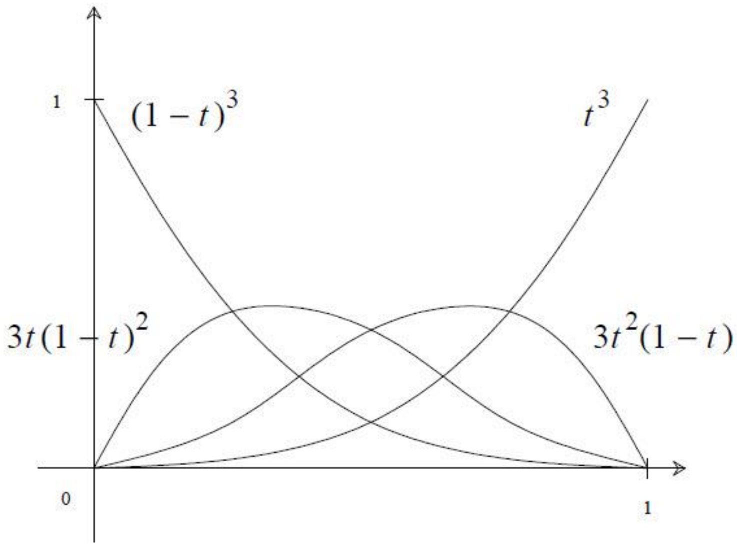


Matrix form:

$$\mathbf{C}(t) = [1, t, t^2, t^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

Review: Bezier Curve Segment of Degree 3

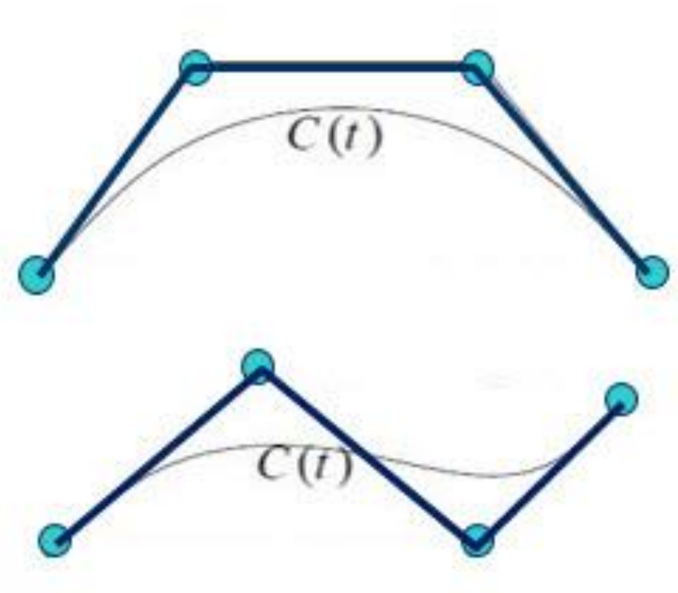
- $\mathbf{P}_i = (x_i, y_i)$ are called **control points**
- The polygon $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$ is called the **control polygon**
- The weights $(1-t)^3$, $3t(1-t)^2$, $3t^2(1-t)$, and t^3 are called **blending functions**



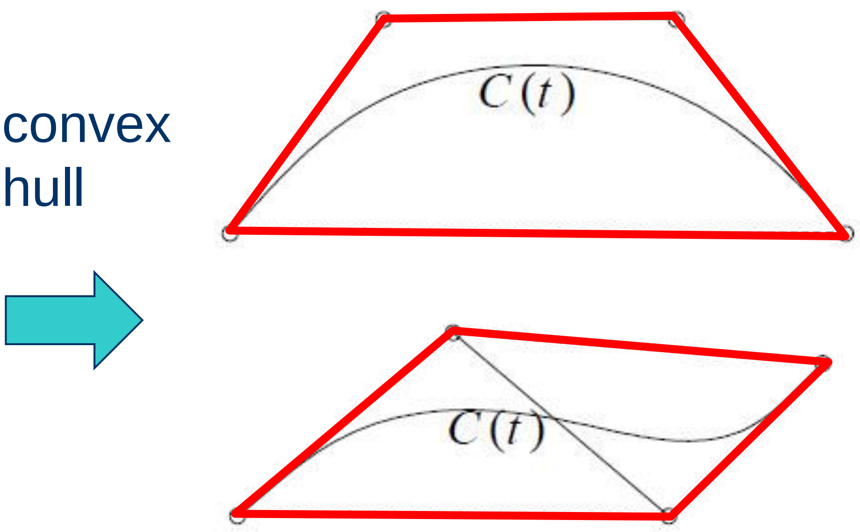
Notes:
- Blending functions are always non-negative
- Blending functions always sum to 1

Review: Bezier Curve Segment of Degree 3

- A Bezier curve always starts at **P0 and ends** at **P3**
- A Bezier curve is tangent to the control polygon at the endpoints
- Bezier curve segments satisfy **convex hull property**
- Bezier curves have intuitive appeal for interactive users



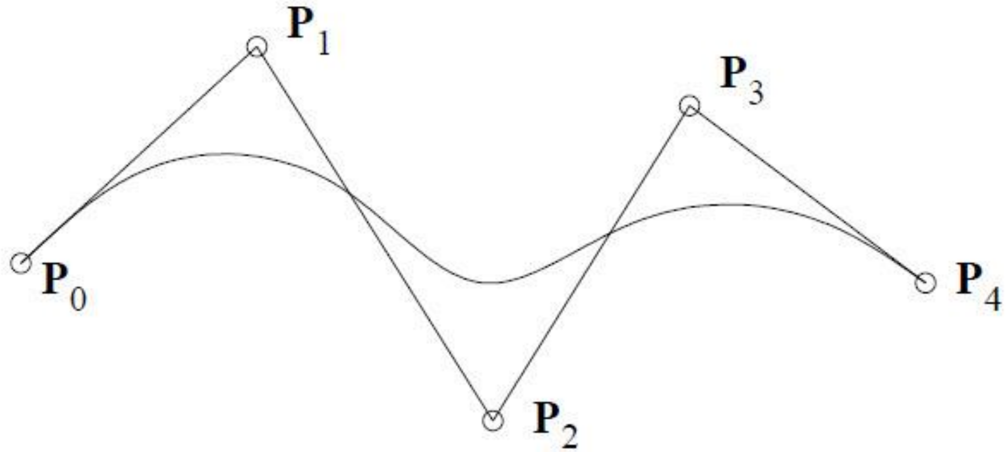
convex
hull



Review: General Bezier Curve Segments

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t)\,\mathbf{P}_i = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{P}_i,$$

where $0 \le t \le 1$ and $\binom{n}{i} \equiv \frac{n!}{i!\,(n-i)!}$. $B_{i,n}(t)$ are again called **blending functions** and $\mathbf{P}_i$ **control points.**



Review: General Bezier Curve Segments

- All the properties mentioned on page 5 hold for general

A recurrance relation:

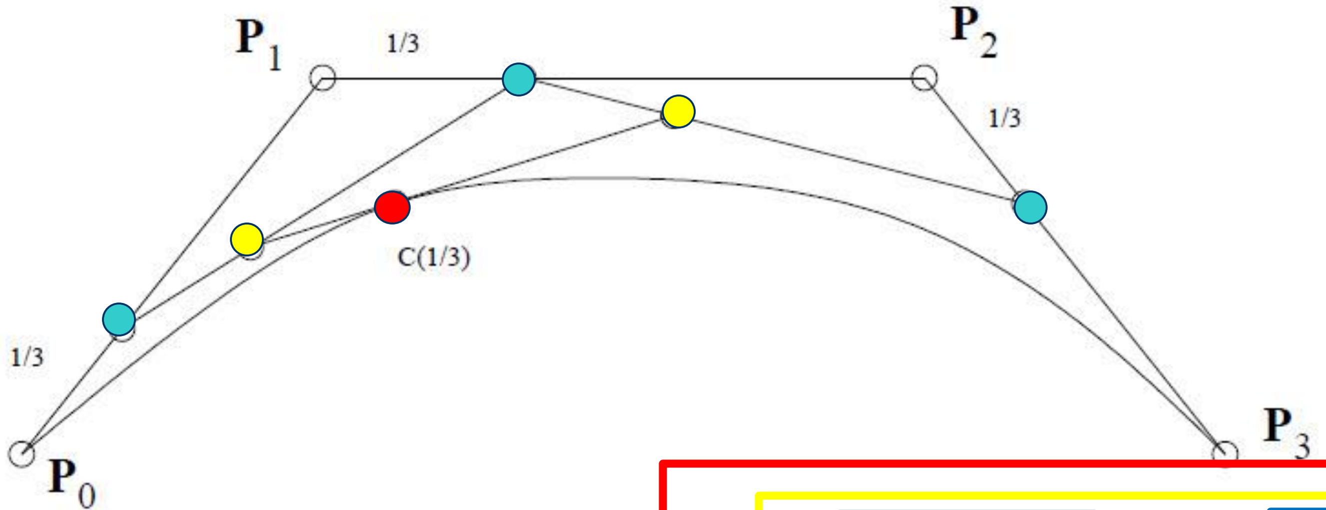
$$C(t) = \boxed{(1-t)} \left(\sum_{i=0}^{n-1} B_{i,n-1}(t)\, \mathbf{P}_i \right) + \boxed{t} \left(\sum_{i=0}^{n-1} B_{i,n-1}(t)\, \mathbf{P}_{i+1} \right)$$

$$= \boxed{(1-t)} \cdot \left[\sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i}\, \mathbf{P}_i \right]$$

$$+ \boxed{t} \left[\sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i}\, \mathbf{P}_{i+1} \right]$$

Review: General Bezier Curve Segments

- Curve computation

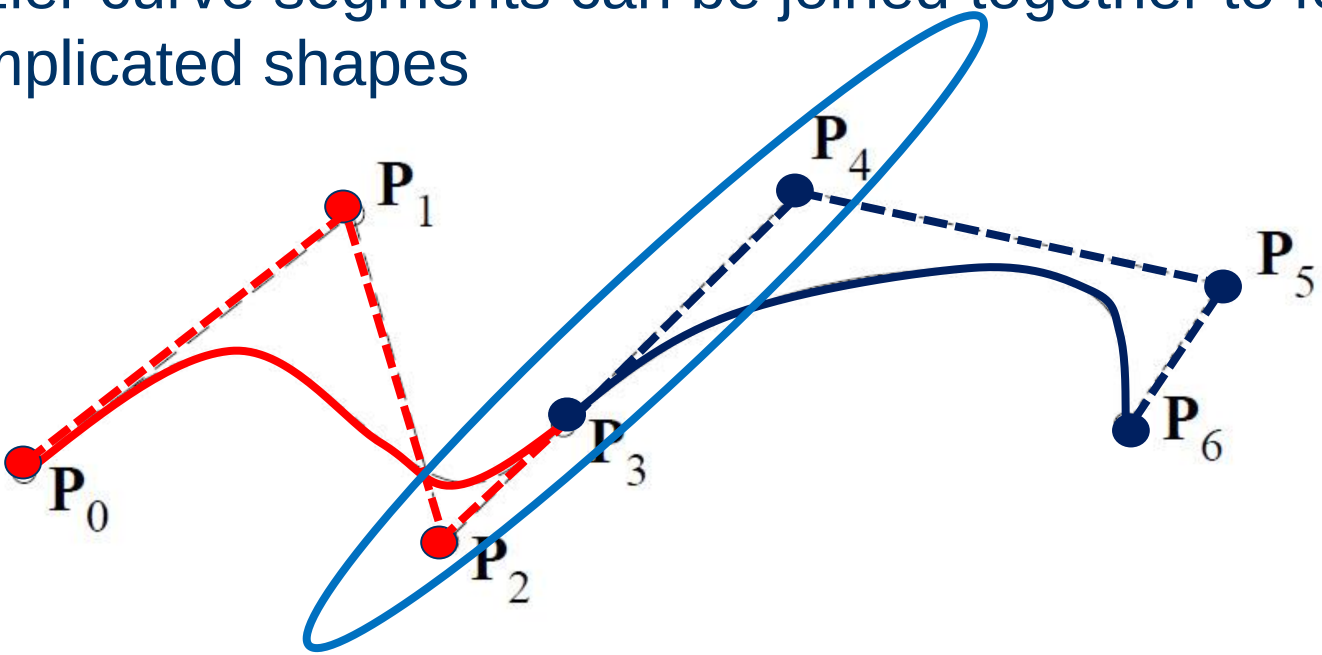


If degree = 3 then

$$C(\tfrac{1}{3}) = \frac{2}{3}\left[\frac{2}{3}\left(\boxed{\frac{2}{3}\mathbf{P}_0 + \frac{1}{3}\mathbf{P}_1}\right) + \frac{1}{3}\left(\boxed{\frac{2}{3}\mathbf{P}_1 + \frac{1}{3}\mathbf{P}_2}\right)\right] + \frac{1}{3}\left[\frac{2}{3}\left(\boxed{\frac{2}{3}\mathbf{P}_1 + \frac{1}{3}\mathbf{P}_2}\right) + \frac{1}{3}\left(\boxed{\frac{2}{3}\mathbf{P}_2 + \frac{1}{3}\mathbf{P}_3}\right)\right]$$

Review: Composite Bezier Curves

- Bezier curve segments can be joined together to form complicated shapes



P0, P1, P2, and **P3** are control points of the 1st segment
P3, P4, P5, and **P6** are control points of the 2nd segment
P2, P3, and **P4** are collinear (to guarantee smooth joint)

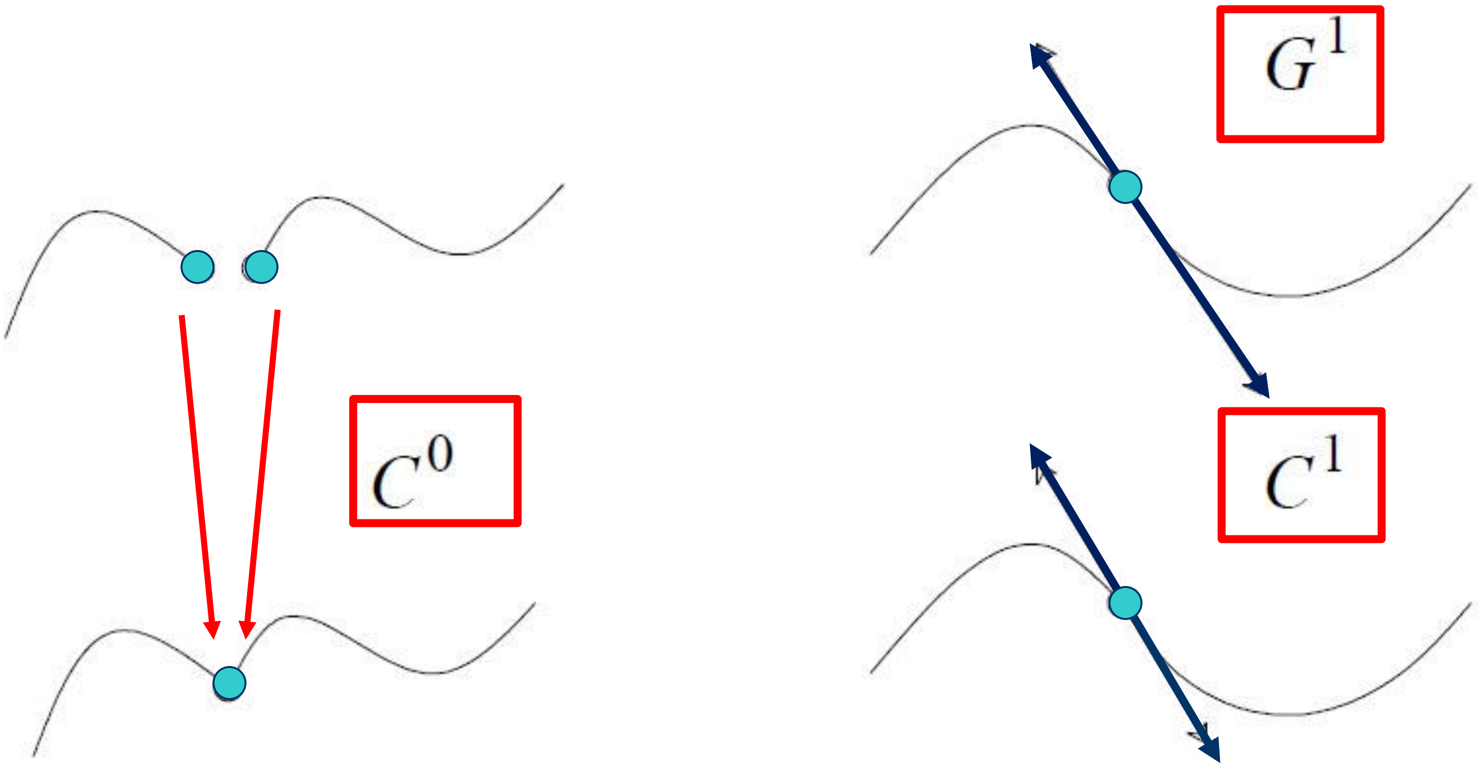
Review: Composite Bezier Curves

- Smoothness (continuity) at Join Points:

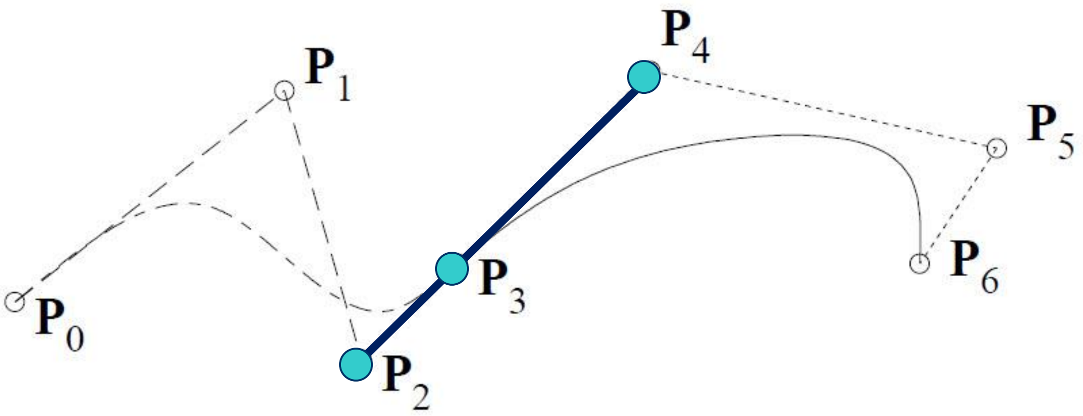
  C^0 : the endpoints coincide

  G^1 : tangents have the same slope

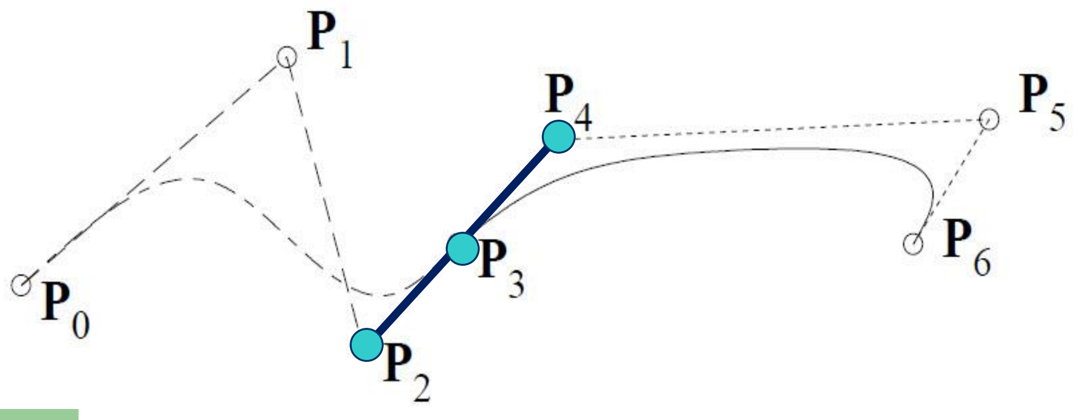
  C^1 : first derivatives on both segments match at join point



Review: Composite Bezier Curves



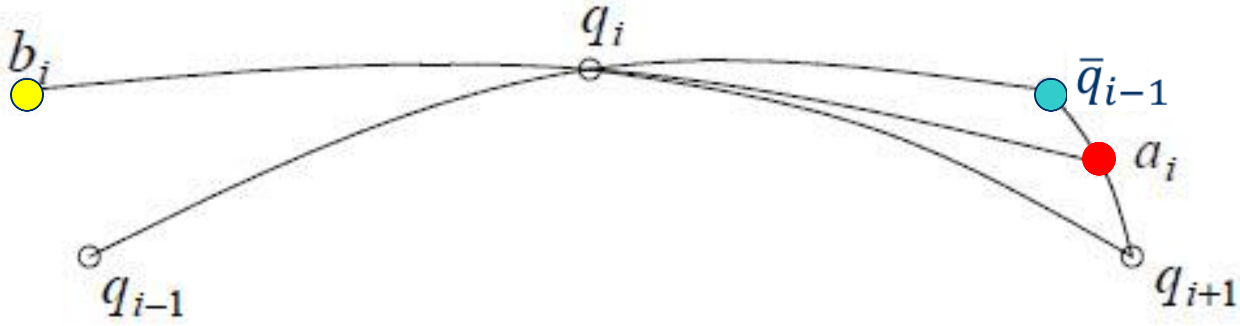
- G1-continuity: **P2, P3,** and **P4** are collinear



- C1-continuity: **P2, P3,** and **P4** are collinear and **P3** is the midpoint of **P2P4**

(i) Shoemake's approach

For each set of 3 consecutive key quaternions q_{i-1}, q_i and q_{i+1}, construct a_i and b_i as:



$$\boxed{a_i = Bisect(Double(q_{i-1}, q_i), q_{i+1})}$$

$$\boxed{b_i = Double(a_i, q_i)}$$

where
$$\begin{cases} Double(p, q) = 2(\, p \cdot q \,)\, q - p\,; \\ Bisect(p, q) = \dfrac{p+q}{|p+q|} \end{cases}$$

$$Double(p,q) = 2(\,p \cdot q\,)\,q - p\,;$$
$$Bisect(p,q) = \frac{p+q}{|p+q|}$$

What do they mean?

Let *D =Double(p,q) .* Then *q* is the midpoint of the Circular arc from *p* to *D, i.e.,*

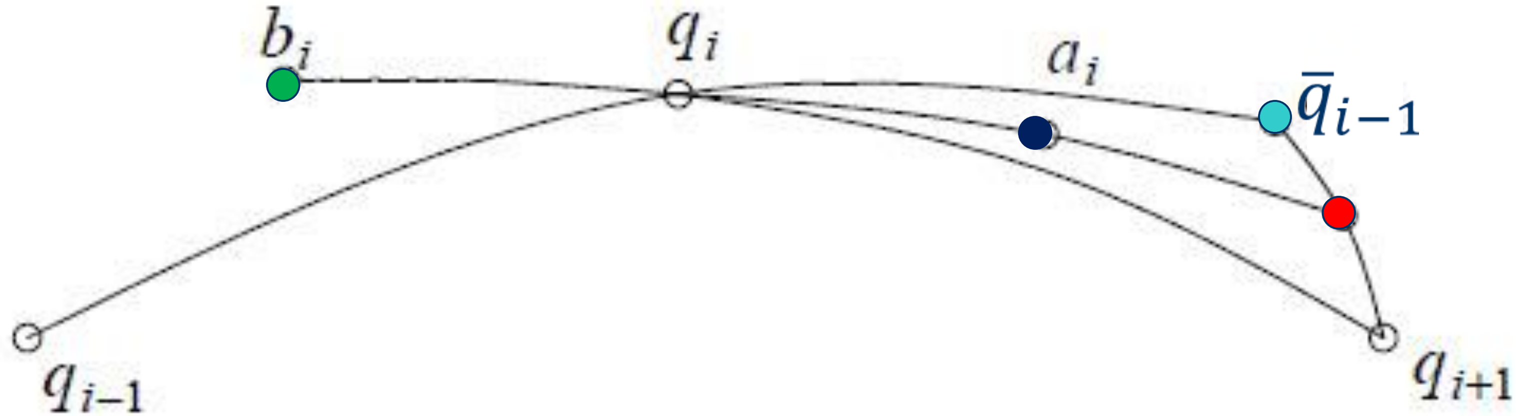
$$q = \frac{p+D}{|p+D|} = \frac{p+D}{2\cos\theta} = \frac{p+D}{2(p \cdot q)}$$

Hence, $D = 2(p \cdot q)q - p$

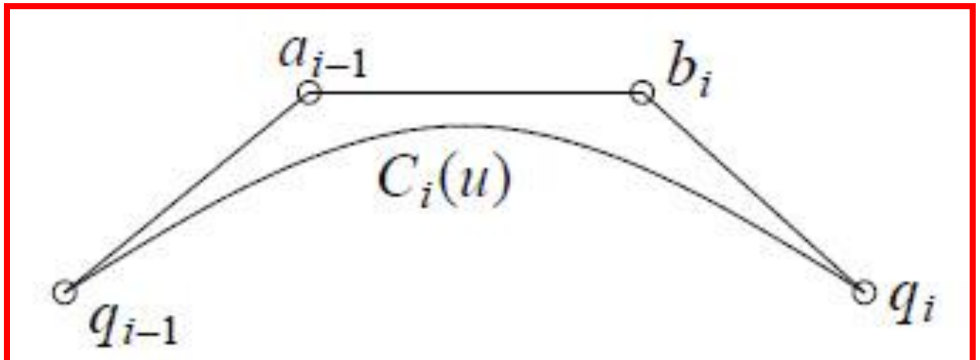
(i) Approach II

$$a_i = Bisect(q_i, Bisect(Double(q_{i-1}, q_i), q_{i+1}))$$

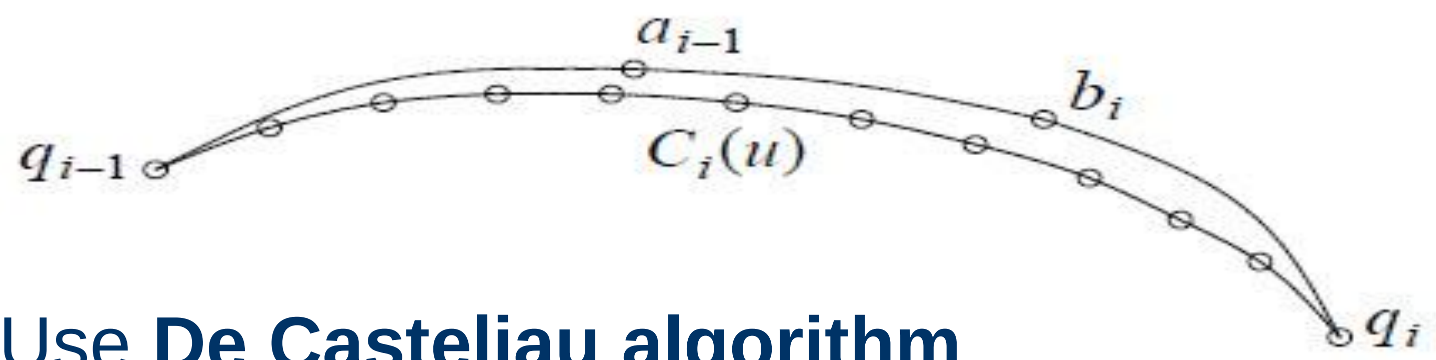
$$b_i = Double(a_i, q_i)$$



Then, use q_{i-1}, a_{i-1}, b_i and q_i as the control points of segment i



Generation of the segment $C_i(u)$



Use **De Casteljau algorithm**

e.g., $\boxed{C_i(1/3) = ?}$

$p_1^1 = slerp(q_{i-1}, a_{i-1}, 1/3)$

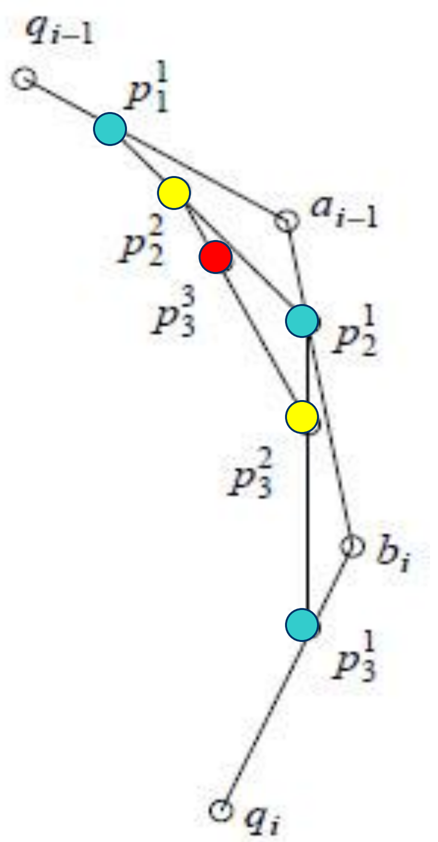
$p_2^1 = slerp(a_{i-1}, b_i, 1/3)$

$p_3^1 = slerp(b_i, q_i, 1/3)$

$p_2^2 = slerp(p_1^1, p_2^1, 1/3)$

$p_3^2 = slerp(p_2^1, p_3^1, 1/3)$

$p_3^3 = slerp(p_2^2, p_3^2, 1/3)$

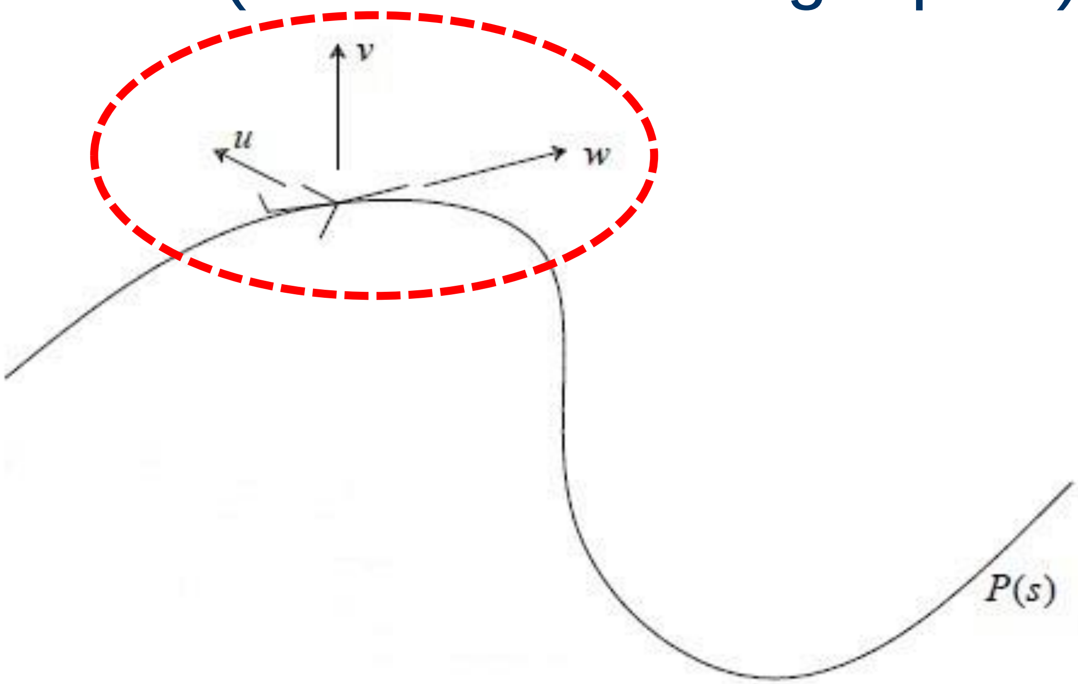


Path Following

Issues: *orientation handling, path smoothing, path along a surface*

Orientation Handling:

- define a local coordinate system (*u, v, w*) for the camera (as it travels along a path)



Frenet Frame:

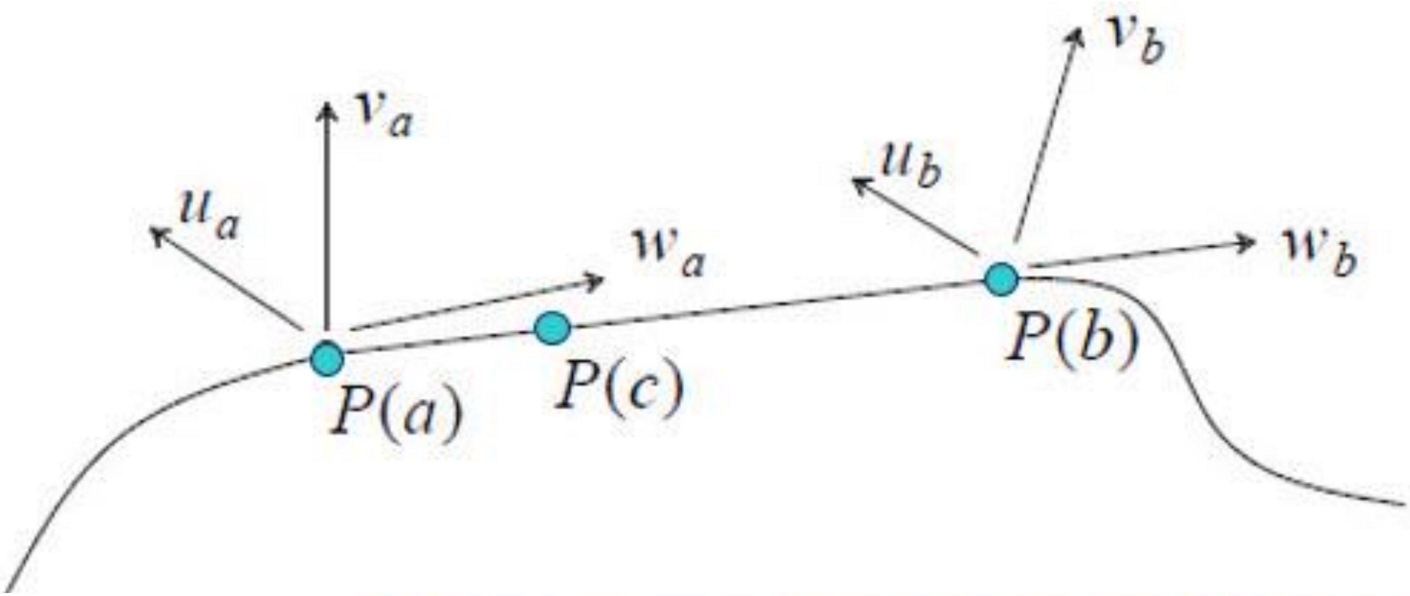
$$w = P'(s)$$

$$u = P'(s) \times P''(s)$$

$$v = w \times u$$

Solution for

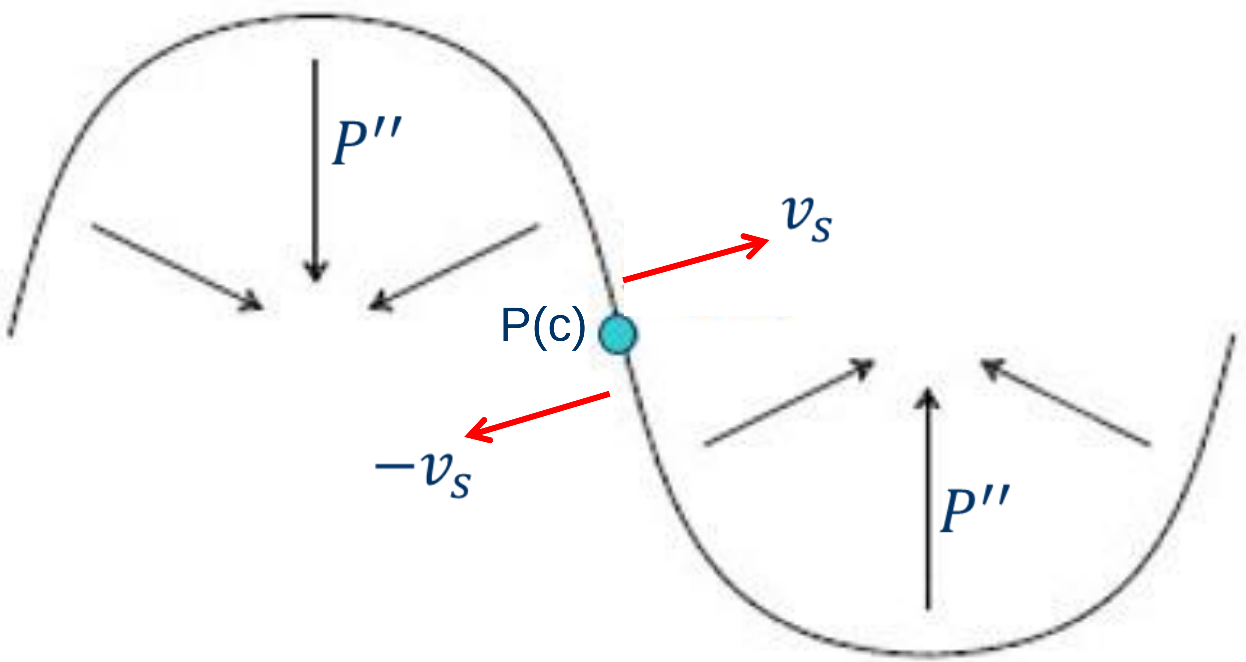
(i) when $P''(s) = 0$ for a segment



if $P''(s) = 0$ for $a < s < b$ then how should a local coordinate system (u_c, v_c, w_c) for $P(c)$, $a < s < b$, be defined?

$$\boxed{\begin{array}{l} w_c = w_a = w_b \\ \text{interpolate } v_a \text{ and } v_b \text{ to get } v_c \end{array}}$$

Solution for
(ii) when $P''(s)$ is not continuous at a point



Change v_s to $-v_s$ when $s > c$

Main problem with using the Frenet frame as the local coordinate frame to define the orientation of the camera or object following the path is:

the resulting motions are too extreme and not natural looking

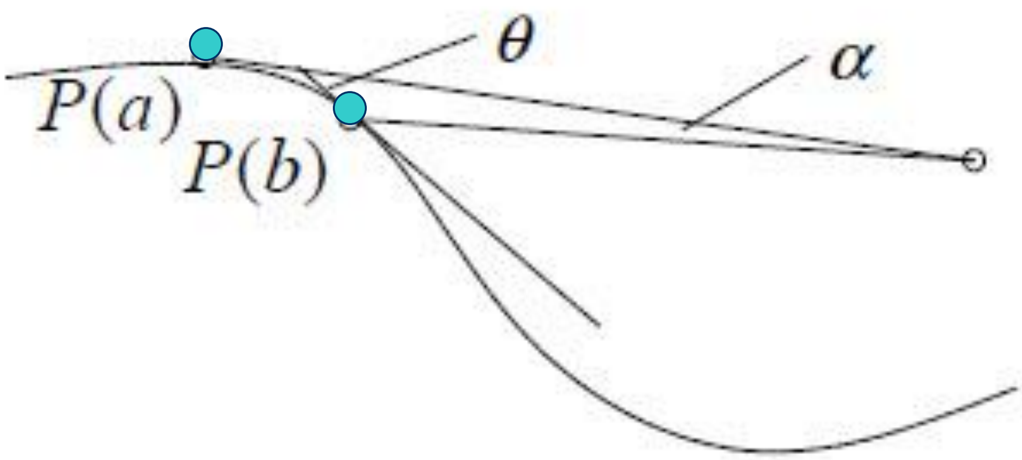
Using the w-axis (tangent vector) as the view direction of a camera can be undesirable. Why?

Often, the tangent vector does not appear to correspond to the direction of “where it’s going” even though it is in the instantaneous (analytic) sense.

The more natural orientation, for someone riding in a car or riding a bike, would be to look further ahead along the curve rather than to look tangential to the curve.

Solution for

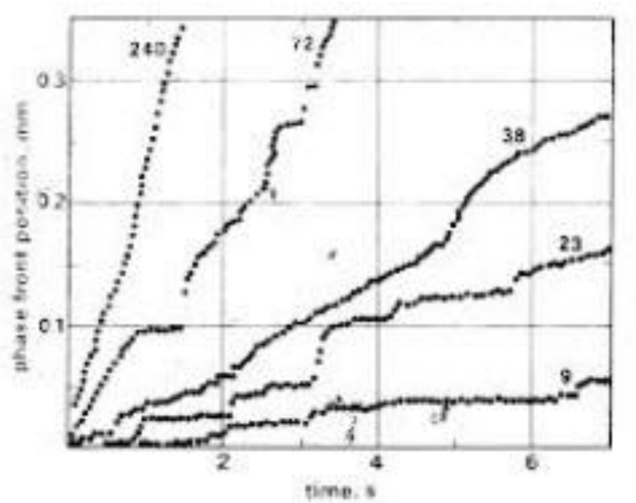
(ii) the resulting motions are too extreme and not natural looking

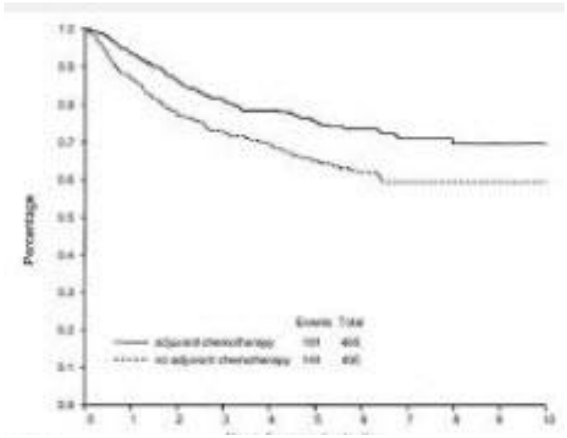


Define view vector as *COI - POS*

Smoothing a Path

- to remove the jerkiness of a path whose points are generated by a digitizing process

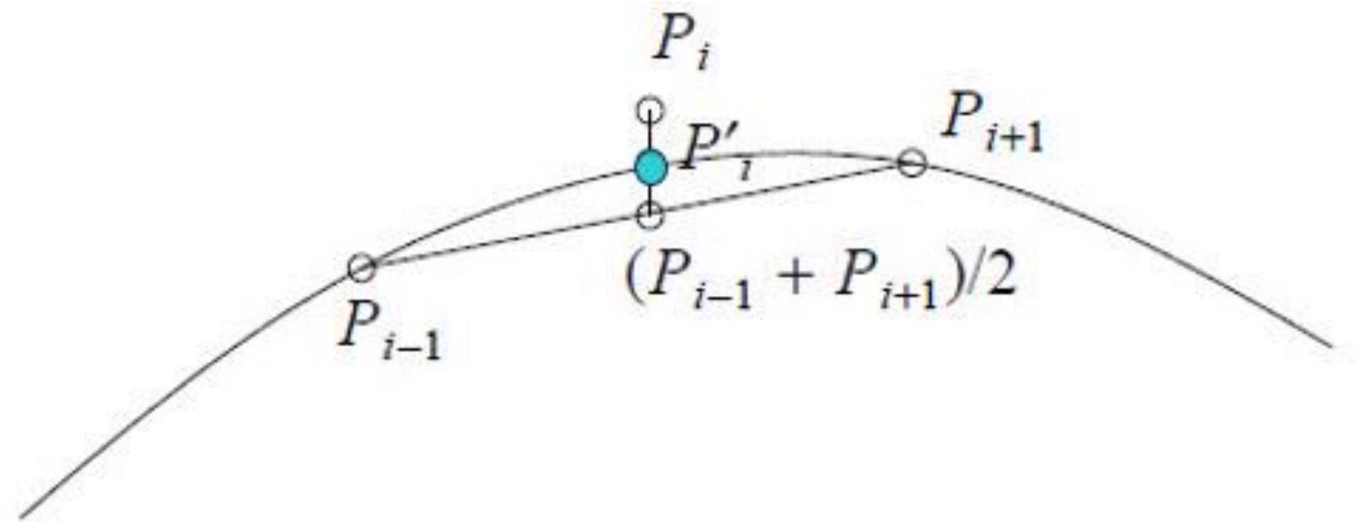




- usually, local methods are used

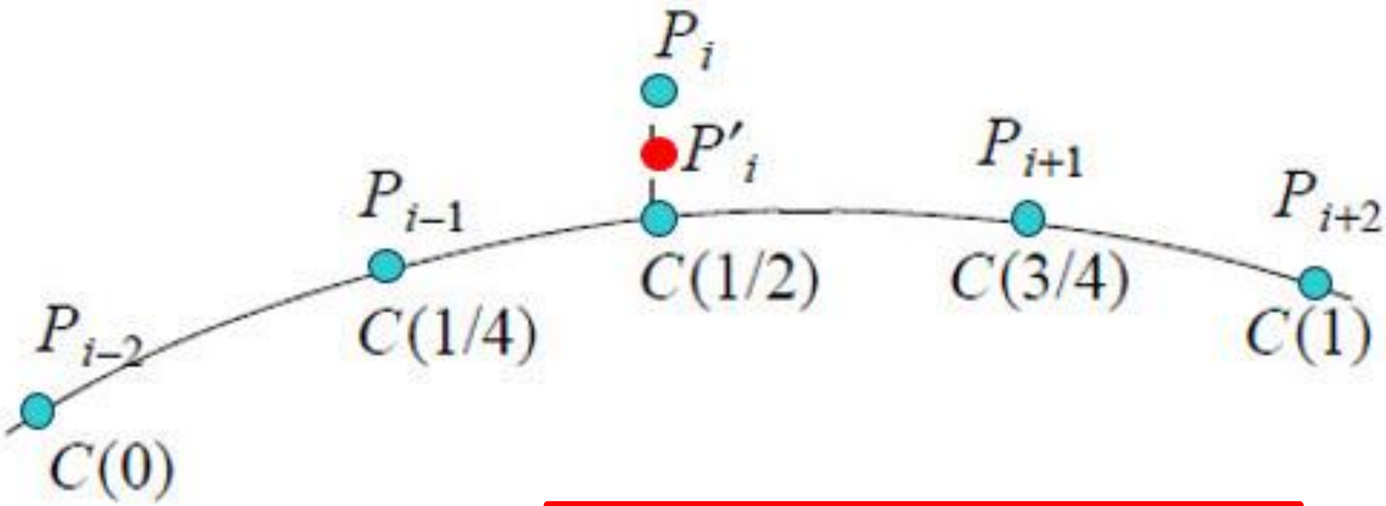
Smoothing a Path

(i) Smoothing with *linear interpolation* of adjacent points



repeated applications would flatten out the curve

(ii) Smoothing with *cubic interpolation* of adjacent points



Construct $\boxed{C(t) = at^3 + bt^2 + ct + d}$

such that $C(0) = P_{i-2}, \quad C(1/4) = P_{i-1}$

$C(3/4) = P_{i+1}, \quad C(1) = P_{i+2}$

Compute *C(1/2) ;* define $P'_i = \dfrac{C(1/2) + P_i}{2}$

$$\boxed{\begin{aligned} C(1/2) &= ? \\ &= \frac{P_{i-1} + P_{i+1}}{2} + \frac{P_{i-1} - P_{i-2}}{6} + \frac{P_{i+1} - P_{i+2}}{6} \end{aligned}}$$

Let $C(1/2) = x \cdot C(0) + y \cdot C(1/4) + z \cdot C(3/4) + w \cdot C(1)$

we have
$$\begin{cases} xd + yd + zd + wd = d \\ (y/64)a + (27z/64)a + wa = (1/8)a \\ (y/16)b + (9z/16)b + wb = (1/4)b \\ (y/4)c + (3z/4)c + wc = (1/2)c \end{cases}$$

or
$$\begin{cases} x + y + z + w = 1 \\ y/64 + 27z/64 + w = 1/8 \\ y/16 + 9z/16 + w = 1/4 \\ y/4 + 3z/4 + w = 1/2 \end{cases}$$

Solving for x, y, z and w, we get

$$\text{x} = \text{-}1/6, \quad \text{y} = 2/3, \quad \text{z} = 2/3, \quad \text{w} = \text{-}1/6$$

Hence, $\text{C}(1/2) = -C(0)/6 + 2C(1/4)/3 + 2C(3/4)/3 - C(1)/6$

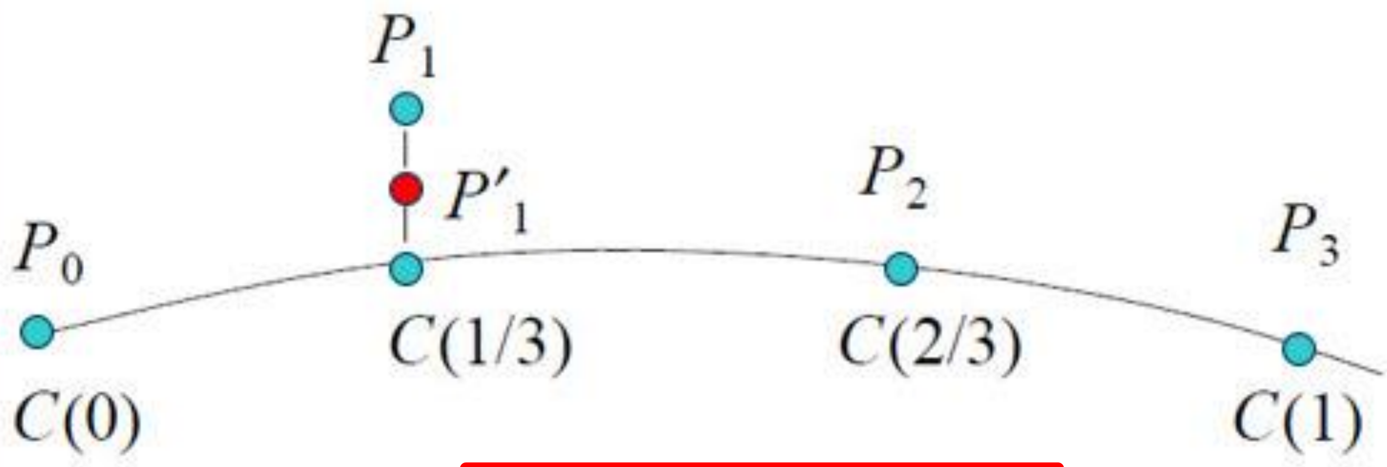
$$= -P_{i-2}/6 + 2P_{i-1}/3 + 2P_{i+1}/3 - P_{i+2}/6$$

$$C(t) = \boldsymbol{a}t^3 + \boldsymbol{b}t^2 + \boldsymbol{c}t + \boldsymbol{d}, \qquad \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in R^3$$

$$C(1/2) = xC(0) + yC(1/4) + zC(3/4) + wC(1)$$

$\boldsymbol{a}/8$	$x\boldsymbol{a} \cdot 0$	$y\boldsymbol{a}/64$	$27z\boldsymbol{a}/64$	$w\boldsymbol{a}$
$+$	$+$	$+$	$+$	$+$
$\boldsymbol{b}/4$	$x\boldsymbol{b} \cdot 0$	$y\boldsymbol{b}/16$	$9z\boldsymbol{b}/16$	$w\boldsymbol{b}$
$+$	$+$	$+$	$+$	$+$
$\boldsymbol{c}/2$	$x\boldsymbol{c} \cdot 0$	$y\boldsymbol{c}/4$	$3z\boldsymbol{c}/4$	$w\boldsymbol{c}$
$+$	$+$	$+$	$+$	$+$
$\boldsymbol{d}$	$x\boldsymbol{d}$	$y\boldsymbol{d}$	$z\boldsymbol{d}$	$w\boldsymbol{d}$

At left end:



Construct $\boxed{C(t) = at^2 + bt + c}$ such that

$$C(0) = P_0, \quad C(2/3) = P_2, C(1) = P_3$$

Compute C(1/3) define $P'_1 = \dfrac{C(1/3) + P_1}{2}$

$$\boxed{\begin{aligned} C(1/3) &= ? \\ &= P_2 + (P_0 - P_3)/3 \end{aligned}}$$

$\boxed{\text{Question: how should } P_0 \text{ be adjusted?}}$ t, UK

Let $C(1/3) = x \cdot C(0) + y \cdot C(2/3) + z \cdot C(1)$

we have
$$\begin{cases} xc + yc + zc = c \\ (4y/9)a + za = (1/9)a \\ (2y/3)b + zb = (1/3)b \end{cases}$$

or
$$\begin{cases} x + y + z = 1 \\ 4y/9 + z = 1/9 \\ 2y/3 + z = 1/3 \end{cases}$$

Solving for x, y and z, we get

$$x = 1/3, \; y = 1, \; z = -1/3$$

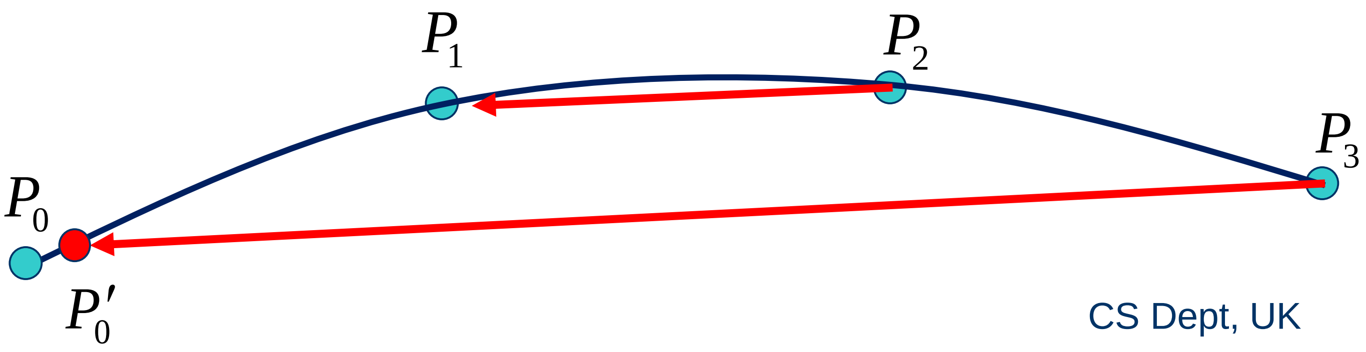
Hence, $C(1/3) = C(0)/3 + C(2/3) - C(1)/3$

$$= P_0/3 + P_2 - P_3/3$$

At left end: $\boxed{\text{Question: how should } P_0 \text{ be adjusted?}}$

1. This point can be left alone if it represents hard constraint
2. Parabolic interpolation can be used to generate estimate for this point

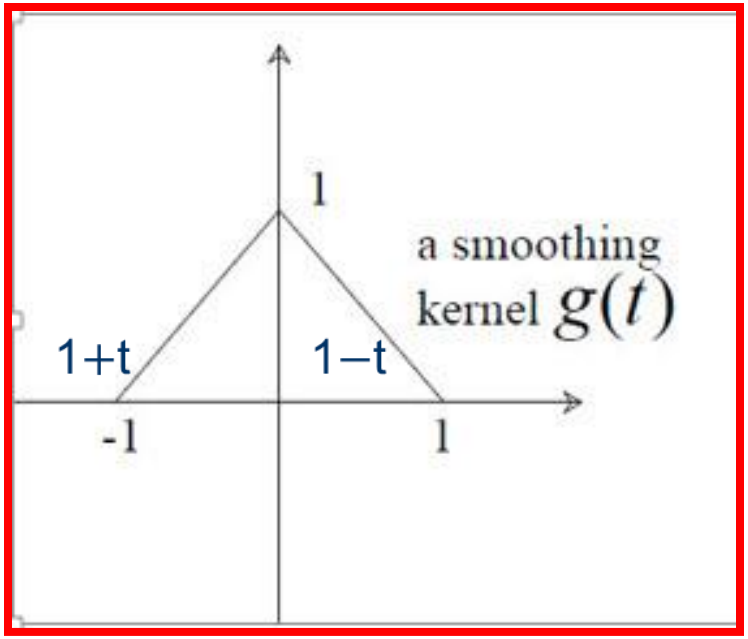
$$\boxed{\text{For example}: \quad P_0' = P_3 + 3(P_1 - P_2)}$$

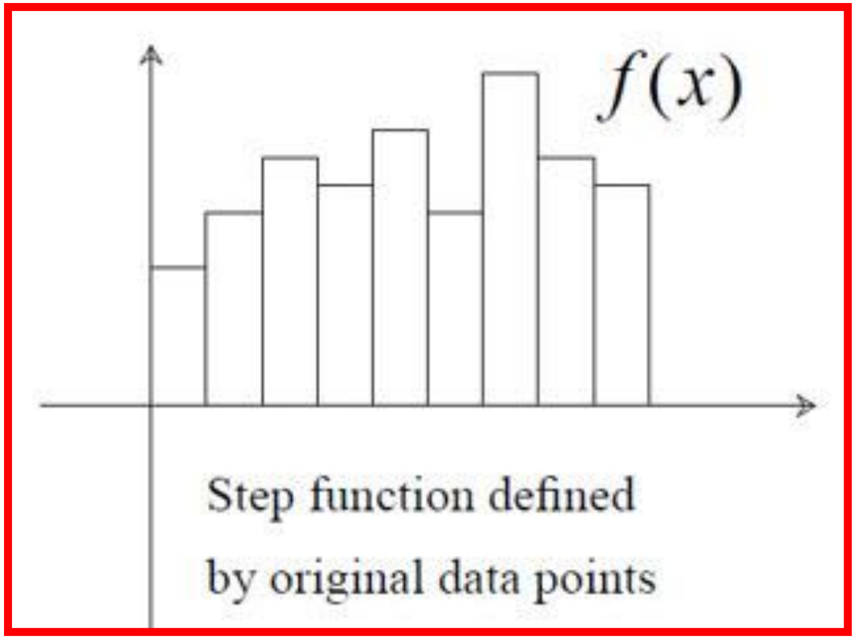


When the data to be smoothed can be viewed as the value of a function, i.e., $y_i=f(x_i)$

(iii) Smoothing with *convolution kernels*

- new point is generated by applying a smoothing kernel to the data points viewed as a step function



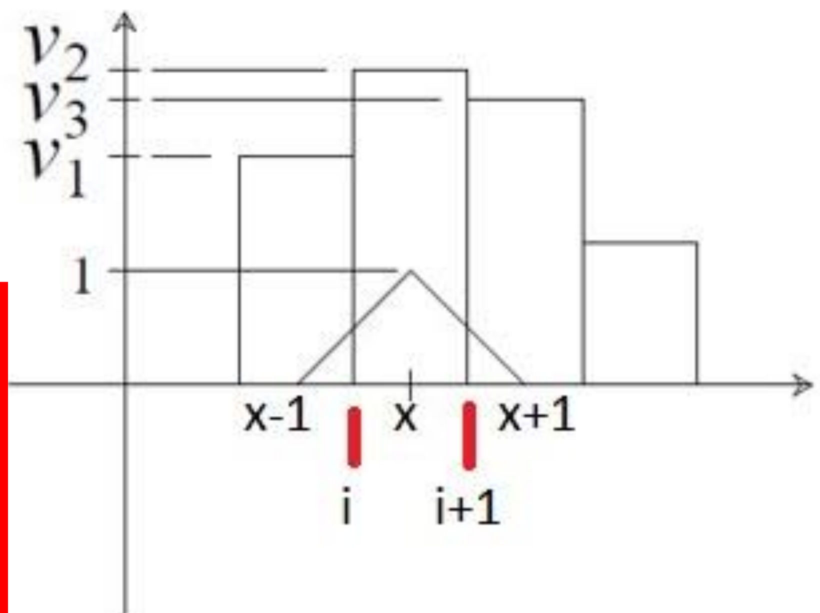


(iii) Smoothing with *convolution kernels*

- new point is generated by applying a smoothing kernel to the data points viewed as a step function

$$\boxed{P(x) = \int_{-s}^{s} f(t)\,g(x-t)\,dt}$$

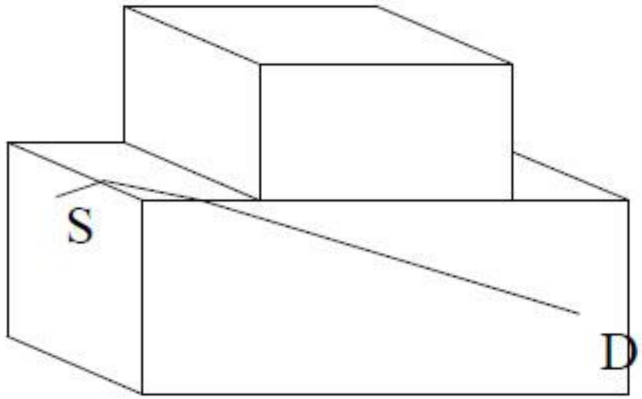
$$\boxed{\begin{aligned} P(x) = (v_1 - v_2)\frac{(1-x+i)^2}{2} \\ +(v_3 - v_2)\frac{(x-i)^2}{2} + v_2 \end{aligned}}$$



Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

a. plane intersection
b. greedy algorithm
c. shortest path

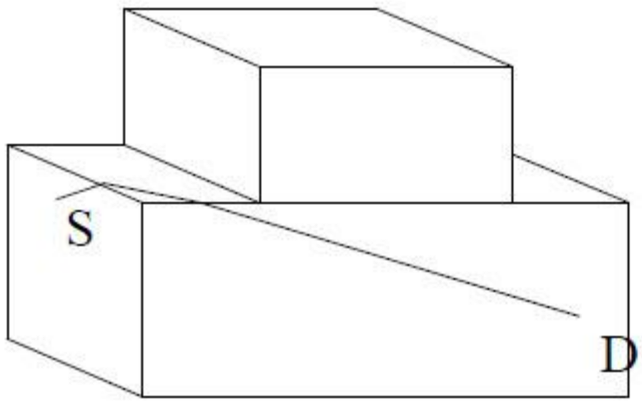


Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

a. plane intersection

Determine a plane that contains the start point and the destination point and is *generally perpendicular* to the surface



Average of the two vertex normals

How to improve this algorithm?

initially the start vertex

Determining a Path along a Surface:

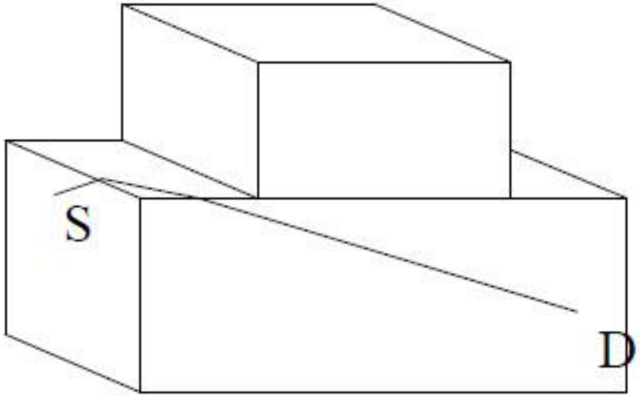
b. **greedy algorithm**

For each edge emanating from the current vertex, calculate the projection of the edge onto the straight line between the current vertex and the destination vertex.
Divide this distance by the length of the edge to get the cosine of the angle between the edge and the straight line. The edge with the largest cosine is the edge most in the direction of the straight line; choose this edge to add to the path.

Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

c. **shortest path**

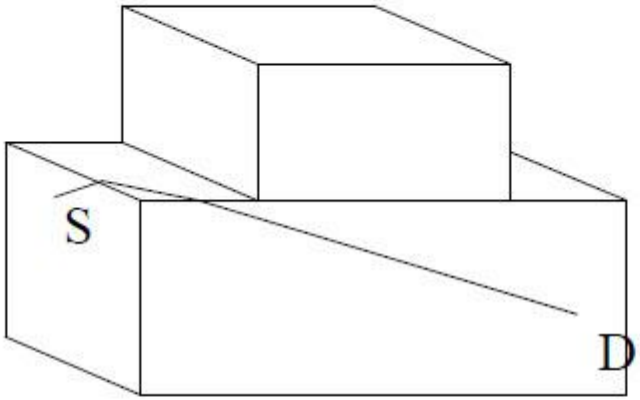


Unfold the faces of the mesh to be on a plane.
The shortest path is a straight line between the start vertex and the destination vertex that lies within the unfolded mesh.

Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

c. **shortest path**

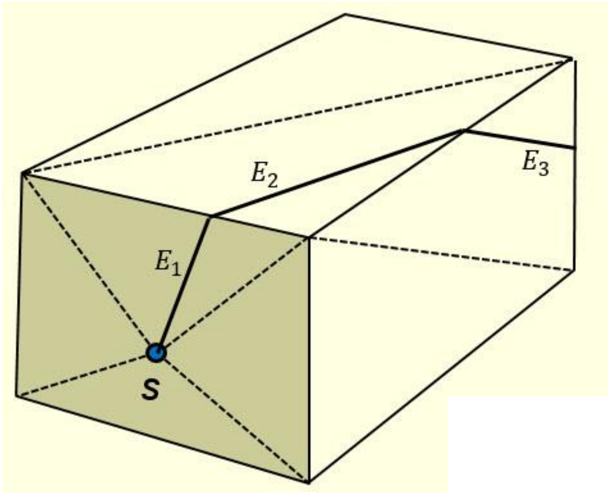


Jingdon Chen, Yijie Han,
Shortest Path on a Polyhedron,
Part I: Computing Shortest Path,
International Journal of Computational Geometry
and Applications, 6, 2 (1992), 127-144.

Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

c. **shortest path**



Basic idea:

1. Triangulate all the faces
2. Triangulate the face that contains S so that *S* becomes a vertex
3. Unfolding, using the following approach

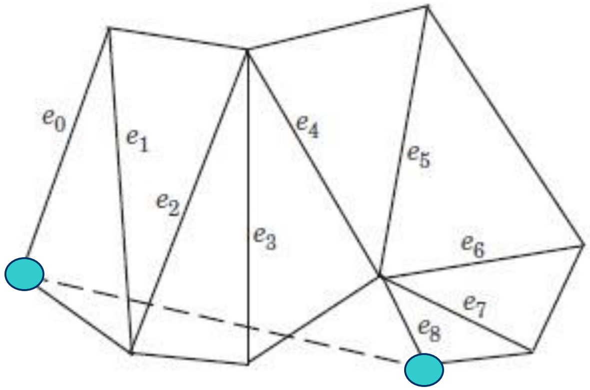
Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

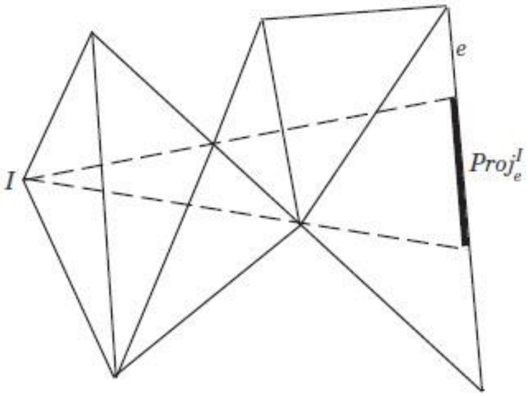
c. shortest path

Unfolding strategy:

1. Avoid situation such as the one shown on the right
2. Use geodesic path



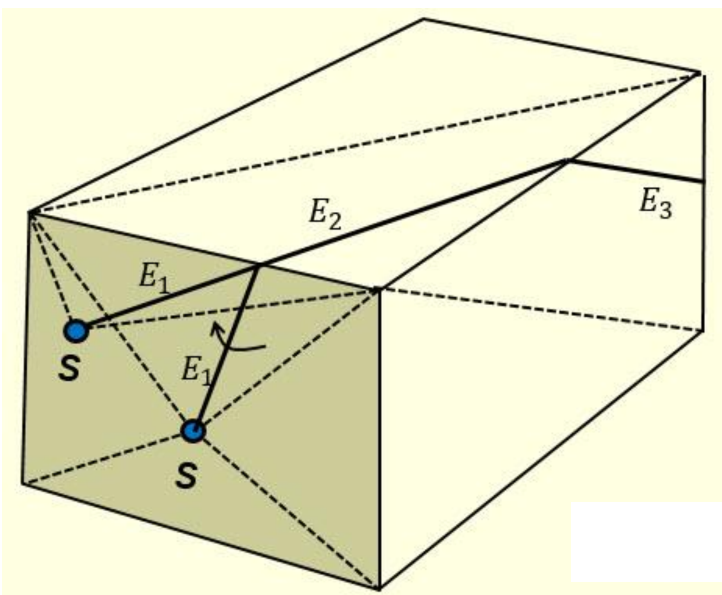
No shortest paths can pass through e_0, e_1, ... e_8 .

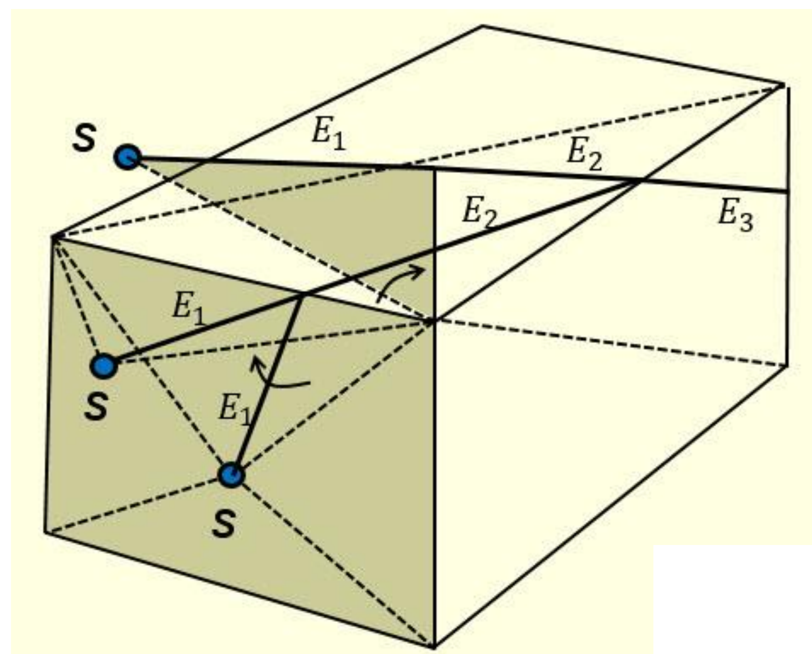


Determining a Path along a Surface:

(i) along a *polygonal surface mesh*

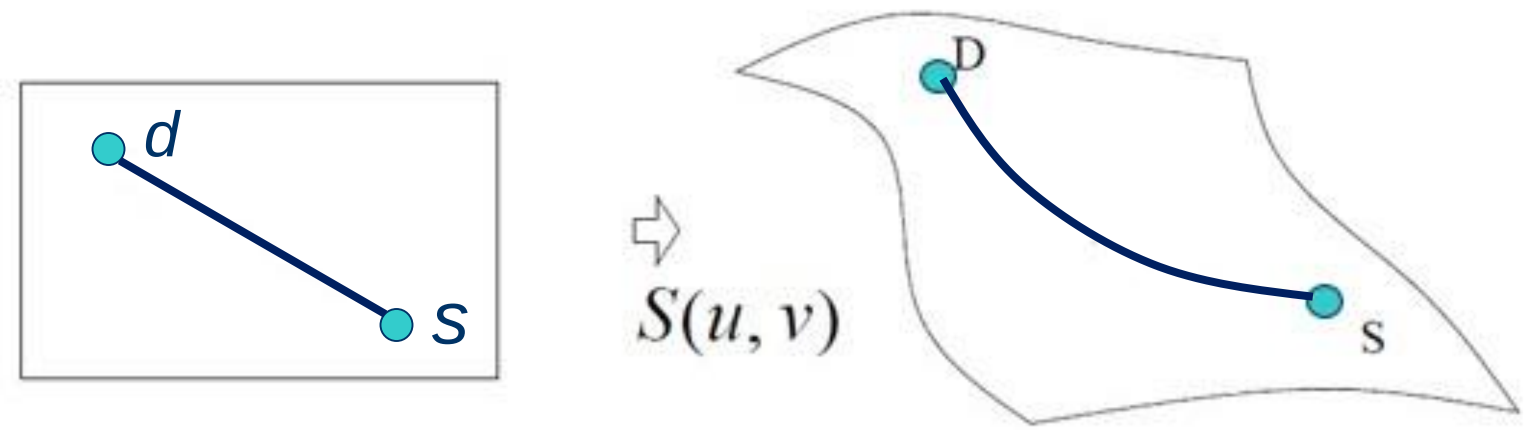
Unfolding strategy:





Determining a Path along a Surface:

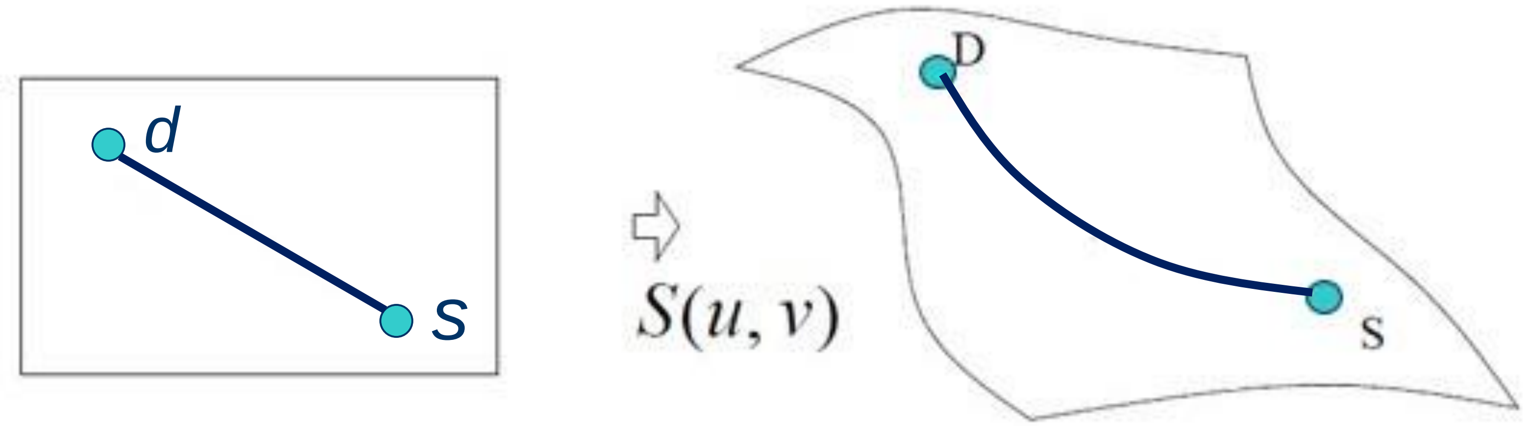
(ii) along a *high-order parametric surface*



construct a line in parameter space and transfer to the surface.
(Question: how to find d and s?)

Determining a Path along a Surface:

(ii) along a *high-order parametric surface*



1. Use *midpoint subdivision technique* to refine the surface
2. Use convex hull concept to find *d* and *s*

Path Finding:

finding a collision-free path in a given environment

A topic usually addressed in the robotics literature.

Complexity of the problem increases when the environment is not stationary, and the problem becomes more complex if the obstacles' movement is not predictable.

No good solutions to this problem have been found yet, even though some (computation intensive) greedy algorithms have been proposed.

Path Finding:

finding a collision-free path in a given environment

Recently, also considered in graphics, such as walk through a plaza or a room with a lot of people moving around.

End of Interpolation III