COSE436

Lecture 20:
Curves & Surfaces II

Nov 21, 2024

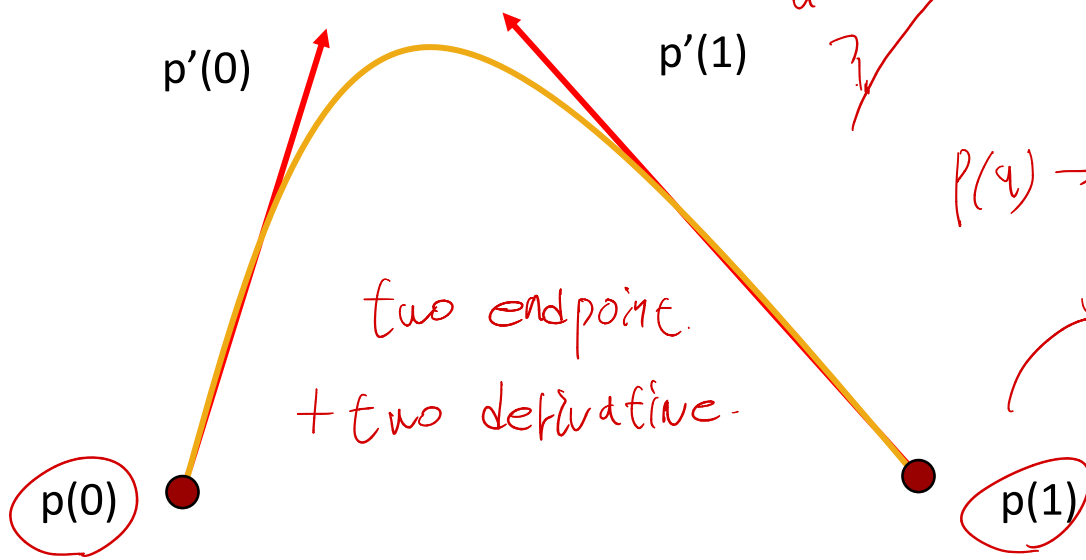
Won-Ki Jeong
(wkjeong@korea.ac.kr)

KOREA UNIVERSITY

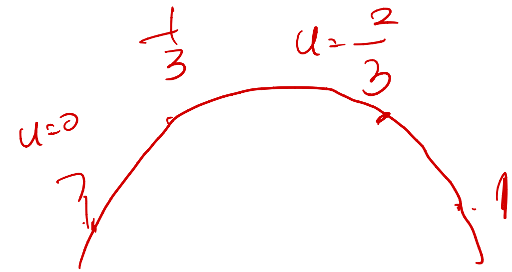
Outlines

- Curves and surfaces
  - Hermite
  - Bezier
  - Splines

"Hermite Form"



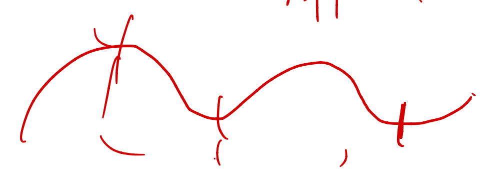
two endpoint
+ two derivative.



Interp.

$p(u) \rightarrow c_0 + c_1 u + c_2 u^2 + c_3 u^3$

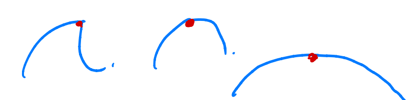
Approx



Joint

continuity.

C_0 C_1 C_2.



- Use two interpolating conditions and two derivative conditions per segment
- Ensures continuity and first derivative continuity between segments

Hermite Form Equations

Interpolating conditions are the same at ends

$$p(0) = p_0 = c_0$$
$$p(1) = p_3 = c_0+c_1+c_2+c_3$$

Differentiating we find $p'(u) = c_1+2uc_2+3u^2c_3$

Evaluating at end points

$$p'(0) = p'_0 = c_1$$
$$p'(1) = p'_3 = c_1+2c_2+3c_3$$

Matrix Form

- We find $\mathbf{c}=\mathbf{M}_H\mathbf{q}$ where $\mathbf{M}_H$ is the Hermite matrix

$$\mathbf{q} = \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c} \Rightarrow \mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

Hermite Blending Polynomials

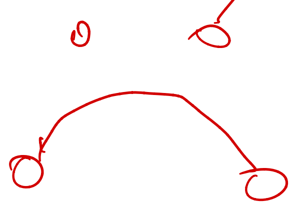
$$p(u) = u^T \mathbf{M_H q} = \mathbf{b}(u)^T \mathbf{q}$$

$$\mathbf{b}(u) = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix} \begin{matrix} p_0 \\ p_1 \\ p_0' \\ p_1' \end{matrix}$$

No zeros in [0,1], much smoother than interpolation blending polynomials

Hermite Blending Polynomial

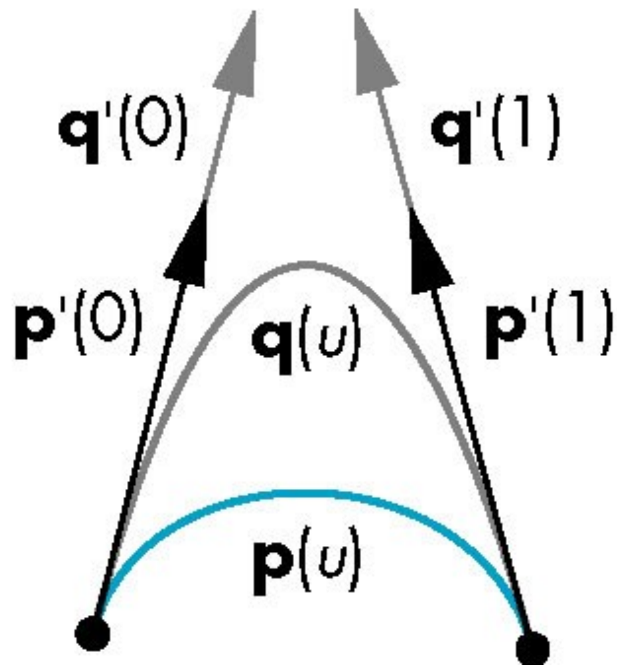
- Although Hermit blending functions are smooth, it is not used directly in Computer Graphics and CAD because <u>we usually have control points rather than derivatives</u>
- However, the Hermite form is the basis of the Bezier form



멀어지면
곡선 그리기가
끌어옴.

Hermite Form Example

- Here the p and q have the same tangents at the ends of the segment but different derivatives
- Generate different Hermite curves
- This techniques is used in drawing applications

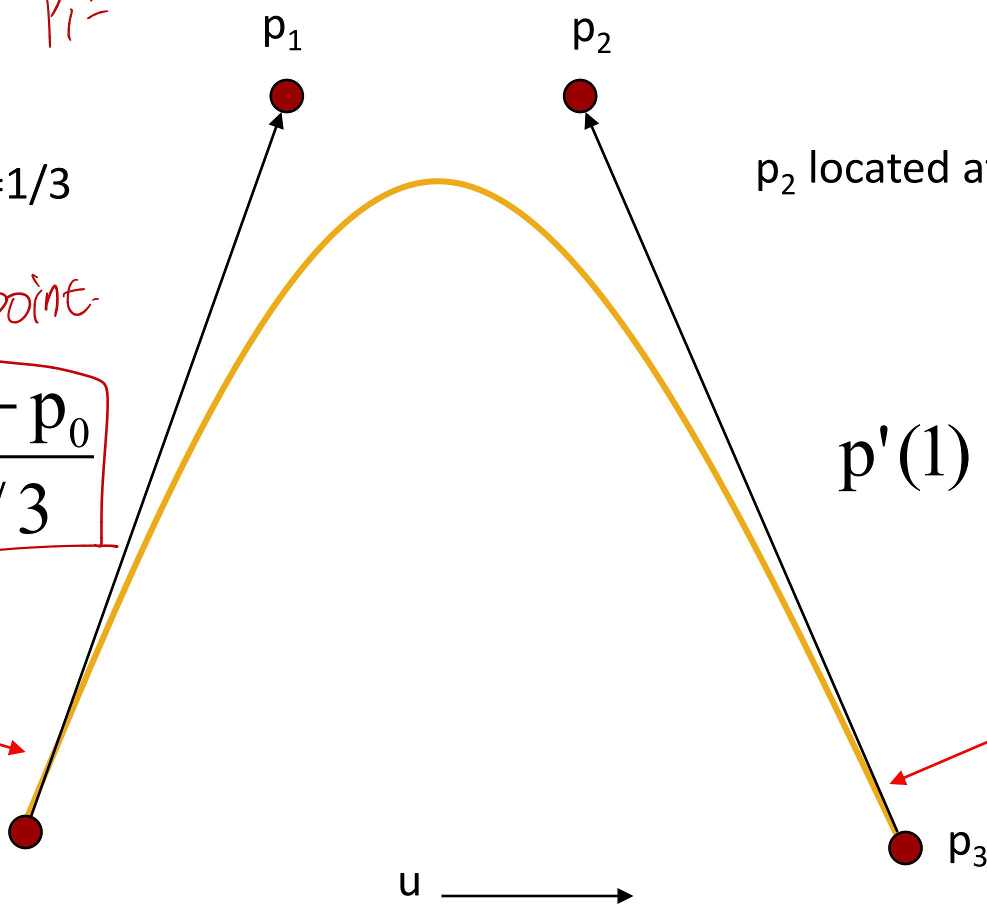


Bézier's Idea

- In graphics and CAD, we do not usually have derivative data
- Bezier suggested using the same 4 data points as with the cubic interpolating curve to approximate the derivatives in the Hermite form

Approximating Derivatives

$P_1 =$ $u = \frac{1}{3}$



p_1 located at u=1/3

p_2 located at u=2/3

not derivative

point

$$p'(0) \approx \boxed{\frac{p_1 - p_0}{1/3}}$$

$$p'(1) \approx \frac{p_3 - p_2}{1/3}$$

slope p'(0)

slope p'(1)

Bézier Equations

- Interpolating conditions are the same

$$p(0) = p_0 = c_0$$
$$p(1) = p_3 = c_0+c_1+c_2+c_3$$

- Approximating derivative conditions

$$p'(0) = (p_1 - p_0) / (1/3) = c_0$$
$$p'(1) = (p_3 - p_2) / (1/3) = c_1+2c_2+3c_3$$

- Solve three linear systems of four equations and four unknowns for $\mathbf{c}=\mathbf{M}_B\mathbf{p}$

Bézier Matrix

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$p(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_B\mathbf{p} = \mathbf{b}(u)^T\mathbf{p}$$

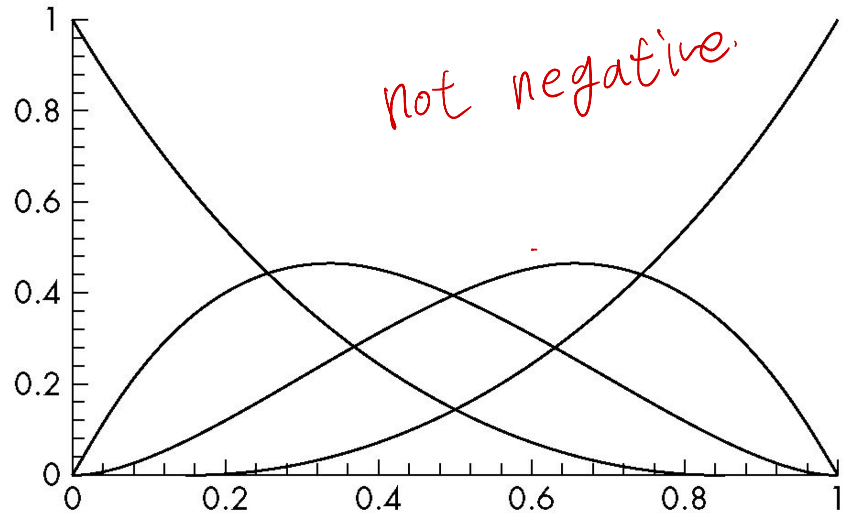
p_0
p_1
p_2
p_3

blending functions

Bézier Blending Functions

Nice!

$$\mathbf{b}(u) = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix}$$



Note that all zeros are at 0 and 1 which forces the functions to be smooth over (0,1)

Bernstein Polynomials

- The blending functions are a special case of the Bernstein polynomials

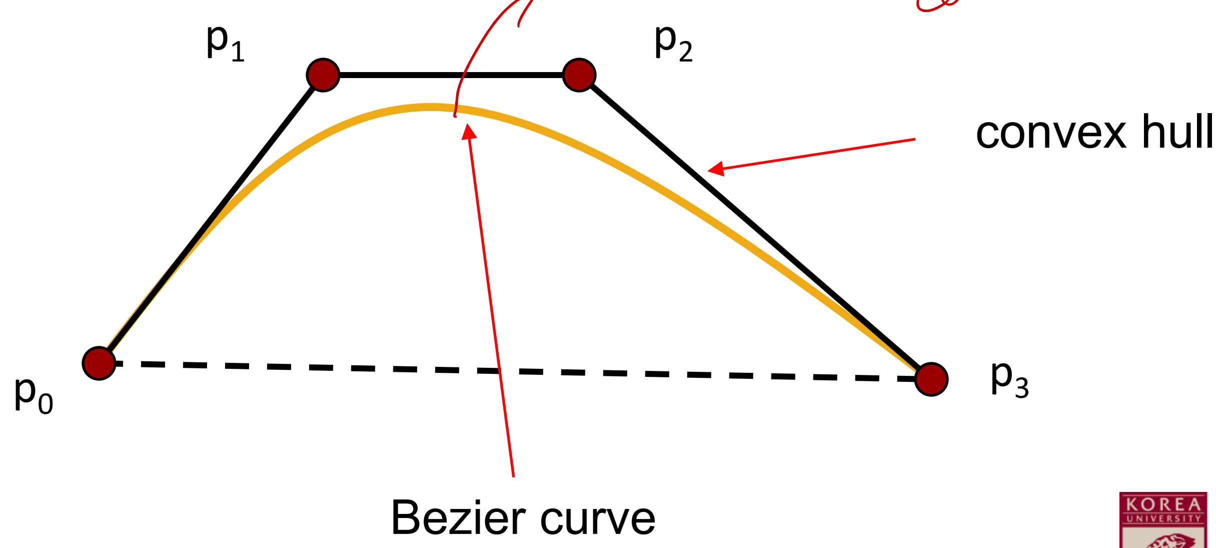
$$b_{\mathrm{kd}}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k}$$

- These polynomials give the blending polynomials for any degree Bezier form
  – All zeros at 0 and 1
  – For any degree they all sum to 1 : $\sum_{i=1}^{d} b_{id}(u) = 1$
  – They are all between 0 and 1 inside (0,1)

Convex Hull Property

- The properties of the Bernstein polynomials ensure that all Bezier curves lie in the convex hull of their control points
- Hence, even though we do not interpolate all the data, we cannot be too far away

이 곡선 위의 점이 항상
Convex Hull 내부.

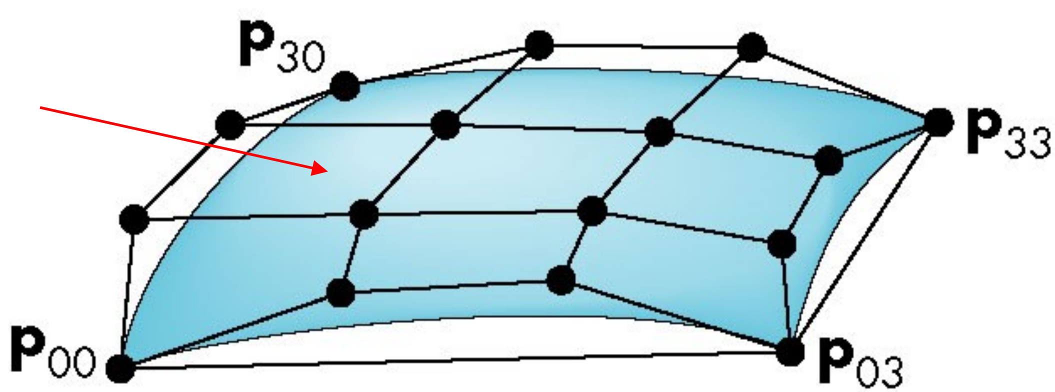


Bézier Patches

- Using same data array **P**=[p_{ij}] as with interpolating form, using bézier blending function

$$p(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\,b_j(v)\,p_{ij} = u^T \mathbf{M}_B \mathbf{P} \mathbf{M}_B^T v$$

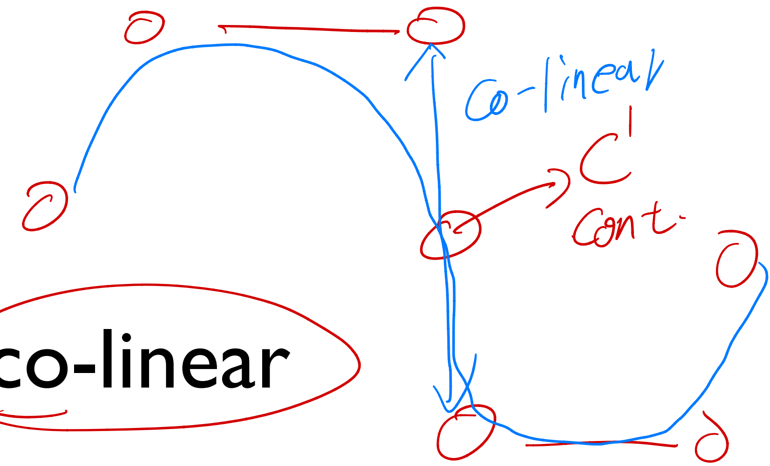
Patch lies in convex hull



Bézier Curve/Surface Analysis

- Interpolating end points
- C^0 continuous at joint
- C^1 if end line segments are co-linear
- Increasing Bezier degree does not increase continuity at joint (why?) → No C^2 cont (2번 미분하는 경우는 식에 없음?)
  - Better to connect lower degree Bezier for local control

한 번에 움직임은 최대 두개의 부분만 영향을 줌.



KOREA UNIVERSITY

Bézier Surface

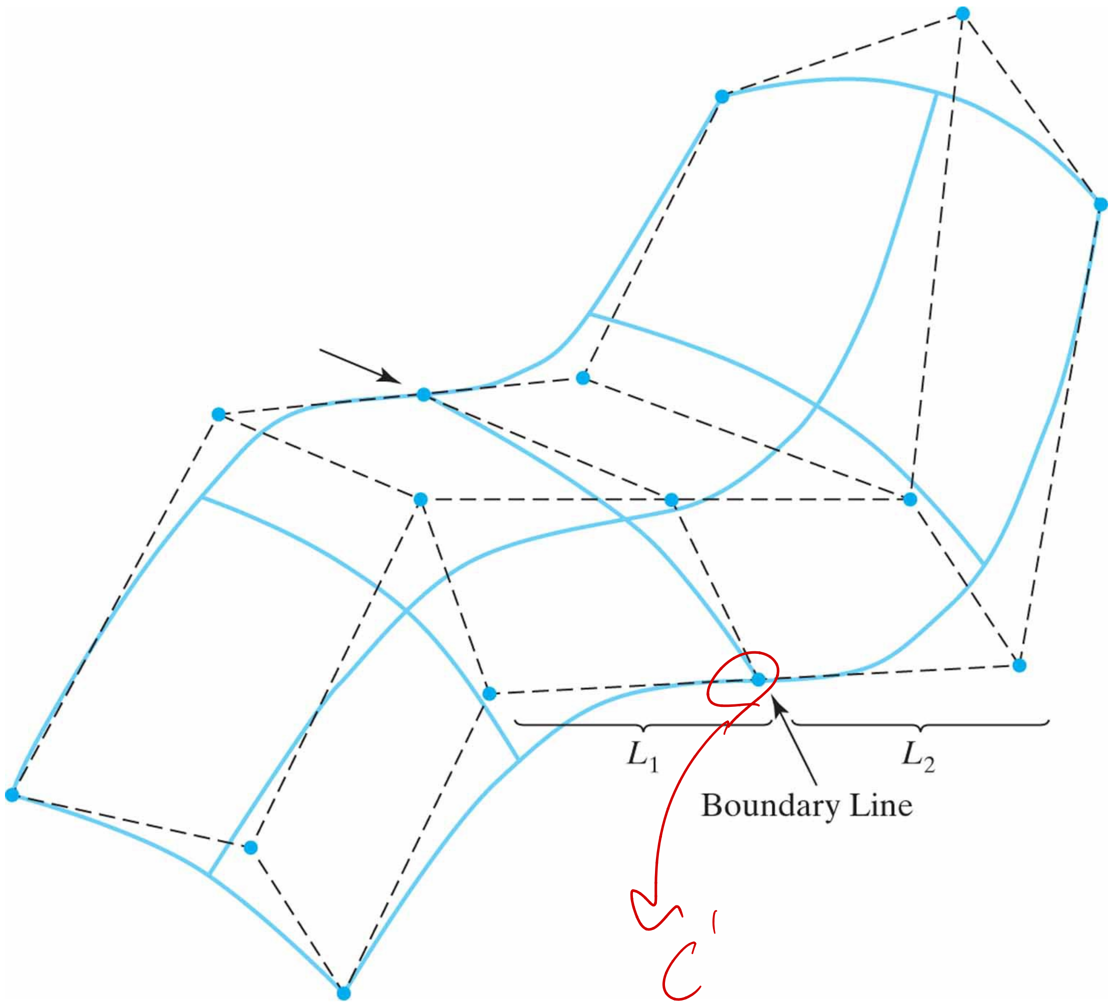


Image courtesy of Hearn

Splines

- Approximating
- Smooth joint
  - C^2 continuous
- Compact support

Cubic B-Spline

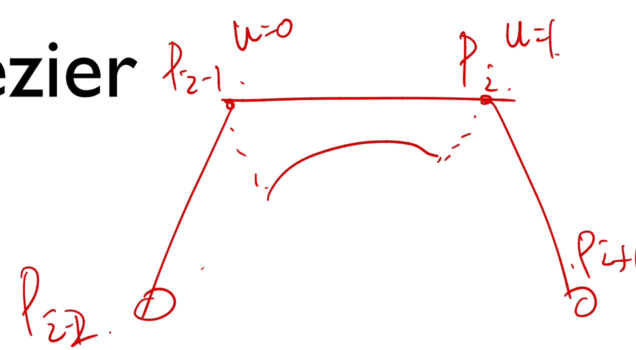
- Basis splines: use the data at $\mathbf{p}=[p_{i-2}\ p_{i-1}\ p_i\ p_{i+1}]^T$ to define curve only between p_{i-1} and p_i

Bezier는 끝점만 보간.

- C^2 at interior points

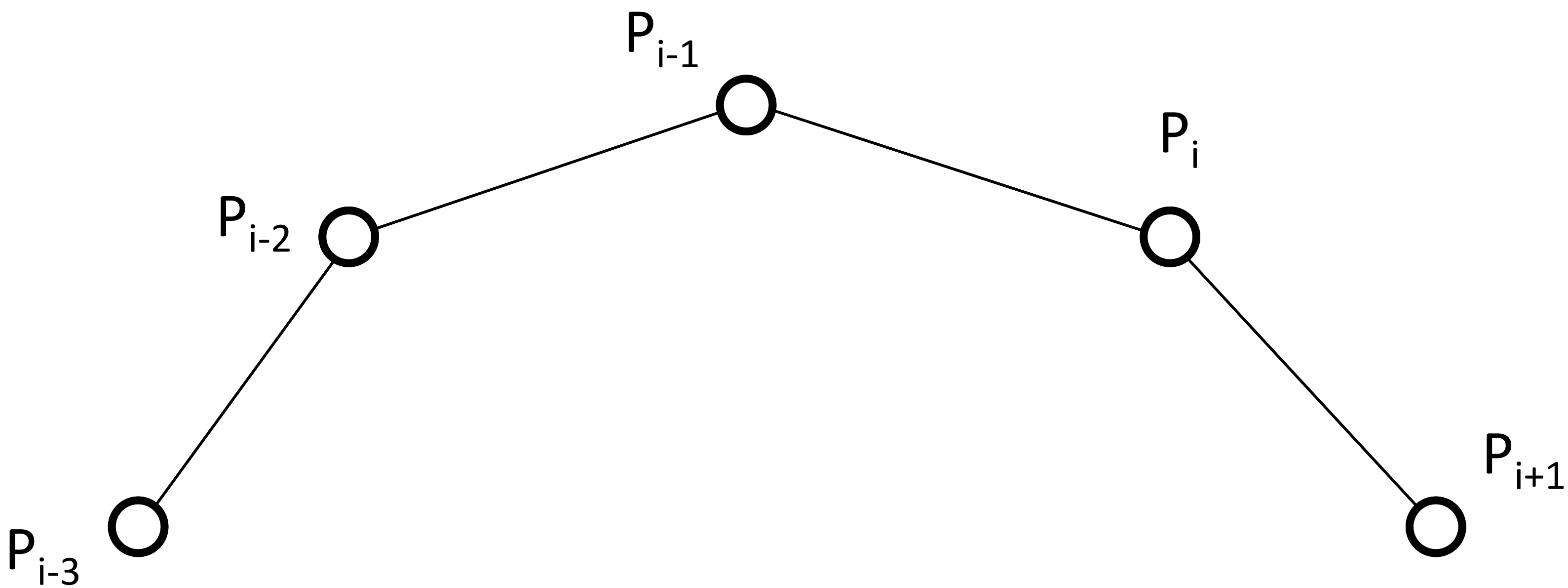
4개의 점이 $\frac{1}{3}$ 도막만 만듦.

- Cost is 3 times as much work for curves
  - For surfaces, we do 9 times as much work
- Add one new point each time rather than three as in Cubic Bézier



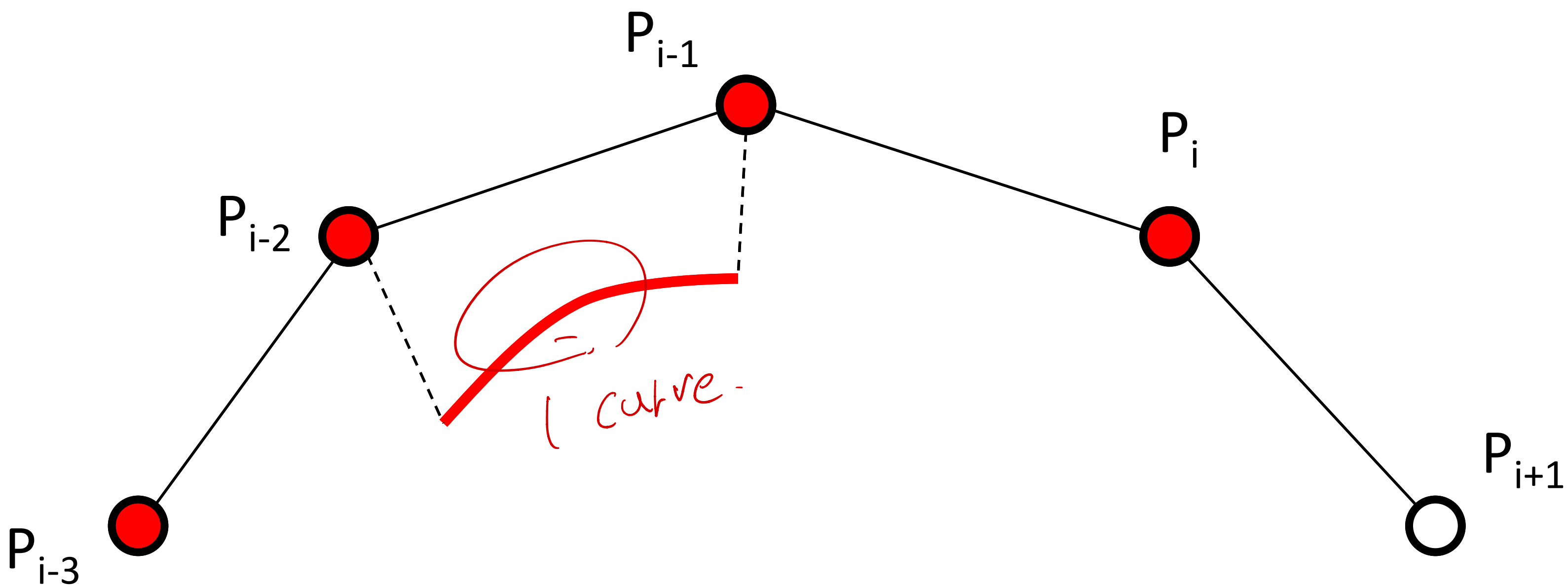
Cubic B-Spline

- Four control points make a curve segment



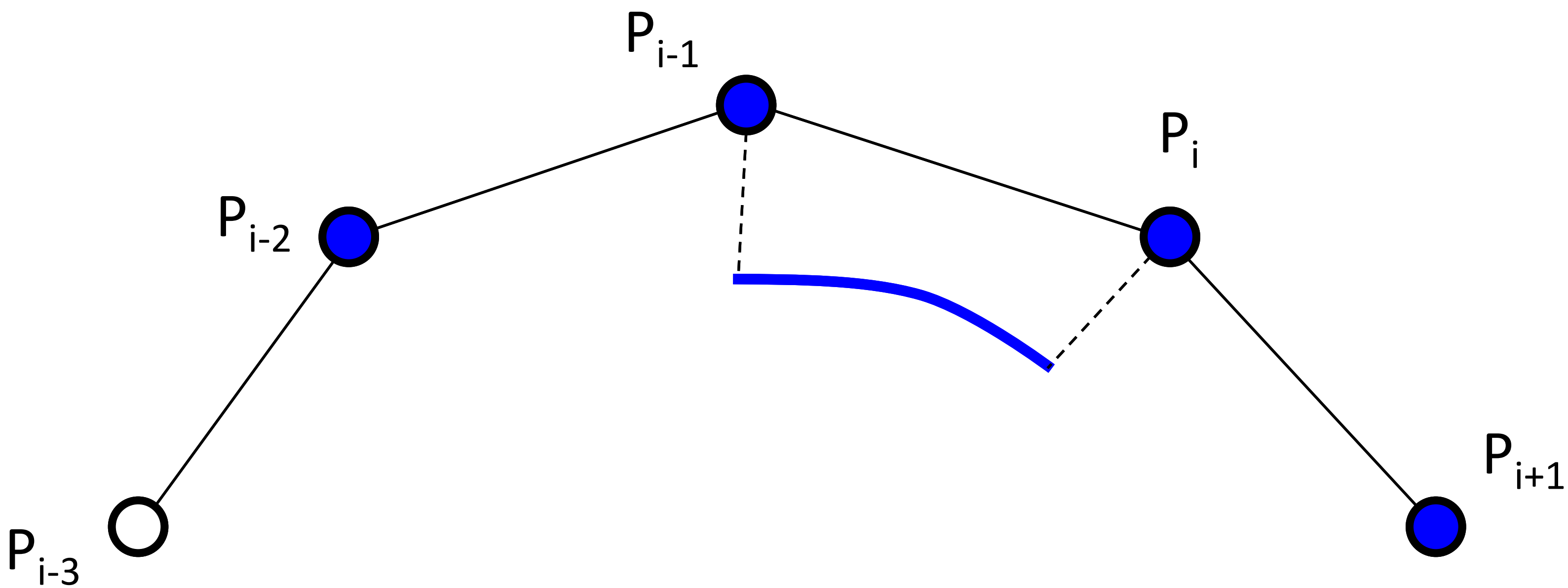
Cubic B-Spline

- Four control points make a curve segment



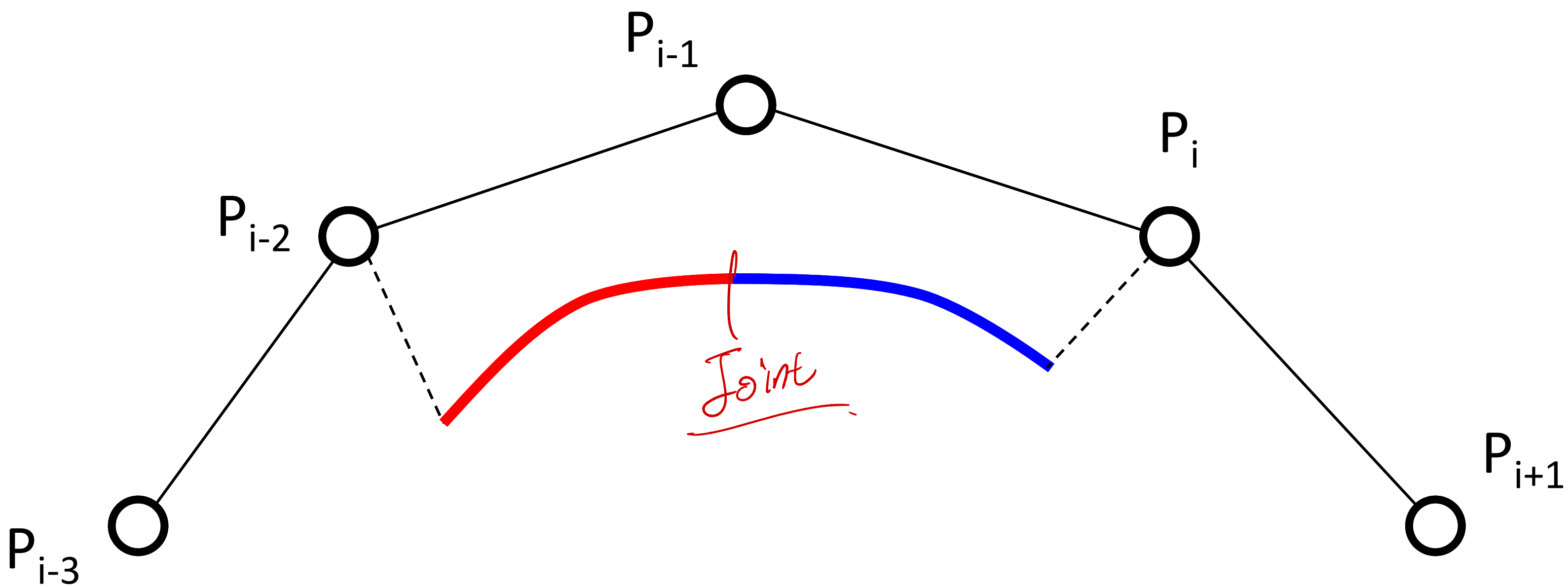
Cubic B-Spline

- Four control points make a curve segment



Cubic B-Spline

- Four control points make a curve segment



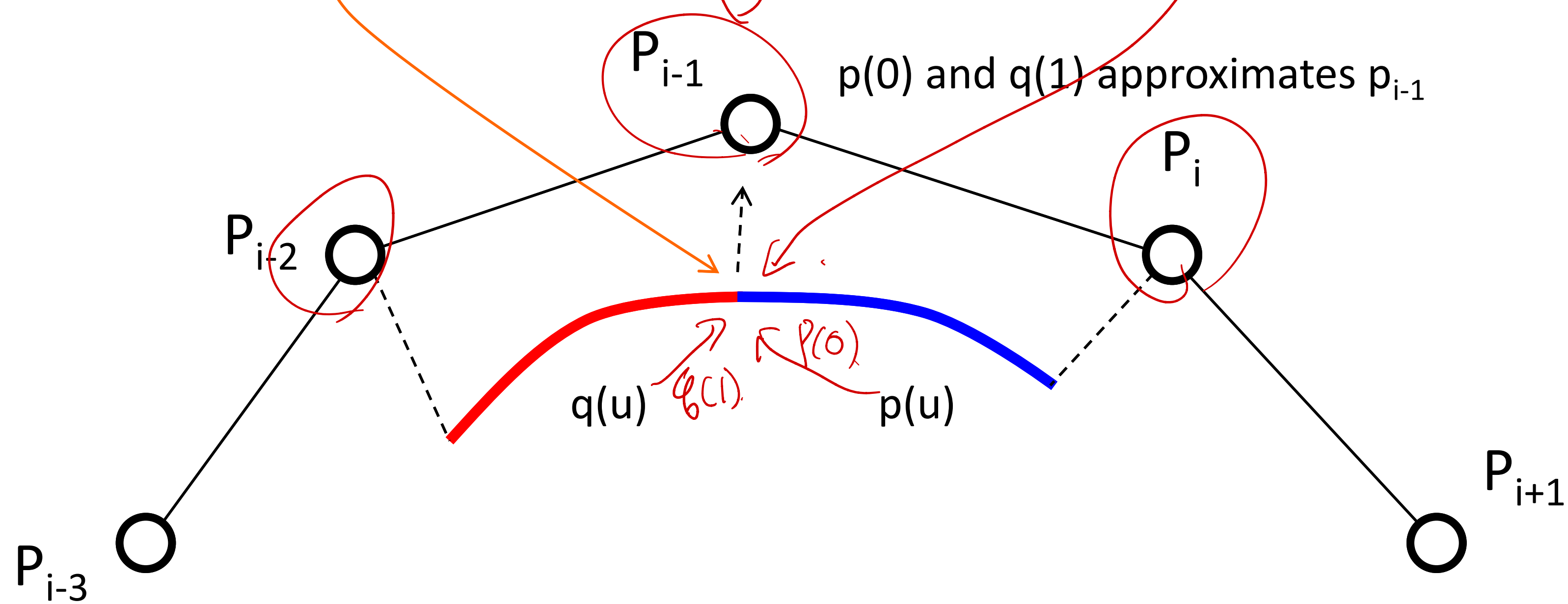
Deriving Cubic B-spline

- Consider points
  - pi-2, pi-1, pi, pi+1 : p(0) approx pi-1, p(1) approx pi
  - pi-3, pi-2, pi-1, pi : q(0) approx pi-2, q(1) approx pi-1
- Condition 1 : p(0)=q(1)
  - Symmetry: p(0) = q(1) = 1/6(pi-2 + 4 pi-1 + pi)
- Condition 2 : p'(0)=q'(1)
  - Geometry: p'(0) = q'(1) = 1/2 ((pi – pi-1) + (pi-1 – pi-2)) = 1/2 (pi – pi-2)

End-point Constraints

$$q(1) = p(0) = \frac{1}{6}(p_{i-2} + 4p_{i-1} + p_i) = c_0 \qquad p(u) = \sum_{k=0}^{3} c_k u^k$$

$$q'(1) = p'(0) = \frac{1}{2}(p_i - p_{i-2}) = c_1$$

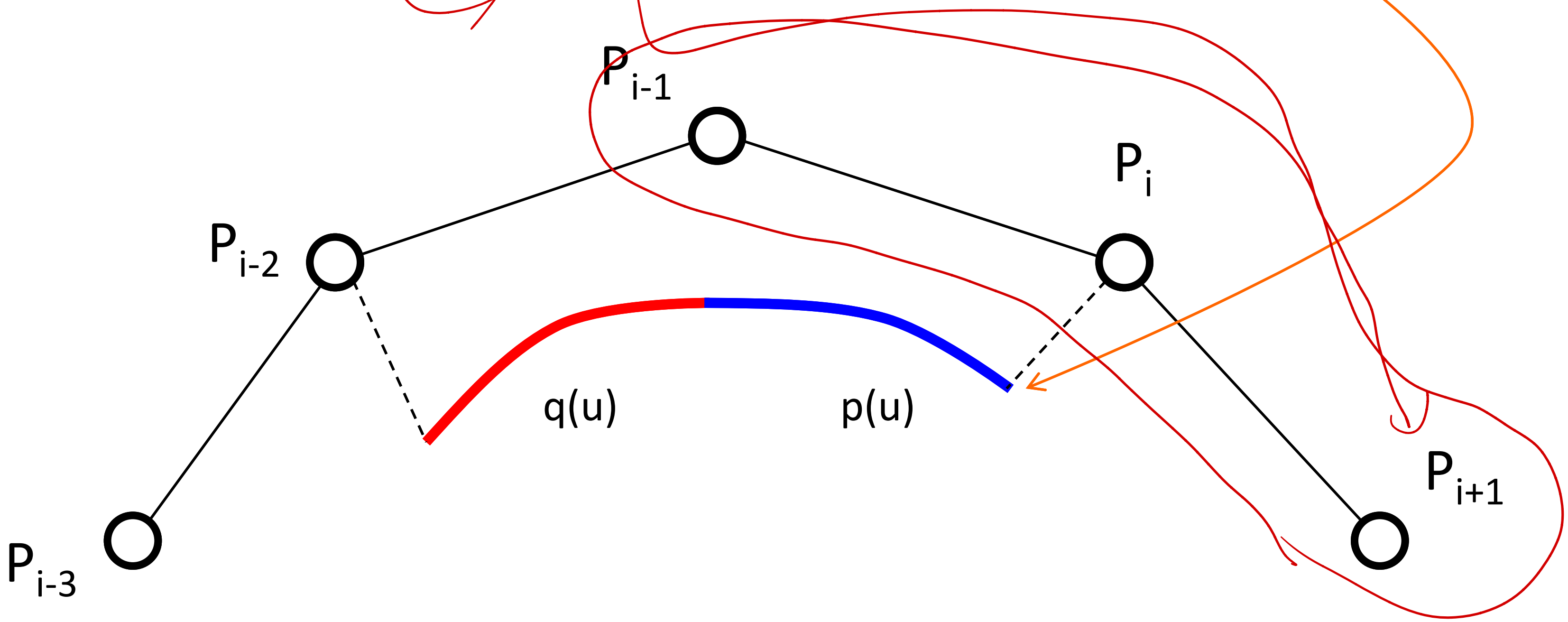


KOREA UNIVERSITY

End-point Constraints

$$p(1) = \frac{1}{6}(p_{i-1} + 4p_i + p_{i+1}) = c_0 + c_1 + c_2 + c_3$$

$$p'(1) = \frac{1}{2}(p_{i+1} - p_{i-1}) = c_1 + 2c_2 + 3c_3$$

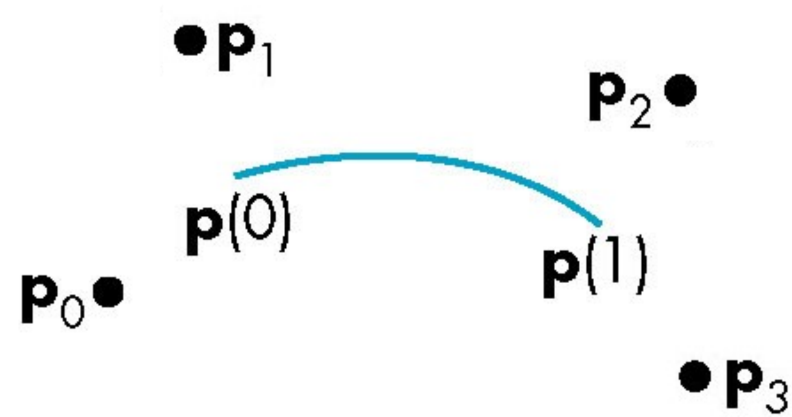


Cubic B-spline

b-func.

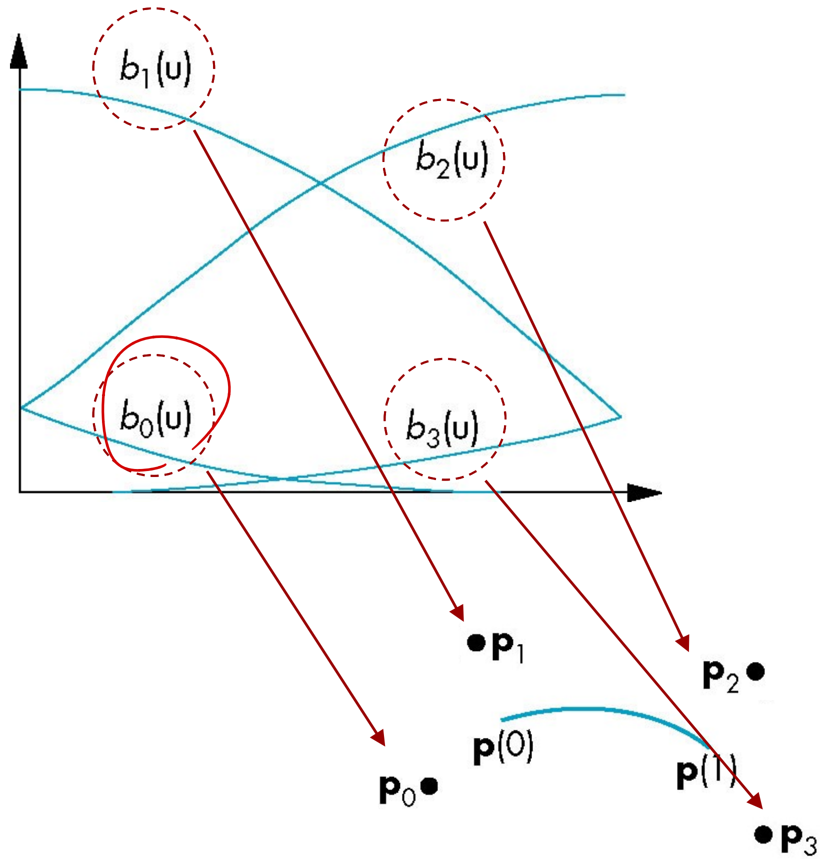
$$p(u) = \mathbf{u}^T\,\mathbf{c} = \mathbf{u}^T\mathbf{M}_S\mathbf{p} = \boxed{\mathbf{b}(u)^T}\,\mathbf{p}$$

$$\mathbf{M}_S = \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$



Blending Functions

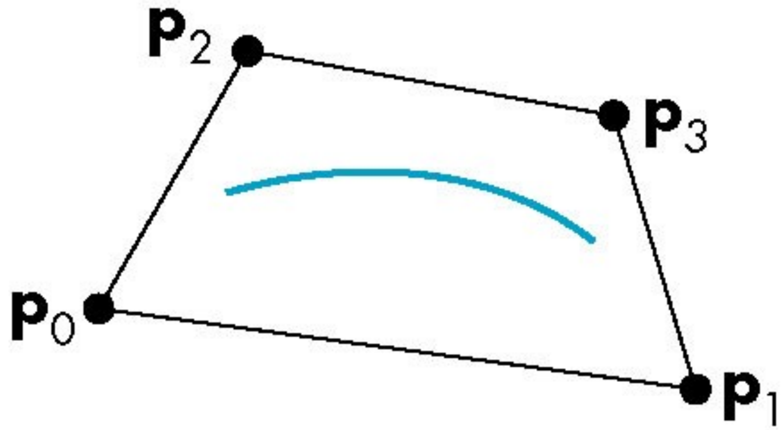
$$\mathbf{b}(u) = \frac{1}{6}\begin{bmatrix} (1-u)^3 \\ 4-6u^2+3u^3 \\ 1+3u+3u^2-3u^2 \\ u^3 \end{bmatrix}$$



$$p(u) = \mathbf{u}^T\mathbf{M}_S\mathbf{p} = \mathbf{b}(u)^T\mathbf{p}$$

Convex Hull Property

- For $0 \leq u \leq 1$, have $0 \leq b_k(u) \leq 1$, $\text{sum}(b_k(u))=1$
- $p(u) = b_{i-2}(u)p_{i-2} + b_{i-1}(u)p_{i-1} + b_i(u)p_i + b_{i+1}(u)p_{i+1}$



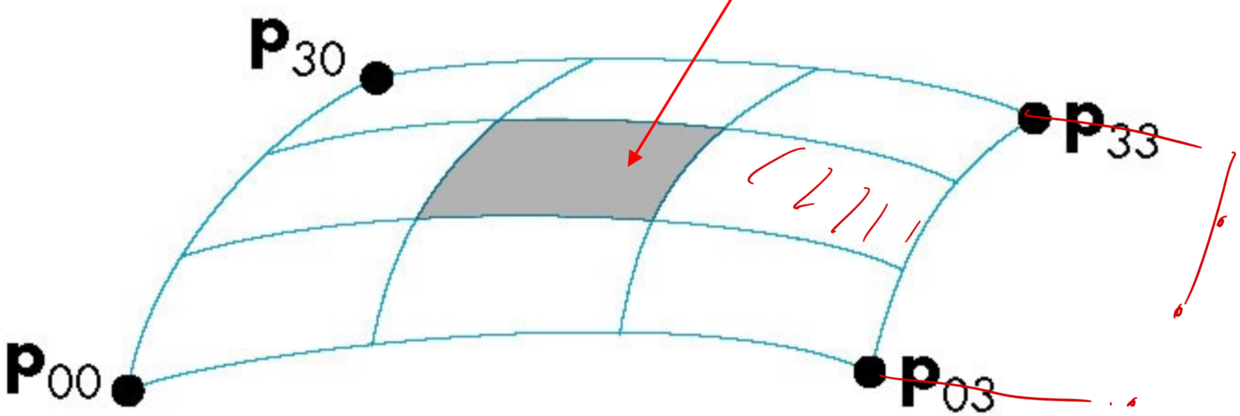
B-Spline Patches

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(v)\, p_{ij} = \underbrace{u^T \mathbf{M}_S \mathbf{P} \mathbf{M}_S^T v}$$

Matrix

defined over only 1/9 of region

⇒ No change.



결국 4개를 가지고, 한 면만.
결정가능.

Splines and Basis

p(u). p_0 p_1 p_2 p_3

- If we examine the cubic B-spline from the perspective of each control (data) point, each interior point contributes (through the blending functions) to four segments
- We can rewrite p(u) in terms of the data points as

→ Basis function.. 각 점에 곱하고 더함

$$p(u) = \sum B_i(u)\, p_i$$

p_0 p_1 p_2

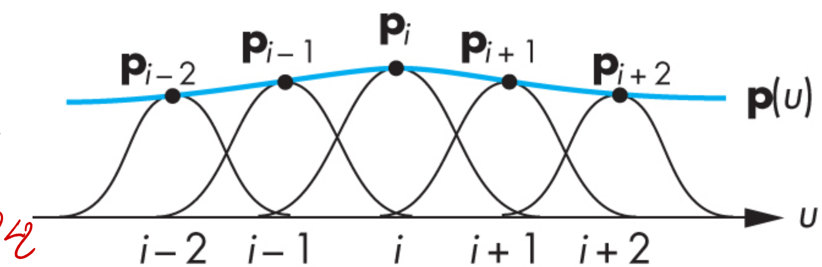
defining the basis functions $\{B_i(u)\}$

Basis Functions

특정 점이 변화할때 영향을 미치는 곡선의 범위.

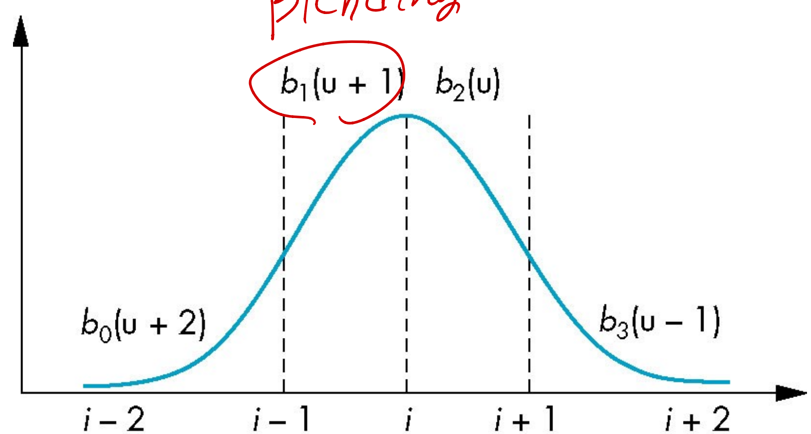
In terms of the blending polynomials,
Total contribution $B_i(u)p_i$ of p_i is given by

이 점이 곡선 전체에 미치는 영향.



$$
B_i(u) = \begin{cases} 0 & u < i-2 \\ b_0(u+2) & i-2 \le u < i-1 \\ b_1(u+1) & i-1 \le u < i \\ b_2(u) & i \le u < i+1 \\ b_3(u-1) & i+1 \le u < i+2 \\ 0 & u \ge i+2 \end{cases}
$$

Blending



Generalizing Splines

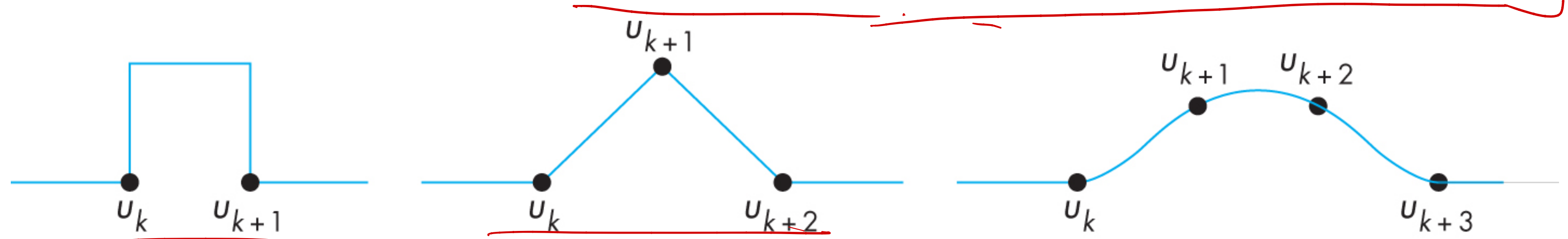
- Generalize from cubic to any degree
- Generalize to different basis function
  - Cox-deBoor recursion

$$p(u) = \sum_{i=0}^{n} B_{i,d}(u) P_i$$

$$B_{k,0}(u) = \begin{cases} 1, & \text{if } u_k \le u \le u_{k+1} \\ 0, & \text{otherwise} \end{cases}$$

Basis의 재귀적 정의.

$$\boxed{B_{k,d}(u) = \frac{u - u_k}{u_{k+d} - u_k} B_{k,d-1}(u) + \frac{u_{k+d+1} - u}{u_{k+d+1} - u_{k+1}} B_{k+1,d-1}(u)}$$



box func

hat func.

order↑.

Quadratic B-spline

d=2, n=4

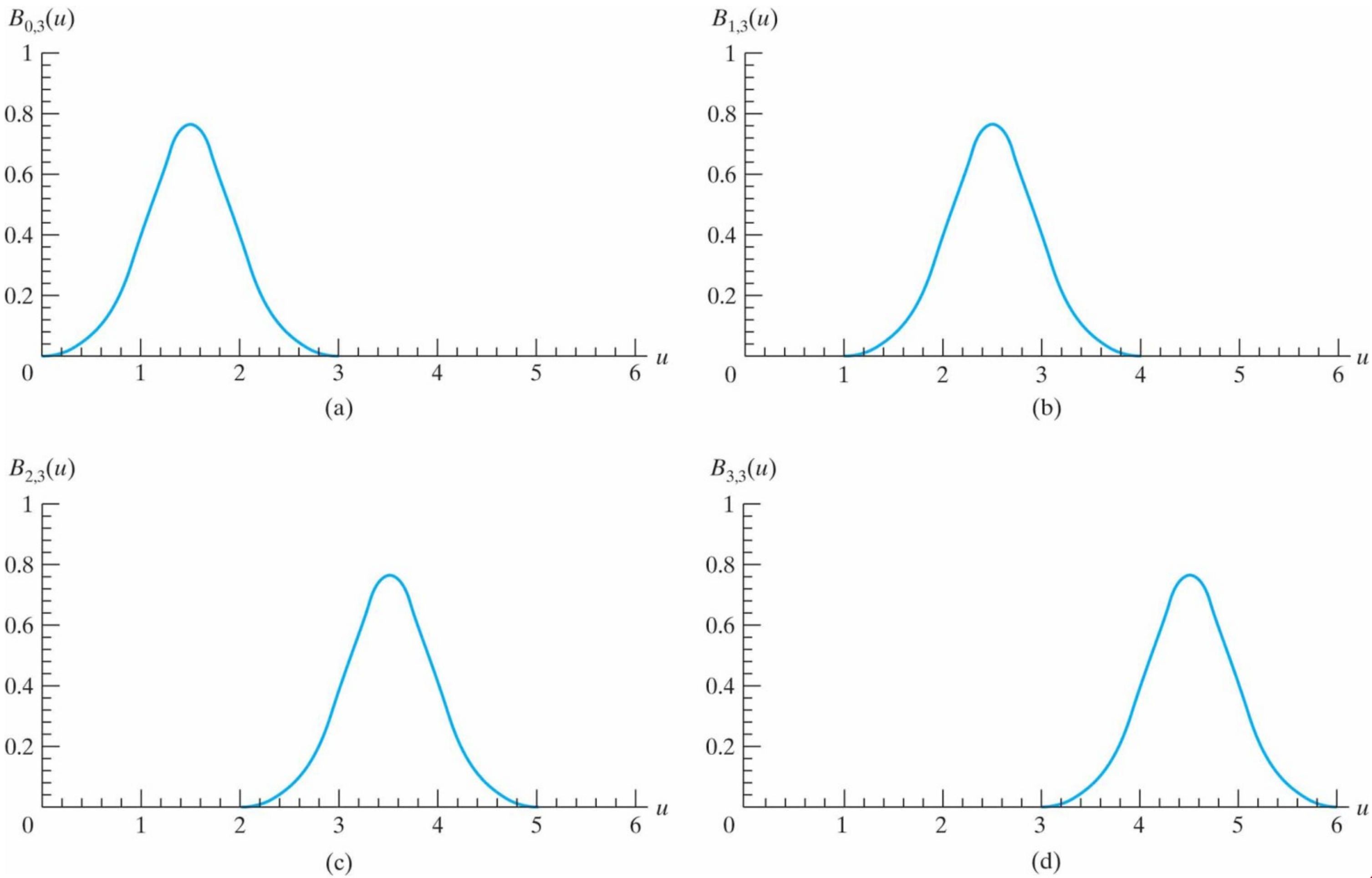
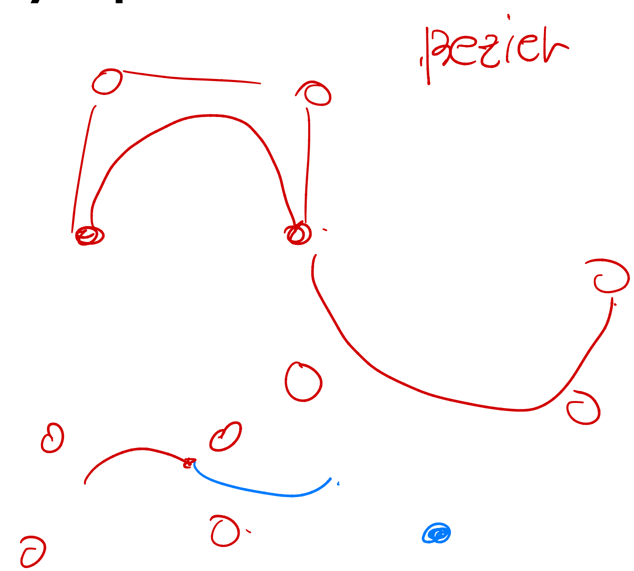


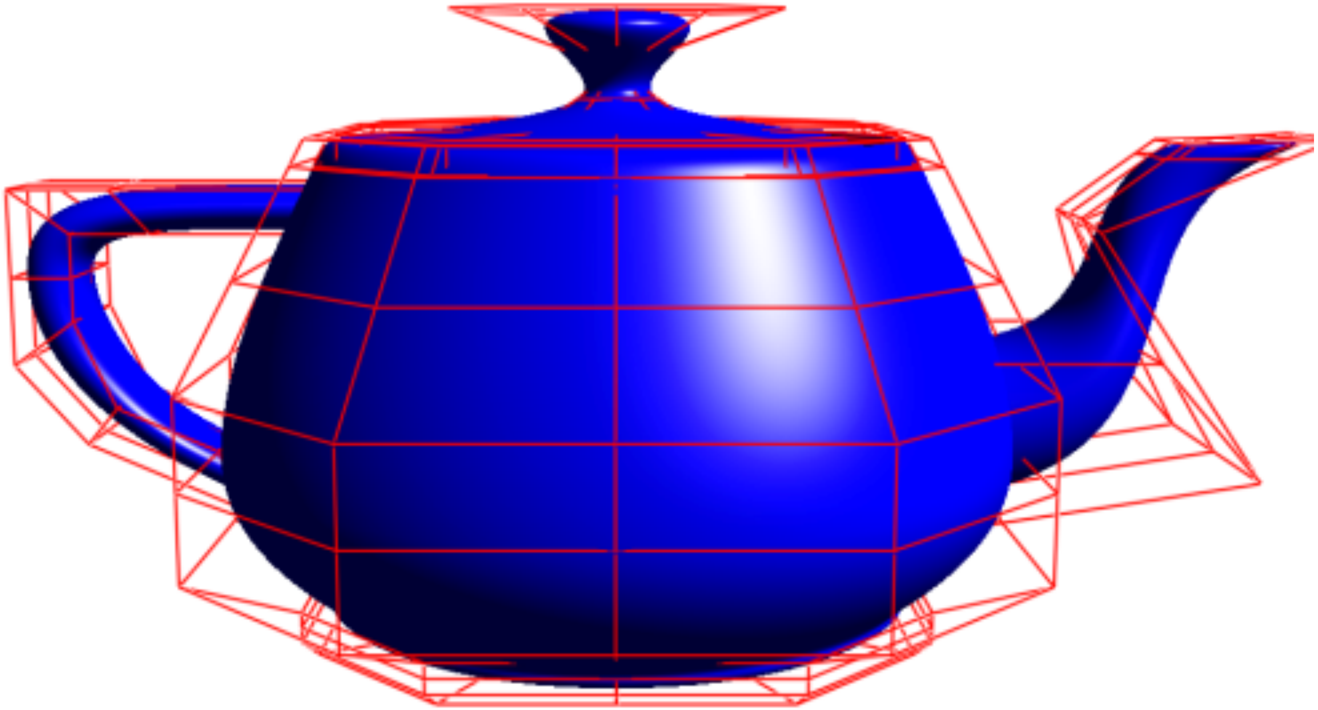
Image courtesy of Hearn

Cubic B-Spline Summary

- Expensive than Bezier to evaluate
- Smoother at joint point (C^2)
- Local control
  - Compact support defined by spline basis
- Easy to add points
  - Degree does not increase
- Convex hull property



Questions?



Bezier surface rendering of Utah teapot