학부3학년 교과목 "전산선박설계(Computer Aided ship design)" 강의 교재

Naval Architecture & Ocean Engineering

Computer Aided Ship design

-Part II. Hull Form Modeling-

September, 2009

Prof. Kyu-Yeul Lee

Department of Naval Architecture and Ocean Engineering,
Seoul National University of College of Engineering

Seoul National Univ.

@SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

Naval Architecture & Ocean Engineering

Ch 3. 곡면(Surfaces)

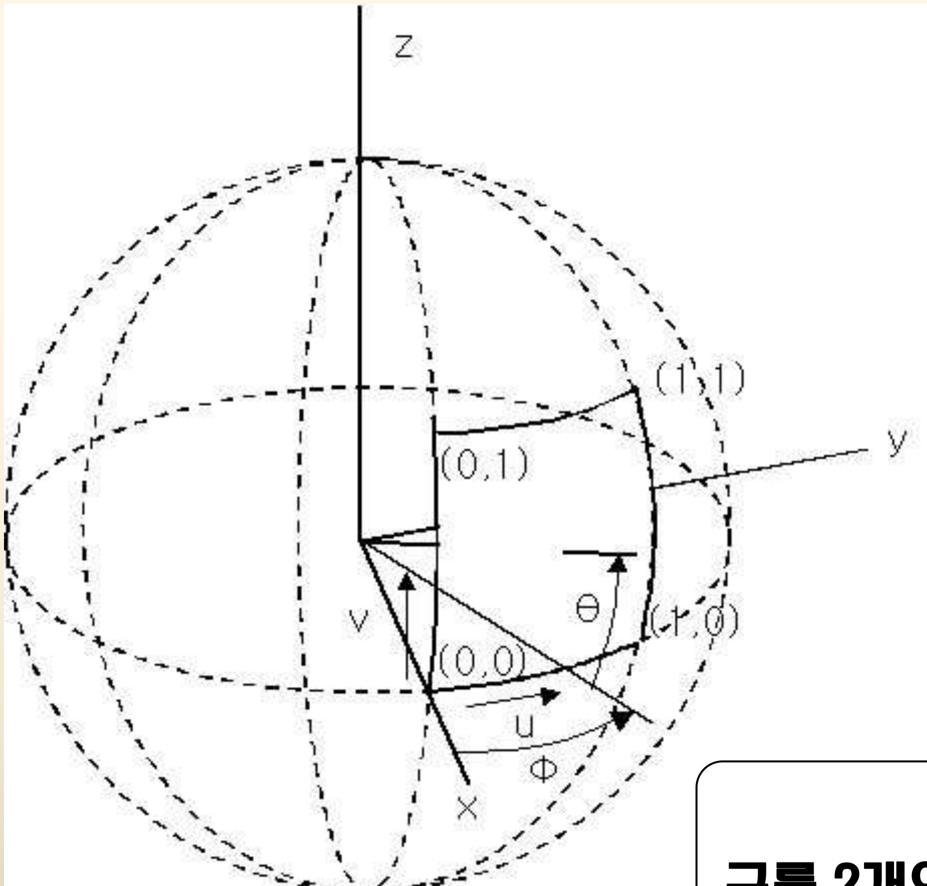
Seoul National Univ.

@SDAL Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.1 Parametric Surfaces

Seoul National Univ.
@SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.1 Parametric Surfaces



	곡면
양함수	$z = \pm\sqrt{d^2 - x^2 - y^2}$
음함수	$x^2 + y^2 + z^2 = d^2$
매개변수	$x = d\cos\phi\cos\theta$ $y = d\sin\phi\cos\theta$ $z = d\sin\theta$
	$\mathbf{r} = r(x(\phi,\theta), y(\phi,\theta), z(\phi,\theta))$

구를 2개의 매개변수 $(\phi,\ \theta)$ 로 표현하면 간단히 수식화 할 수 있다.

3.2 Bezier surfaces

3.2.1 Generation of Bezier surfaces by de Casteljau algorithm

3.2.1 Generation of Bezier surfaces by de Casteljau algorithm

3.2.1.1 Bi-linear Bezier Surface Patch

3.2.1.2 Bi-quadratic Bezier Surface Patch

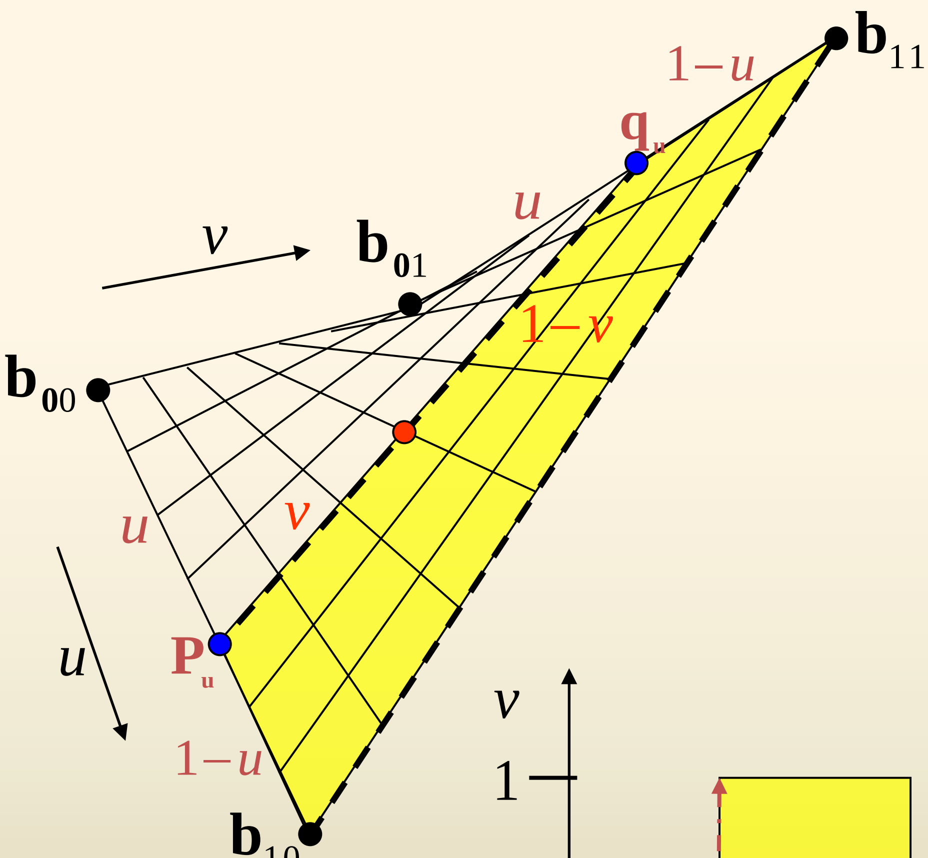
3.2.1.3 Bi-Cubic Bezier Surface Patch

Seoul National Univ.
@SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.2.1.1 Bi-linear Bezier Surface Patch

- Given: 2x2 Bezier control point
- Find: Points on bi-linear Bezier Surface Patch

방법: u, v 방향으로 'de Casteljau algorithm' 사용



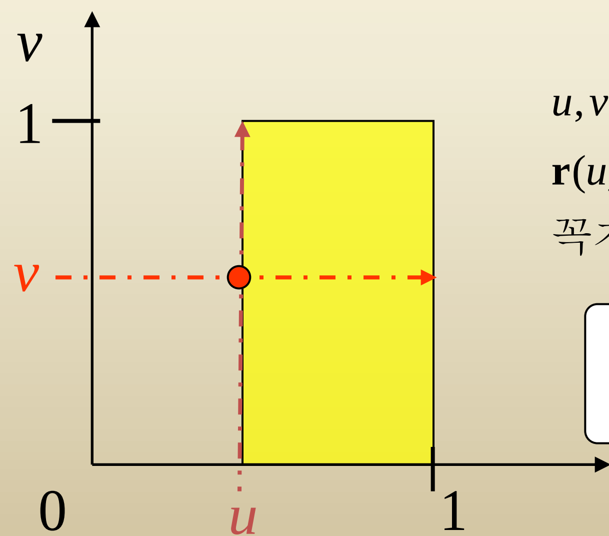
$$\mathbf{P}_u = (1-u)\mathbf{b}_{00} + u\,\mathbf{b}_{10}$$

$$\mathbf{q}_u = (1-u)\mathbf{b}_{01} + u\,\mathbf{b}_{11}$$

$$\mathbf{r}(u,v) = (1-v)\,\mathbf{P}_u + v\,\mathbf{q}_u$$

$$\mathbf{r}(u,v) = (1-v)(1-u)\mathbf{b}_{00} + (1-v)\,u\,\mathbf{b}_{10} + v\,(1-u)\mathbf{b}_{01} + v\,u\,\mathbf{b}_{11}$$

$$r(u,v) = \begin{bmatrix}(1-v) & v\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} \\ \mathbf{b}_{01} & \mathbf{b}_{11}\end{bmatrix}\begin{bmatrix}(1-u) \\ u\end{bmatrix}$$



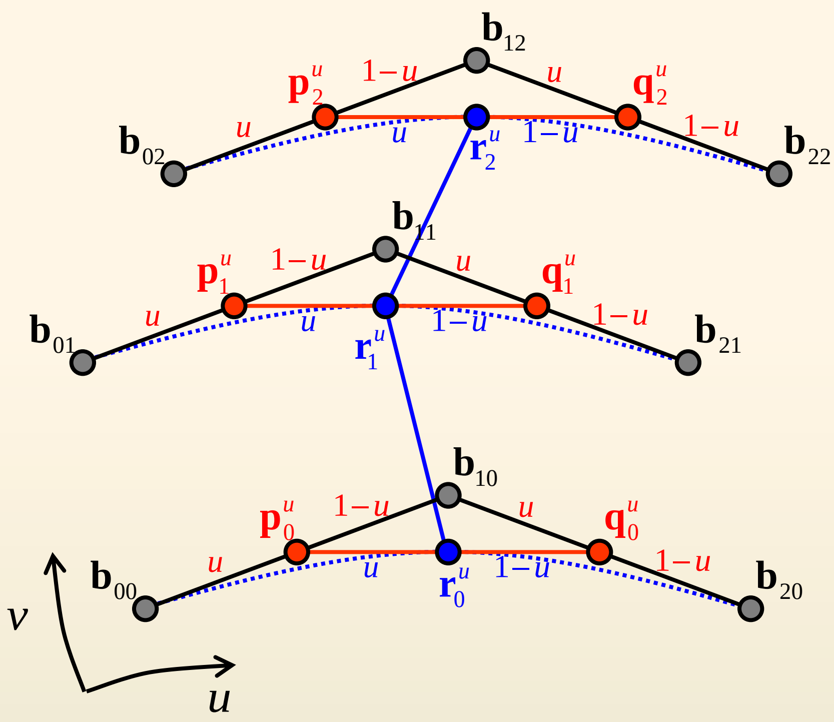
u,v를 0에서 1까지 증가하며 $\mathbf{r}(u,v)$를 계산하면 점 $\mathbf{b}_{00},\mathbf{b}_{10},\mathbf{b}_{01},\mathbf{b}_{11}$ 을 꼭지점으로 하는 곡면을 얻을 수 있다.

2x2개의 Bezier 조정점을 이용하여 Bi-linear Interpolation 으로 곡면식을 구할 수 있다.

3.2.1.2 Bi-quadratic Bezier Surface Patch

- Given: 3x3 Bezier control point
- Find: Points on bi-quadratic Bezier Surface Patch

방법: u, v 방향으로 'de Casteljau algorithm' 사용



$$\mathbf{p}_0^u = (1-u)\mathbf{b}_{00} + u\mathbf{b}_{10}$$
$$\mathbf{q}_0^u = (1-u)\mathbf{b}_{10} + u\mathbf{b}_{20}$$
$$\mathbf{p}_1^u = (1-u)\mathbf{b}_{01} + u\mathbf{b}_{11}$$
$$\mathbf{q}_1^u = (1-u)\mathbf{b}_{11} + u\mathbf{b}_{21}$$
$$\mathbf{p}_2^u = (1-u)\mathbf{b}_{02} + u\mathbf{b}_{12}$$
$$\mathbf{q}_2^u = (1-u)\mathbf{b}_{12} + u\mathbf{b}_{22}$$
$$\mathbf{r}_0^u = (1-u)\mathbf{p}_0^u + u\mathbf{q}_0^u$$
$$\mathbf{r}_1^u = (1-u)\mathbf{p}_1^u + u\mathbf{q}_1^u$$
$$\mathbf{r}_2^u = (1-u)\mathbf{p}_2^u + u\mathbf{q}_2^u$$

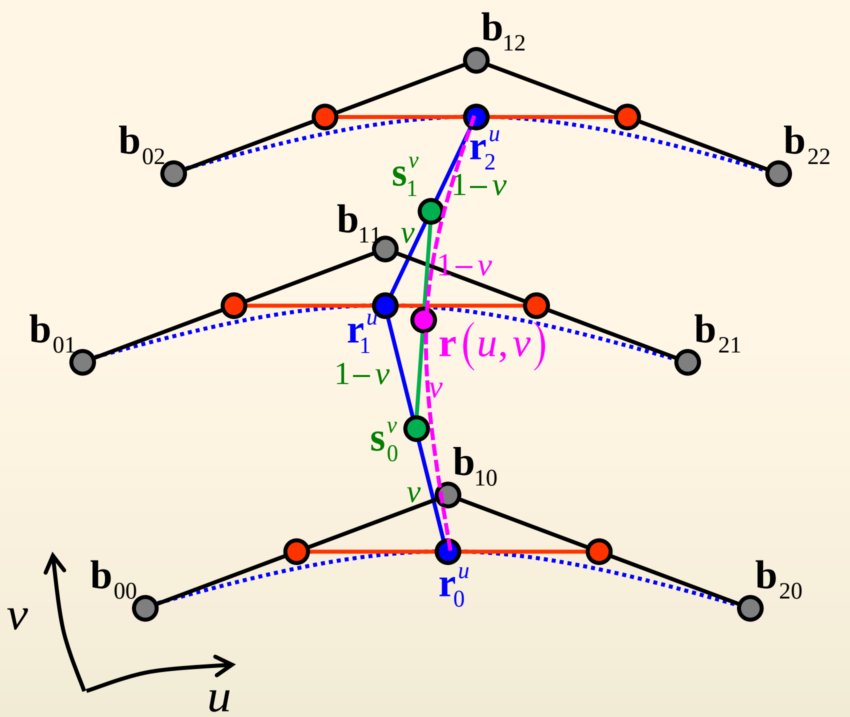
$$\begin{bmatrix} \mathbf{r}_0^u \\ \mathbf{r}_1^u \\ \mathbf{r}_2^u \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} (1-u)^2 \\ 2u(1-u) \\ u^2 \end{bmatrix}$$

$$\mathbf{r}_0^u = (1-u)^2\mathbf{b}_{00} + 2u(1-u)\mathbf{b}_{10} + u^2\mathbf{b}_{20}$$
$$\mathbf{r}_1^u = (1-u)^2\mathbf{b}_{01} + 2u(1-u)\mathbf{b}_{11} + u^2\mathbf{b}_{21}$$
$$\mathbf{r}_2^u = (1-u)^2\mathbf{b}_{02} + 2u(1-u)\mathbf{b}_{12} + u^2\mathbf{b}_{22}$$

3.2.1.2 Bi-quadratic Bezier Surface Patch

- Given: 3x3 Bezier control point
- Find: Points on bi-quadratic Bezier Surface Patch

방법: u, v 방향으로 'de Casteljau algorithm' 사용



$$\mathbf{s}_0^v = (1-v)\mathbf{r}_0^u + v\mathbf{r}_1^u$$

$$\mathbf{s}_1^v = (1-v)\mathbf{r}_1^u + v\mathbf{r}_2^u$$

$$\mathbf{r}(u,v) = (1-v)\mathbf{s}_0^v + v\mathbf{s}_1^v$$

$$\mathbf{r}(u,v) = (1-v)^2\mathbf{r}_0^u + 2v(1-v)\mathbf{r}_1^u + v^2\mathbf{r}_2^u$$

$$\mathbf{r}(u,v) = \begin{bmatrix}(1-v)^2 & 2v(1-v) & v^2\end{bmatrix}\begin{bmatrix}\mathbf{r}_0^u \\ \mathbf{r}_1^u \\ \mathbf{r}_2^u\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{r}_0^u \\ \mathbf{r}_1^u \\ \mathbf{r}_2^u\end{bmatrix} = \begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}(1-u)^2 \\ 2u(1-u) \\ u^2\end{bmatrix}$$

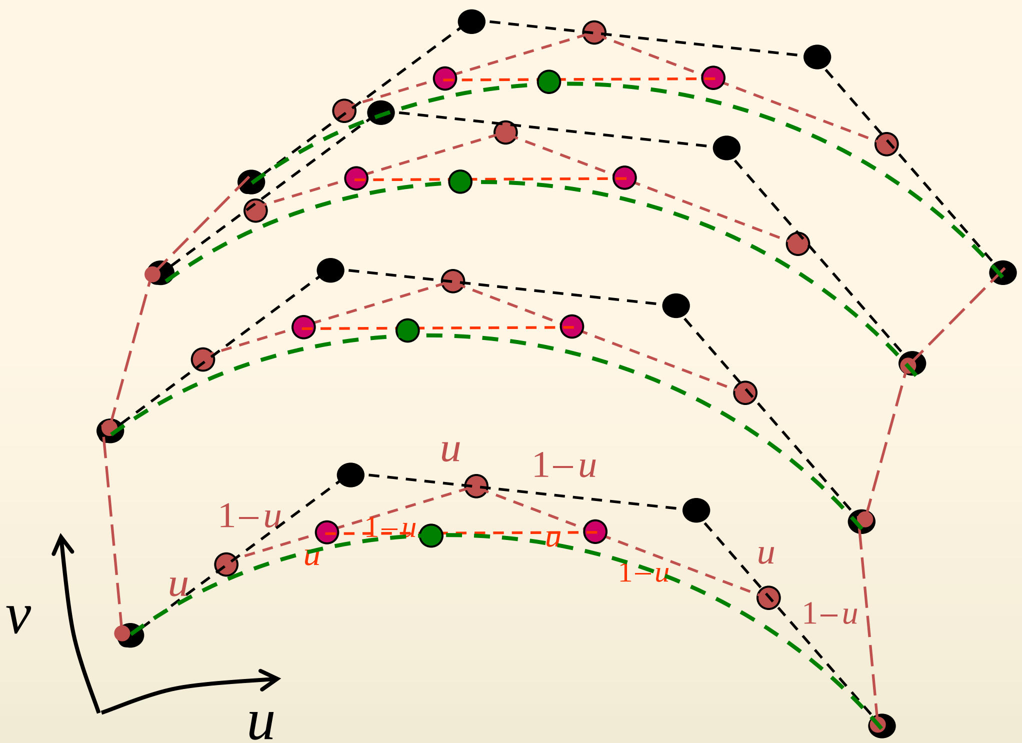
$$\mathbf{r}(u,v) = \begin{bmatrix}(1-v)^2 & 2v(1-v) & v^2\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}(1-u)^2 \\ 2u(1-u) \\ u^2\end{bmatrix}$$

3x3개의 조정점을 이용하여
Bi-Quadratic Bezier Patch를 구할 수 있다.

3.2.1.3 Bi-cubic Bezier Surface Patch

- Given: 4x4 Bezier control point
- Find: Points on bi-cubic Bezier Surface Patch

방법: u, v 방향으로 'de Casteljau algorithm' 사용



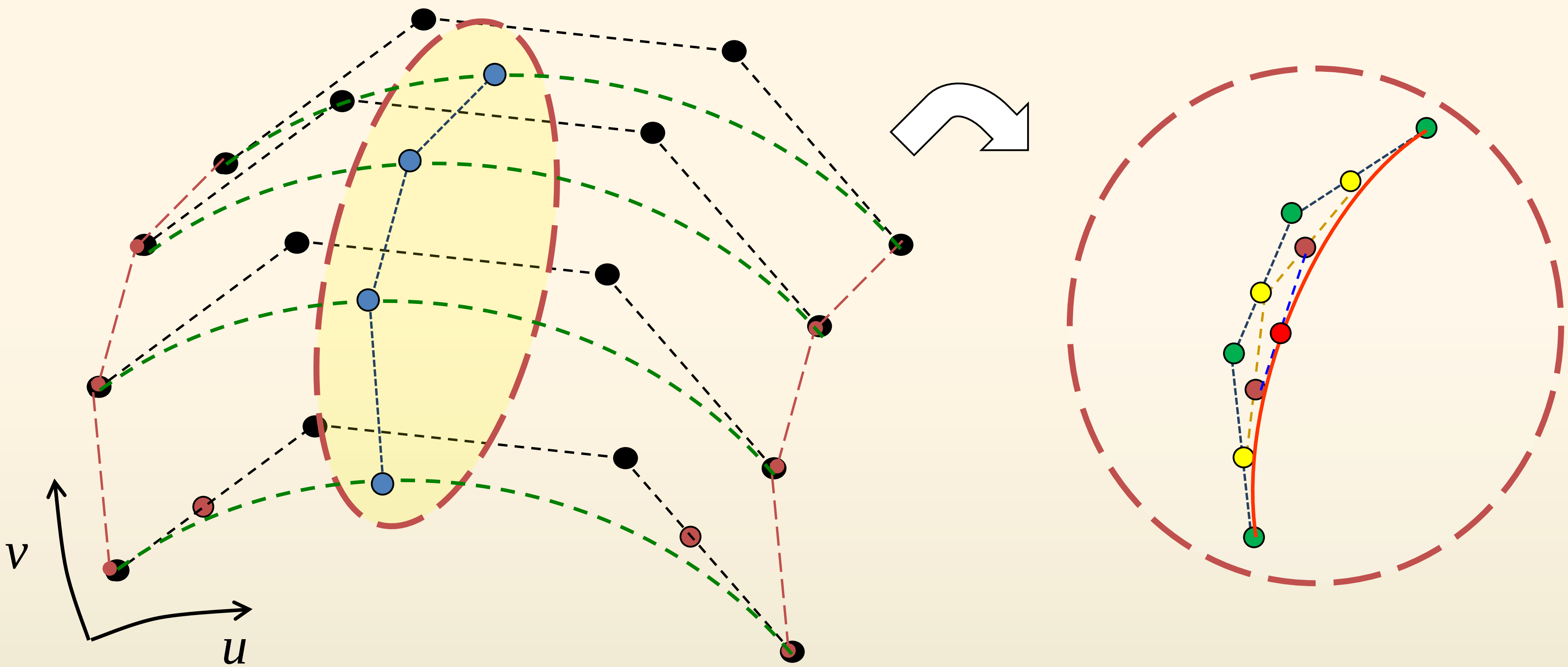
$$\mathbf{b}(u,v)=\begin{bmatrix} B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u) \end{bmatrix}\begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{1,0} & \mathbf{b}_{2,0} & \mathbf{b}_{3,0} \\ \mathbf{b}_{0,1} & \mathbf{b}_{1,1} & \mathbf{b}_{2,1} & \mathbf{b}_{3,1} \\ \mathbf{b}_{0,2} & \mathbf{b}_{1,2} & \mathbf{b}_{2,2} & \mathbf{b}_{3,2} \\ \mathbf{b}_{0,3} & \mathbf{b}_{1,3} & \mathbf{b}_{2,3} & \mathbf{b}_{3,3} \end{bmatrix}\begin{bmatrix} B_0^3(v) \\ B_1^3(v) \\ B_2^3(v) \\ B_3^3(v) \end{bmatrix}$$

4x4개의 조정점을 이용하여 Bi-Cubic Bezier Patch를 구할 수 있다.

3.2.1.3 Bi-cubic Bezier Surface Patch

- Given: 4x4 Bezier control point
- Find: Points on bi-cubic Bezier Surface Patch

방법: u, v 방향으로 'de Casteljau algorithm' 사용



$$\mathbf{b}(u,v)=\begin{bmatrix} B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u) \end{bmatrix}\begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{1,0} & \mathbf{b}_{2,0} & \mathbf{b}_{3,0} \\ \mathbf{b}_{0,1} & \mathbf{b}_{1,1} & \mathbf{b}_{2,1} & \mathbf{b}_{3,1} \\ \mathbf{b}_{0,2} & \mathbf{b}_{1,2} & \mathbf{b}_{2,2} & \mathbf{b}_{3,2} \\ \mathbf{b}_{0,3} & \mathbf{b}_{1,3} & \mathbf{b}_{2,3} & \mathbf{b}_{3,3} \end{bmatrix}\begin{bmatrix} B_0^3(v) \\ B_1^3(v) \\ B_2^3(v) \\ B_3^3(v) \end{bmatrix}$$

4x4개의 조정점을 이용하여 Bi-Cubic Bezier Patch를 구할 수 있다.

3.2.2 Generation of Bezier surfaces by tensor-product approach

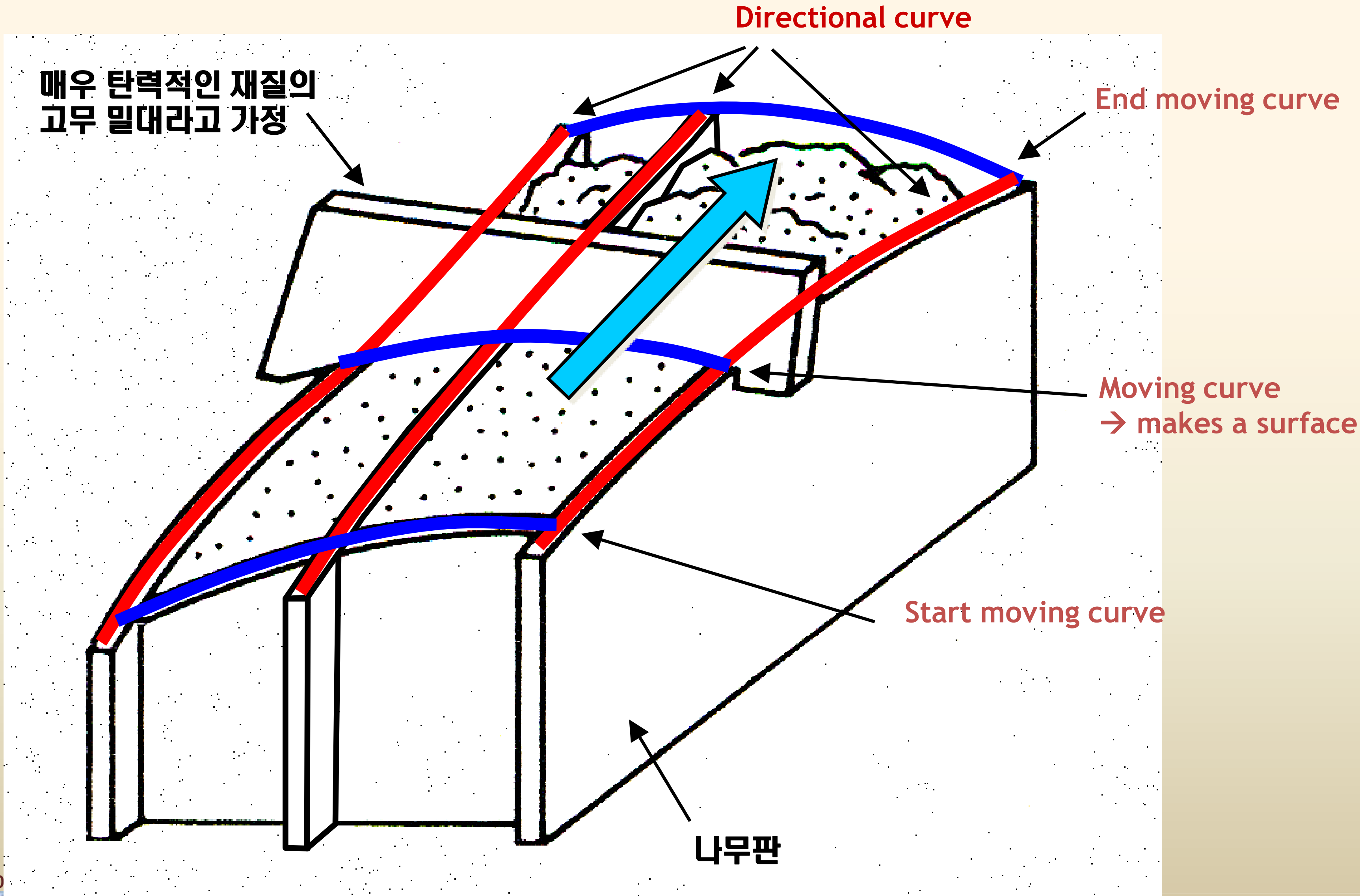
3.2.2.1 Tensor-product approach

3.2.2.2 Tensor-product biquadratic Bezier surface

3.2.2.3 Tensor-product bicubic Bezier surface

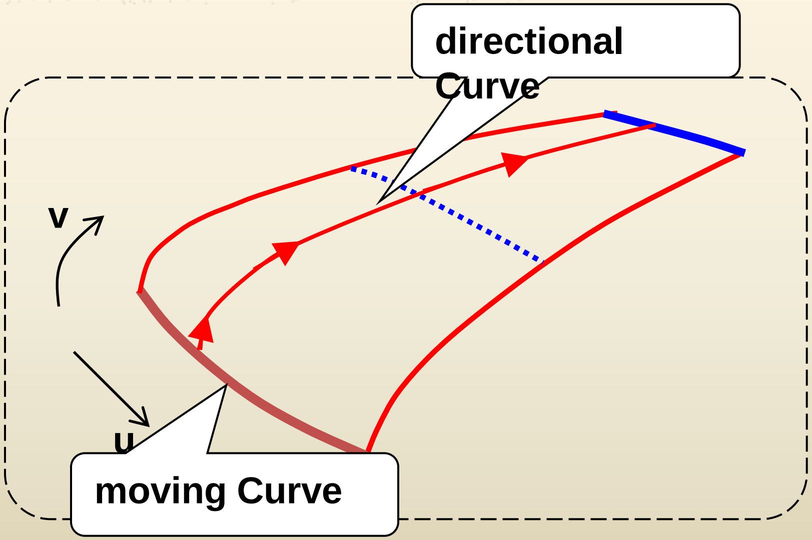
Seoul National Univ.
@SDAL Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.4.1 Tensor product approach (1)

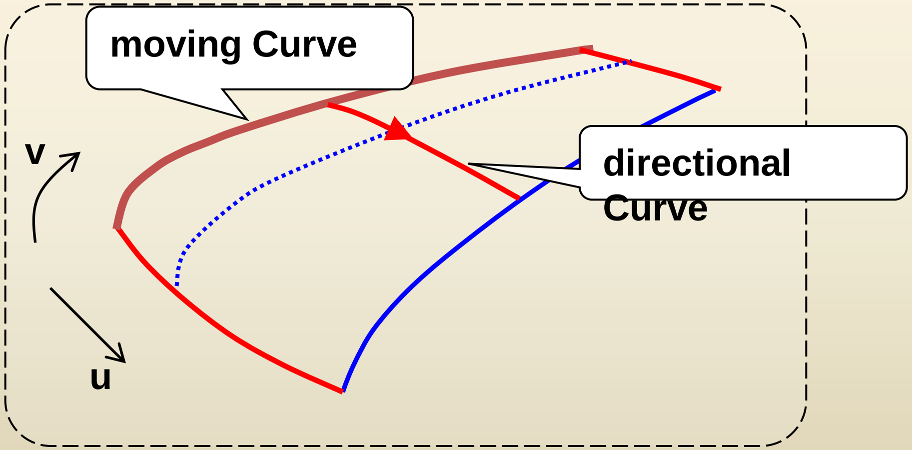


3.4.1 Tensor product approach (2)

- moving curve가 일정한 차수의 Bezier curve 이고, moving curve의Bezier control points의 궤적을 나타내는 directional curve도 Bezier curve일 때, 이러한 방법으로 생성되는 곡면을 "Tensor product Bezier surface" 라고 한다.



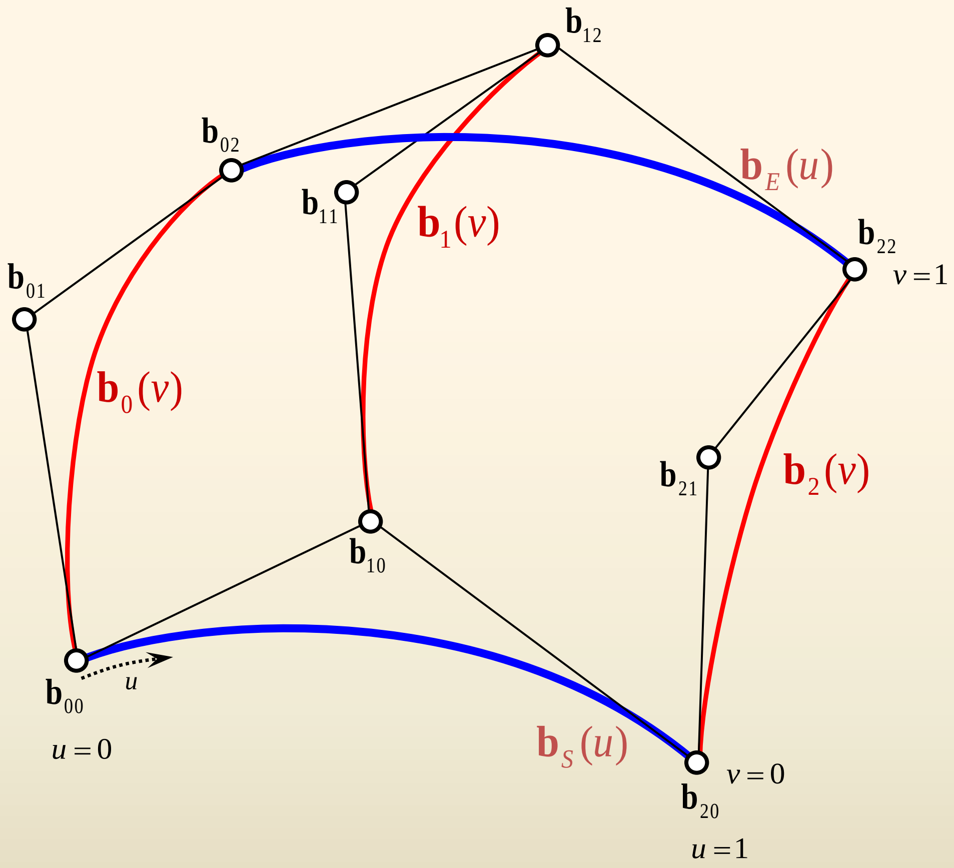
r(*u*) 곡선을 *v* 방향으로 sweeping



r(*v*) 곡선을 *u* 방향으로 sweeping

3.4.2 Tensor-product bi-quadratic Bezier surface (1)

- Given: Control Points of bi-quadratic Bezier Surface
- Find: Points on bi-quadratic Bezier Surface



☑ Given 3x3 Points $\mathbf{b}_{ij}$,

☑ Generate start/end moving curves and directional curves in quadratic Bezier form

$$\mathbf{b}_E(u) = \mathbf{b}_{02}B_0^2(u) + \mathbf{b}_{12}B_1^2(u) + \mathbf{b}_{22}B_2^2(u)$$

$$\mathbf{b}_S(u) = \mathbf{b}_{00}B_0^2(u) + \mathbf{b}_{10}B_1^2(u) + \mathbf{b}_{20}B_2^2(u)$$

$$\mathbf{b}_0(v) = \mathbf{b}_{00}B_0^2(v) + \mathbf{b}_{01}B_1^2(v) + \mathbf{b}_{02}B_2^2(v)$$

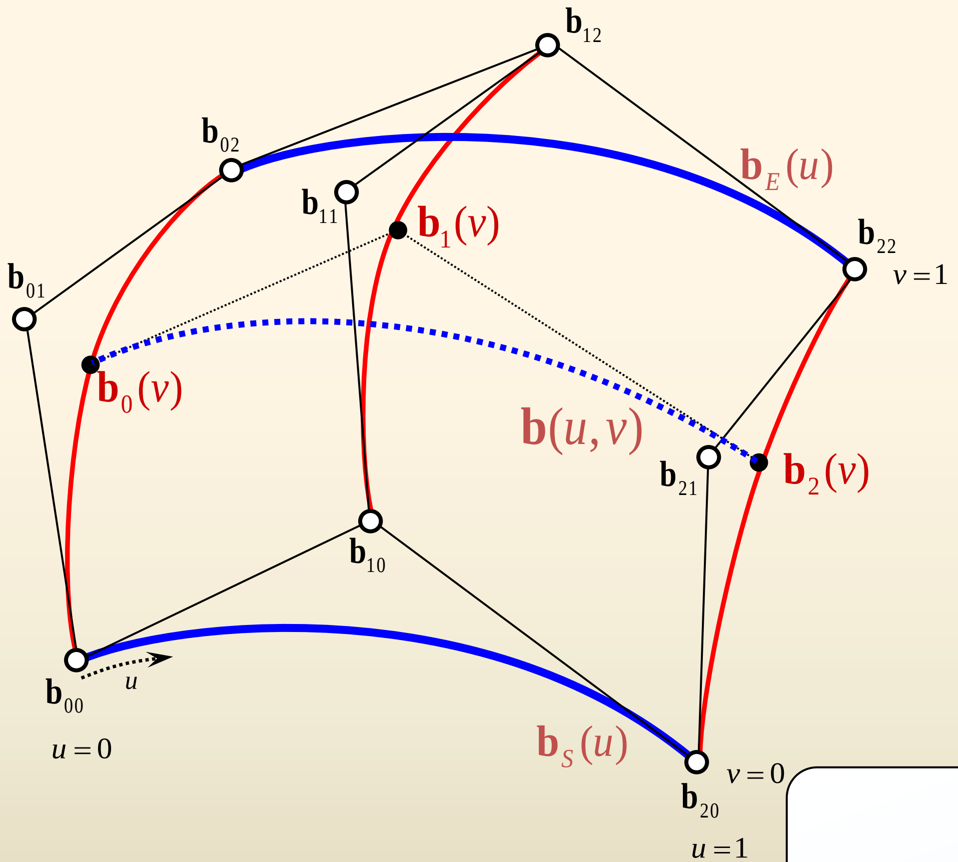
$$\mathbf{b}_1(v) = \mathbf{b}_{10}B_0^2(v) + \mathbf{b}_{11}B_1^2(v) + \mathbf{b}_{12}B_2^2(v)$$

$$\mathbf{b}_2(v) = \mathbf{b}_{20}B_0^2(v) + \mathbf{b}_{21}B_1^2(v) + \mathbf{b}_{22}B_2^2(v)$$

$$\begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} B_0^2(v) \\ B_1^2(v) \\ B_2^2(v) \end{bmatrix}$$

3.4.2 Tensor-product bi-quadratic Bezier surface (2)

- Given: Control Points of bi-quadratic Bezier Surface
- Find: Points on bi-quadratic Bezier Surface



☑ Given 3x3 Points $\mathbf{b}_{ij}$,

☑ Moving curve can be represented in the following form:

$$\mathbf{b}(u,v)=\mathbf{b}_0(v)B_0^2(u)+\mathbf{b}_1(v)B_1^2(u)+\mathbf{b}_2(v)B_2^2(u)$$

$$=\begin{bmatrix}B_0^2(u) & B_1^2(u) & B_2^2(u)\end{bmatrix}\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\end{bmatrix}=\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}B_0^2(v)\\ B_1^2(v)\\ B_2^2(v)\end{bmatrix}$$

$$\mathbf{b}(u,v)=\begin{bmatrix}B_0^2(u) & B_1^2(u) & B_2^2(u)\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}B_0^2(v)\\ B_1^2(v)\\ B_2^2(v)\end{bmatrix}$$

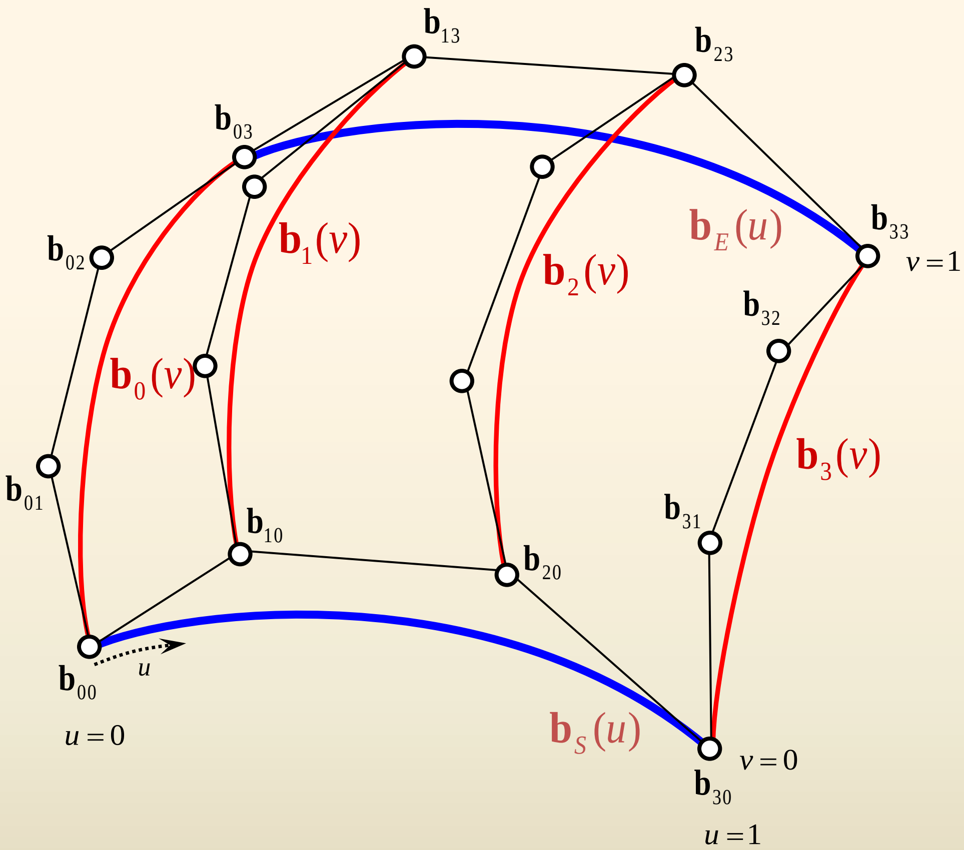
$$=\sum_{j=0}^{2}\sum_{i=0}^{2}\mathbf{b}_{ij}B_i^2(u)B_j^2(v)$$

Bezier surface control points

3.4.2 Tensor-product bi-cubic Bezier surface (1)

- Given: Control Points of bi-cubic Bezier Surface

- Find: Points on bi-cubic Bezier Surface



☑ **Given 4x4 Points $\mathbf{b}_{ij}$,**

☑ **Generate start/end moving curves and directional curves in cubic Bezier form**

$$\mathbf{b}_E(u) = \mathbf{b}_{03}B_0^3(u) + \mathbf{b}_{13}B_1^3(u) + \mathbf{b}_{23}B_2^3(u) + \mathbf{b}_{33}B_3^3(u)$$

$$\mathbf{b}_S(u) = \mathbf{b}_{00}B_0^3(u) + \mathbf{b}_{10}B_1^3(u) + \mathbf{b}_{20}B_2^3(u) + \mathbf{b}_{30}B_3^3(u)$$

$$\mathbf{b}_0(v) = \mathbf{b}_{00}B_0^3(v) + \mathbf{b}_{01}B_1^3(v) + \mathbf{b}_{02}B_2^3(v) + \mathbf{b}_{03}B_3^3(v)$$

$$\mathbf{b}_1(v) = \mathbf{b}_{10}B_0^3(v) + \mathbf{b}_{11}B_1^3(v) + \mathbf{b}_{12}B_2^3(v) + \mathbf{b}_{13}B_3^3(v)$$

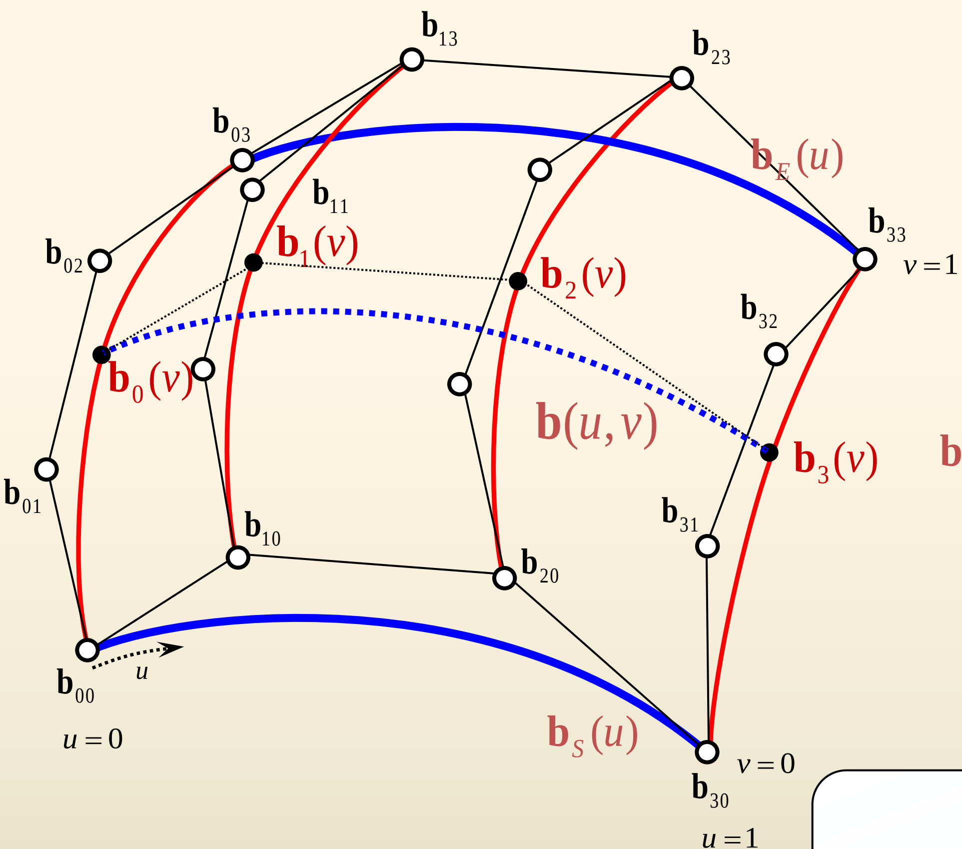
$$\mathbf{b}_2(v) = \mathbf{b}_{20}B_0^3(v) + \mathbf{b}_{21}B_1^3(v) + \mathbf{b}_{22}B_2^3(v) + \mathbf{b}_{23}B_3^3(v)$$

$$\mathbf{b}_3(v) = \mathbf{b}_{30}B_0^3(v) + \mathbf{b}_{31}B_1^3(v) + \mathbf{b}_{32}B_2^3(v) + \mathbf{b}_{33}B_3^3(v)$$

$$\begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \\ \mathbf{b}_3(v) \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23} \\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33} \end{bmatrix} \begin{bmatrix} B_0^3(v) \\ B_1^3(v) \\ B_2^3(v) \\ B_3^3(v) \end{bmatrix}$$

3.4.2 Tensor-product bi-cubic Bezier surface (2)

- Given: Control Points of bi-cubic Bezier Surface
- Find: Points on bi-cubic Bezier Surface



☑ Given 4x4 Points $\mathbf{b}_{ij}$,

☑ Moving curve can be represented in the following form:

$$\mathbf{b}(u,v) = \mathbf{b}_0(v)B_0^3(u) + \mathbf{b}_1(v)B_1^3(u) + \mathbf{b}_2(v)B_2^3(u) + \mathbf{b}_3(v)B_3^3(u)$$

$$= \begin{bmatrix} B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \\ \mathbf{b}_3(v) \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \\ \mathbf{b}_3(v) \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23} \\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33} \end{bmatrix} \begin{bmatrix} B_0^3(v) \\ B_1^3(v) \\ B_2^3(v) \\ B_3^3(v) \end{bmatrix}$$

$$\mathbf{b}(u,v) = \begin{bmatrix} B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23} \\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33} \end{bmatrix} \begin{bmatrix} B_0^3(v) \\ B_1^3(v) \\ B_2^3(v) \\ B_3^3(v) \end{bmatrix}$$

$$= \sum_{j=0}^{3} \sum_{i=0}^{3} \mathbf{b}_{ij} B_i^3(u) B_j^3(v)$$

3.3 B-spline surfaces

3.3.1 Generation of B-spline surfaces by tensor-product approach

3.3.2 B-spline surface Interpolation

3.3.1 Generation of B-spline surfaces by tensor-product approach

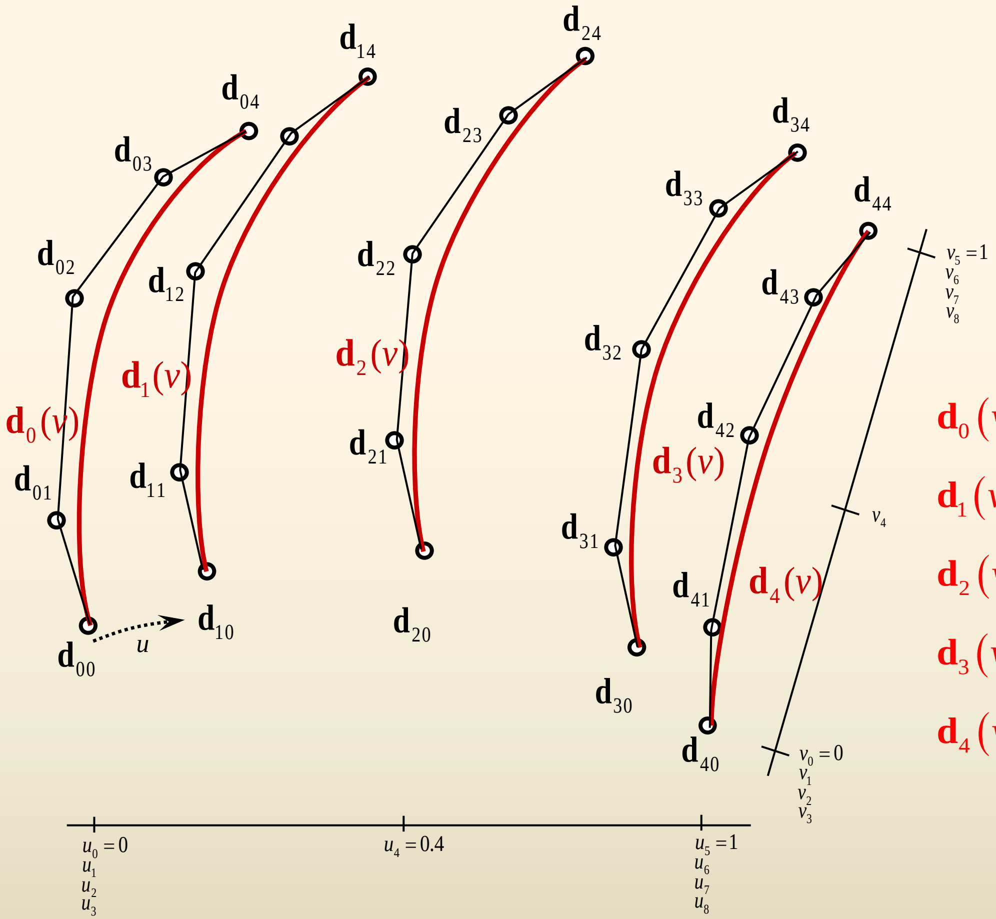
3.3.1.1 Tensor-product bicubic B-spline surface

3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface

Seoul National Univ.
@SDAL
Advanced Ship Design Automation Lab.
http://asdal.snu.ac.kr

3.3.1.1 Tensor-product bicubic B-spline surface (1)

- Given: Control Points of bicubic B-spline surface
- Find: Points on bicubic B-spline surface



☑ **Given 5x5 Control Points $\mathbf{d}_{ij}$, u-knots, v-knots, u-degree(=3), v-degree(=3),**

☑ **Generate start/end moving curves and directional curves in cubic B-spline form:**

$$\mathbf{d}_0(v) = \mathbf{d}_{00}N_0^3(v) + \mathbf{d}_{01}N_1^3(v) + \mathbf{d}_{02}N_2^3(v) + \mathbf{d}_{03}N_3^3(v) + \mathbf{d}_{04}N_4^3(v)$$

$$\mathbf{d}_1(v) = \mathbf{d}_{10}N_0^3(v) + \mathbf{d}_{11}N_1^3(v) + \mathbf{d}_{12}N_2^3(v) + \mathbf{d}_{13}N_3^3(v) + \mathbf{d}_{14}N_4^3(v)$$

$$\mathbf{d}_2(v) = \mathbf{d}_{20}N_0^3(v) + \mathbf{d}_{21}N_1^3(v) + \mathbf{d}_{22}N_2^3(v) + \mathbf{d}_{23}N_3^3(v) + \mathbf{d}_{24}N_4^3(v)$$

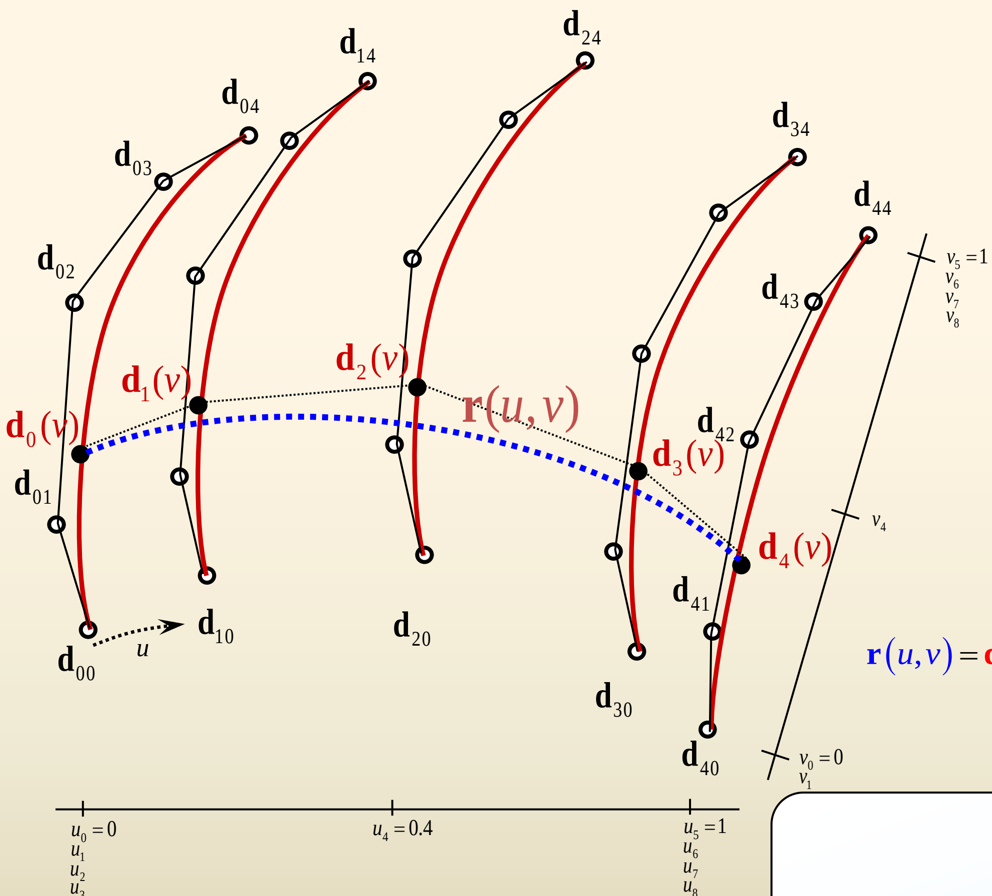
$$\mathbf{d}_3(v) = \mathbf{d}_{30}N_0^3(v) + \mathbf{d}_{31}N_1^3(v) + \mathbf{d}_{32}N_2^3(v) + \mathbf{d}_{33}N_3^3(v) + \mathbf{d}_{34}N_4^3(v)$$

$$\mathbf{d}_4(v) = \mathbf{d}_{40}N_0^3(v) + \mathbf{d}_{41}N_1^3(v) + \mathbf{d}_{42}N_2^3(v) + \mathbf{d}_{43}N_3^3(v) + \mathbf{d}_{44}N_4^3(v)$$

$$\begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

3.3.1.1 Tensor-product bicubic B-spline surface (1)

- Given: Control Points of bicubic B-spline surface
- Find: Points on bicubic B-spline surface



☑ **Given 5x5 Control Points $\mathbf{d}_{ij}$, u-knots, v-knots, u-degree(=3), v-degree(=3),**

☑ **Moving curve can be represented in the following form:**

$$= \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix}$$

$$\mathbf{r}(u,v) = \mathbf{d}_0(v) N_0^3(u) + \mathbf{d}_1(v) N_1^3(u) + \mathbf{d}_2(v) N_2^3(u) + \mathbf{d}_3(v) N_3^3(u) + \mathbf{d}_4(v) N_4^3(u)$$

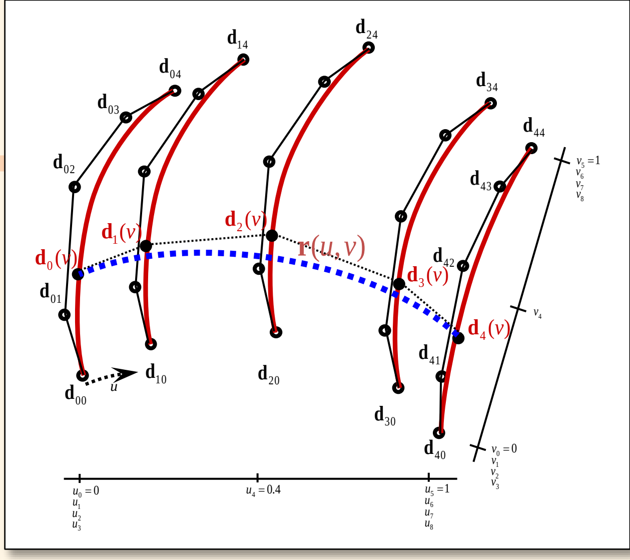
$$\begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

$$= \sum_{j=0}^{5} \sum_{i=0}^{5} \mathbf{d}_{ij} N_i^3(u) N_j^3(v)$$

3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface

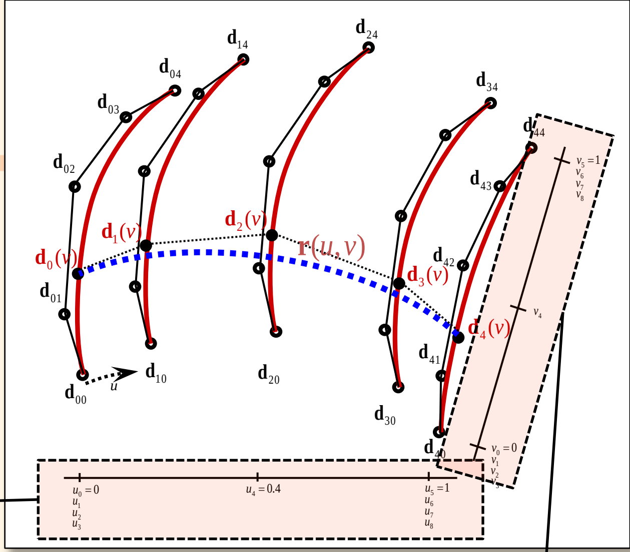
- Member Variables of Class




```
class CBsplineSurface
{
public:
        // member variables
        int m_nDegree;                 → 차수          (3차)

        double* m_pKnot_U;             → u 방향 Knot와 개수     (9개)
        int m_nNumOfKnot_U;

        double* m_pKnot_V;             → v 방향 Knot와 개수     (9개)
        int m_nNumOfKnot_V;

        Vector** m_pCP;                → Control Point, u, v 방향 개수
        int m_nNumOfCP_U;                  (u 방향 5개, v 방향 5개)
        int m_nNumOfCP_V;

        // member functions
        …
};
```


3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface

- Member Functions of Class



***u* 방향 Knot**

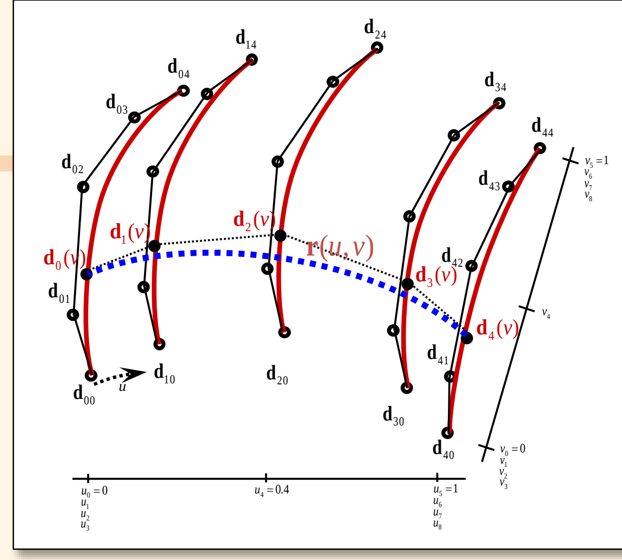
***v* 방향 Knot**

```
class CBsplineSurface
{
public:
        // member variables
        …

        // member functions
        …

        void SetKnot(double* pKnot_U, int nNumOfKnot_U, double* pKnot_V, int nNumOfKnot_V);

        double N(int n, int i, double u, int uv);

        Vector GetPoint(double u, double v);
};
```

➔ Knot 설정

3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface

- Member Functions of Class




```
class CBsplineSurface
{
public:
        // member variables
        …

        // member functions
        …

        void SetKnot(double* pKnot_U, int nNumOfKnot_U, double* pKnot_V, int
nNumOfKnot_V);

        double N(int n, int i, double u, int uv);

        Vector GetPoint(double u, double v);
};
```

u, *v* 방향을 구분하는 flag

➔ B-spline Basis Function 계산

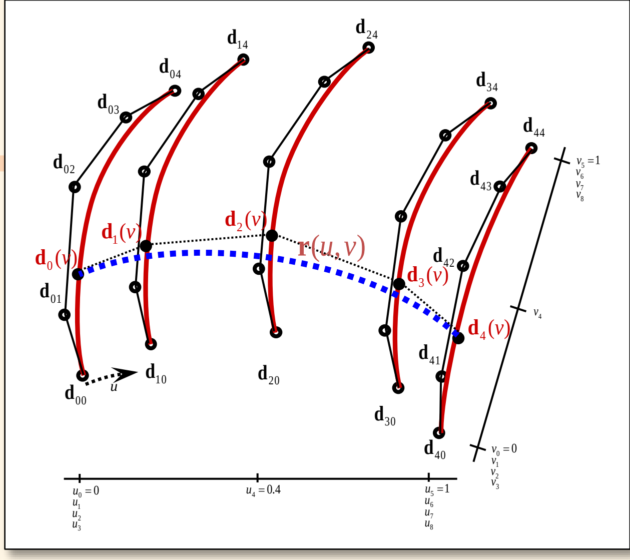
B-spline Basis Function 계산

(Cox-de Boor Recurrence Formula)

$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u)$$

$$N_i^0(u) = \begin{cases} 1 & \text{if } \ u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface
- Member Functions of Class




```
class CBsplineSurface
{
public:
        // member variables
        …

        // member functions
        …

        void SetKnot(double* pKnot_U, int nNumOfKnot_U, double* pKnot_V, int
nNumOfKnot_V);

        double N(int n, int i, double u, int uv);

        Vector GetPoint(double u, double v);
};
```

➔ **Parameter *u*, *v*에 대한 곡면 상의 점 계산**

$$\mathbf{r}(u,v)=\begin{bmatrix}N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u)\end{bmatrix}\begin{bmatrix}\mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04}\\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14}\\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24}\\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34}\\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44}\end{bmatrix}\begin{bmatrix}N_0^3(v)\\ N_1^3(v)\\ N_2^3(v)\\ N_3^3(v)\\ N_4^3(v)\end{bmatrix}$$

$$=\sum_{j=0}^{5}\sum_{i=0}^{5}\mathbf{d}_{ij}N_i^3(u)N_j^3(v)$$

3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface

- Member Function Example 'GetPoint'

➔ Parameter *u*, *v*에 대한 곡면 상의 점 계산

```
Vector CBsplineSurface::GetPoint(double u, double v)
{
  // return value
  Vector r_u_v(0.0, 0.0, 0.0);

  // get curve
  for (int i=0; i<m_nNumOfCP_U; i++)
  {
    Vector r_v(0.0, 0.0, 0.0);

    for (int j=0; j<m_nNumOfCP_V; j++)
    {
      r_v = r_v + m_pCP[i][j] * N(m_nDegree, j, v, ID_V);
    }

    r_u_v = r_u_v + N(m_nDegree, i, u, ID_U) * r_v;
  }

  return r_u_v;
}
```

$$\mathrm{d}_0(v) = \mathrm{d}_{00}N_0^3(v) + \mathrm{d}_{01}N_1^3(v) + \mathrm{d}_{02}N_2^3(v) + \mathrm{d}_{03}N_3^3(v) + \mathrm{d}_{04}N_4^3(v)$$

$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

$$= \sum_{j=0}^{5} \sum_{i=0}^{5} \mathbf{d}_{ij} N_i^3(u) N_j^3(v)$$

3.3.2 B-spline surfaces Interpolation

3.3.2.1 bicubic B-spline surface interpolation 개요

3.3.2.2 bicubic B-spline surface interpolation 상세 과정

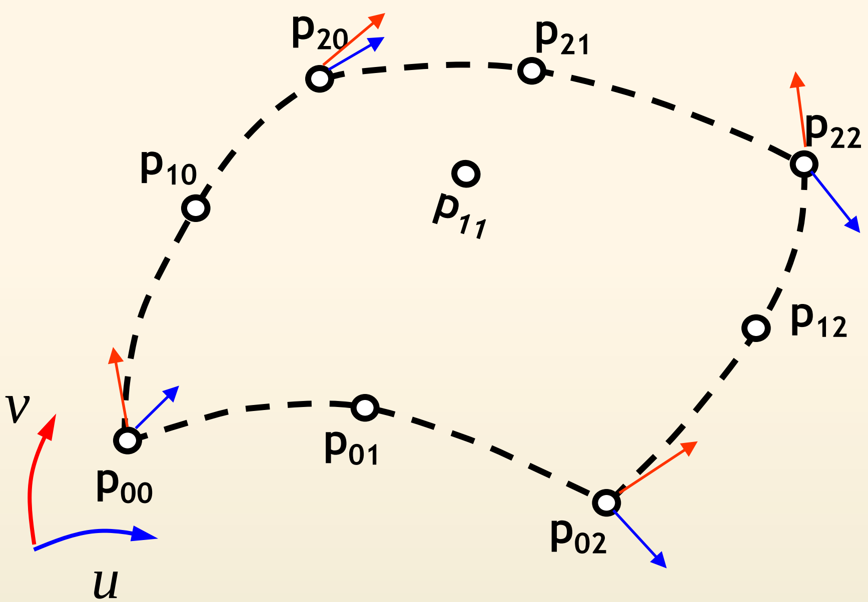
3.3.2.3 Sequences of Finding knots

3.3.2.4 Knot 간격차이가 주는 영향

3.3.2.5 Example of bicubic B-spline Surface Interpolation

3.3.2.1 bicubic B-spline Surface Interpolation (1)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

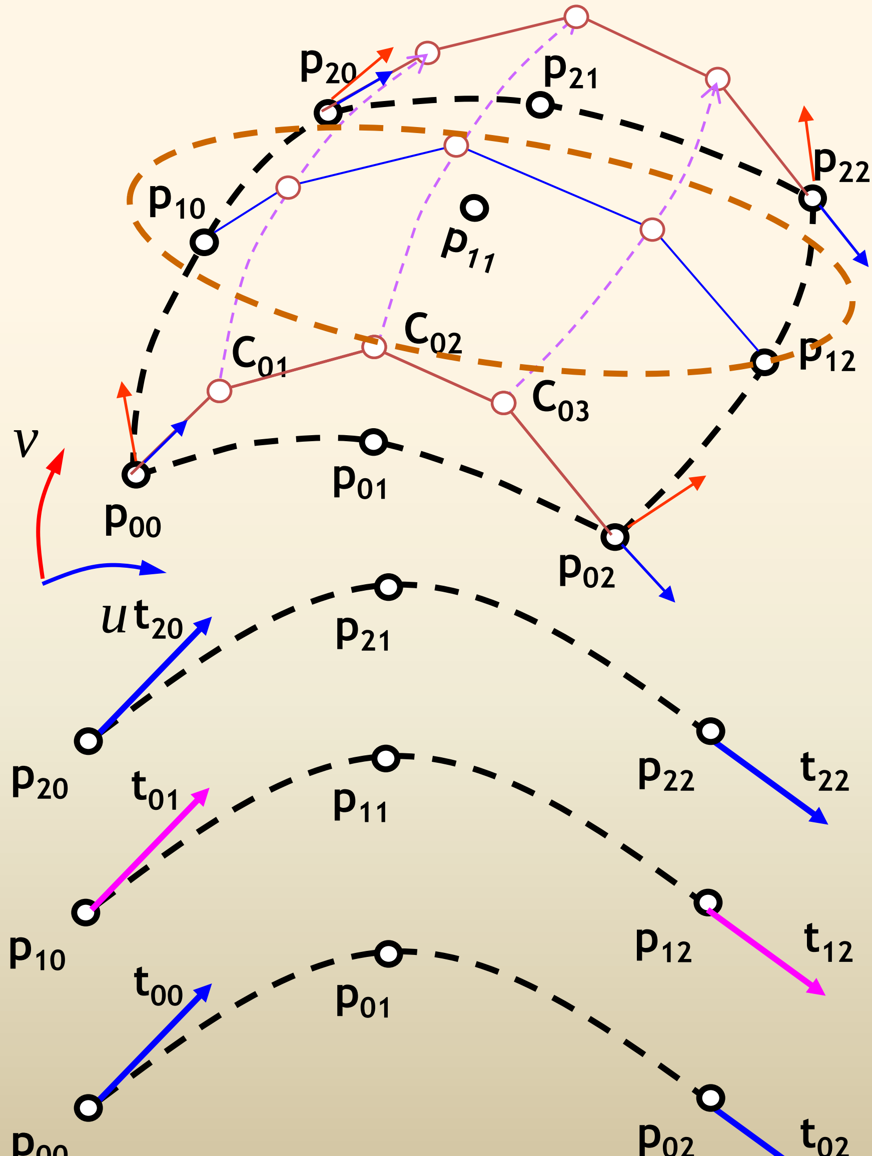


$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} & d_{04} \\ d_{10} & d_{11} & d_{12} & d_{13} & d_{14} \\ d_{20} & d_{21} & d_{22} & d_{23} & d_{24} \\ d_{30} & d_{31} & d_{32} & d_{33} & d_{34} \\ d_{40} & d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

5x5 개의 조정점을 구하면 곡면을 생성할 수 있다.

3.3.2.1 bicubic B-spline Surface Interpolation (2)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface



곡선상의 점($P_{i,j}$)과 접선벡터 ($t_{i,j}$) 으로부터 중간 조정점($C_{i,j}$)을 구한다.

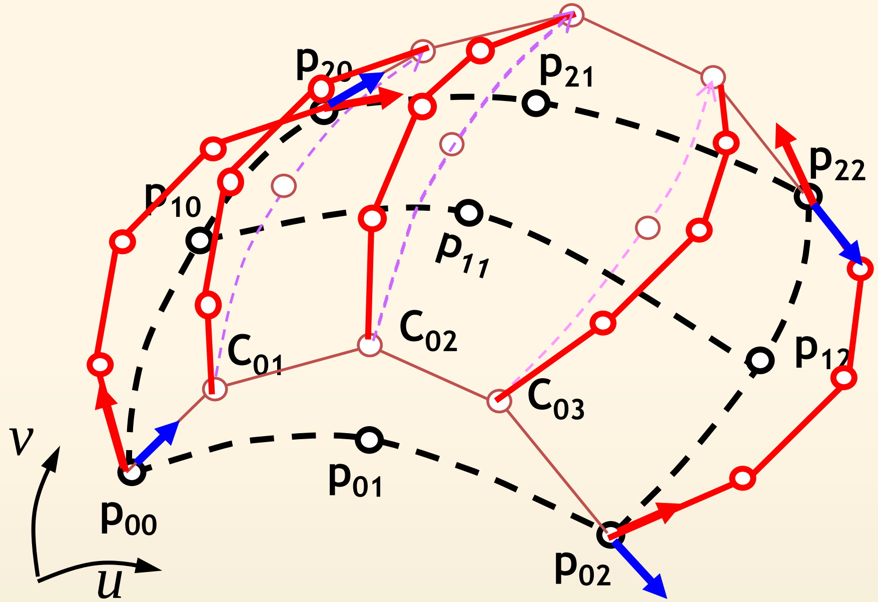
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{3}{\Delta_s} & \frac{3}{\Delta_s} & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & 0 \\ 0 & 0 & 0 & -\frac{3}{\Delta_E} & \frac{3}{\Delta_E} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{C}_{00} \\ \mathbf{C}_{01} \\ \mathbf{C}_{02} \\ \mathbf{C}_{03} \\ \mathbf{C}_{04} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{00} \\ \mathbf{t}_{00} \\ \mathbf{P}_{01} \\ \mathbf{t}_{02} \\ \mathbf{P}_{02} \end{bmatrix}$$

Bessel end condition으로 접선벡터($t_{i,j}$)를 구한다

Bessel end condition:
곡선을 지나는 3점을 2차식으로 보간(interpolation) 한 후, 곡선의 끝점에서의 1차미분값을 구하는 방법

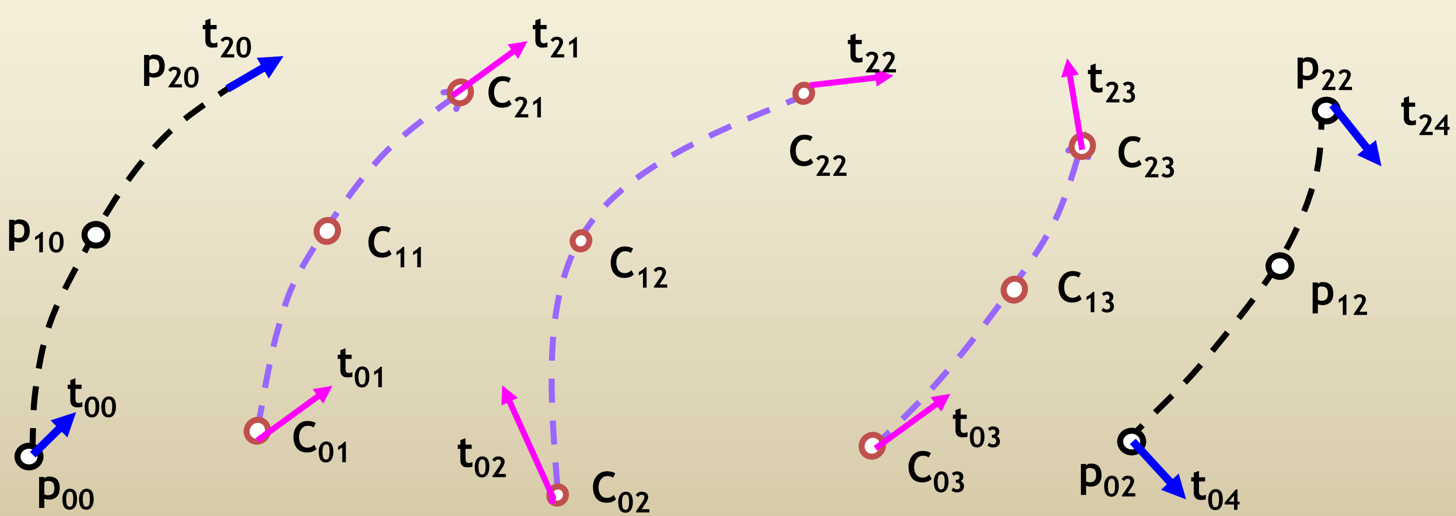
3.3.2.1 bicubic B-spline Surface Interpolation (3)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface



중간 조정점($C_{i,j}$)과 접선 벡터 ($t_{i,j}$) 으로부터 B-spline조정점($d_{i,j}$)을 구한다.

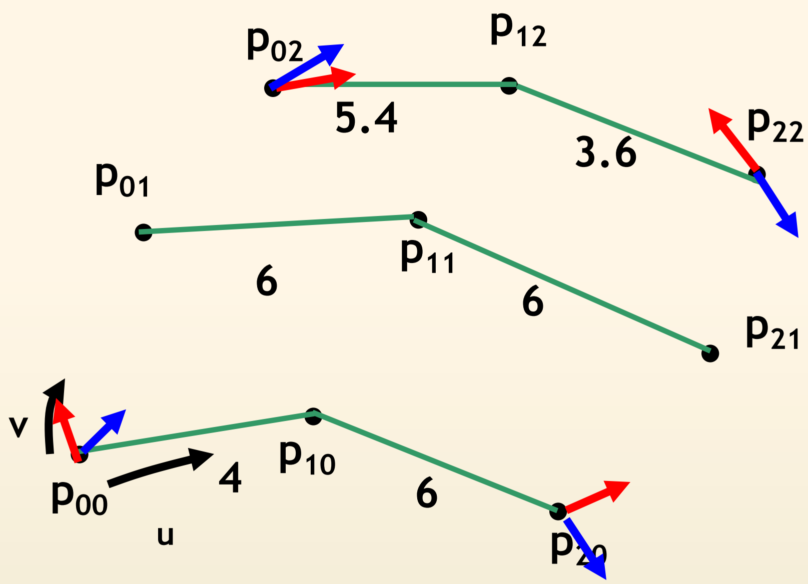
$$\begin{bmatrix} 1&0&0&0&0\\ -\frac{3}{\Delta_s}&\frac{3}{\Delta_s}&0&0&0\\ 0&\alpha&\beta&\gamma&0\\ 0&0&0&-\frac{3}{\Delta_E}&\frac{3}{\Delta_E}\\ 0&0&0&0&1\end{bmatrix}\begin{bmatrix}d_{02}\\d_{12}\\d_{22}\\d_{32}\\d_{42}\end{bmatrix}=\begin{bmatrix}C_{02}\\t_{02}\\C_{12}\\t_{22}\\C_{22}\end{bmatrix}$$



Bessel end condition으로 접선벡터($t_{i,j}$)를 구한다

3.3.2.2 bicubic B-spline Surface Interpolation (4)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

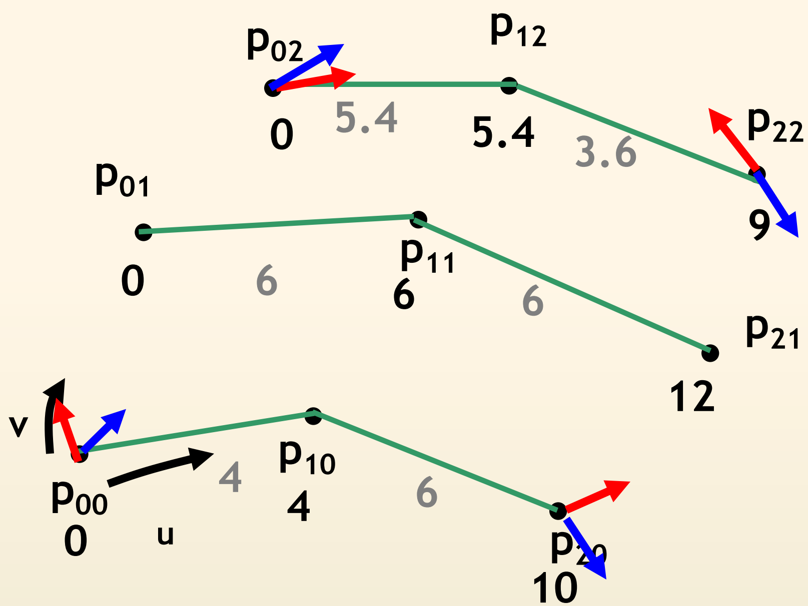


1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다

❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

3.3.2.2 bicubic B-spline Surface Interpolation (5)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

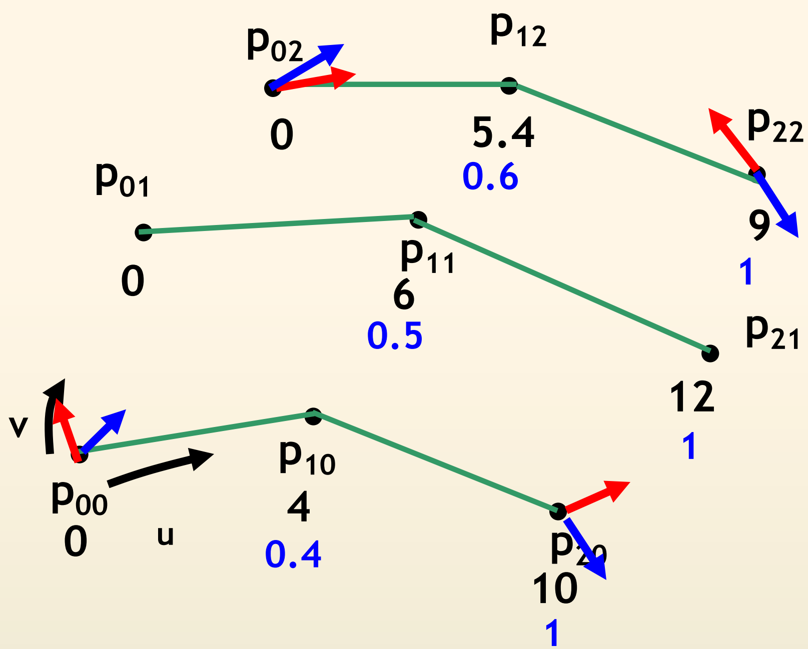


1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다

❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

3.3.2.2 bicubic B-spline Surface Interpolation (6)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

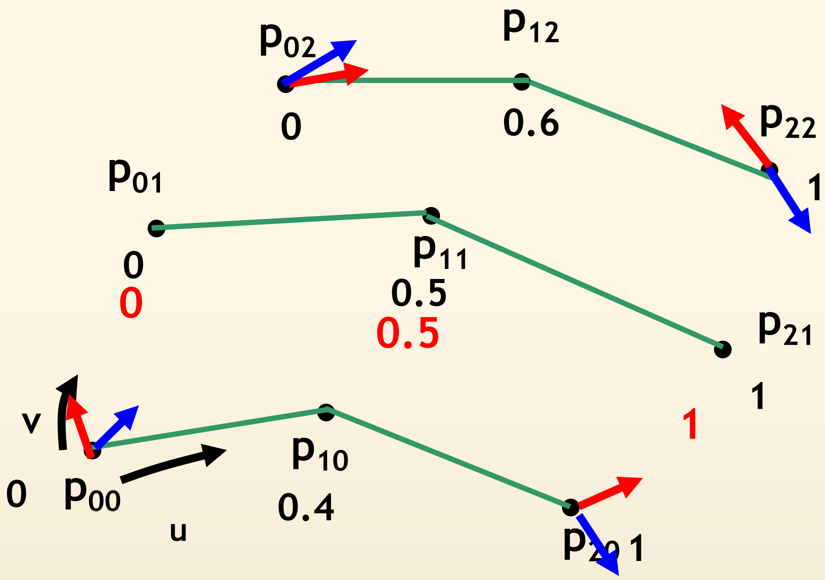


1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다

❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

3.3.2.2 bicubic B-spline Surface Interpolation (7)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

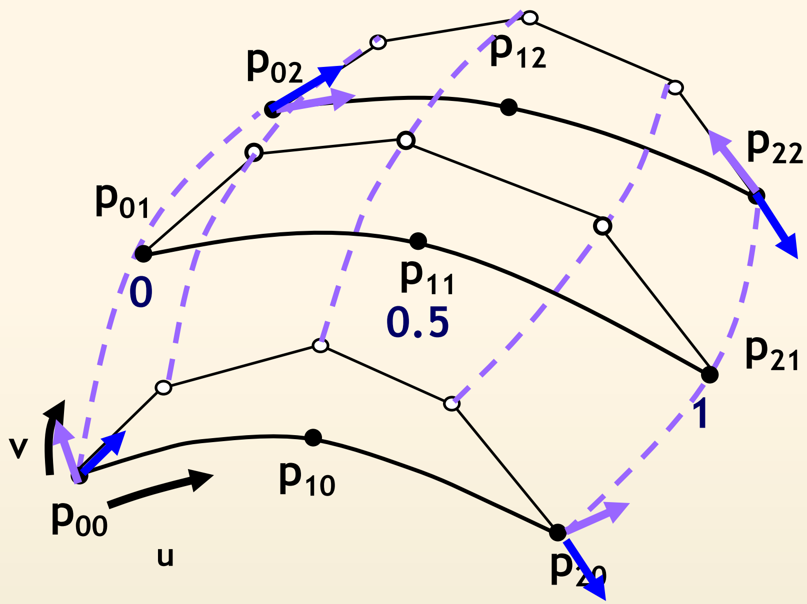


1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균 u방향 knot값을 계산한다

❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

3.3.2.2 bicubic B-spline Surface Interpolation (8)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface

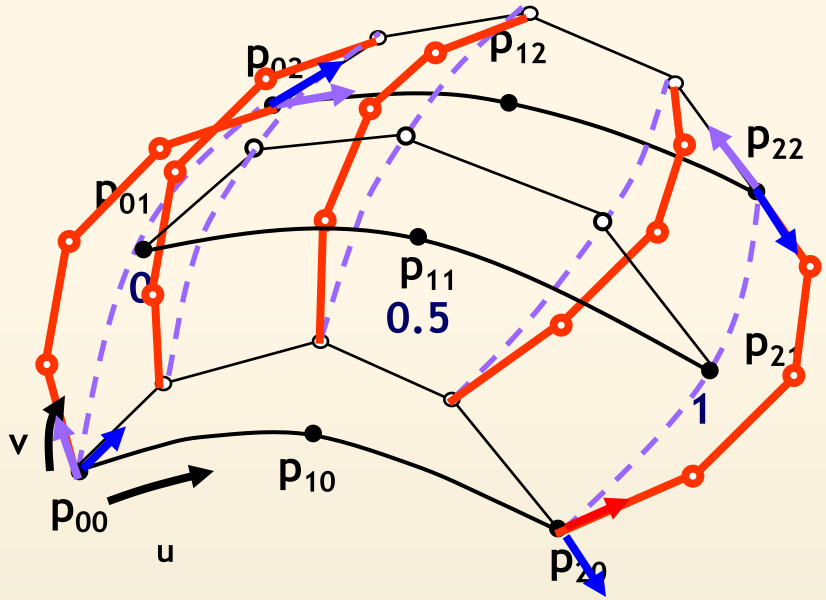


❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들으로 v방향으로 평균 u방향 knot 값을 계산한다
2. u 방향 점들을 보간하는 B-spline 곡선을 계산
   - 최종적인 u방향 knot와 점들을 지나는 B-spline 곡선과 그 조정점을 계산한다
3. v 방향 knot 간격을 u방향과 동일한 방법으로 구함
4. u 방향 B-spline 곡선의 조정점을 보간하는 v방향 B-spline 곡선을 계산
   - u방향 B-spline 곡선의 조정점에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (보라색점선) 을 생성한다. 이 곡선의 조정점 (빨간점) 이 최종 B-spline 곡면의 조정점이다

3.3.2.2 bicubic B-spline Surface Interpolation (9)

- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface



1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균하여 최종적인 u 방향 knot값을 계산한다
2. u 방향 점들을 보간하는 B-spline 곡선을 계산
   - 최종적인 u방향 knot와 점들을 지나는 B-spline 곡선과 그 조정점을 계산한다
3. v 방향 knot 간격을 u방향과 동일한 방법으로 구함
4. u 방향 B-spline 곡선의 조정점을 보간하는 v방향 B-spline 곡선을 계산
   - u방향 B-spline 곡선의 조정점에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (보라색점선) 을 생성한다. 이 곡선의 조정점 (빨간점) 이 최종 B-spline 곡면의 조정점이다

❑ Given
- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

3.3.2.3 Sequences of Finding Knot

B-spline **곡선**에서의 Knot 간격 예측

⇓

주어진 곡선상의 점들로부터 Chord Length를 구함

$$\frac{\Delta_i}{\Delta_{i+1}} = \sqrt{\frac{|p_{i-1} - p_{i-2}|}{|p_i - p_{i-1}|}},\ (i = 2,3,....,n+2)$$

⇓

곡선상의 점,접선벡터와 위의 Knot 간격을 이용하여 B-spline 조정점을 구할 수 있음

Tensor Product방식으로 정의된 spline **곡면**에서의 Knot 간격예측

⇓

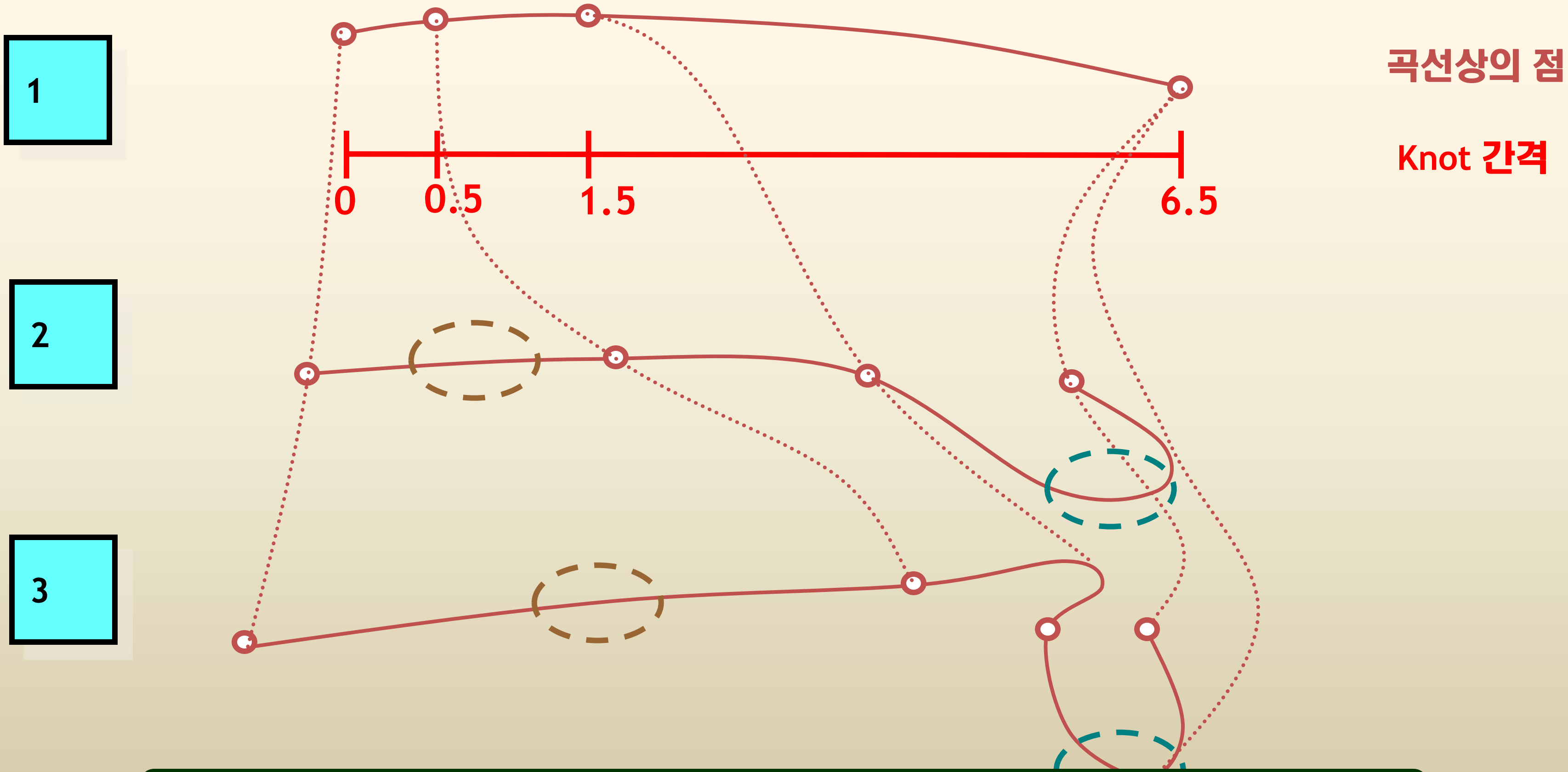
주어지는 곡면상의 점이 GRID와 같이 균일하게 주어져야 이상적인 곡면 재현 가능

⇓

즉, 주어진 곡선들의 Knot 간격이 모두 일정해야 함.

3.3.2.4 Knot 간격 차이가 주는 영향

점과 점 사이의 knot 간격은 그 점 사이를 지나가는 데 걸리는 시간과 같은 개념이다.



그러므로 knot간격비가 비슷할 수록 곡면의 왜곡이 적다.

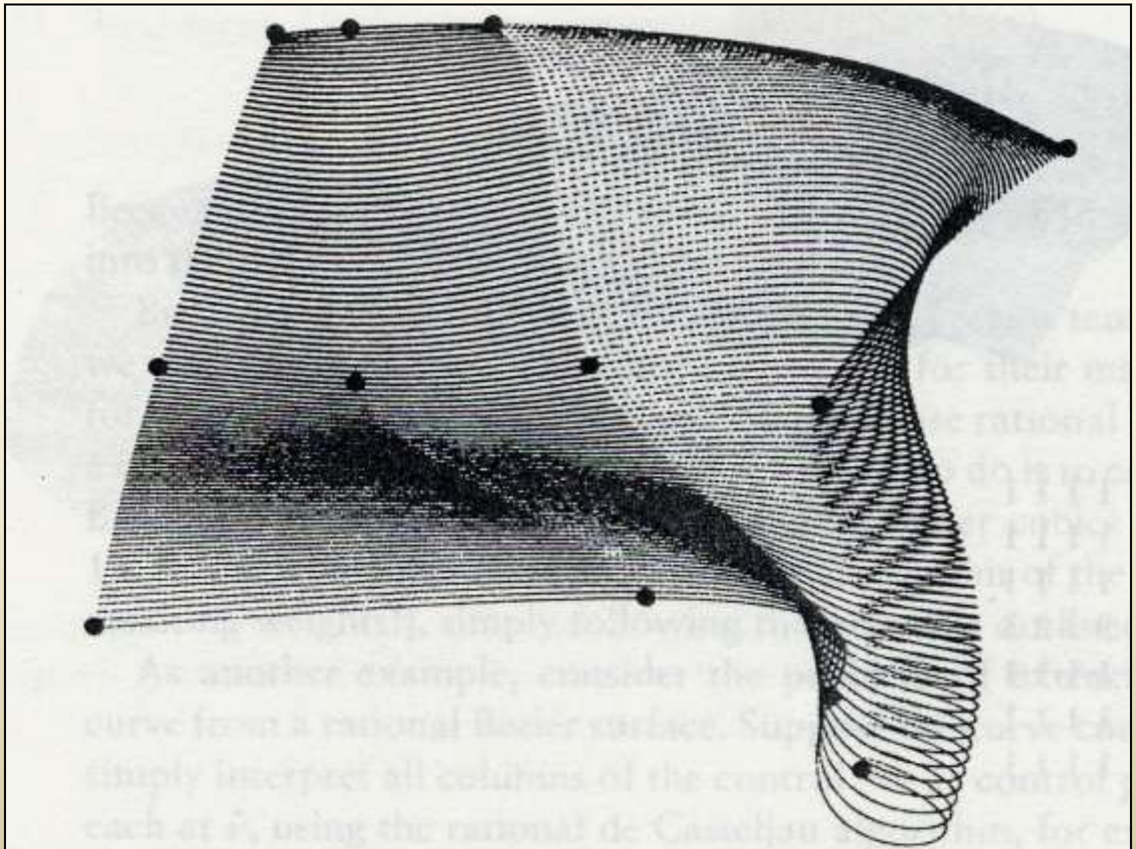
3.3.2.4 Knot 간격 차이가 주는 영향

점과 점 사이의 knot 간격은 그 점 사이를 지나가는 데 걸리는 시간과 같은 개념이다.

1

2

3



곡선상의 점

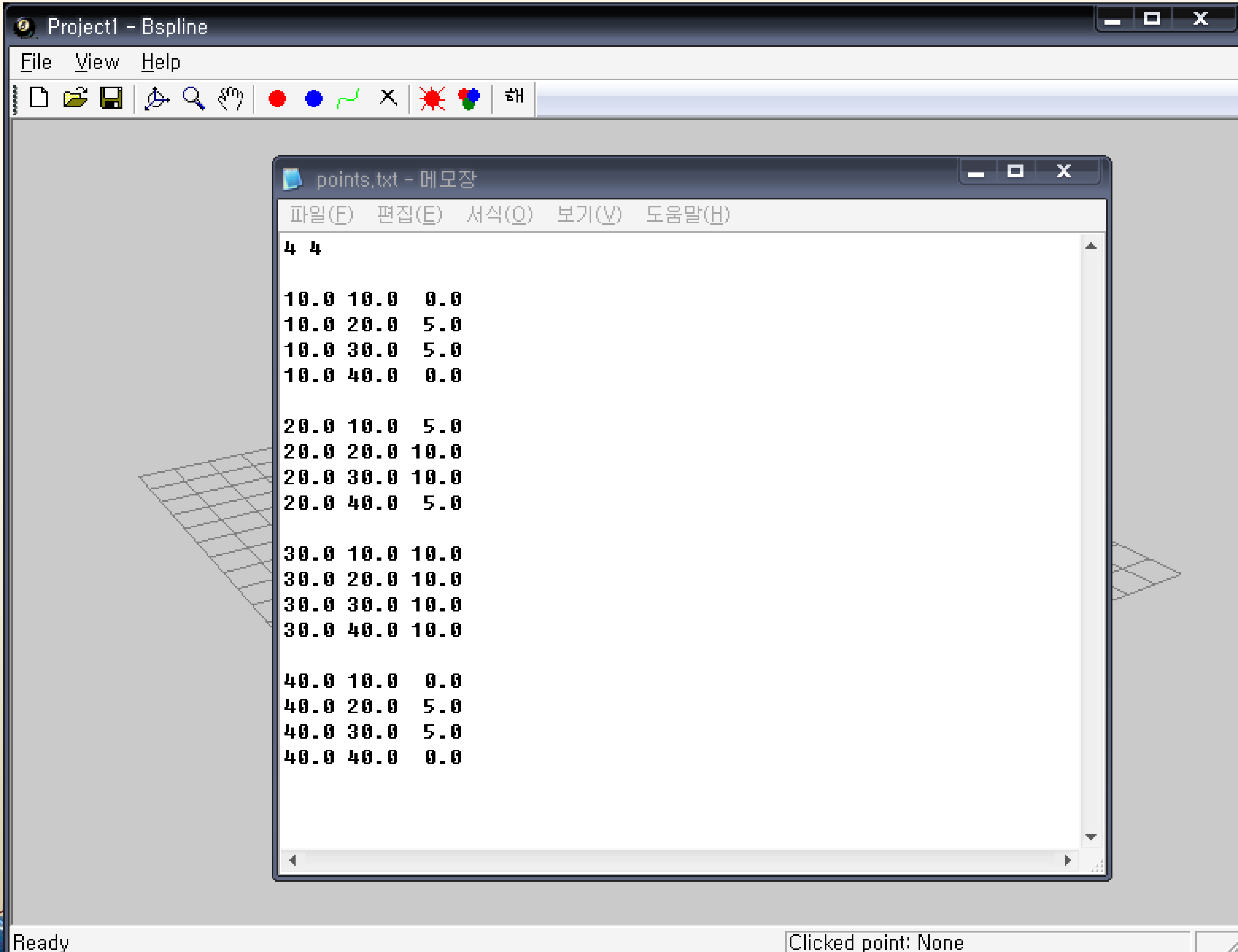
Knot 간격

그러므로 knot간격비가 비슷할 수록 곡면의 왜곡이 적다.

3.3.2.5 Example of bicubic B-spline Surface Interpolation (1)

- Given: Points on Surface
- Find: bicubic B-spline Surface (Control Points of bicubic B-spline Surface)

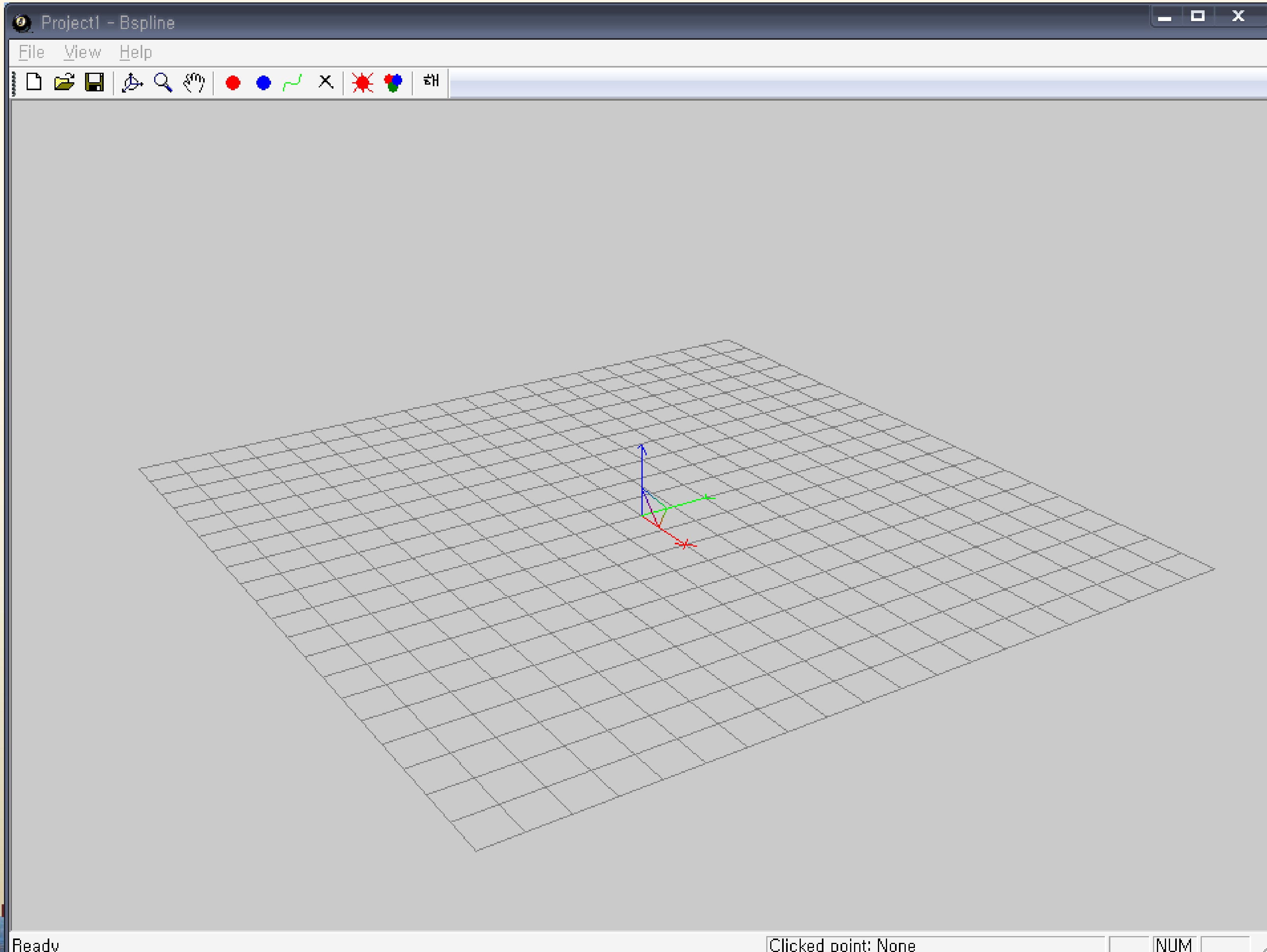
일반 3차원 곡면 형상 예시



3.3.2.5 Example of bicubic B-spline Surface Interpolation (2)

- Given: Points on Surface
- Find: bicubic B-spline Surface (Control Points of bicubic B-spline Surface)

선박 선형 곡면 예시



3.3.2.5 Sample code of bicubic B-spline Surface Interpolation (1)

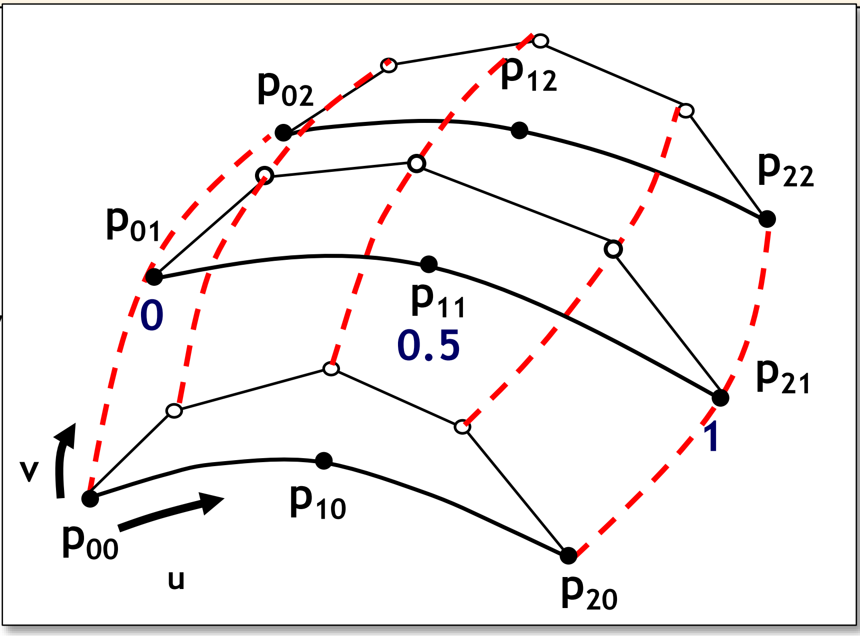
```
void BicubicBsplineSurface::Interpolate(Vector **pFittingPoint, int nU, int nV) {
  // Generate u-Knot
    if(m_UKnot) delete[] m_UKnot;
    m_nUKnot = (m_nU - 2) + 2*(3+1);
    m_UKnot = new double [m_nUKnot];

    // Initial u-Knot
    double** tmpUKnots;
    tmpUKnots = new double*[nV];
    for(int j = 0; j < nV; j++){
        tmpUKnots[j] = new double[nU];
        for(int i = 0; i < nU; i++){
            tmpUKnot[j][i] = …;    // chord length or centripetal
        }
    }
    // generate average u-Knot
   for(int i = 0; i < nU; i++){
        m_UKnot[i] = 0;
        for(int j=0; j<nV; j++) {        m_UKnot[i] += tmpUKnot[j][i];    }
        m_UKnot[i] = m_UKnot[i] / nV;
  }
```

➔ Chord length를 이용하여 u, v 방향 Knot 생성

➔ Knot간 간격의 평균값으로 Knot 재 설정

3.3.2.5 Sample code of bicubic B-spline Surface Interpolation (2)

```
// Interpolate u-directional B-spline curve
CubicBsplineCurve* u_curve = new CubicBsplineCurve[nV
for(int j = 0; j < nV; j++){
    u_curve[j].SetKnot( m_UKnot );
    u_curve[j].Interpolate(  pFittingPoint[j], nU );
}

// Generate v-directional Fitting Point
int nvFittingPoint = u_curve[0].m_nControlPoint;
Vector** vFittingPoint = new Vector [ nvFittingPoint ];
for(int j=0; j < nvFittingPoint; j++){
    vFittingPoint[j] = new Vector[ nV ];
    for( int i = 0; i < nV; i++){
         vFittingPoint[j][i] = u_curve[i].m_ControlPoint[j];
    }
}
……………………….
```

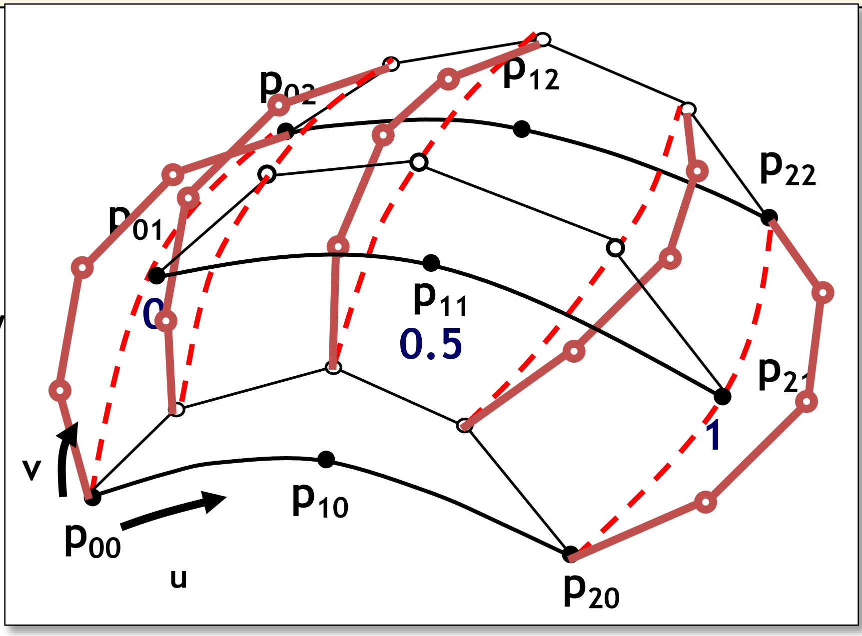


u 방향 B-spline curve 생성

3.3.2.5 Sample code of bicubic B-spline Surface Interpolation (2)

```
// Interpolate u-directional B-spline curve
 CubicBsplineCurve* u_curve = new CubicBsplineCurve[nV
 for(int j = 0; j < nV; j++){
     u_curve[j].SetKnot( m_UKnot );
     u_curve[j].Interpolate(  pFittingPoint[j], nU );
 }
```

```
// Generate v-directional Fitting Point
int nvFittingPoint = u_curve[0].m_nControlPoint;
Vector** vFittingPoint = new Vector [ nvFittingPoint ];
for(int j=0; j < nvFittingPoint; j++){
    vFittingPoint[j] = new Vector[ nV ];
    for( int i = 0; i < nV; i++){
         vFittingPoint[j][i] = u_curve[i].m_ControlPoint[j];
    }
}
………………………
```



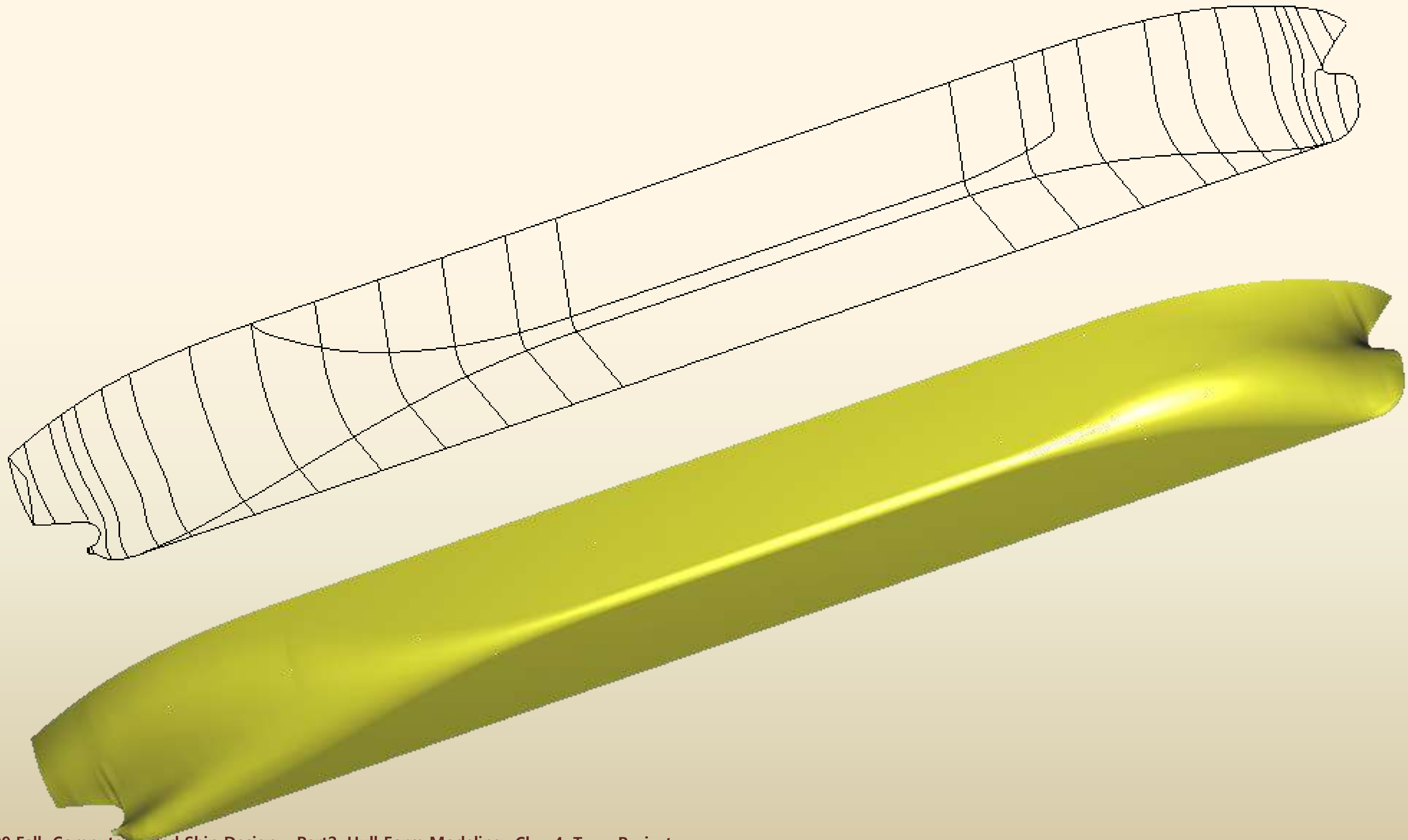
u 방향 B-spline curve 의 조정점을 Fitting Point로 하여 v 방향 B-spline Curve 생성

Ch 4. Term Project

Generation Ship hull surfaces by interpolating given points

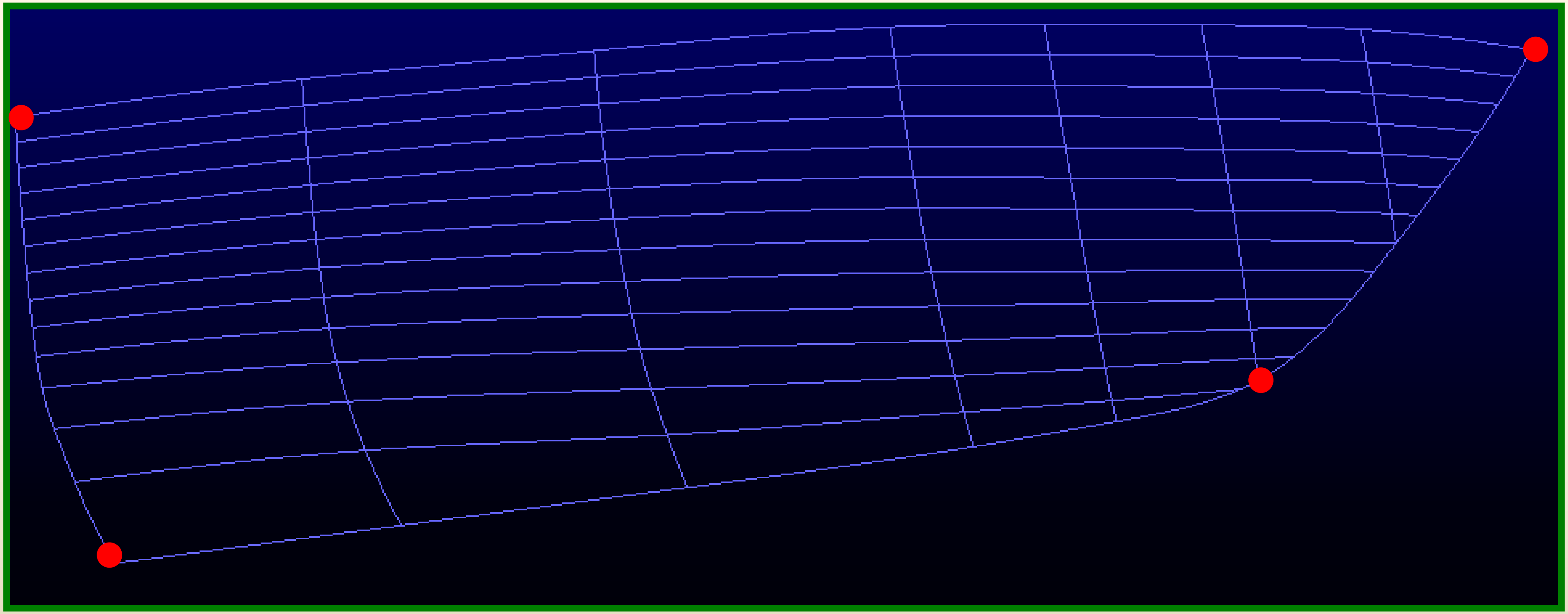
- Given: $\mathbf{P}_{i,j}$
- Find : $\mathbf{d}_{i,j}$

4.1 단일 B-spline 곡면 patch를 이용한 선형곡면 생성 프로그램 구현



4.2 간단한 요트 형상의 선수부 곡선그물망 형상

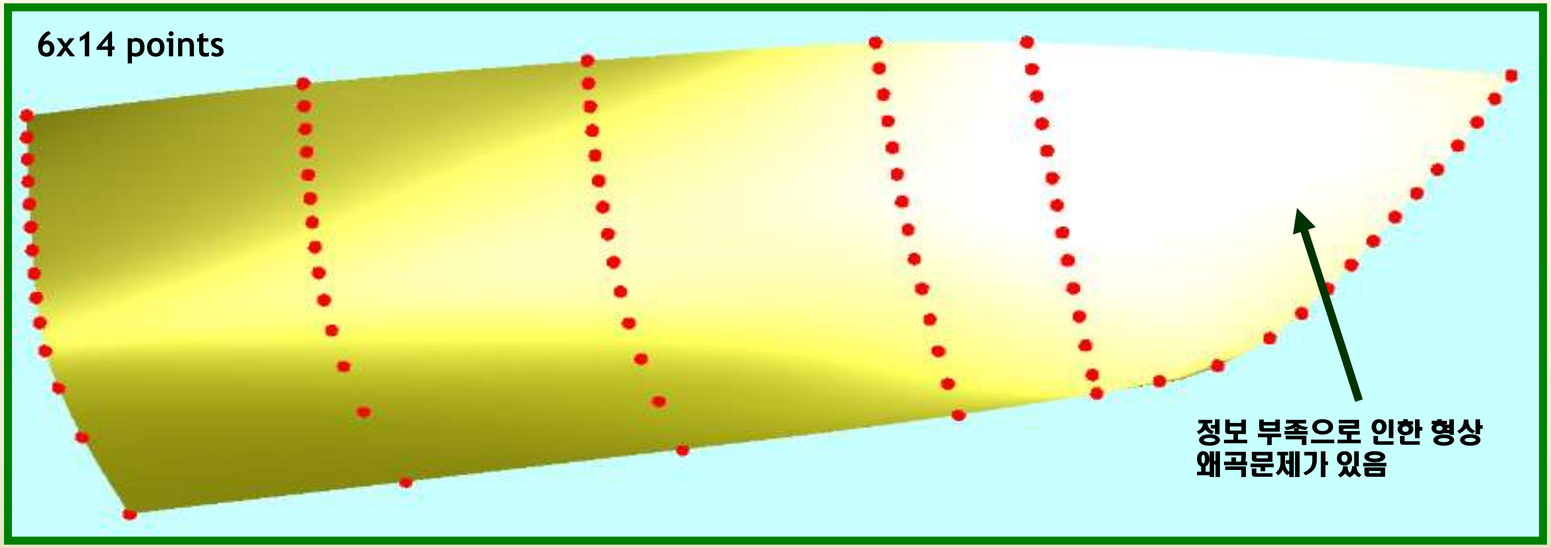
*** 출처 : 서울대 조선해양공학과 2005년 2학년 교과목 『 조선해양공학계획』 강좌 중에 학생들이 설계한 선형**



사각형 패치의 꼭지점 결정

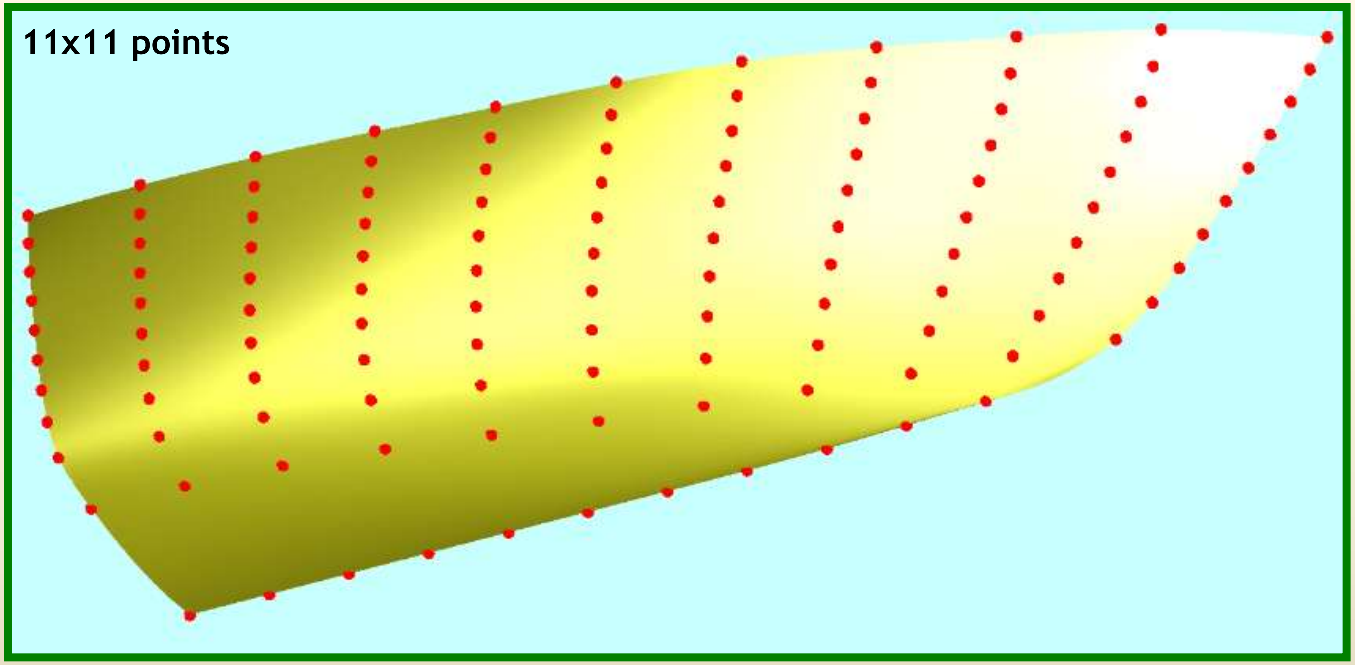
4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (1)

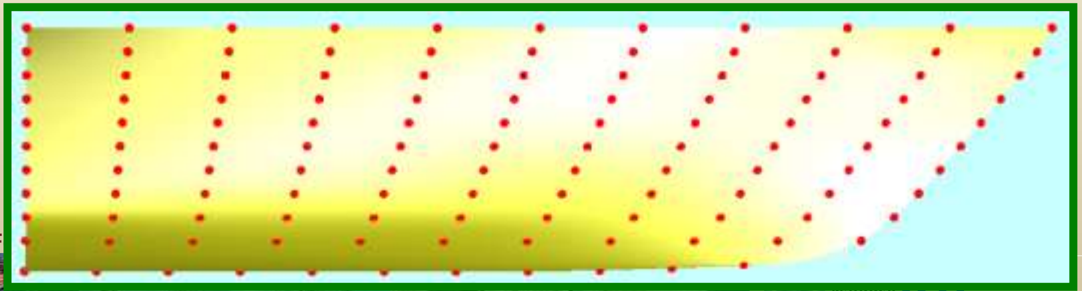
- Offset table 형식으로 점을 추출한 후,
  이 점들로 부터 bicubic B-spline 선형곡면을 생성한 결과



4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (2)

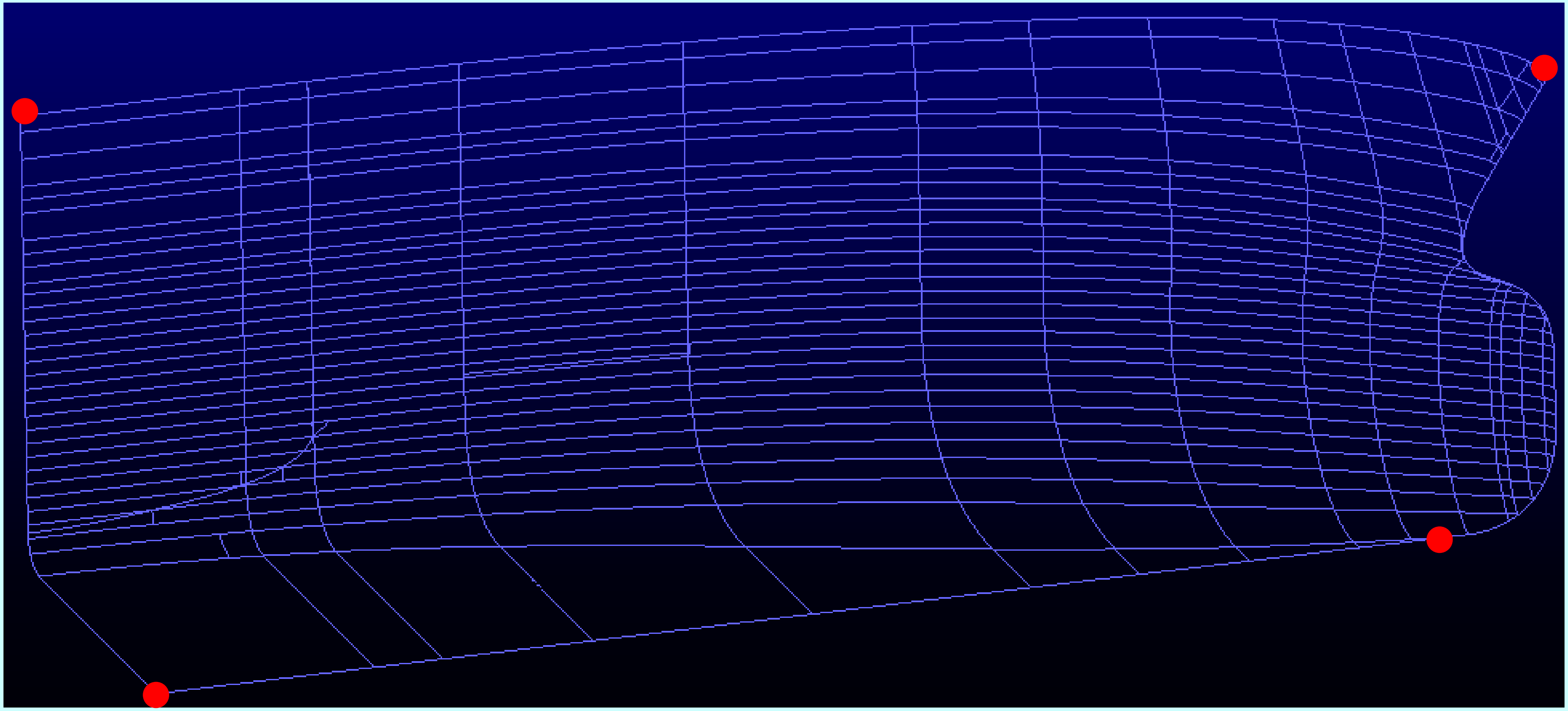
- ❑ 점들의 x좌표 사이의 거리가 일정하도록 점을 추출한 후, 이 점들로부터 bicubic B-spline 선형곡면을 생성한 결과





4.4 구상선수를 갖는 단축선의 선수부 곡선그물망 형상

선수부



4.5 구상선수부 곡선그물망으로부터 선수부 선형곡면 생성결과

- 주어진 곡선그물망 이외의 보조선을 생성하여 곡선 보간에 적합한 점 data를 생성한 후, 점 data로부터 선수부 선형곡면을 생성한 결과