[2008][04-2]

Computer aided ship design

Part 1. Curve & Surface

October 2008

Prof. Kyu-Yeul Lee

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

Advanced
Ship
Design
Automation
Laboratory

Ch 3. 곡면(Surfaces)

3.1 Parametric Surfaces

3.2 Bezier Surfaces

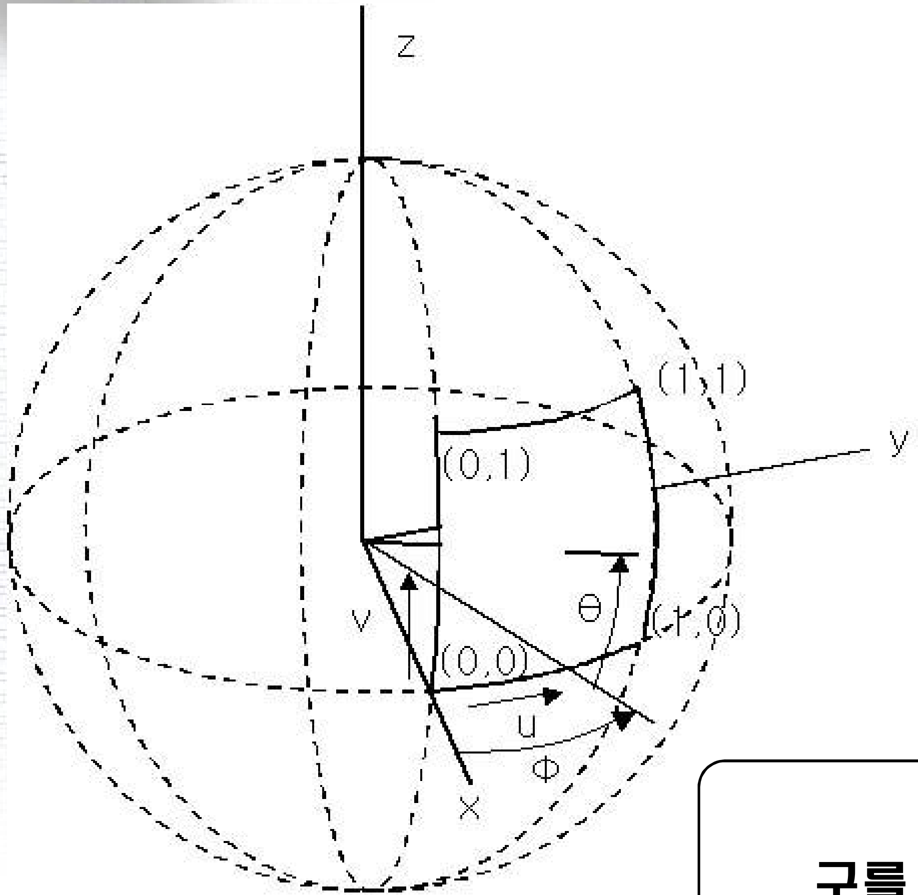
3.3 B-spline surfaces

Advanced
Ship
Design
Automation
Laboratory

3.1 Parametric Surfaces

Advanced
Ship
Design
Automation
Laboratory

2.1 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 (ϕ, θ) 로 표현하면 간단히 수식화 할 수 있다.

3.2 Bezier surfaces

3.2.1 Generation of Bezier surfaces
by de Casteljau algorithm

Advanced
Ship
Design
Automation
Laboratory

3.2.1 Generation of Bezier surfaces by de Casteljau algorithm

3.2.1.1 Bi-linear Bezier Surface Patch

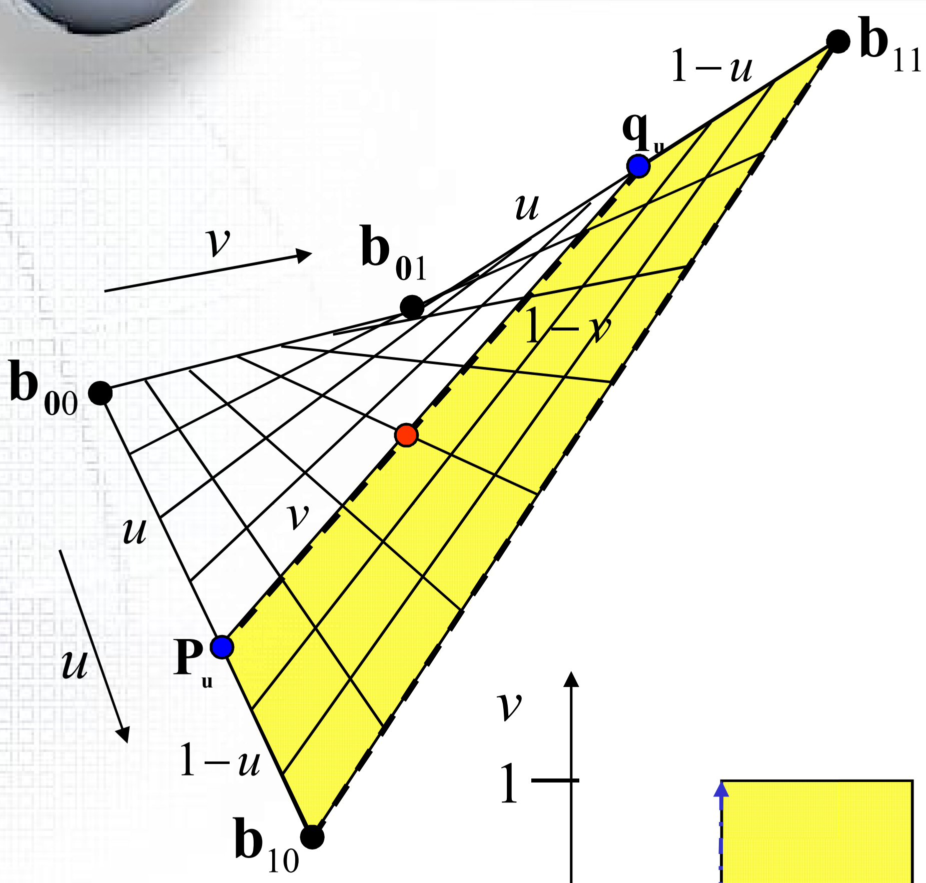
3.2.1.2 Bi-quadratic Bezier Surface Patch

3.2.1.3 Bi-Cubic Bezier Surface Patch

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**



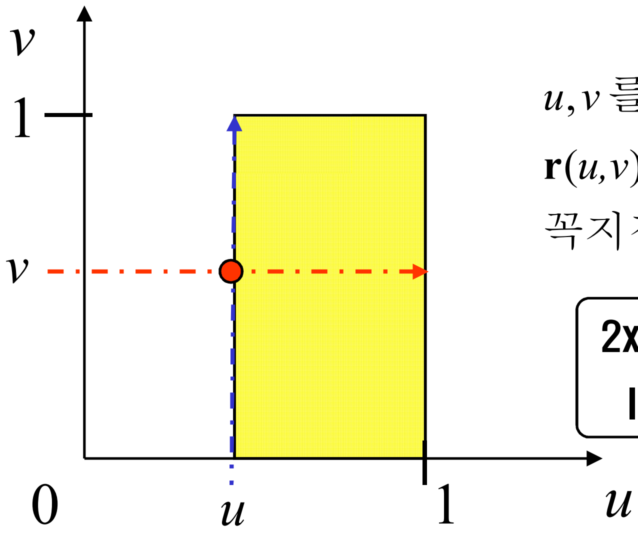
$$\mathbf{P}_u = (1-u)\mathbf{b}_{00} + u\,\mathbf{b}_{10}$$

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

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

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

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

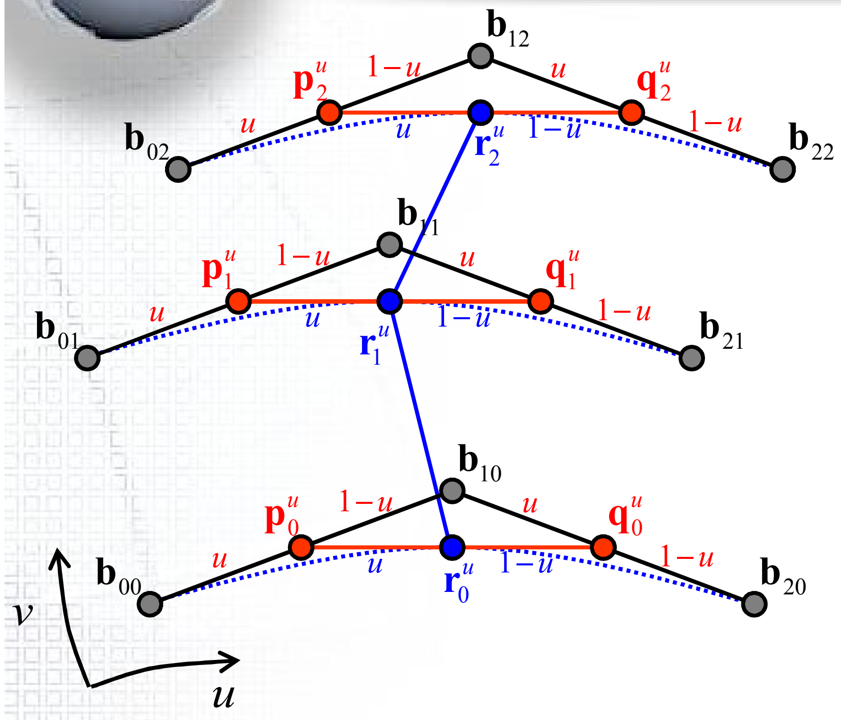


u, v를 0에서 1까지 증가하며 $\mathbf{r}(u,v)$를 계산하면 점 $\mathbf{b}_{00}, \mathbf{b}_{10}, \mathbf{b}_{01}, \mathbf{b}_{11}$ 을 꼭지점으로 하는 곡면을 얻을 수 있다.

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



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

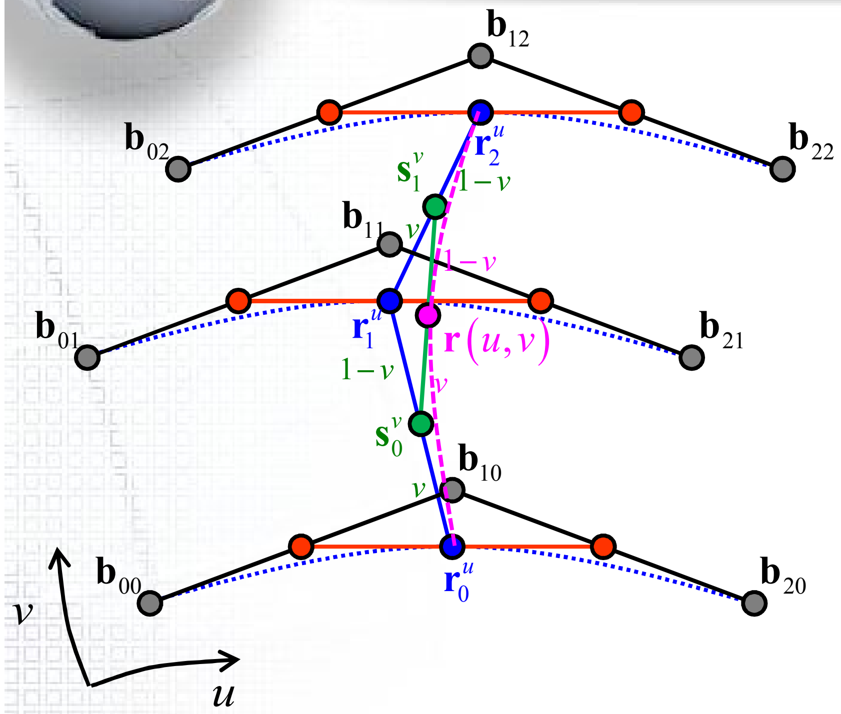
$$\begin{bmatrix} \mathbf{r}_0^u \\ \mathbf{r}_1^u \\ \mathbf{r}_2^u \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} (1-u)^2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

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

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



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

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

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

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

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

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

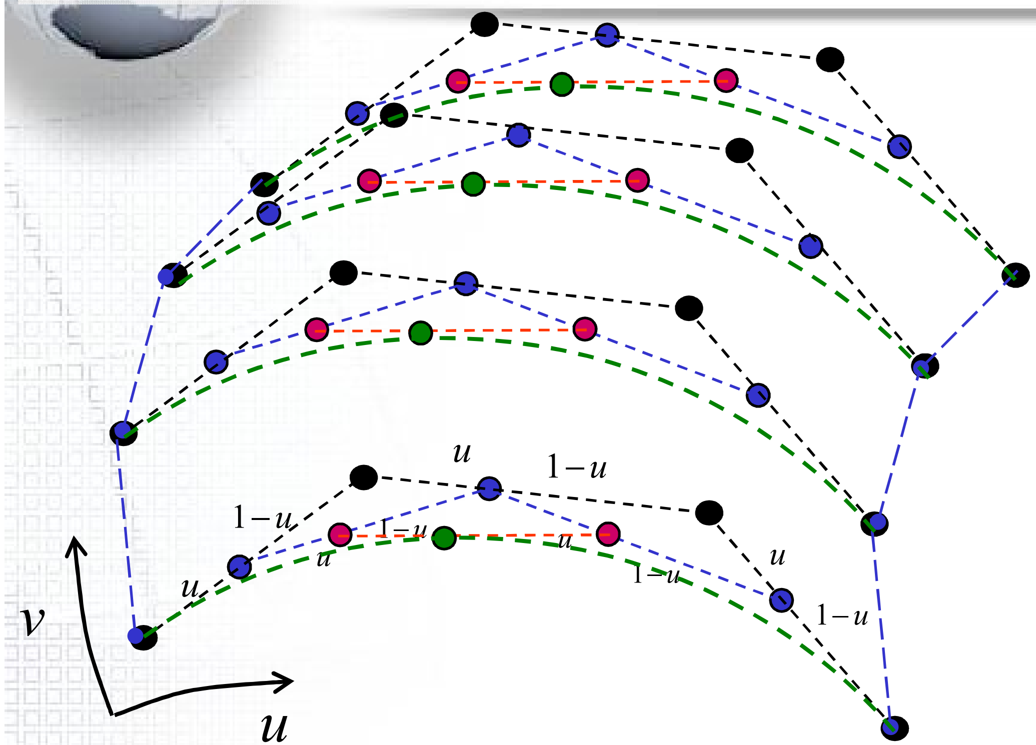
$$\mathbf{r}(u,v) = \begin{bmatrix}(1-v)^2 & 2v(1-v) & v^2\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}(1-u)^2 \\ 2u(1-u) \\ u^2\end{bmatrix}$$

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



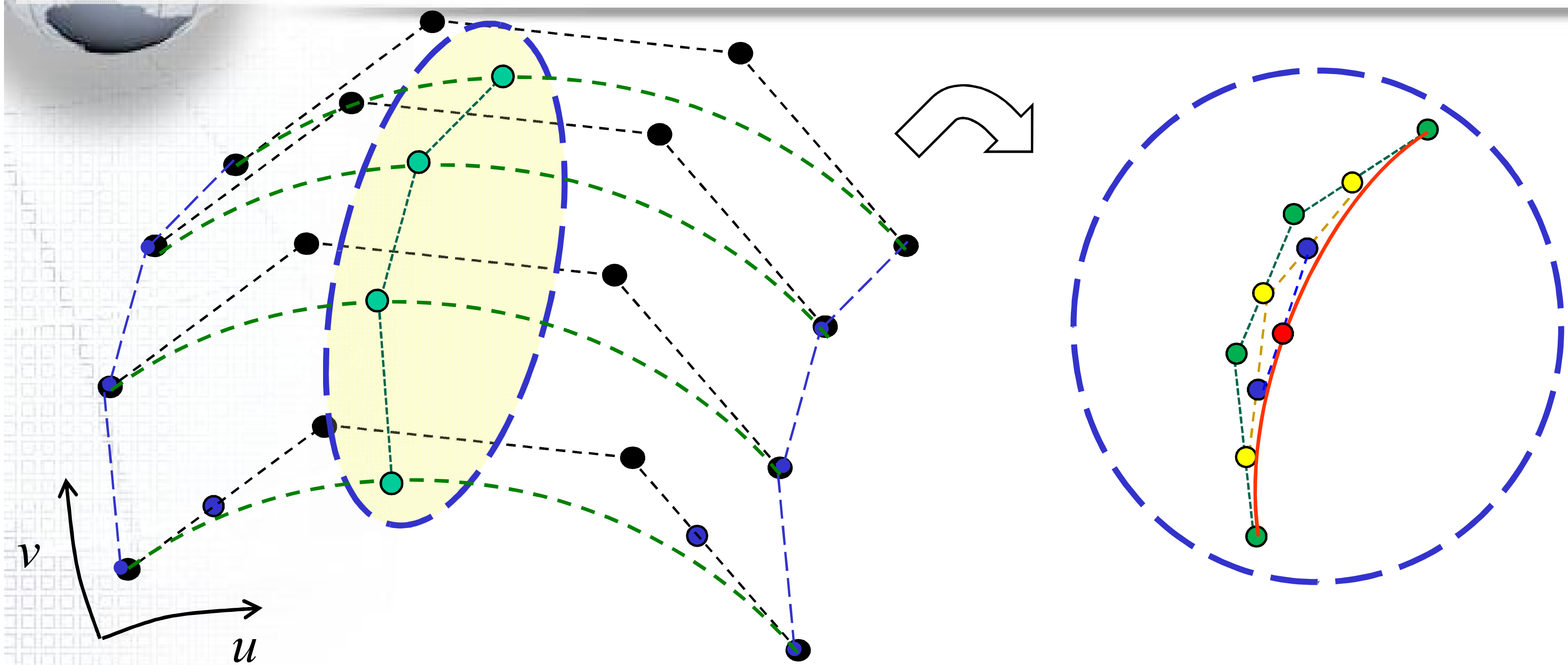
$$\mathbf{b}(u,v) = \begin{bmatrix} B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} & \mathbf{b}_{0,3} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} & \mathbf{b}_{1,3} \\ \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{b}_{2,2}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

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



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

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

3.3 B-spline surfaces

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

3.3.2 B-spline surface Interpolation

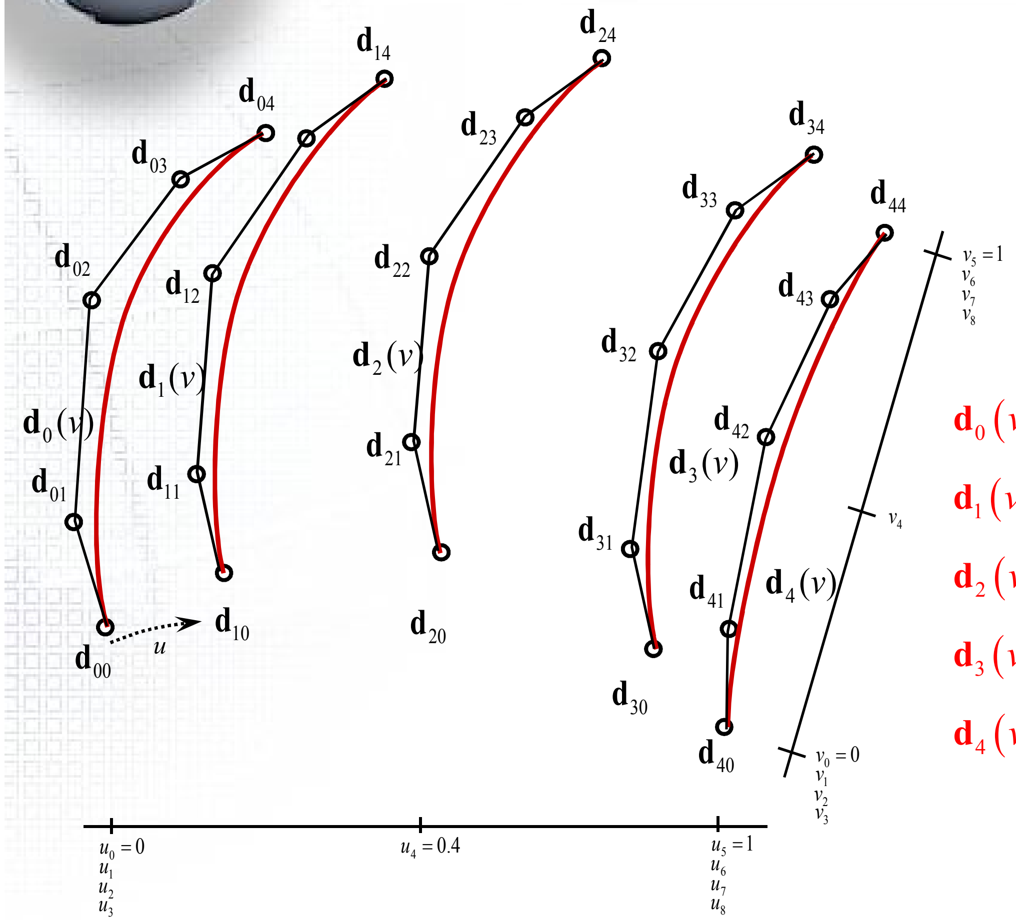
3.3.1 Generation of B-spline surfaces by tensor-product approach

3.3.1.1 Tensor-product bicubic B-spline surface

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

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:

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

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

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

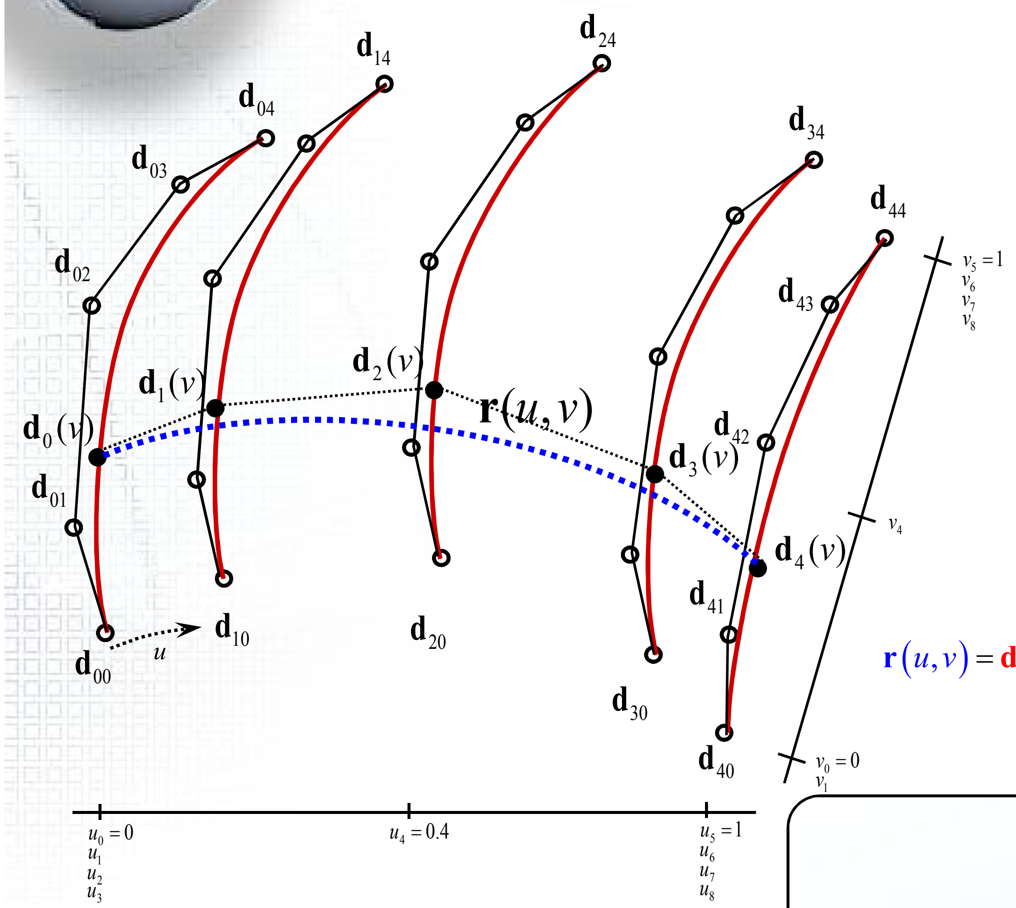
$$\mathbf{d}_3(v) = \mathbf{d}_{30}N_0^3(v) + \mathbf{d}_{31}N_1^3(v) + \mathbf{d}_{32}N_2^3(v) + \mathbf{d}_{33}N_3^3(v) + \mathbf{d}_{34}N_4^3(v)$$

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

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

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:

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

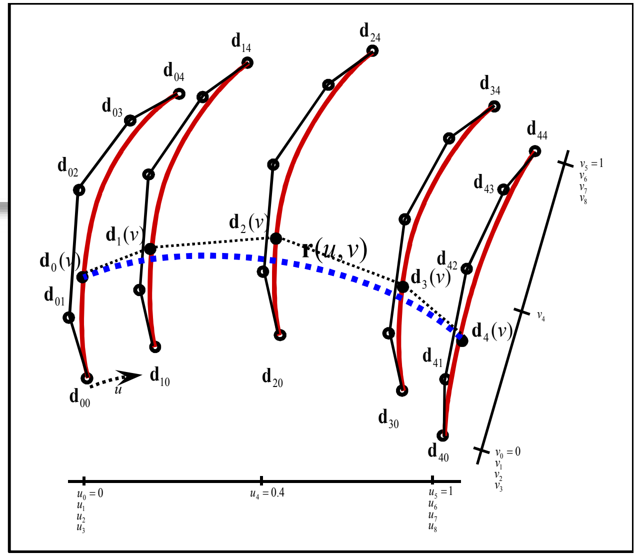
$$\mathbf{r}(u,v) = \mathbf{d}_0(v)N_0^3(u) + \mathbf{d}_1(v)N_1^3(u) + \mathbf{d}_2(v)N_2^3(u) + \mathbf{d}_3(v)N_3^3(u) + \mathbf{d}_4(v)N_4^3(u)$$

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

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

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

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;

        double* m_pKnot_U;
        int m_nNumOfKnot_U;

        double* m_pKnot_V;
        int m_nNumOfKnot_V;

        Vector** m_pCP;
        int m_nNumOfCP_U;
        int m_nNumOfCP_V;

        // member functions
        …
};
```

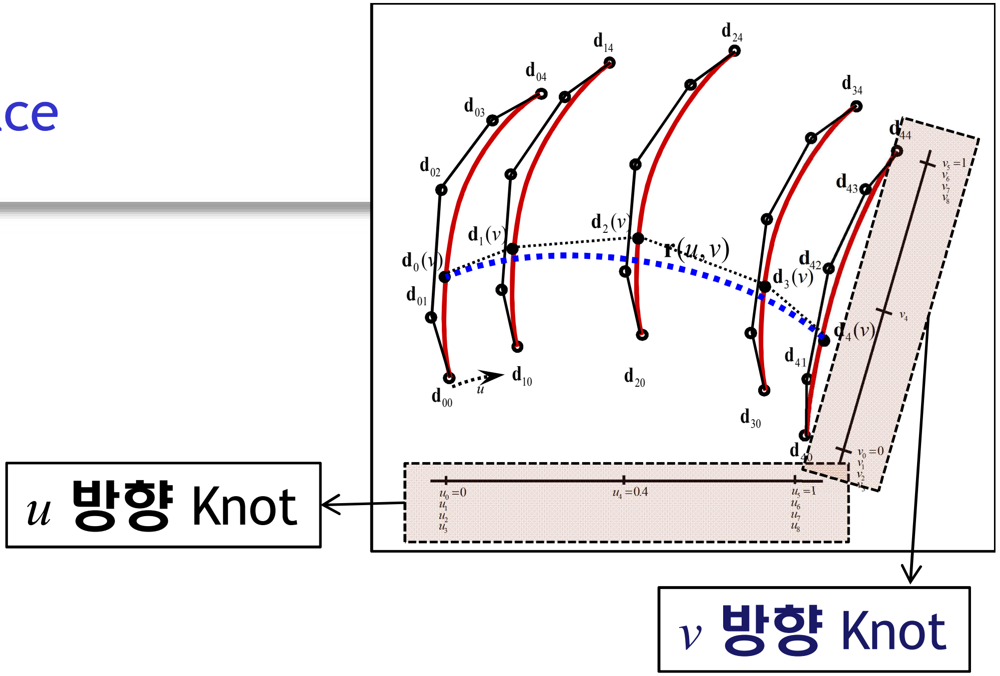
➔ 차수 (3차)

➔ u 방향 Knot와 개수 (9개)

➔ v 방향 Knot와 개수 (9개)

➔ Control Point, u, v 방향 개수 (u 방향 5개, v 방향 5개)

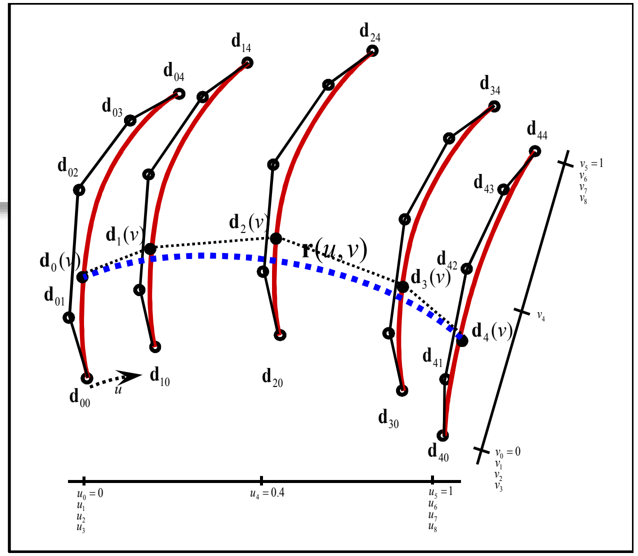
3.3.1.2 Programming Guide of Tensor-product bicubic B-spline surface
- Member Functions of Class




```
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);
};
```

➔ 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

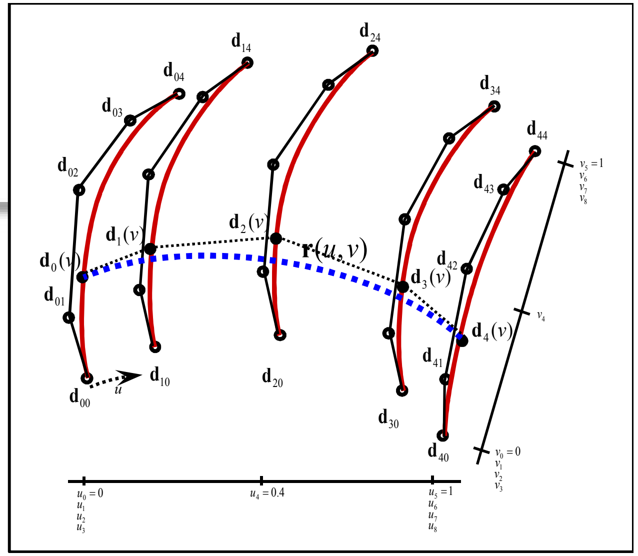
➔ B-spline Basis Function 계산

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, v에 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, v에 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;
}
```

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

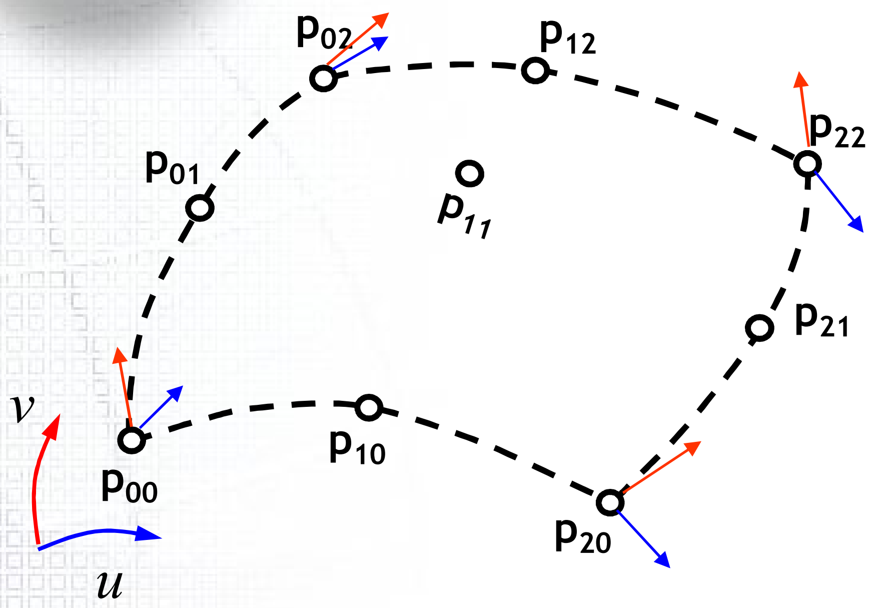
$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\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

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

3.3.2 B-spline surfaces Interpolation

3.3.2.1 bicubic B-spline Surface Interpolation 개요 (1)

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

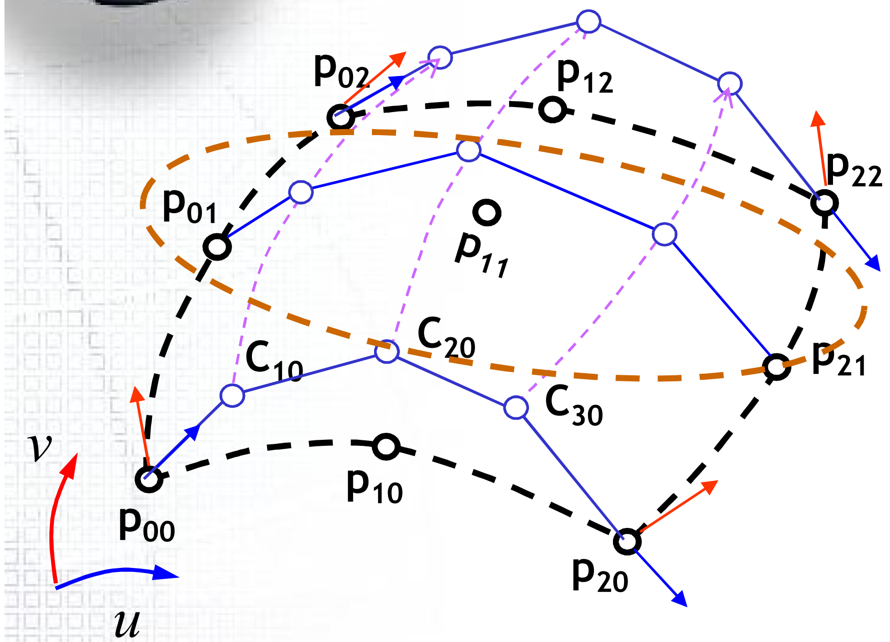


$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{0,0} & \mathbf{d}_{1,0} & \mathbf{d}_{2,0} & \mathbf{d}_{3,0} & \mathbf{d}_{4,0} \\ \mathbf{d}_{0,1} & \mathbf{d}_{1,1} & \mathbf{d}_{2,1} & \mathbf{d}_{3,1} & \mathbf{d}_{4,1} \\ \mathbf{d}_{0,2} & \mathbf{d}_{1,2} & \mathbf{d}_{2,2} & \mathbf{d}_{3,2} & \mathbf{d}_{4,2} \\ \mathbf{d}_{0,3} & \mathbf{d}_{1,3} & \mathbf{d}_{2,3} & \mathbf{d}_{3,3} & \mathbf{d}_{4,3} \\ \mathbf{d}_{0,4} & \mathbf{d}_{1,4} & \mathbf{d}_{2,4} & \mathbf{d}_{3,4} & \mathbf{d}_{4,4} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

5x5 개의 조정점을 구하면 곡면을 생성할 수 있다.

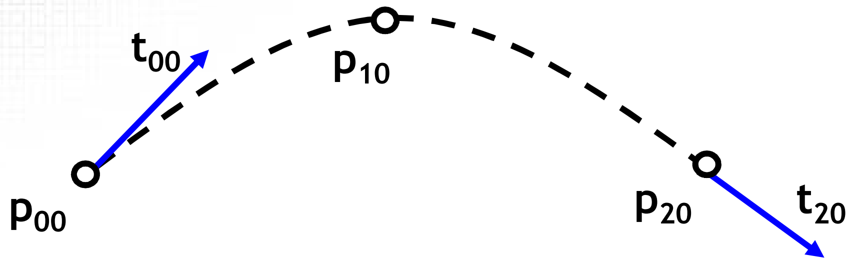
3.3.2.1 bicubic B-spline Surface Interpolation 개요 (2)

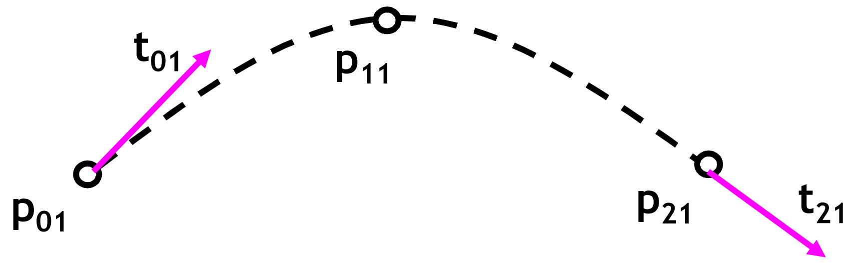
- Given: 곡면상의 9개 점과 4 꼭지점에서의 u, v방향의 접선벡터
- Find: Control Points of bicubic B-spline Surface



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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{3}{\Delta_s} & \frac{3}{\Delta_s} & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & 0 \\ 0 & 0 & 0 & -\frac{3}{\Delta_E} & \frac{3}{\Delta_E} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_{00} \\ C_{10} \\ C_{20} \\ C_{30} \\ C_{40} \end{bmatrix} = \begin{bmatrix} P_{00} \\ t_{00} \\ P_{10} \\ t_{20} \\ P_{20} \end{bmatrix}$$





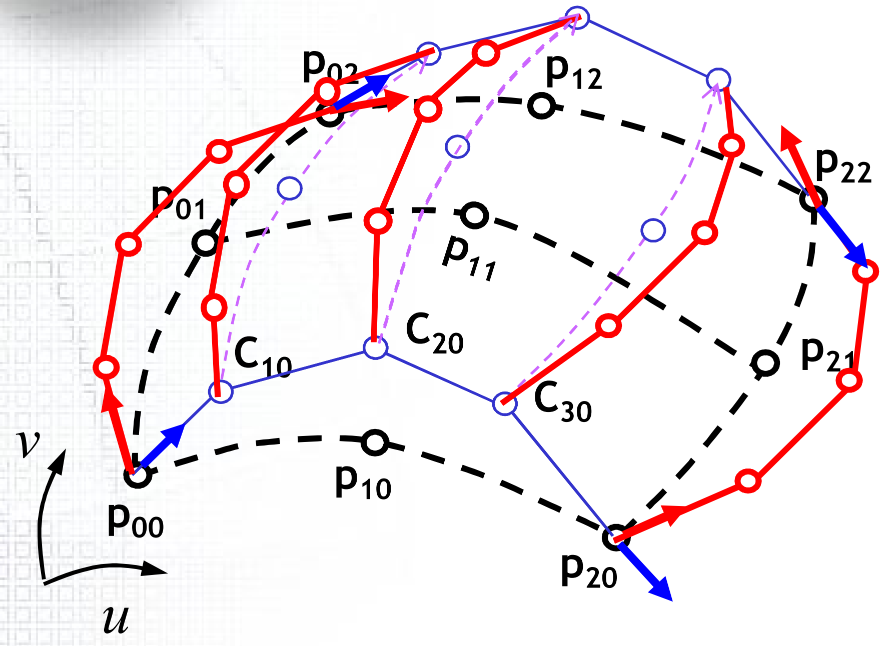
Bessel end condition:

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

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

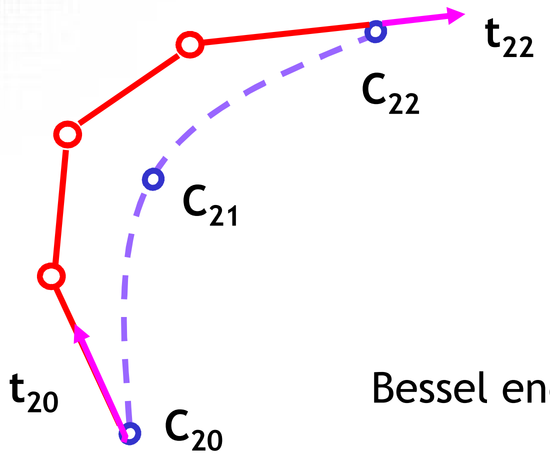
3.3.2.1 bicubic B-spline Surface Interpolation 개요 (3)

- 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.

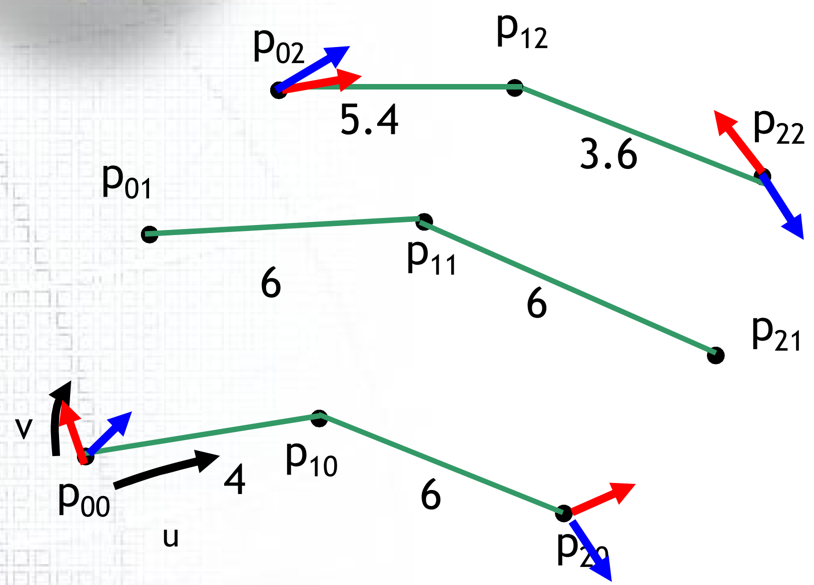
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{3}{\Delta_s} & \frac{3}{\Delta_s} & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & 0 \\ 0 & 0 & 0 & -\frac{3}{\Delta_E} & \frac{3}{\Delta_E} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}_{20} \\ \mathbf{d}_{21} \\ \mathbf{d}_{22} \\ \mathbf{d}_{23} \\ \mathbf{d}_{24} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{20} \\ \mathbf{t}_{20} \\ \mathbf{C}_{21} \\ \mathbf{t}_{22} \\ \mathbf{C}_{22} \end{bmatrix}$$



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

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (1)

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



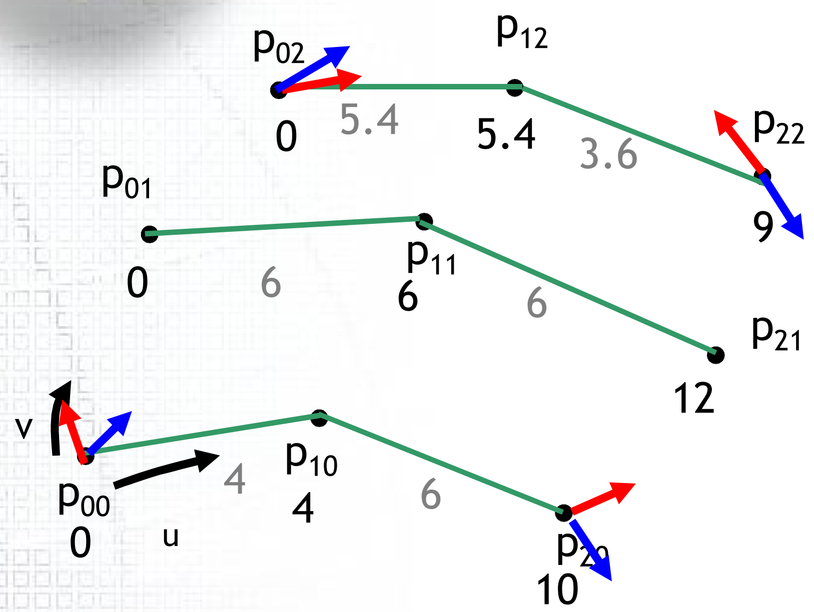
1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다

❑ Given

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

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (2)

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



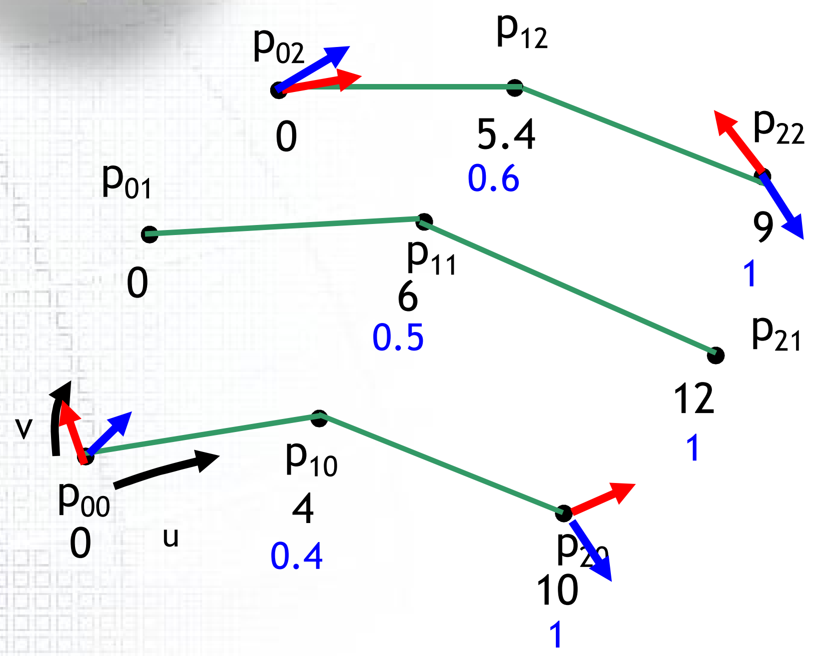
❑ Given

- 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

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (3)

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



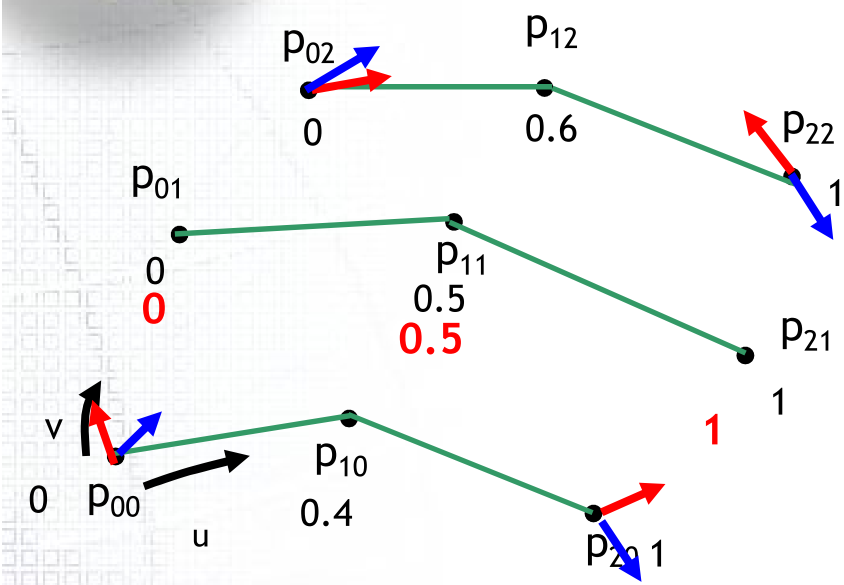
1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다

❑ Given

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

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (4)

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



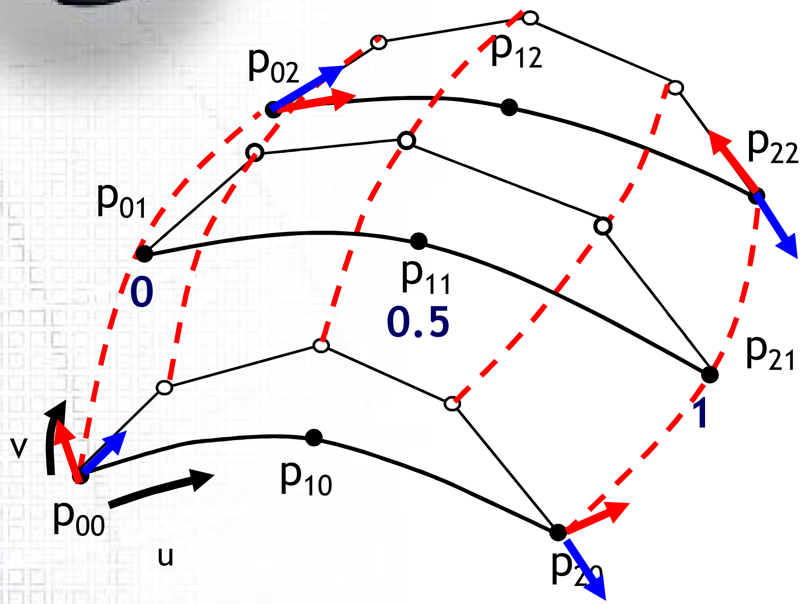
1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균하여 최종적인 u방향 knot값을 계산한다

❑ Given

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

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (5)

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



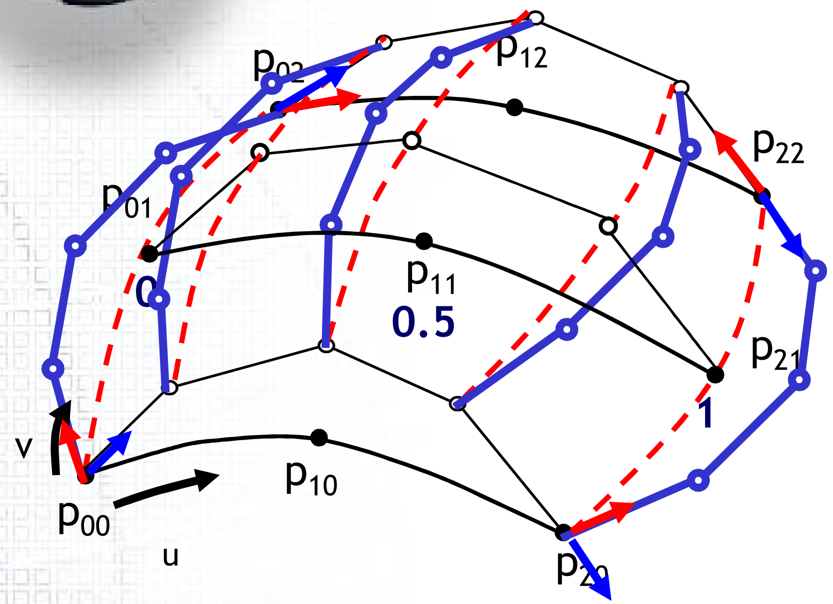
❑ Given

- 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 knot값들을 v방향으로 평균하여 최종적인 u방향 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 u뱡향과 동일한 방법으로 구함
4. u 방향 B-spline 곡선의 조정점을 보간하는 v방향 B-spline 곡선을 계산
   - u방향 B-spline 곡선의 조정정에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (빨간점선) 을 생성한다. 이 곡선의 조정점 (파란점) 이 최종 B-spline 곡면의 조정점이다

3.3.2.2 bicubic B-spline Surface Interpolation 상세 과정 (6)

- 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 knot값들을 v방향으로 평균하여 최종적인 u방향 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 조점정에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (빨간점선) 을 생성한다. 이 곡선의 조정점 (파란점) 이 최종 B-spline 곡면의 조정점이다

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

⇩

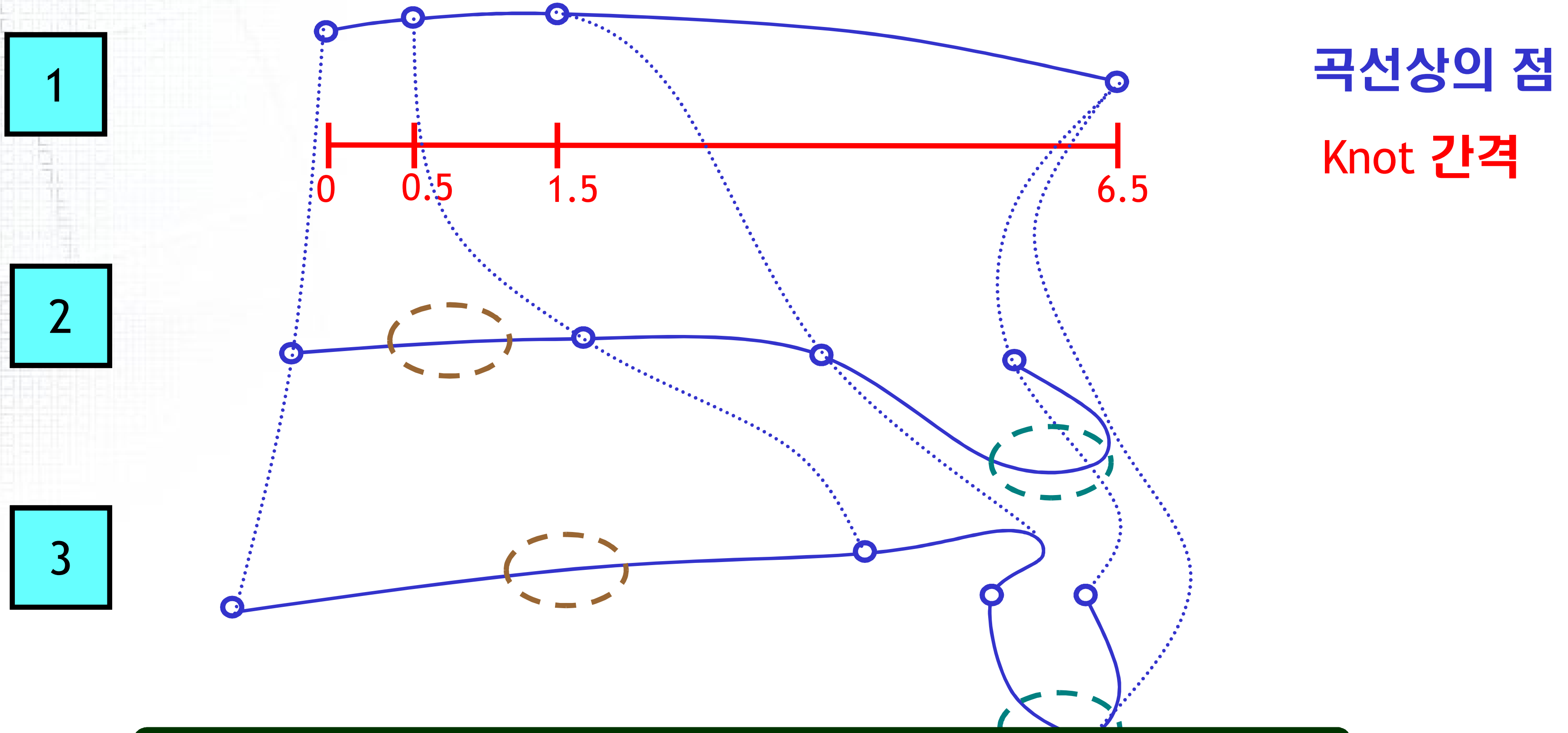
주어지는 곡면상의 점이 GRID와 같이 균일하게 주어져야 이상적인 곡면재현 가능

⇩

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.



** CAGD

그러므로 knot간격비가 비슷할 수록 곡면의 왜곡이 적다.

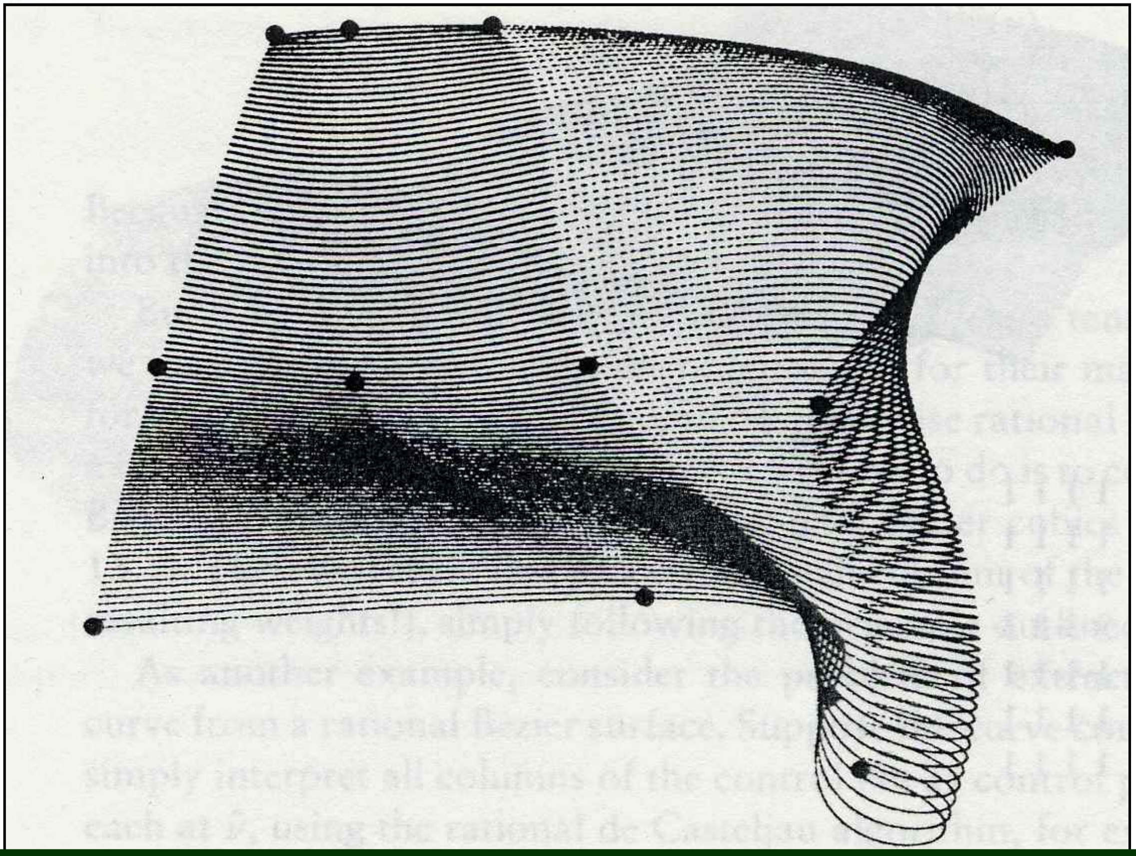
3.3.2.4 Knot 간격 차이가 주는 영향

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

1

2

3



곡선상의 점

Knot 간격

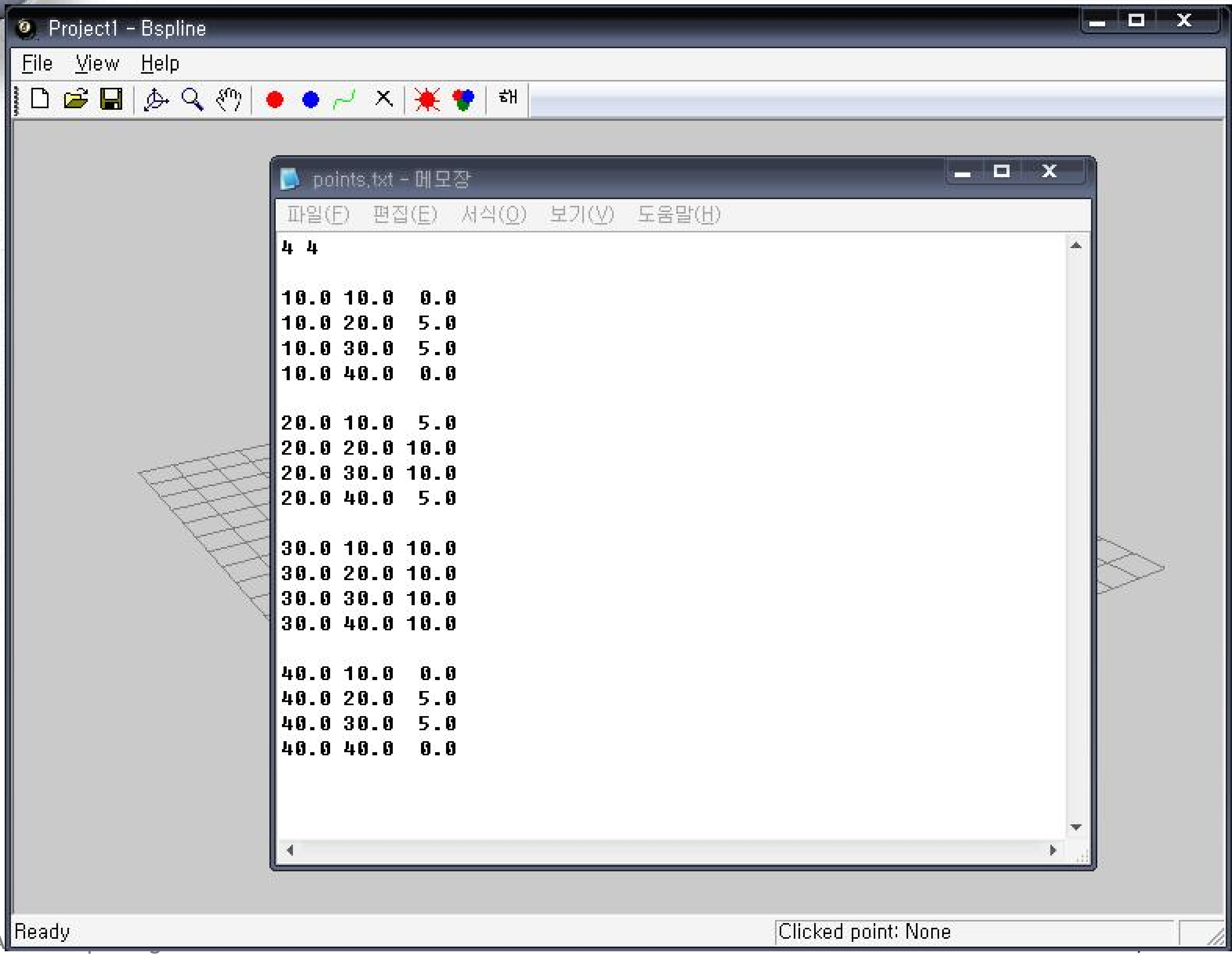
** CAGD

그러므로 knot간격비가 비슷할 수록 곡면의 왜곡이 적다.

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

일반 3차원 곡면 형상 예시

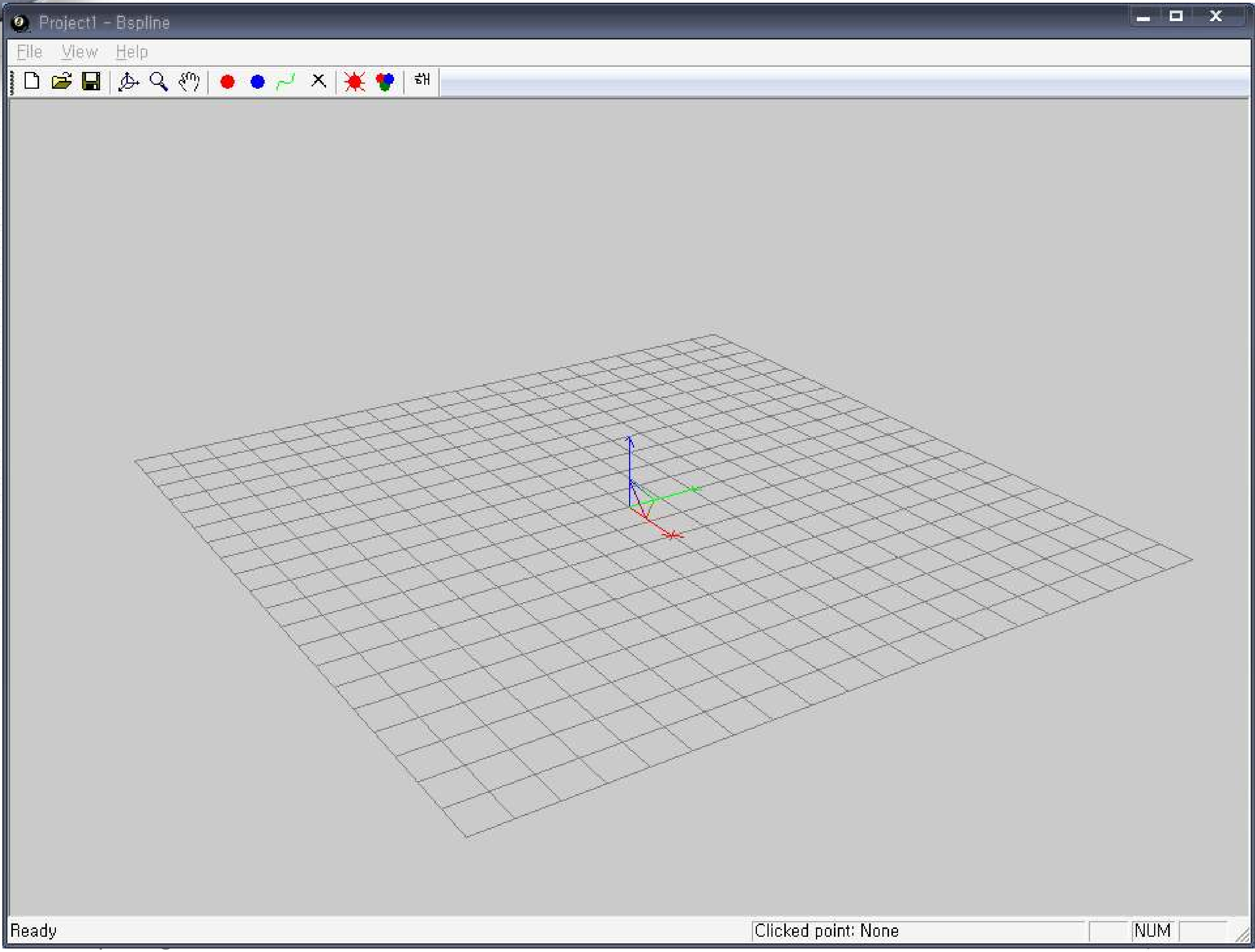
- Given: Points on Surface
- Find: bicubic B-spline Surface (Control Points of bicubic B-spline Surface)



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;
   }
```

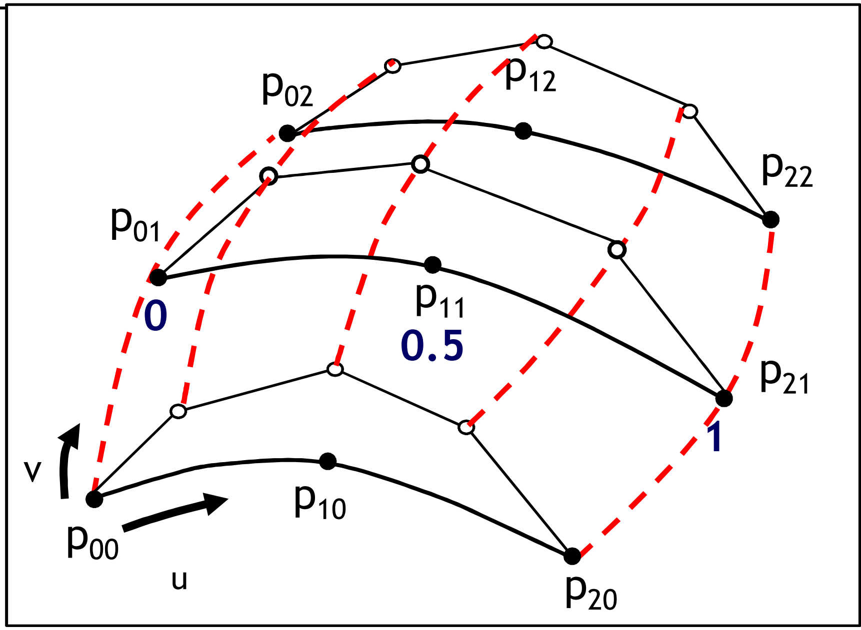
➔ Chord length를 이용하여 u, v 방향 Knot 생성

➔ 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];
    }
}
.........................
```



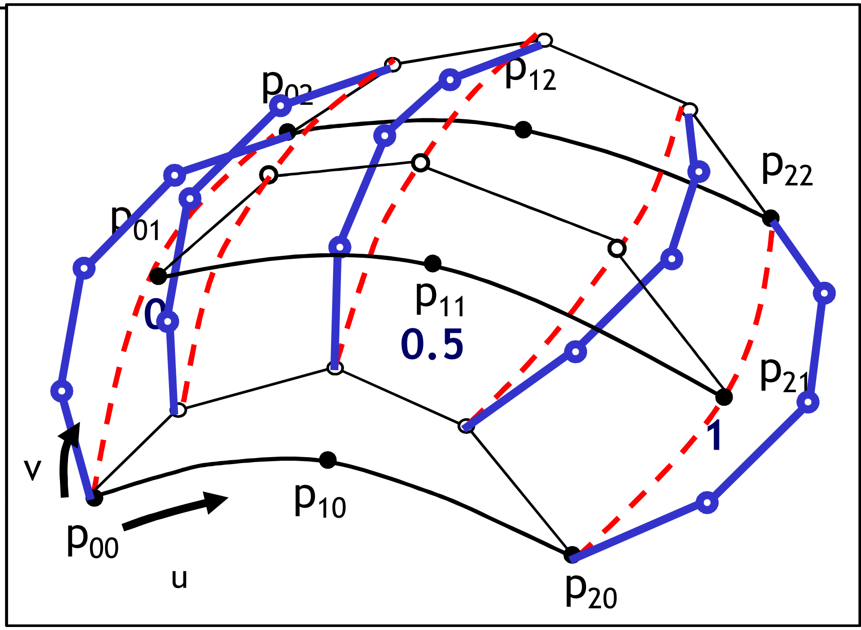
u 방향 B-spline curve 생성

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

3.4. Generation of Bezier surfaces by tensor-product approach

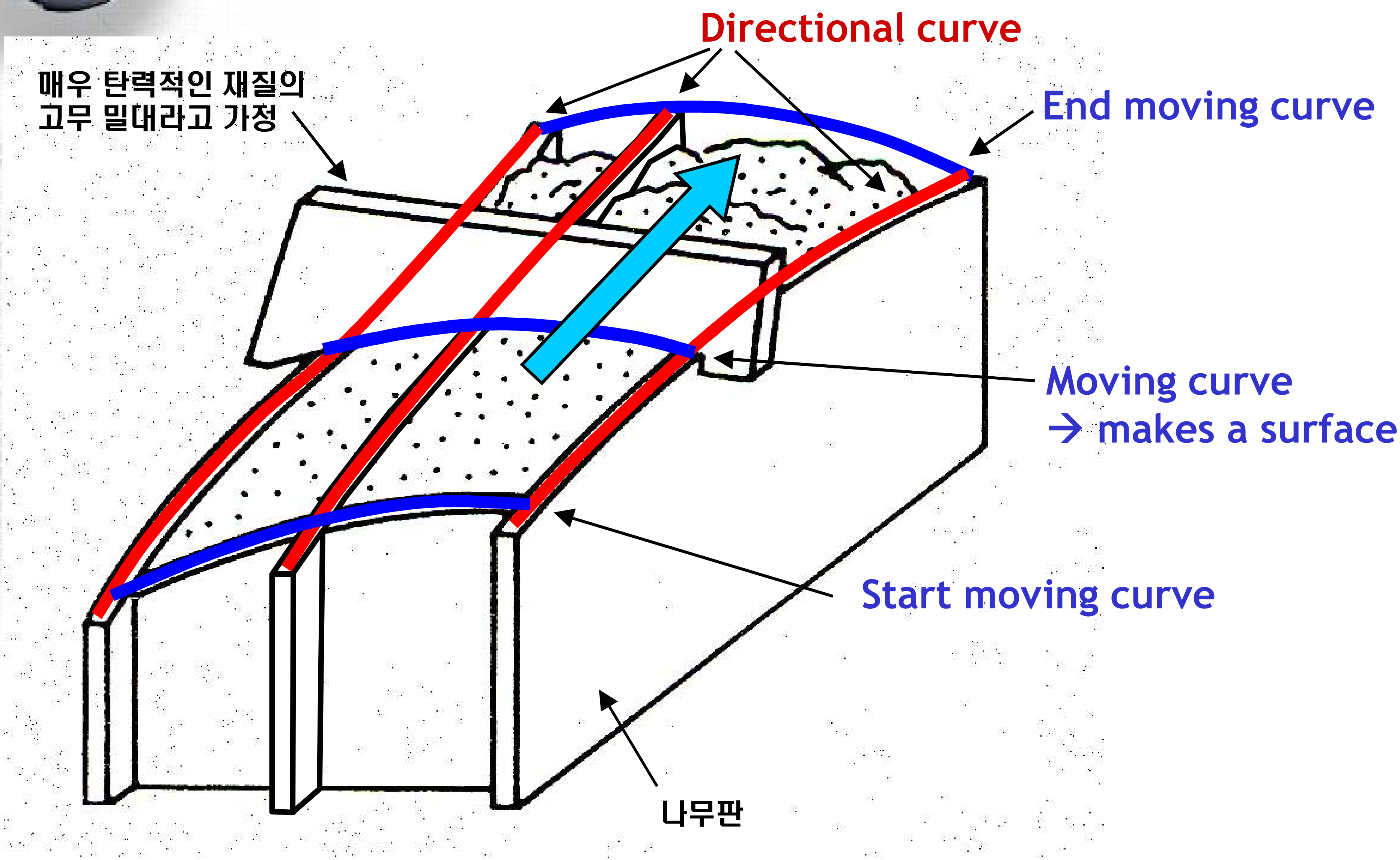
3.4.1 Tensor-product approach

3.4.2 Tensor-product biquadratic Bezier surface

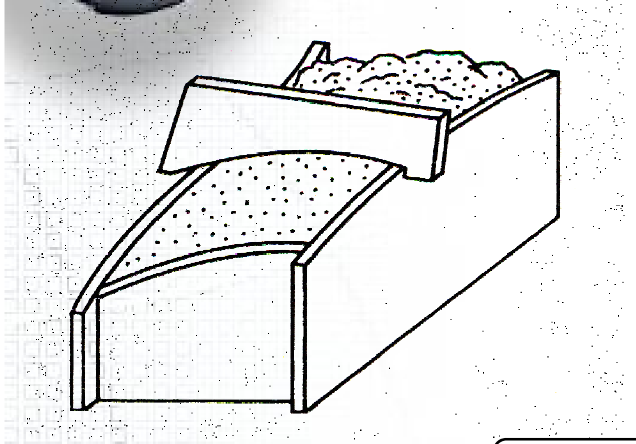
3.4.3 Tensor-product bicubic Bezier surface

Advanced
Ship
Design
Automation
Laboratory

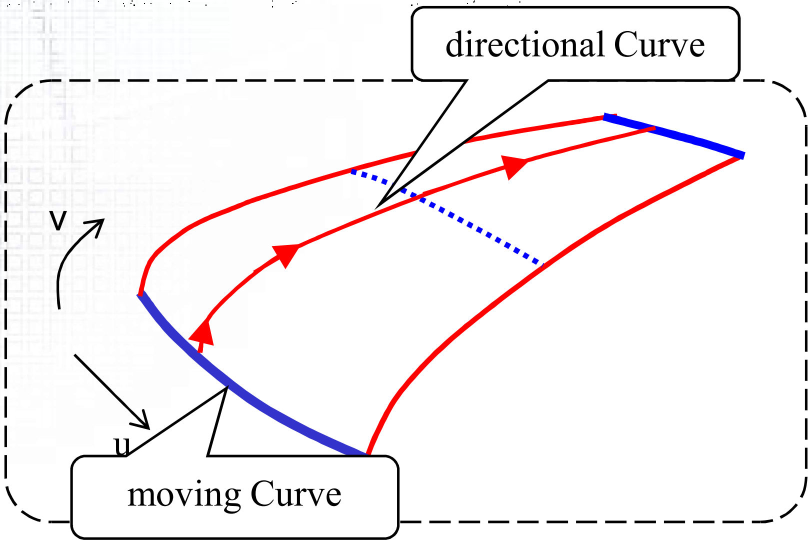
3.4.1 Tensor product approach (1)



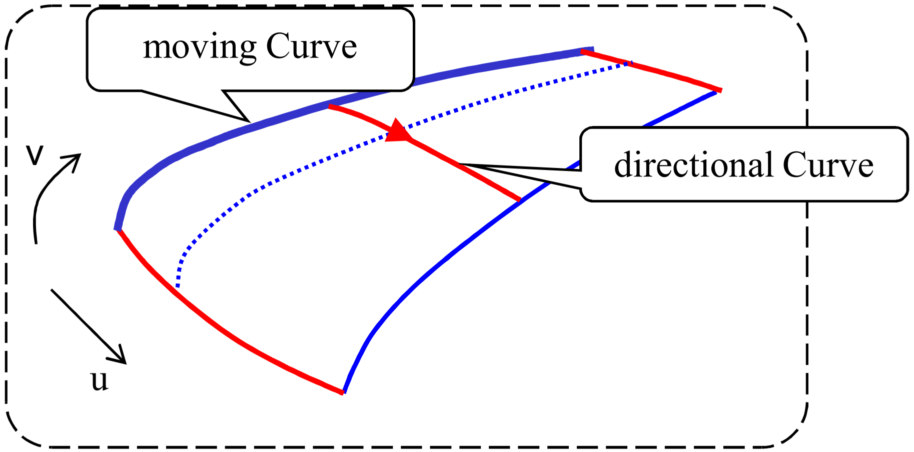
3.4.1 Tensor product approach (2)



- 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

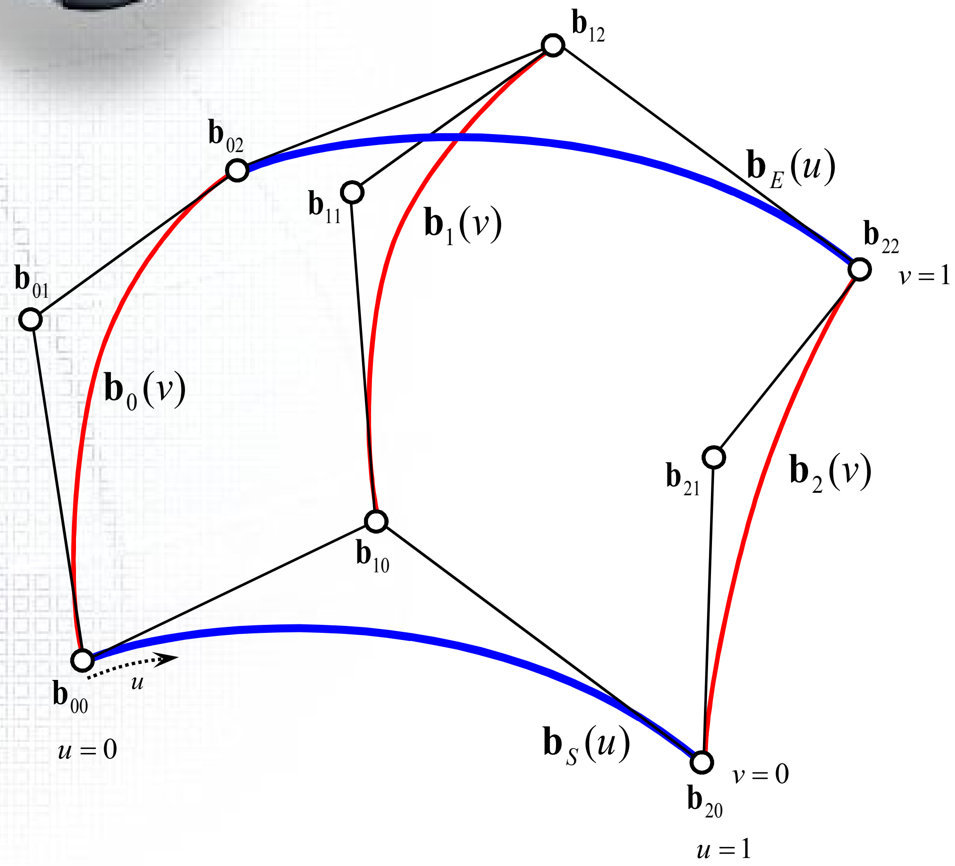


r(v) 곡선을 u 방향으로 sweeping

* CAGD, Ch14.3, Tensor Product Approach

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

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

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

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

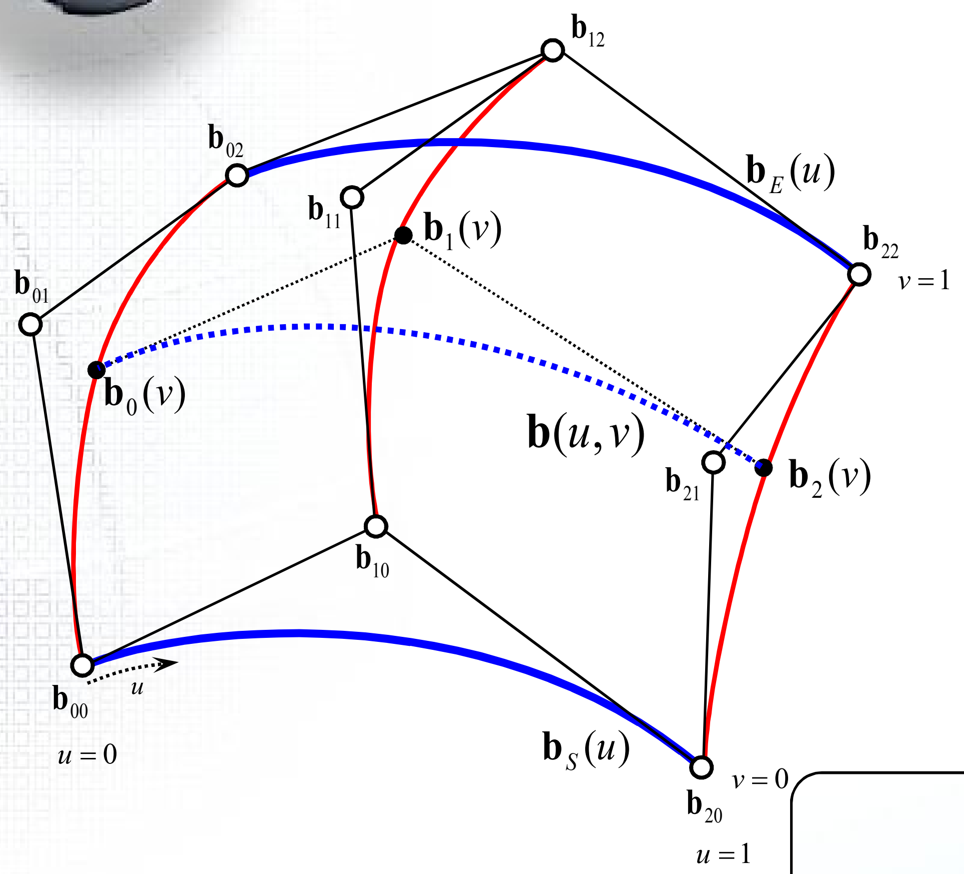
$$\mathbf{b}_1(v) = \mathbf{b}_{10}B_0^2(v) + \mathbf{b}_{11}B_1^2(v) + \mathbf{b}_{12}B_2^2(v)$$

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

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

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:

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

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

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

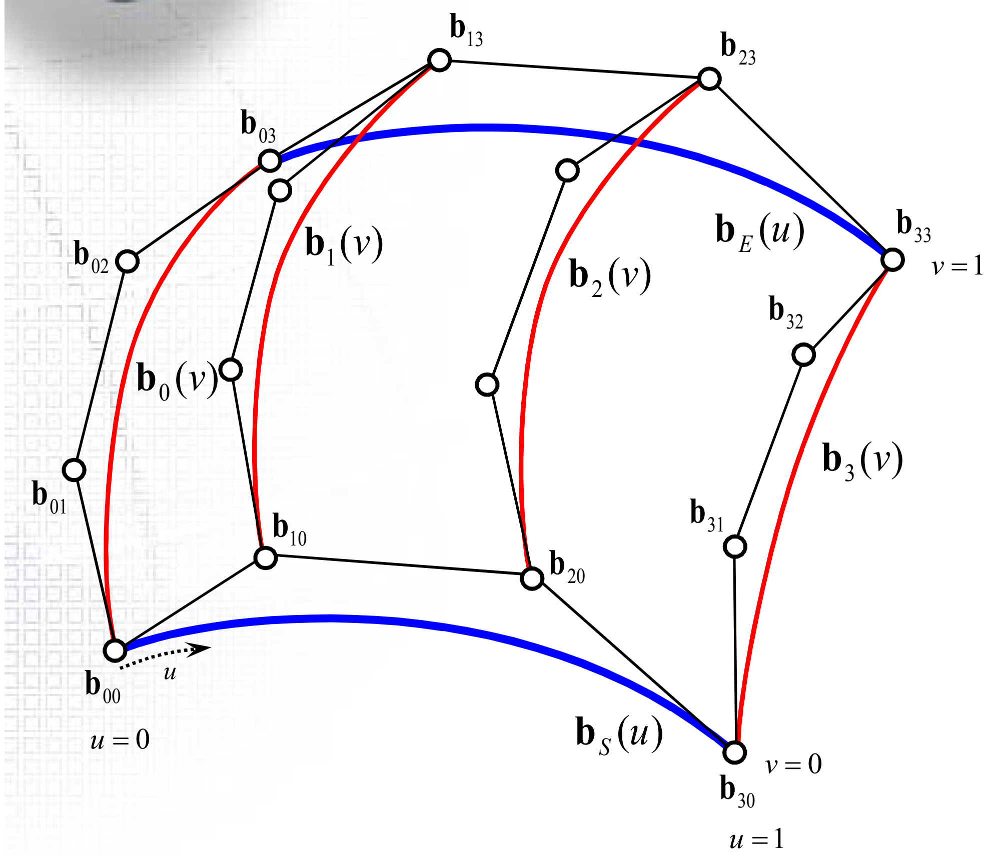
$$\mathbf{b}(u,v) = \begin{bmatrix} B_0^2(u) & B_1^2(u) & B_2^2(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} B_0^2(v) \\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

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

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

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

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

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

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

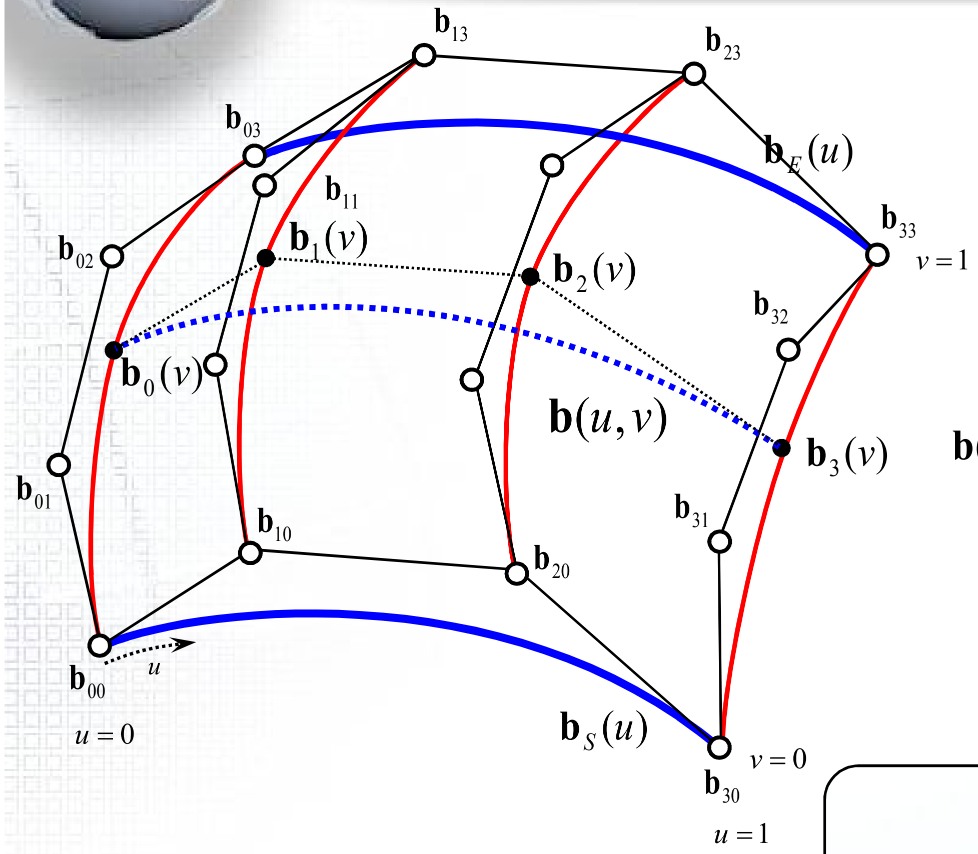
$$\mathbf{b}_2(v)=\mathbf{b}_{20}B_0^3(v)+\mathbf{b}_{21}B_1^3(v)+\mathbf{b}_{22}B_2^3(v)+\mathbf{b}_{23}B_3^3(v)$$

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

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

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:

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

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

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

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

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

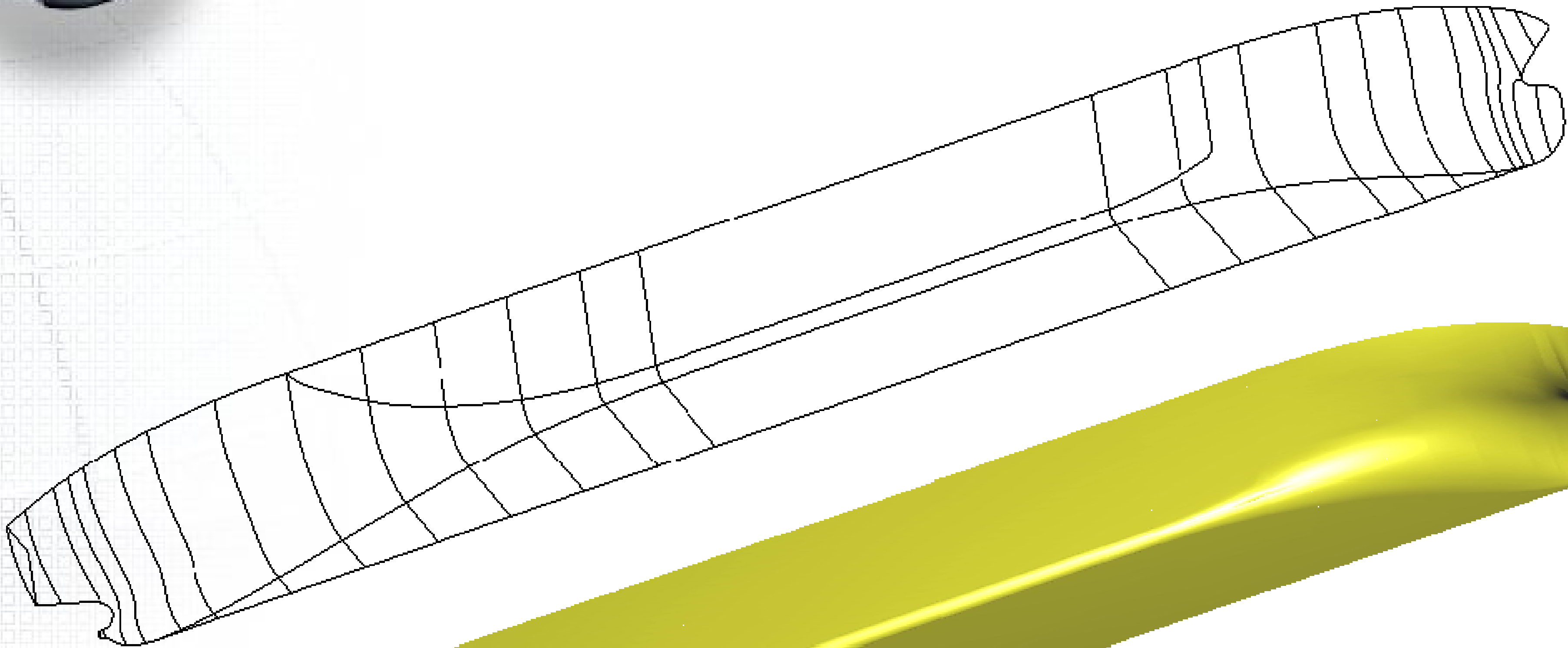
Ch 4. Term Project

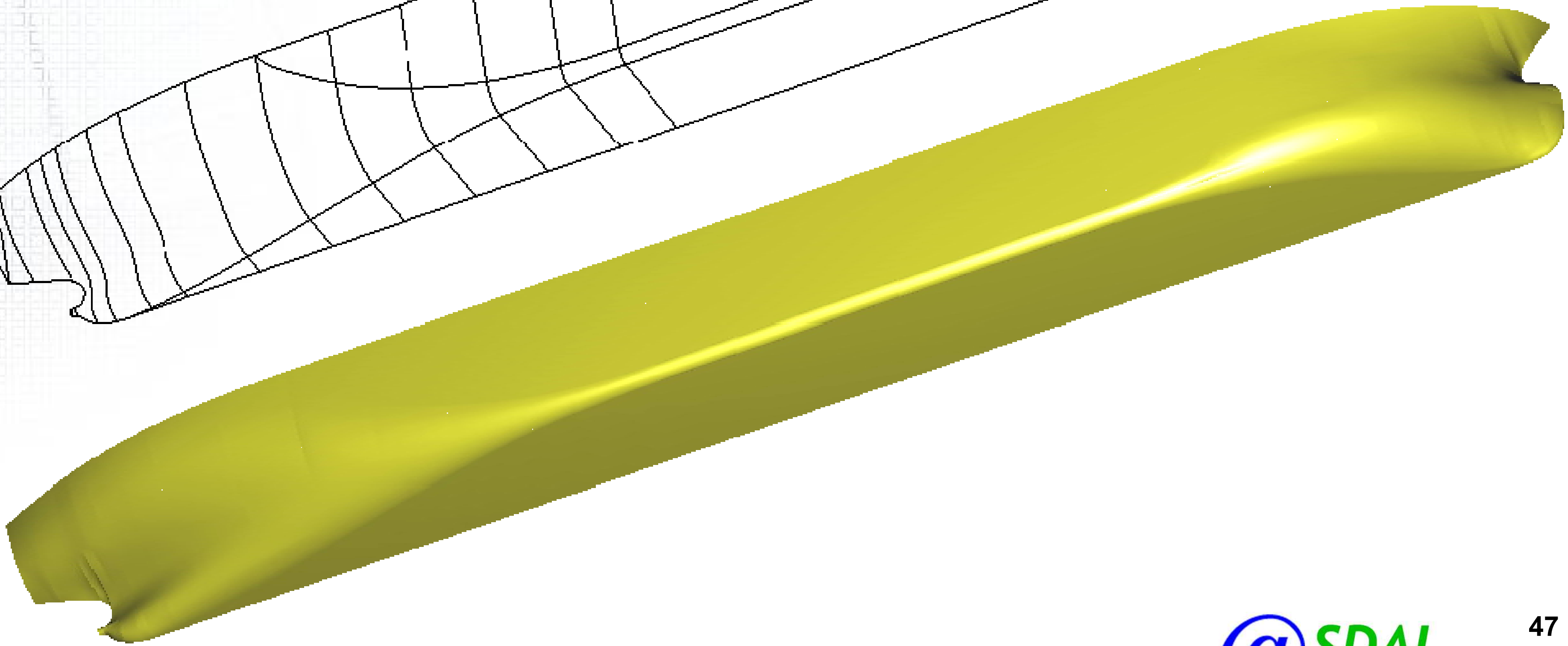
Generation Ship hull surfaces
by interpolating given points

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

Advanced
Ship
Design
Automation
Laboratory

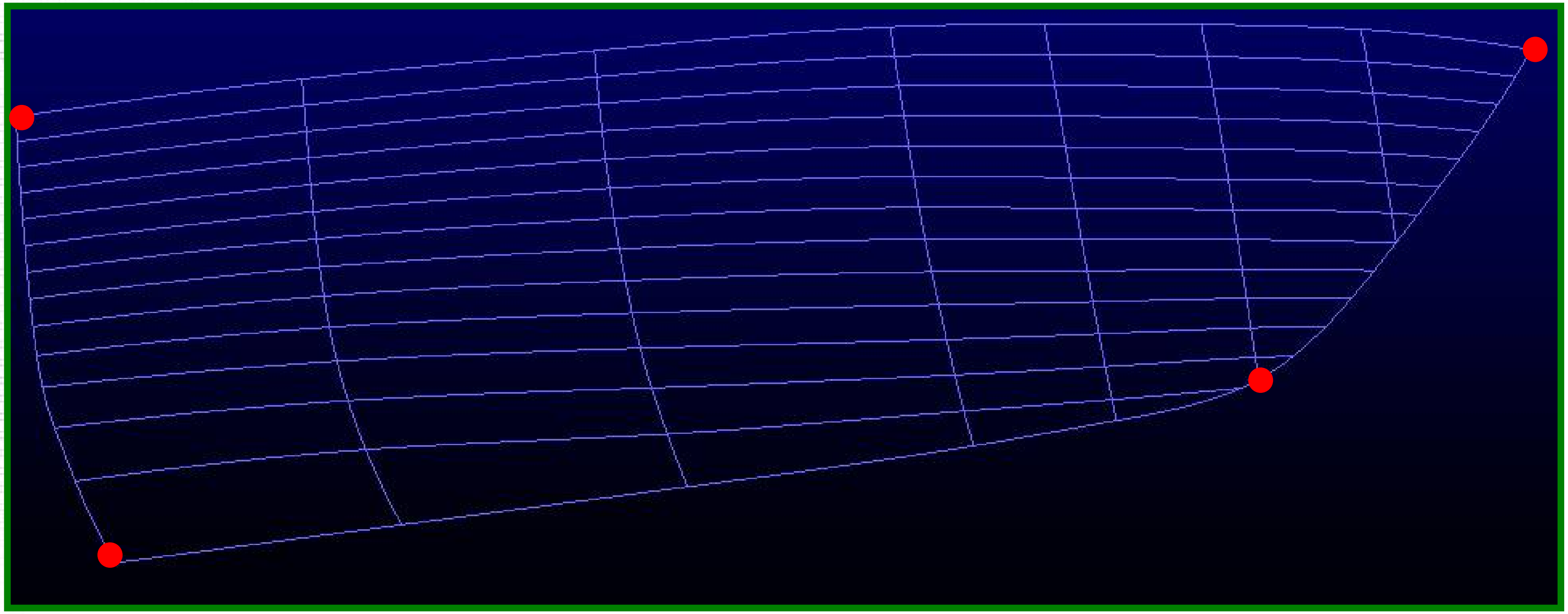
4.1 단일 B-spline 곡면 patch를 이용한 선형곡면 생성 프로그램 구현





4.2 간단한 요트 형상의 선수부 곡선그물망 형상

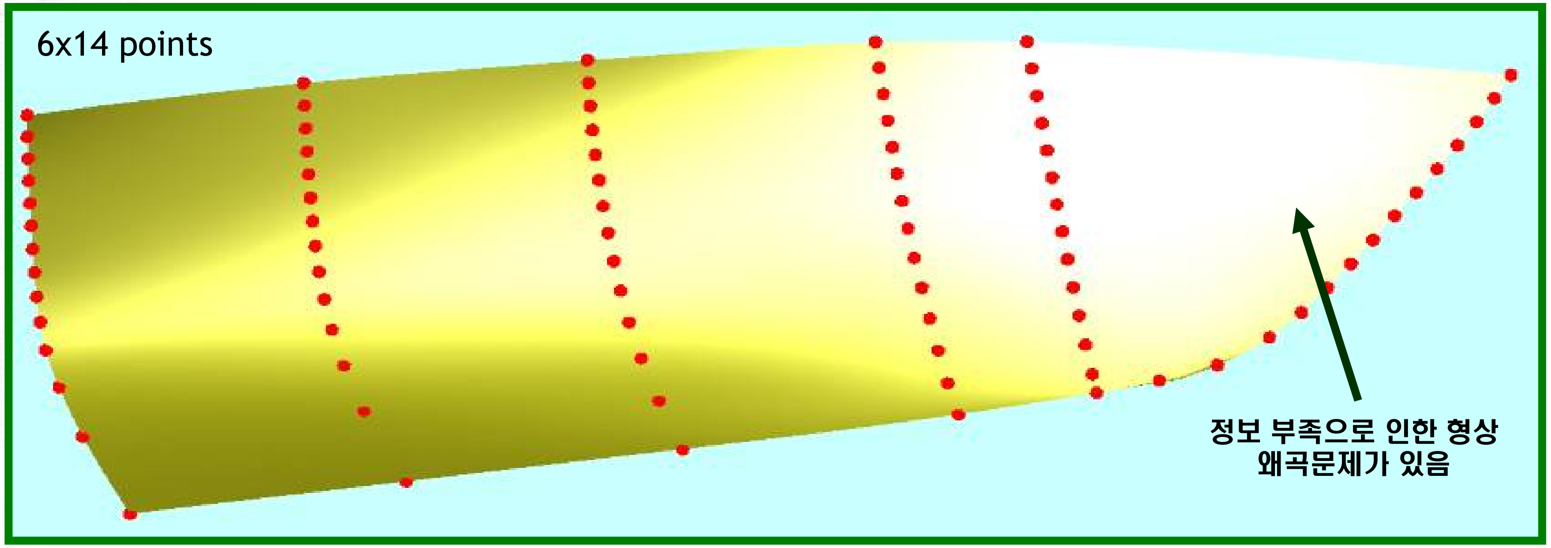
* 출처 : 서울대 조선해양공학과 2005년 2학년 교과목 『 조선해양공학계획』 강좌 중에 학생들이 설계한 선형



사각형 패치의 꼭지점 결정

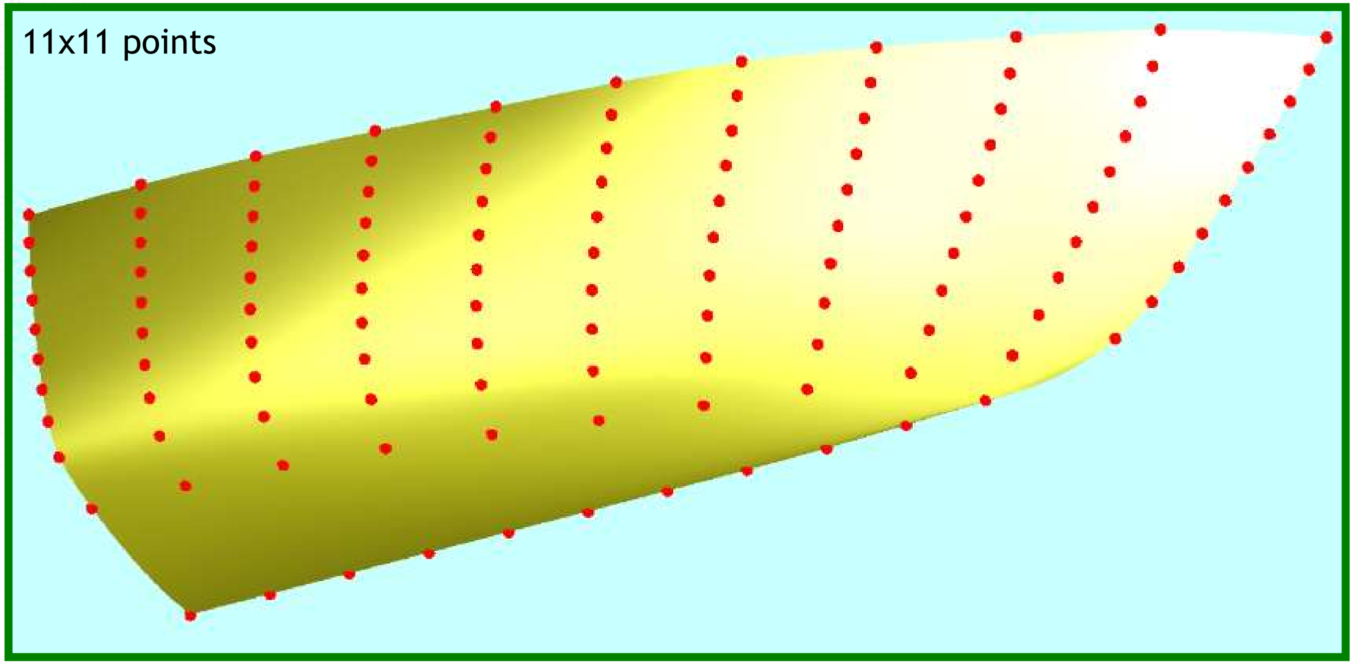
4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (1)

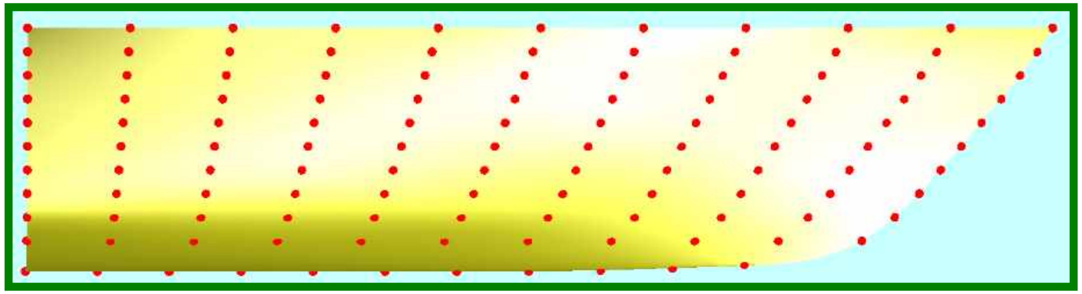
- Offset table 형식으로 점을 추출한 후,
  이 점들로 부터 bicubic B-spline 선형곡면을 생성한 결과



4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (2)

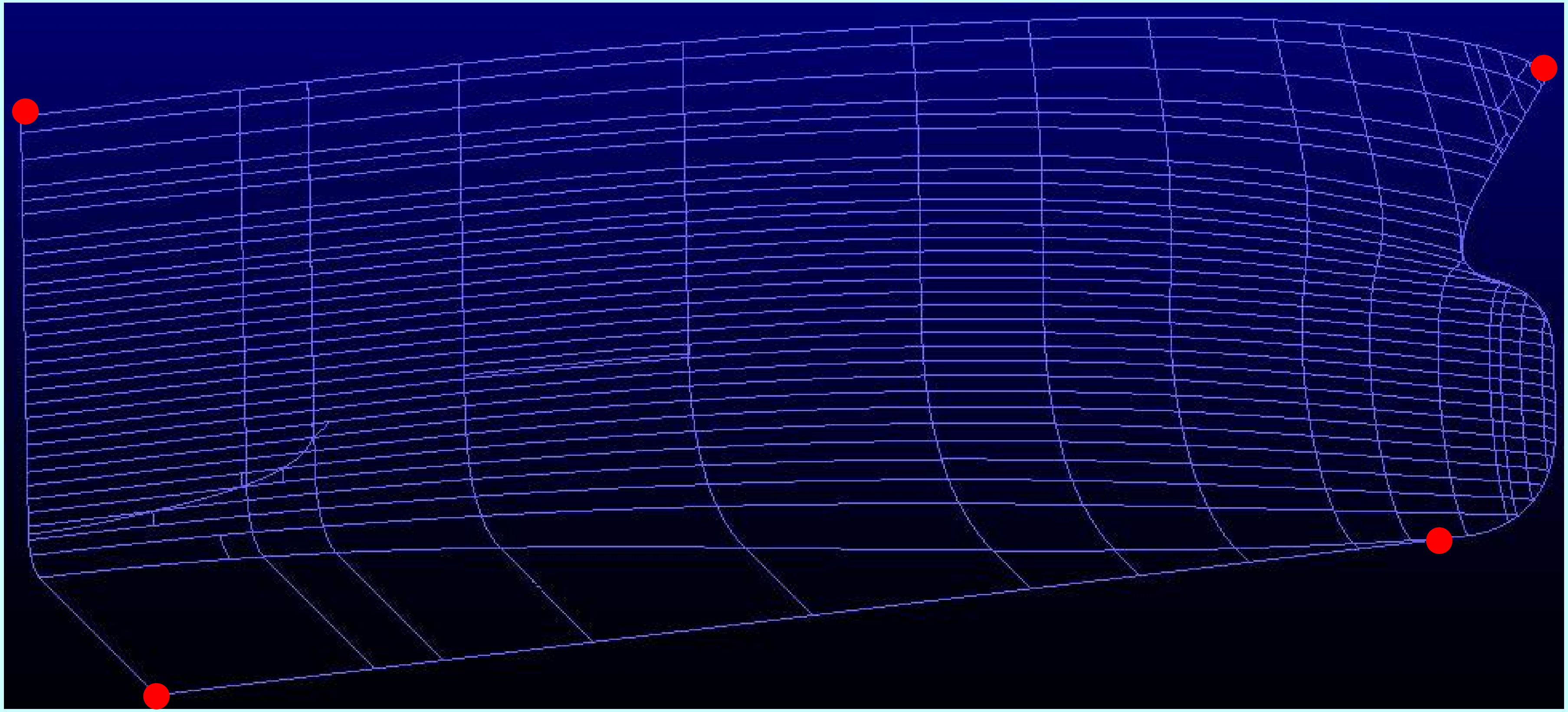
- ❑ 점들의 x좌표 사이의 거리가 일정하도록 점을 추출한 후, 이 점들로부터 bicubic B-spline 선형곡면을 생성한 결과





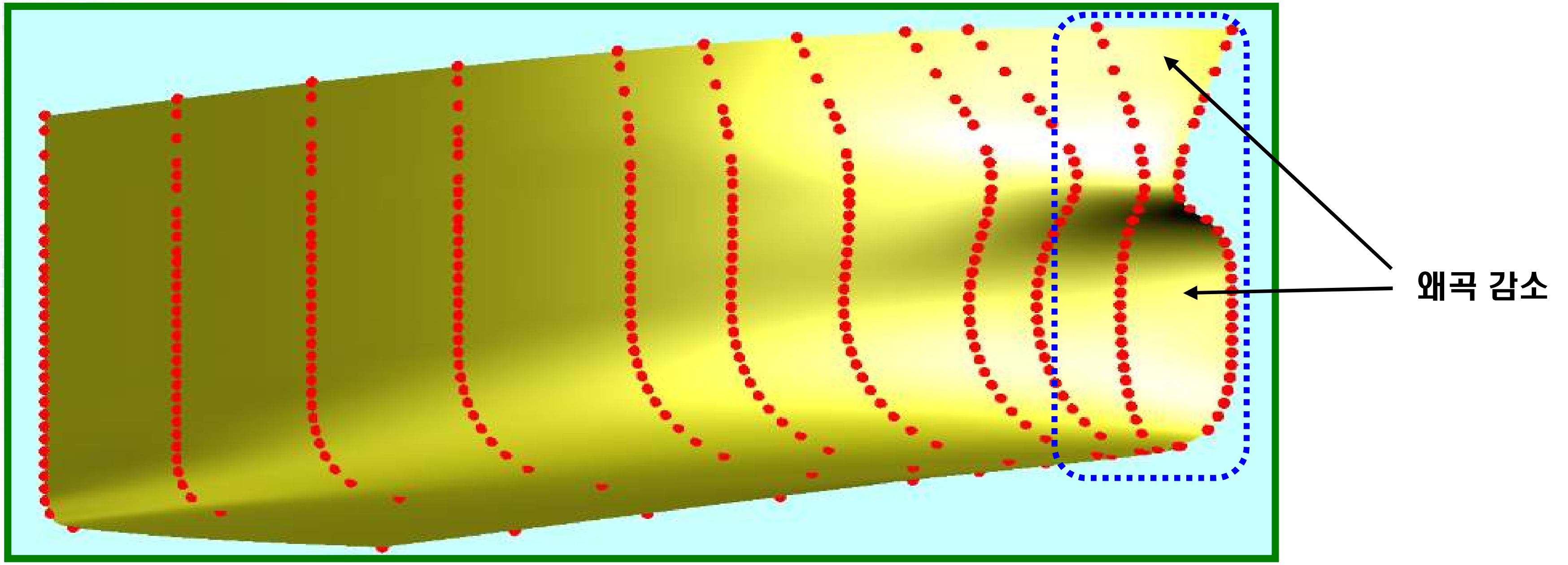
4.4 구상선수를 갖는 단축선의 선수부 곡선그물망 형상

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



참고 문헌

- **이규열, 조두연, 노명일, 차주환, 전산선박설계**, 3th Ed., 2003.9
- G. Farin, Curves and Surfaces for CAGD, 5^{th} Edi., Academic Press, 2002
- G. Farin and D. Hansford, The Essemtials of CAGD, A K Peters, 2000

참고 슬라이드

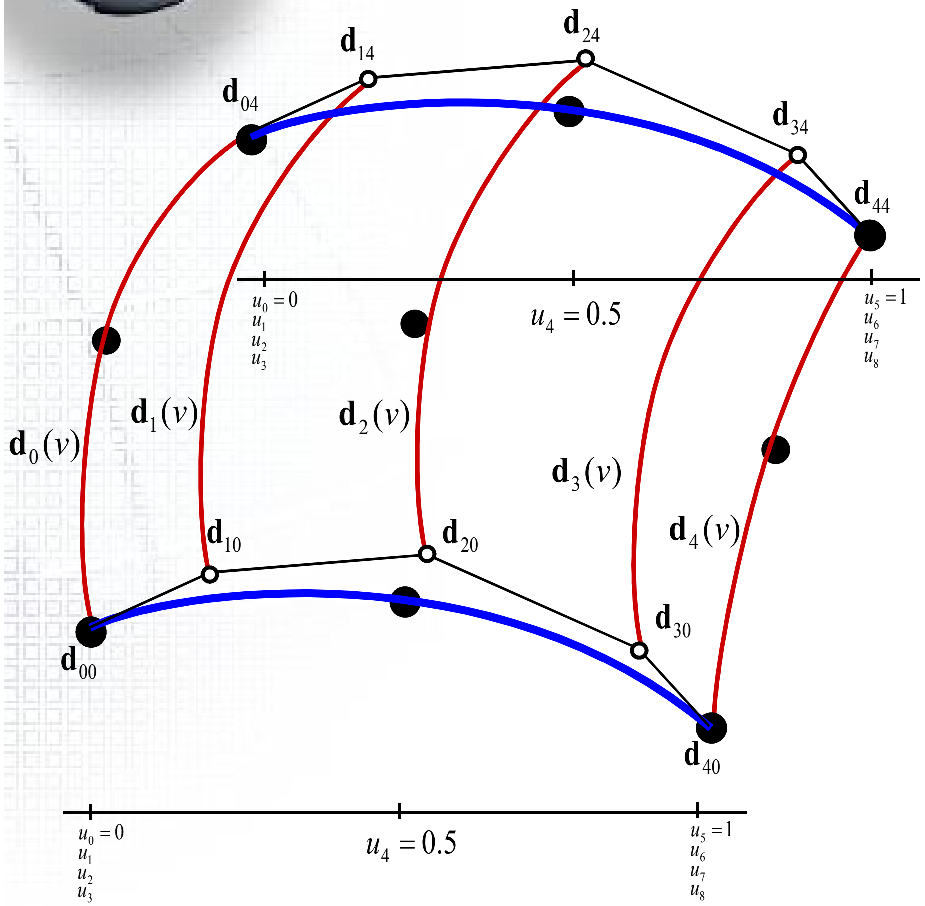
Advanced
Ship
Design
Automation
Laboratory

Bezier Curve and B-spline Curve

항목		Bezier Curve	B-spline Curve
Make Curve	Given	Bezier Control Point $\mathbf{b}_i$ Parameter t Bernstein Polynomial Func. $B_i^n(t)$	B-spline Control Point $\mathbf{d}_i$ Parameter u B-spline Basis Func. $N_i^n(u)$
	Find	Bezier Curve $\mathbf{r}(t)$ $\mathbf{r}(t)=\mathbf{b}_0B_0^n(t)+\mathbf{b}_1B_1^n(t)+.......+\mathbf{b}_nB_n^n(t).$	B-spline Curve $\mathbf{r}(u)$ $\mathbf{r}(u)=\mathbf{d}_0N_0^3(u)+\mathbf{d}_1N_1^3(u)+\mathbf{d}_2N_2^3(u)+\cdots+\mathbf{d}_{D-1}N_{D-1}^3(u)$
Basis Function (*Blending*)		Bernstein Polynomial Function $B_i^n(t)=\binom{n}{i}t^i(1-t)^{n-i},$ $\binom{n}{i}={}_nC_i=\begin{cases}\dfrac{n!}{i!(n-i)!} & \textbf{if } 0\le i\le n\\ \mathbf{0} & \textbf{else}\end{cases}$	B-spline Basis Function (Cox-de boor Recursive Formula) $N_i^n(u)=\dfrac{u-u_{i-1}}{u_{i+n-1}-u_{i-1}}N_i^{n-1}(u)+\dfrac{u_{i+n}-u}{u_{i+n}-u_i}N_{i+1}^{n-1}(u)$ $N_i^0(u)=\begin{cases}1 & \textbf{if } u_{i-1}\le u<u_i\\ 0 & \textbf{else}\end{cases},\ \sum_{i=0}^{D-1}N_i^n(u)=1$
Constructive Approach		de Casteljau Algorithm $\mathbf{b}_i^k(t)=(1-t)\mathbf{b}_i^{k-1}+t\mathbf{b}_{i+1}^{k-1}$	de Boor Algorithm $\mathbf{d}_i^k(u)=\dfrac{u_{i+n-k}-u}{u_{i+n-k}-u_{i-1}}\mathbf{d}_{i-1}^{k-1}(u)+\dfrac{u-u_{i-1}}{u_{i+n-k}-u_{i-1}}\mathbf{d}_i^{k-1}(u)$
Interpolation	Given	Points on Curve: $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$	Points on Curve: $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$
	Find	Bezier Control Point $\mathbf{b}_i$	B-spline Control Point $\mathbf{d}_i$

3.3.2.1 B-spline surface interpolation 과정 (1)

- Given: 곡면 상의 9개 점
- Find: Control Points of bicubic B-spline Surface



- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves

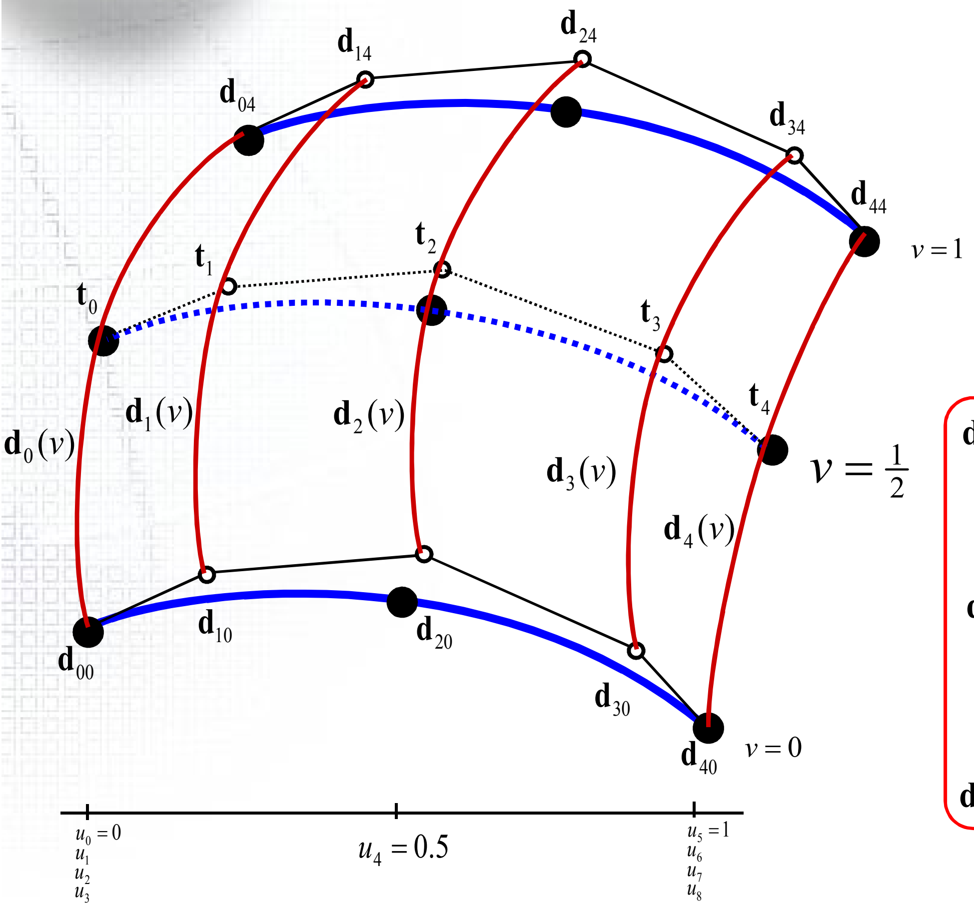
$\mathbf{d}_0(1) = \mathbf{d}_{04} \quad \mathbf{d}_1(1) = \mathbf{d}_{14} \quad \mathbf{d}_2(1) = \mathbf{d}_{24} \quad \mathbf{d}_3(1) = \mathbf{d}_{34} \quad \mathbf{d}_4(1) = \mathbf{d}_{44}$

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

$\mathbf{d}_0(0) = \mathbf{d}_{00} \quad \mathbf{d}_1(0) = \mathbf{d}_{10} \quad \mathbf{d}_2(0) = \mathbf{d}_{20} \quad \mathbf{d}_3(0) = \mathbf{d}_{30} \quad \mathbf{d}_4(0) = \mathbf{d}_{40}$

3.3.2.1 B-spline surface interpolation 과정 (2)

- Given: 곡면 상의 9개 점
- Find: Control Points of bicubic B-spline Surface

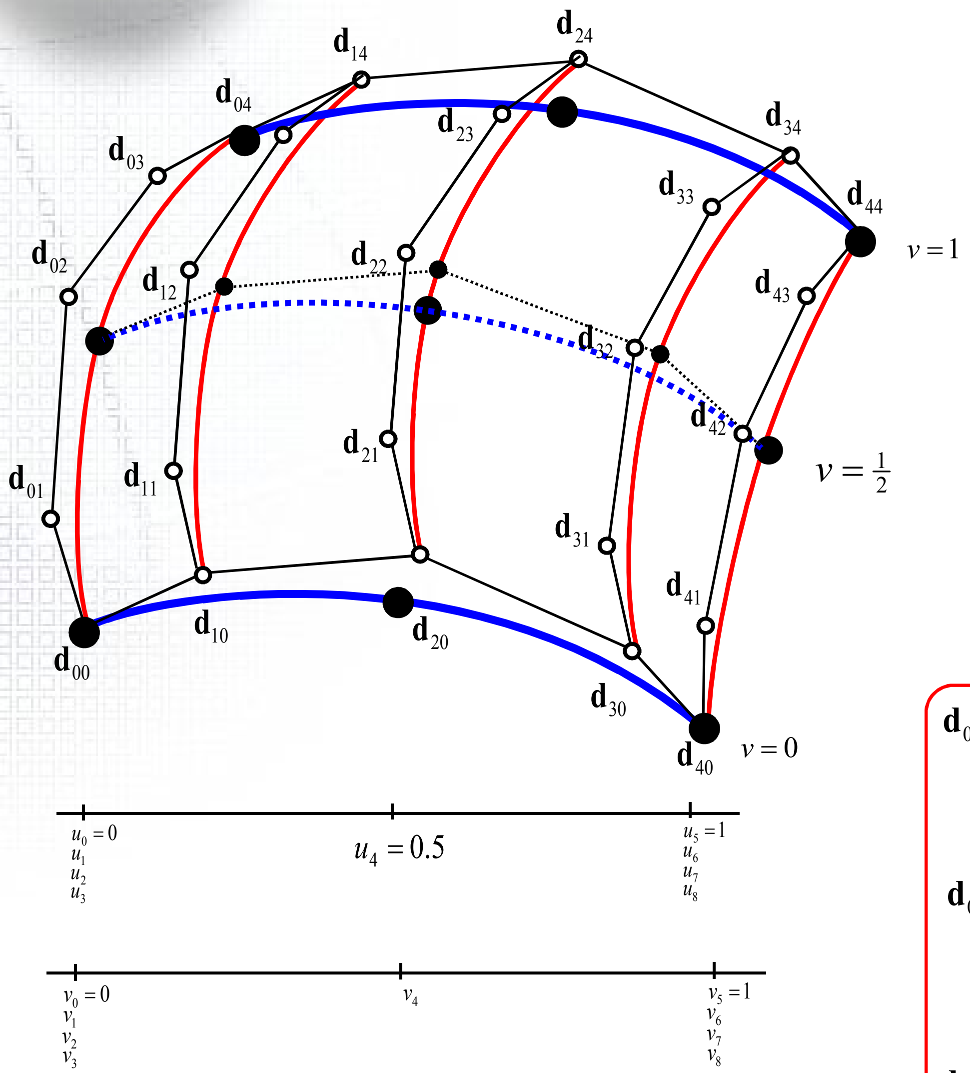


- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves
- (2) interpolate moving curves at v=v* with same u-knots

$\mathbf{d}_0(1)=\mathbf{d}_{04}$	$\mathbf{d}_1(1)=\mathbf{d}_{14}$	$\mathbf{d}_2(1)=\mathbf{d}_{24}$	$\mathbf{d}_3(1)=\mathbf{d}_{34}$	$\mathbf{d}_4(1)=\mathbf{d}_{44}$
$\mathbf{d}_0(\frac{1}{2})=\mathbf{t}_0$	$\mathbf{d}_1(\frac{1}{2})=\mathbf{t}_1$	$\mathbf{d}_2(\frac{1}{2})=\mathbf{t}_2$	$\mathbf{d}_3(\frac{1}{2})=\mathbf{t}_3$	$\mathbf{d}_4(\frac{1}{2})=\mathbf{t}_4$
$\mathbf{d}_0(0)=\mathbf{d}_{00}$	$\mathbf{d}_1(0)=\mathbf{d}_{10}$	$\mathbf{d}_2(0)=\mathbf{d}_{20}$	$\mathbf{d}_3(0)=\mathbf{d}_{30}$	$\mathbf{d}_4(0)=\mathbf{d}_{40}$
$\mathbf{d}_0(v)$	$\mathbf{d}_1(v)$	$\mathbf{d}_2(v)$	$\mathbf{d}_3(v)$	$\mathbf{d}_4(v)$

3.3.2.1 B-spline surface interpolation 과정 (3)

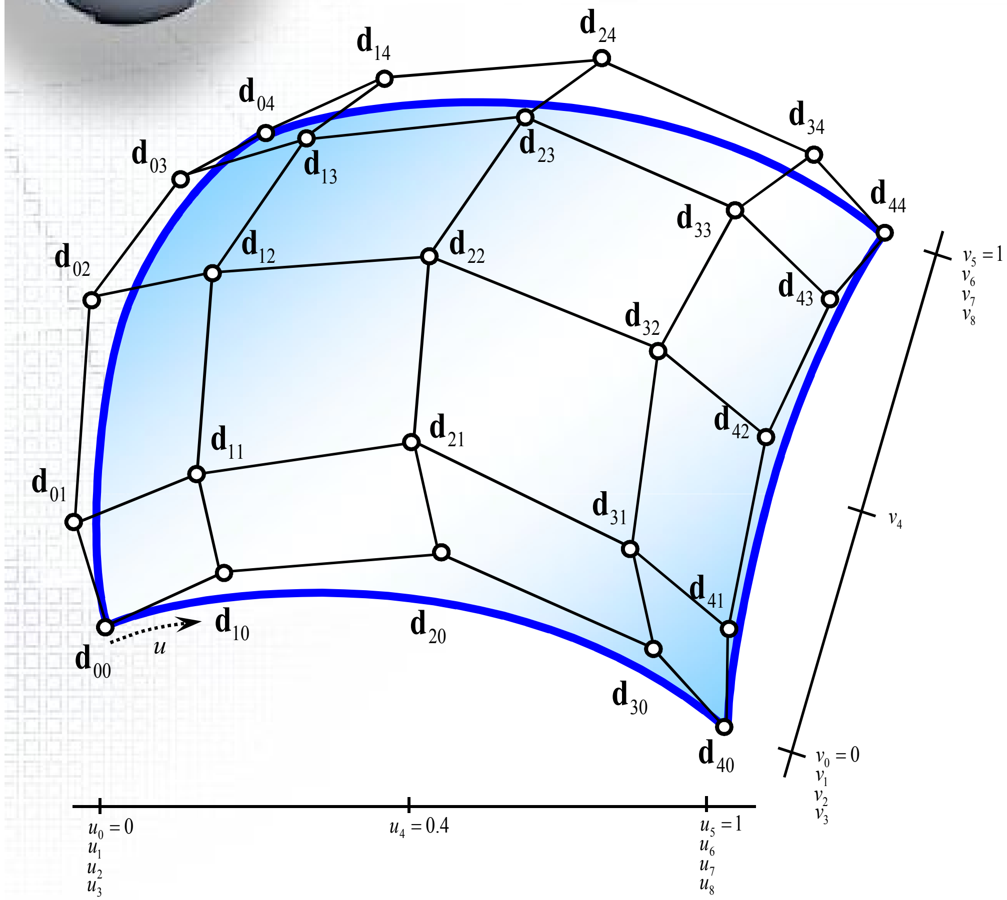
- Given: 곡면 상의 9개 점
- Find: Control Points of bicubic B-spline Surface



- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves
- (2) interpolate moving curves at v=v* with same u-knots
- (3) generate directional B-spline curves with same v-knots by interpolating the control points of the moving B-spline curves
- The control points of the directional curves are the control points of final bicubic B-spline surface

$\mathbf{d}_0(1)=\mathbf{d}_{04}$	$\mathbf{d}_1(1)=\mathbf{d}_{14}$	$\mathbf{d}_2(1)=\mathbf{d}_{24}$	$\mathbf{d}_3(1)=\mathbf{d}_{34}$	$\mathbf{d}_4(1)=\mathbf{d}_{44}$
$\mathbf{d}_0(\frac{1}{2})=\mathbf{t}_0$	$\mathbf{d}_1(\frac{1}{2})=\mathbf{t}_1$	$\mathbf{d}_2(\frac{1}{2})=\mathbf{t}_2$	$\mathbf{d}_3(\frac{1}{2})=\mathbf{t}_3$	$\mathbf{d}_4(\frac{1}{2})=\mathbf{t}_4$
$\mathbf{d}_0(0)=\mathbf{d}_{00}$	$\mathbf{d}_1(0)=\mathbf{d}_{10}$	$\mathbf{d}_2(0)=\mathbf{d}_{20}$	$\mathbf{d}_3(0)=\mathbf{d}_{30}$	$\mathbf{d}_4(0)=\mathbf{d}_{40}$

B-spline surface 정리



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

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

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

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

Programming Guide (1)

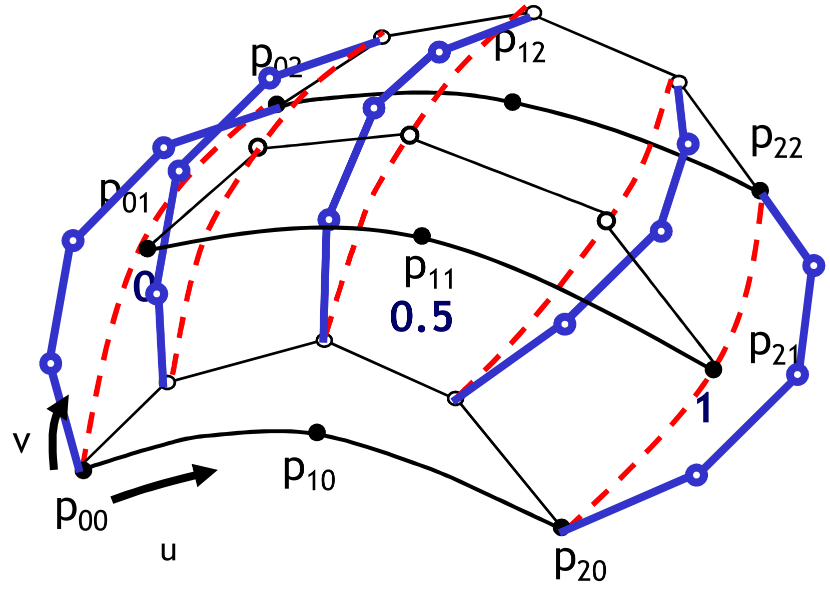
(1) B-spline Surface 생성

(2) u,v 방향 knot 구함

① 점 사이의 거리비를 통해 knot 계산
② 마지막 knot로 나누어 Normalize 함
③ 평균을 구함

(3) u 방향 B-spline 곡선의 Control Point 구함

(4) u방향의 B-spline 곡선의 Control point를 사용하여 v 방향의 B-spline 곡선을 생성하는 Control Point 생성



Programming Guide (2)

Given : 곡면을 지나는 공간상의 점 (MxN 개)

Find : Cubic B-spline Surface의 Control Points

(Q) 4x3의 점이 주어졌을 때, u,v 방향 knot 개수와 Control point 개수는?

(A) u 방향 : 4 + 3 x 2 = 10 개

v 방향 : 3 + 3 x 2 = 9 개

Control Point 개수 : 6 x 5 = 30 개

(Q) MxN의 점이 주어졌을 때, u,v 방향 knot 개수와 Control point 개수는?

(A) u 방향 : M + 3 x 2 = M + 6 개

v 방향 : N + 3 x 2 = N + 9 개

Control Point 개수 : (M+2) x (N+2) 개

Term Project #2
B-Spline Surface Interpolation

Advanced
Ship
Design
Automation
Laboratory

Term Project 2. B-Spline Surface Interpolation
- 과제 개요

과제 목표: 입력 받은 점을 지나는 B-Spline Surface를 Interpolation 한다.

Input Data

5 4

10.0 10.0 0.0

10.0 20.0 5.0

10.0 30.0 5.0

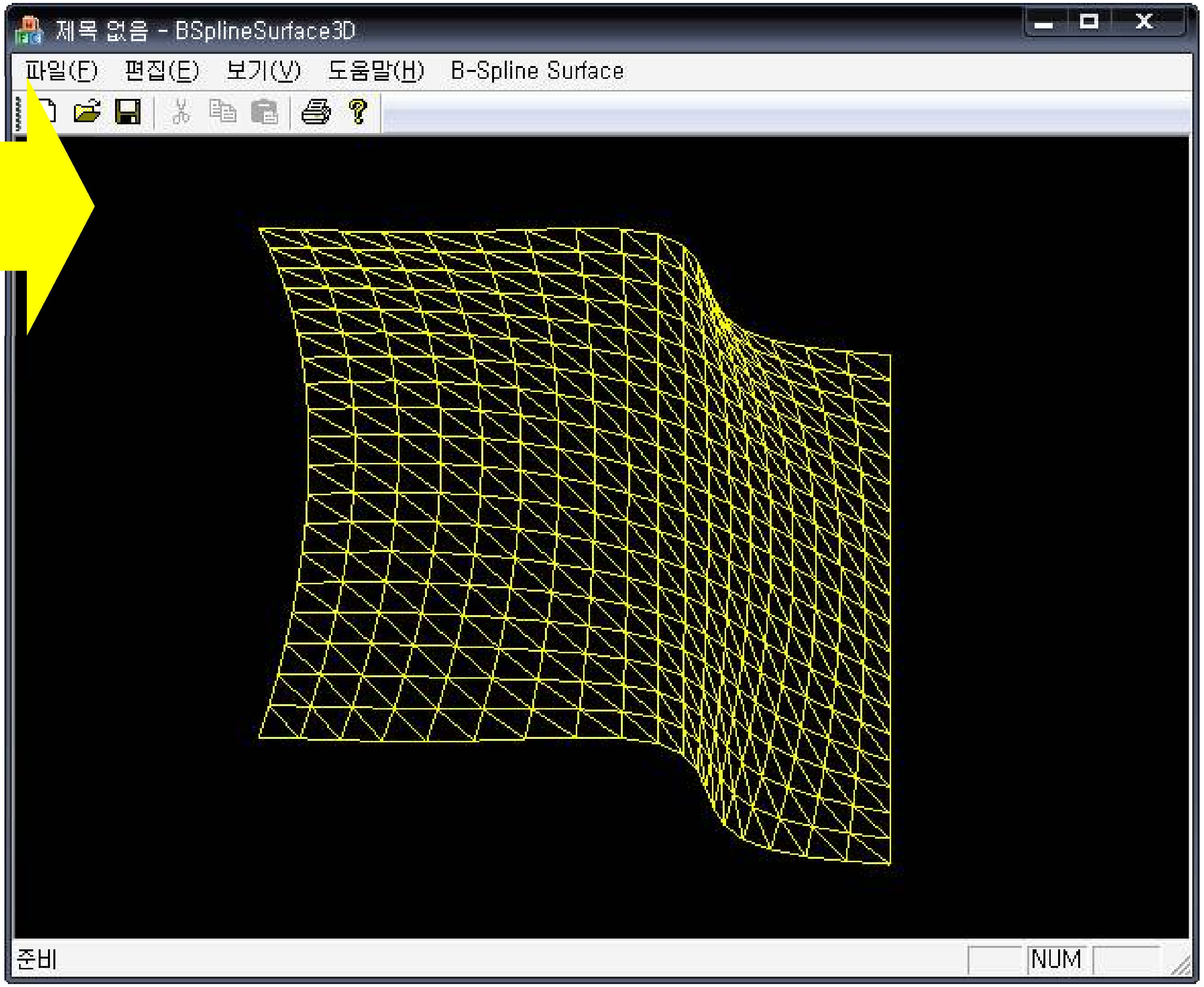
10.0 40.0 0.0

20.0 10.0 5.0

20.0 20.0 10.0

20.0 30.0 10.0

20.0 40.0 5.0



Term Project 2. B-Spline Surface Interpolation
- Input Data Format

Input Data

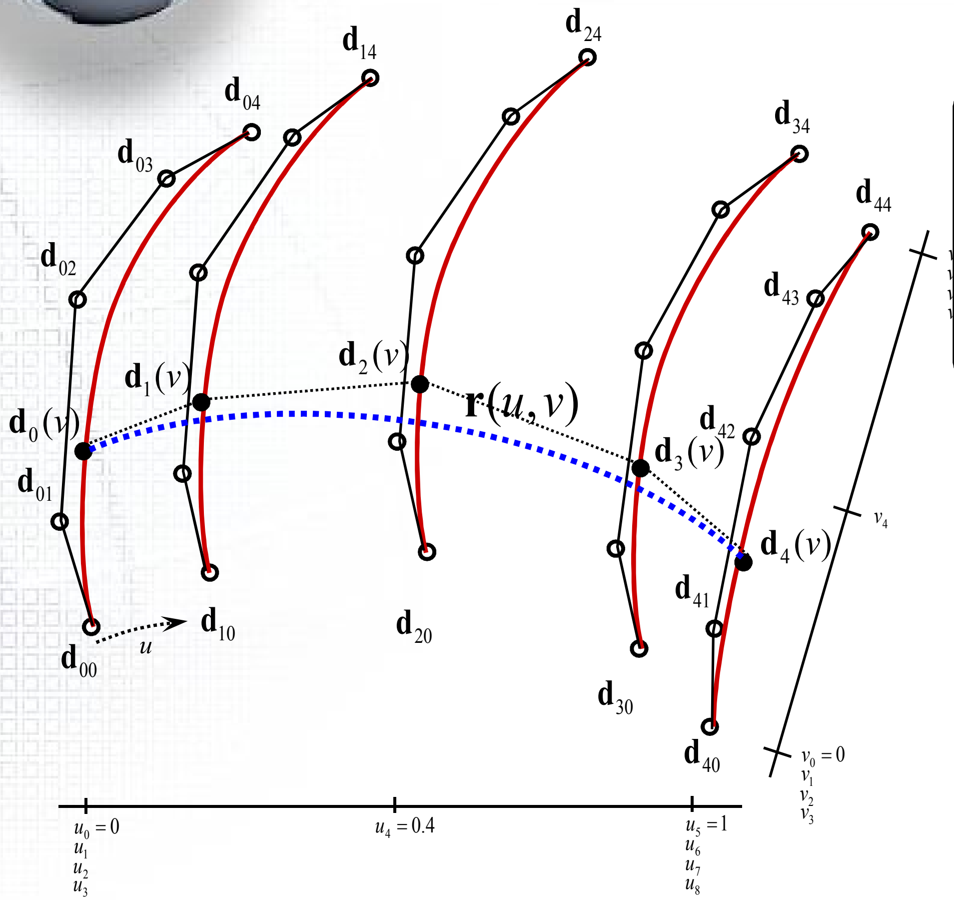
5 4 → 입력 받을 점의 u, v 방향 개수

10.0 10.0 0.0
10.0 20.0 5.0
10.0 30.0 5.0
10.0 40.0 0.0

20.0 10.0 5.0
20.0 20.0 10.0
20.0 30.0 10.0
20.0 40.0 5.0
…

→ 입력 받는 점의 3차원 좌표 (x, y, z)

bicubic B-spline Surface Interpolation



Cox-de Boor Recurrence Formula

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

$$N_i^0(u) = \begin{cases} 1 & \textbf{if } \; u_{i-1} \le u < u_i \\ 0 & \textbf{else} \end{cases} \quad , \sum_{i=0}^{D-1} N_i^n(u) = 1$$

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

Term Project 2. B-Spline Surface Interpolation

- 과제 수행 시 프로그램 작성 부분

```
double CBSplineSurface::N(int n, int i, double u, int uv)
{
    // check invalid condition
    if (m_nNumOfKnot_U < 1) return 0.0;
    if (m_nNumOfKnot_V < 1) return 0.0;
    if (m_nDegree < 2) return 0.0;

    // initialize
    double val = 0.0;

    ///////////////////////////////////////////////////////////////////////////
    // B-Spline basis function 함수를 작성하시오.


    //
    ///////////////////////////////////////////////////////////////////////////

    return val;
}

Vector CBSplineSurface::GetPoint(double u, double v)
{
    // return value
    Vector vec(0.0, 0.0, 0.0);

    // check invalid condition
    if (m_nDegree < 2) return vec;
    if (m_nNumOfCP_U < 1) return vec;
    if (m_nNumOfCP_V < 1) return vec;

    ///////////////////////////////////////////////////////////////////////////
    // parameter u, v가 주어졌을 때 곡면 상의 점을 구하는 함수를 작성하시오.

    //
    ///////////////////////////////////////////////////////////////////////////

    return vec;
}
```

Class	CBSplineSurface
Function	N
내용	u, v parameter**에 따라** B-spline basis function**을 계산**

Cox-de Boor Recurrence Formula

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

$$N_i^0(u) = \begin{cases} 1 & \textbf{if} \;\; u_{i-1} \le u < u_i \\ 0 & \textbf{else} \end{cases}, \quad \sum_{i=0}^{D-1} N_i^n(u) = 1$$

Term Project 2. B-Spline Surface Interpolation
- 과제 수행 시 프로그램 작성 부분

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

    // check invalid condition
    if (m_nDegree < 2) return vec;
    if (m_nNumOfCP_U < 1) return vec;
    if (m_nNumOfCP_V < 1) return vec;

    ////////////////////////////////////////////////////////////////////////////
    // parameter u, v가 주어졌을 때 곡면 상의 점을 구하는 함수를 작성하시오.

    //
    ////////////////////////////////////////////////////////////////////////////

    return vec;
}
```

Class	CBSplineSurface
Function	GetPoint
내용	**주어진** parameter u, v**에 대한 곡면 상의 점을 구하는 함수**

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

Term Project 2. B-Spline Surface Interpolation
- 과제 수행 시 프로그램 작성 부분

```
bool CBSplineSurface::Interpolation(Vector** pPoint, int nNumOfPointU, int nNumOfPointV)
{
    // check invalid condition
    if (pPoint == NULL) return false;
    if (nNumOfPointU < 1) return false;
    if (nNumOfPointV < 1) return false;

    // initialize
    int i = 0;
    int j = 0;

    double* pAvgKnotU = NULL;
    int nNumOfAvgKnotU = 0;

    double* pAvgKnotV = NULL;
    int nNumOfAvgKnotV = NULL;


    //////////////////////////////////////////////////////////////////////////
    // bicubic B-Spline surface interpolation 함수를 작성하시오.

    // create curve (u direction or v direction)


    // set knot using chord length


    // normalize knot


    // set average knot


    // set average knot to each curve


    // interpolation



    // create curve (v direction of u direction)
```

Class	CBSplineSurface
Function	Interpolation
내용	**곡면 상의 주어진 점을 지나는** bicubic B-spline surface**의** Control Point**를 구하는 함수**