Computer Graphics

- Splines-II -

Philipp Slusallek

Overview

- **Last Time**
  - Transformations
- **Today**
  - B-Splines
  - Parametric Surfaces
  - Tensor Product Construction
  - Ray-Tracing
- **Next Lecture**
  - Subdivision Surfaces

B-Splines

- **Goal**
  - Spline curve with local control and high continuity
- **Given**
  - Degree: n
  - Control points: $P_0, \ldots, P_m$ (Control polygon, $m \geq n+1$)
  - Knots: $t_0, \ldots, t_{m+n+1}$ (Knot vector, weakly monotonic)
  - The knot vector defines the parametric locations where segments join
- **B-Spline Curve**

$$\underline{P}(t) = \sum_{i=0}^{m} N_i^n(t)\underline{P}_i$$

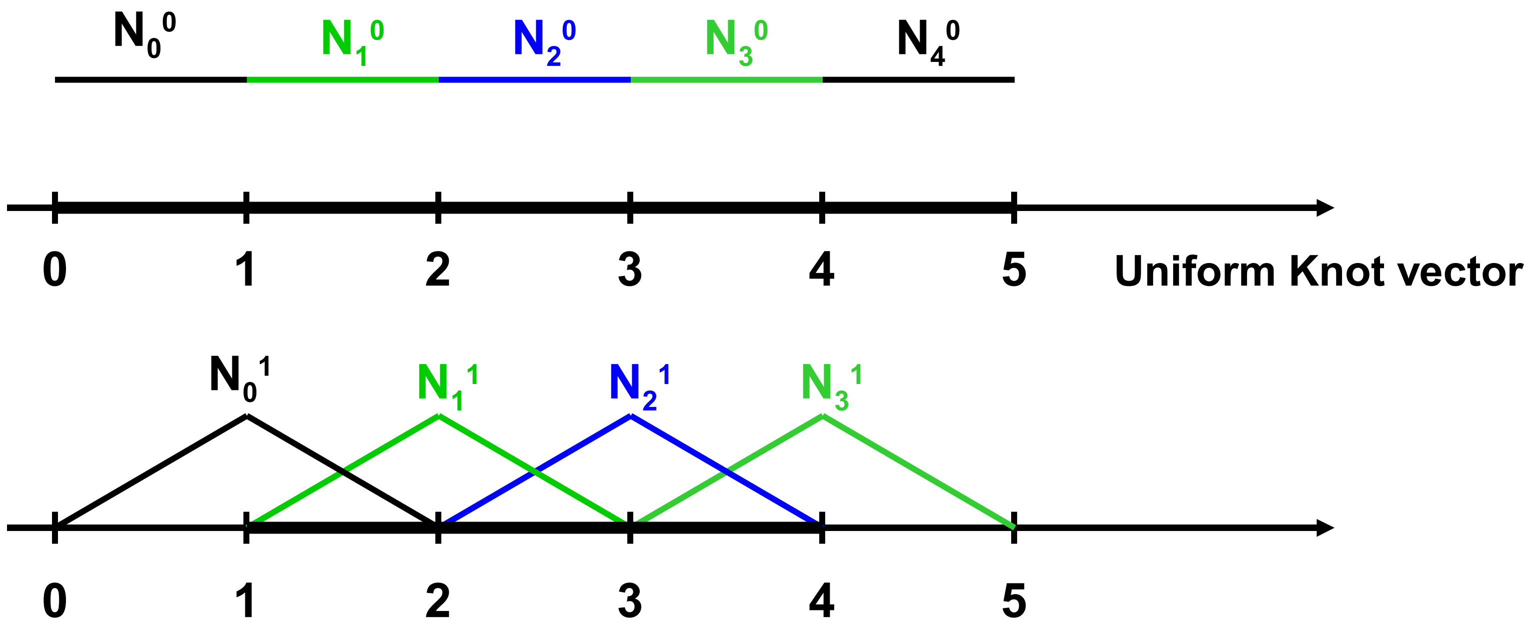
  - Continuity:
    - C_{n-1} at simple knots
    - C_{n-k} at knot with multiplicity k

B-Spline Basis Functions

- **Recursive Definition**

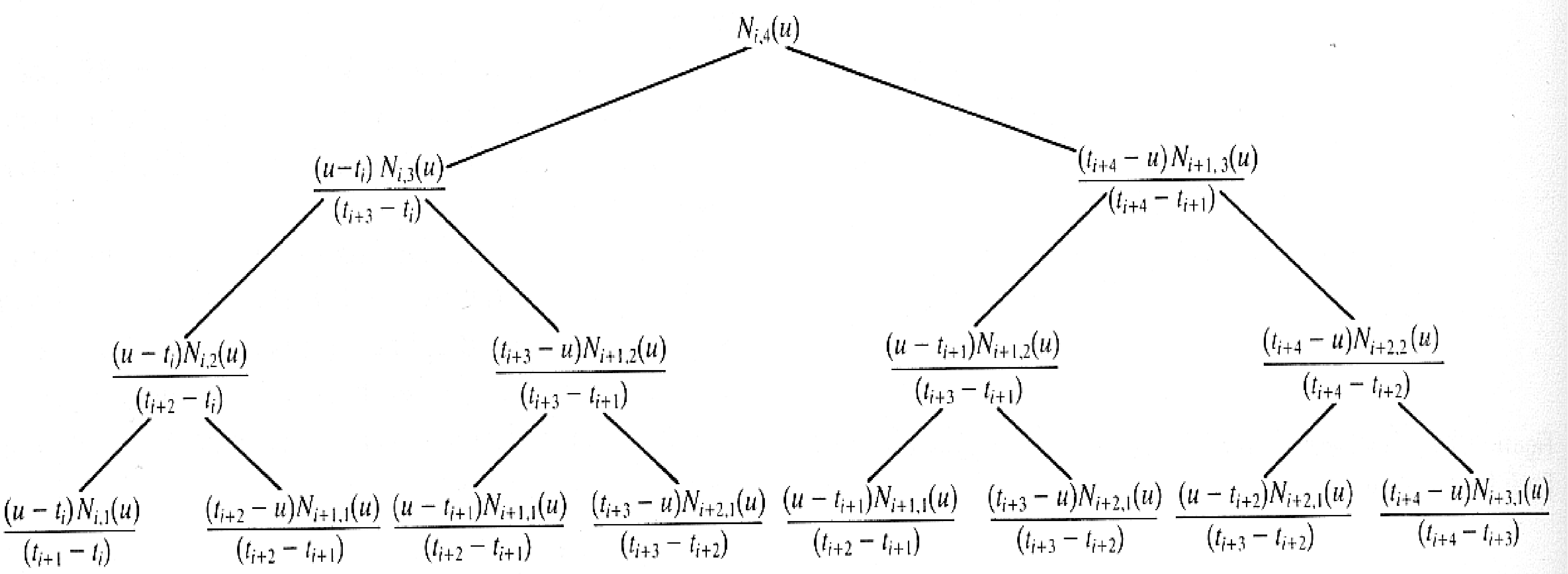
$$N_i^0(t) = \begin{cases} 1 & \text{if } t_i < t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i} N_i^{n-1}(t) - \frac{t - t_{i+n+1}}{t_{i+n+1} - t_{i+1}} N_{i+1}^{n-1}(t)$$



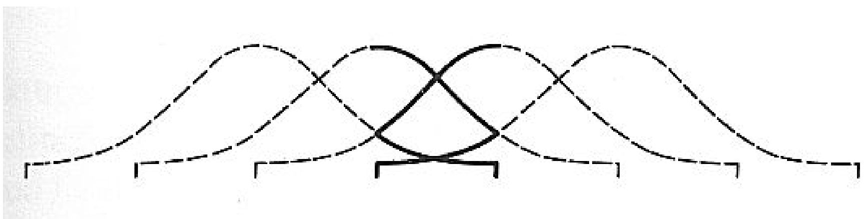
B-Spline Basis Functions

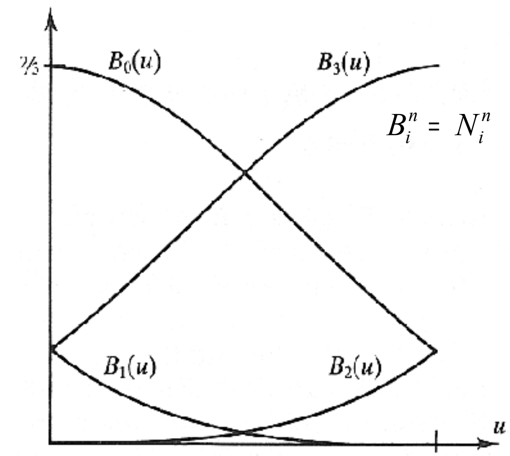
- **Recursive Definition**
  - Degree increases in every step
  - Support increases by one knot interval



B-Spline Basis Functions

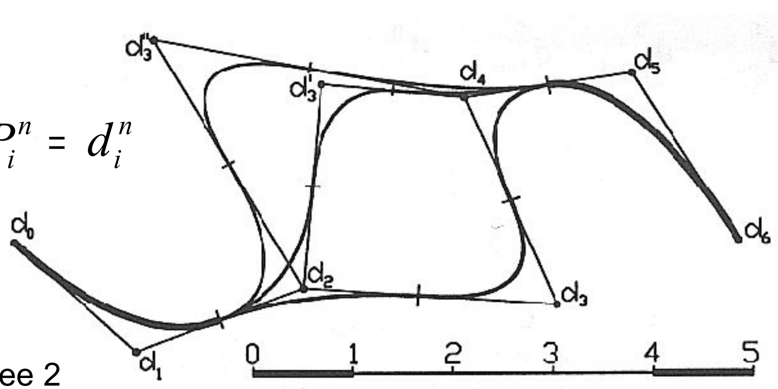
- **Uniform Knot Vector**
  - All knots at integer locations
    - UBS: Uniform B-Spline
  - Example: cubic B-Splines





- **Local Support = Localized Changes**
  - Basis functions affect only (n+1) Spline segments
  - Changes are localized

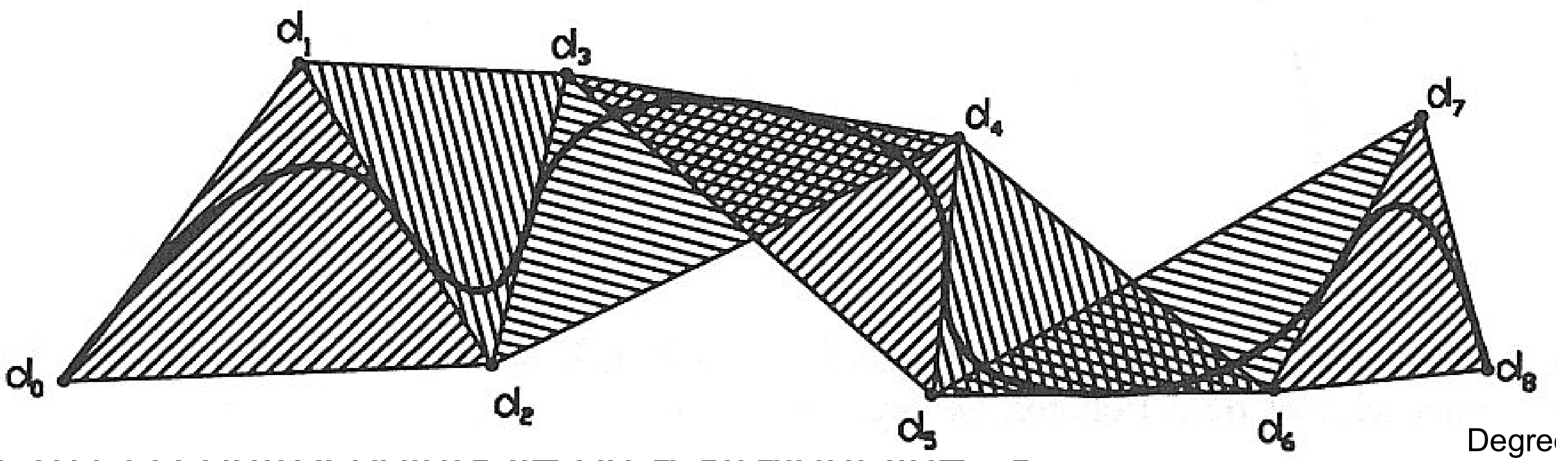
$P_i^n = d_i^n$



Degree 2

B-Spline Basis Functions

- **Convex Hull Property**
  - Spline segment lies in convex Hull of (n+1) control points

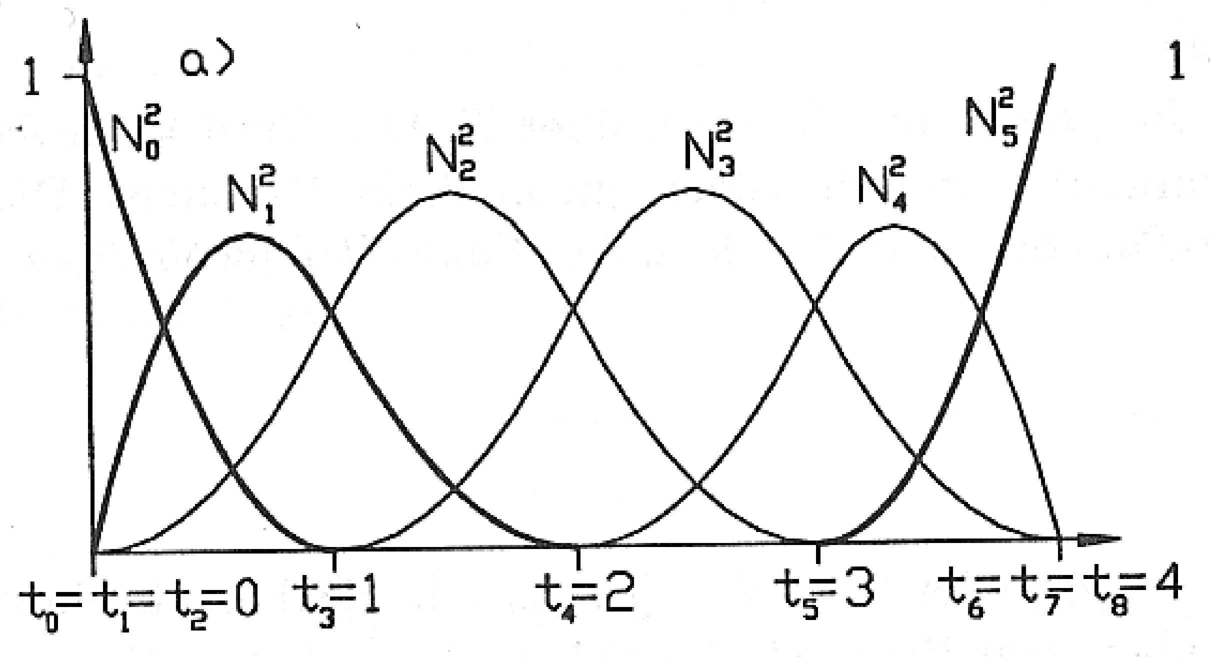


Degree 2

  - (n+1) control points lie on a straight line ➔ curve touches this line
  - n control points coincide ➔ curve interpolates this point and is tangential to the control polygon

Normalized Basis Functions

- **Basis Functions on an Interval**
  - Knots at beginning and end with multiplicity
    - NUBS: Non-uniform B-Splines
  - Interpolation of end points and tangents there
  - Conversion to Bézier segments via knot insertion



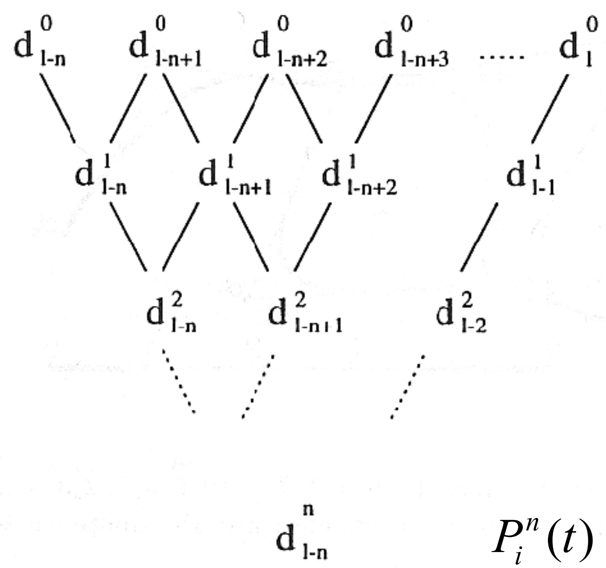
deBoor-Algorithm

- **Recursive Definition of Control Points**
  - Evaluation at t: $t_l < t < t_{l+1}$: $i \in \{l-n, ..., l\}$
    - Due to local support only affected by (n+1) control points

$$\underline{P}_i^r(t) = \left(1 - \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}}\right)\underline{P}_i^{r-1}(t) + \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}}\underline{P}_{i+1}^{r-1}(t)$$

$$\underline{P}_i^0(t) = \underline{P}_i$$

- **Properties**
  - Affine invariance
  - Stable numerical evaluation
    - All coefficients > 0



$$P_i^n(t) = d_i^n$$

Knot Insertion

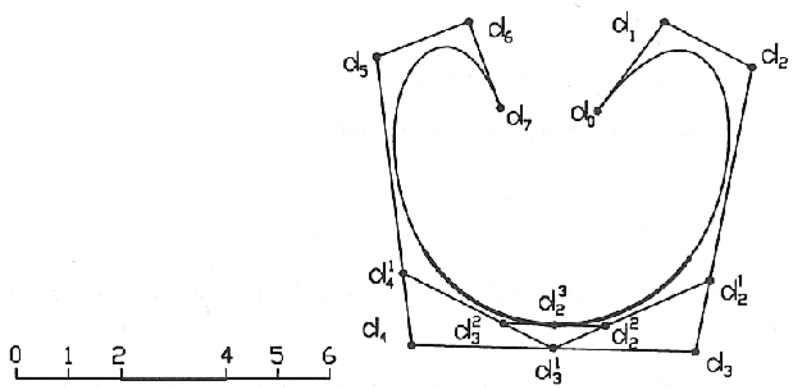
- **Algorithm similar to deBoor**
  - Given a new knot t
    - $t_l \leq t < t_{l+1}: i \in \{l-n, ..., l\}$
  - $T^* = T \cup \{t\}$
  - New representation of the same curve over T*

$$\underline{P}^*(t) = \sum_{i=0}^{m+1} N_i^n(t)\underline{P}_i^*$$

$$P_i^* = (1-a_i)P_{i-1} + a_iP_i$$

$$a_i = \begin{cases} 1 & i \leq l-n \\ \dfrac{t-t_i}{t_{i+n}-t_i} & l-n+1 \leq i \leq l \\ 0 & i \geq l+1 \end{cases}$$

- **Applications**
  - Refinement of curve, display



Consecutive insertion of three knots at t=3 into a cubic B-Spline
First and last knot have multiplicity n
T=(0,0,0,0,1,2,4,5,6,6,6,6), l=5

Conversion to Bézier Spline

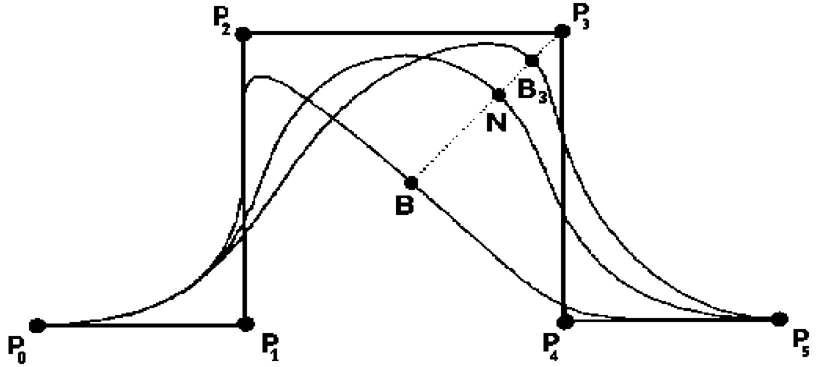
- **B-Spline to Bezier Representation**
  - Remember:
    - Curve interpolates point and is tangential at knots of multiplicity n
  - In more detail: If two consecutive knots have multiplicity n
    - The corresponding spline segment is in Bézier from
    - The (n+1) corresponding control polygon form the Bézier control points

NURBS

- **Non-uniform Rational B-Splines**
  - Homogeneous control points: now with weight w_i
    - $\underline{P}_i{}' = (w_i x_i, w_i y_i, w_i z_i, w_i) = w_i \underline{P}_i$

$$\underline{P}'(t) = \sum_{i=0}^{m} N_i^n(t)\underline{P}'_i$$

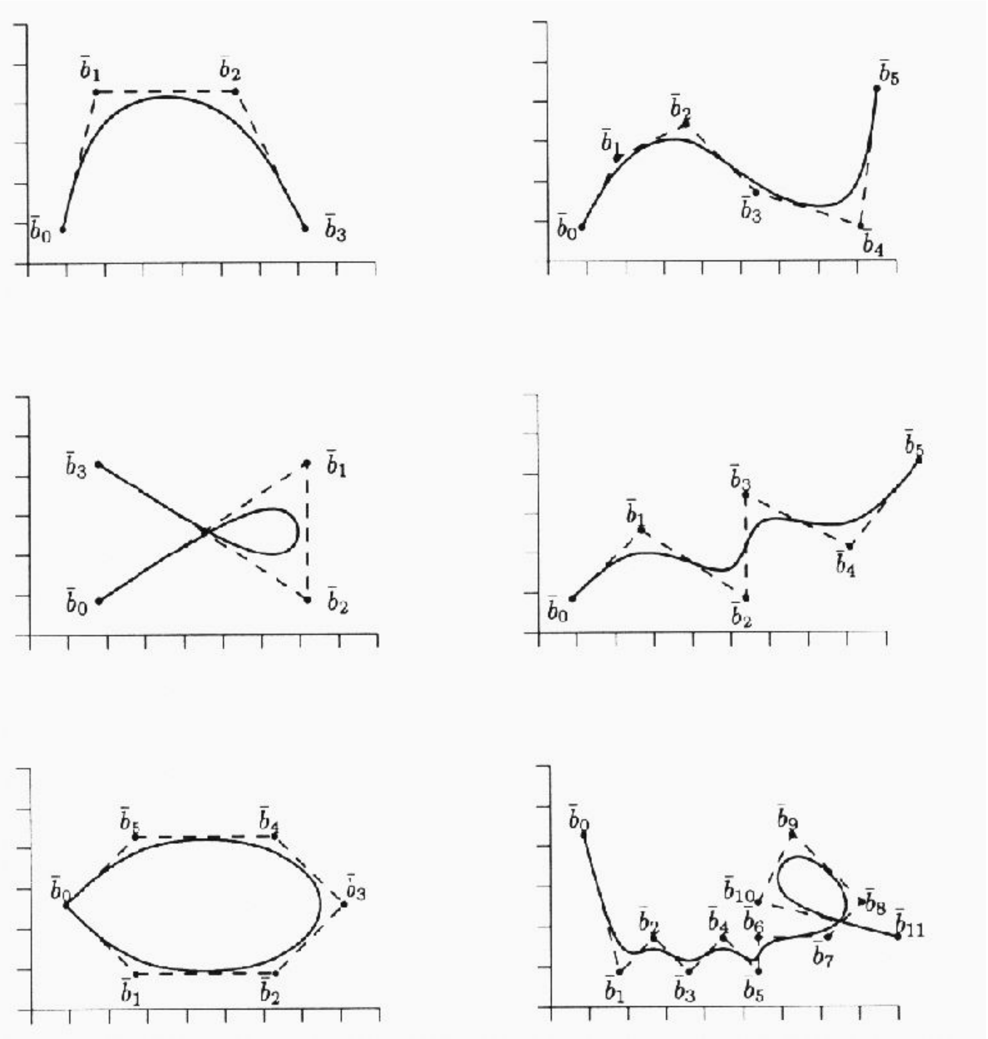
$$\underline{P} = \frac{\sum_{i=0}^{m} N_i^n(t)\underline{P}_i w_i}{\sum_{i=0}^{m} N_i^n(t) w_i} = \sum_{i=0}^{m} R_i^n(t)\underline{P}_i w_i, \quad \text{mit} \quad R_i^n(t) = \frac{N_i^n(t) w_i}{\sum_{i=0}^{m} N_i^n(t) w_i}$$



NURBS

- **Properties**
  - Piecewise rational functions
  - Weights
    - High (relative) weight attract curve towards the point
    - Low weights repel curve from a point
    - Negative weights should be avoided (may introduce singularity)
  - Invariant under projective transformations
  - Variation-Diminishing-Property (in functional setting)
    - Curve cuts a straight line no more than the control polygon does

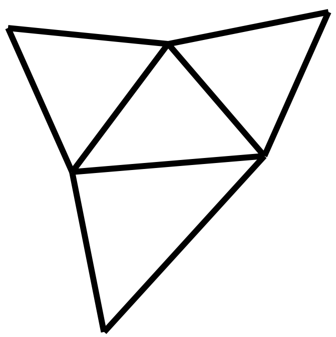
Examples: Cubic B-Splines

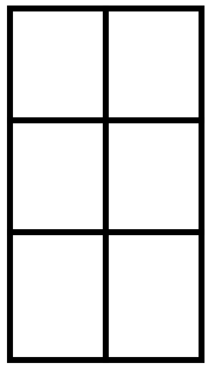


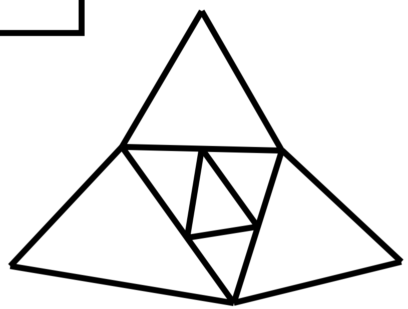
Spline Surfaces

Parametric Surfaces

- **Same Idea as with Curves**
  - $\underline{P}: R^2 \rightarrow R^3$
  - $\underline{P}(u,v) = (x(u,v), y(u,v), z(u,v))^T \in R^3$ (also $P(R^4)$)
- **Different Approaches**
  - Triangular Splines
    - Single polynomial in (u,v) via barycentric coordinates with respect to a reference triangle (e.g. B-Patches)
  - Tensor Product Surfaces
    - Separation into polynomials in u and in v
  - Subdivision Surfaces
    - Start with a triangular mesh in R^3
    - Subdivide mesh by inserting new vertices
      - Depending on local neighborhood
    - Only piecewise parameterization (in each triangle)







Tensor Product Surfaces

- **Idea**
  - Create a “curve of curves"
- **Simplest case: Bilinear Patch**
  - Two lines in space

$$\underline{P}^1(v) = (1-v)\underline{P}_{00} + v\underline{P}_{10}$$

$$\underline{P}^2(v) = (1-v)\underline{P}_{01} + v\underline{P}_{11}$$

  - Connected by lines

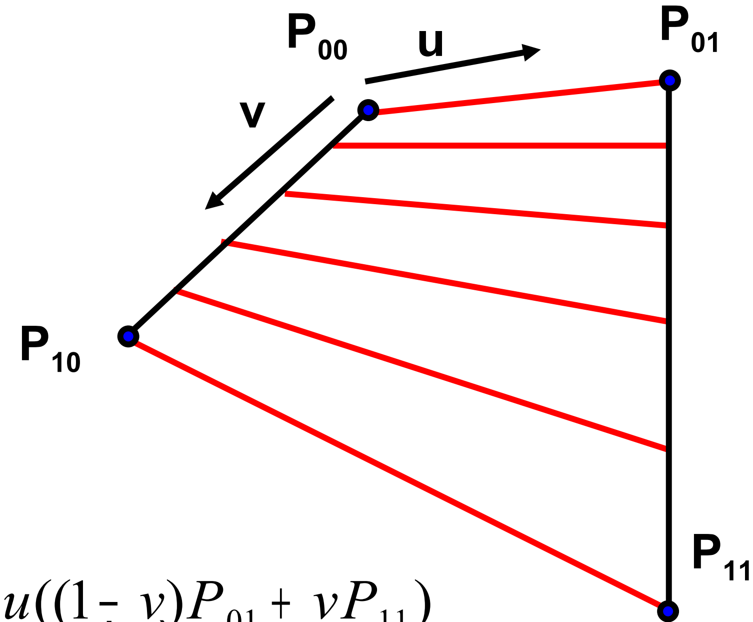
$$\underline{P}(u,v) = (1-u)\underline{P}^1(v) + u\underline{P}^2(v) =$$

$$(1-u)((1-v)\underline{P}_{00} + v\underline{P}_{10}) + u((1-v)\underline{P}_{01} + v\underline{P}_{11})$$

  - Bézier representation (symmetric in u and v)

$$\underline{P}(u,v) = \sum_{i,j=0}^{1} B_i^1(u) B_j^1(v) \underline{P}_{ij}$$

  - Control mesh P_{ij}



Tensor Product Surfaces

- **General Case**
  - Arbitrary basis functions in u and v
    - Tensor Product of the function space in u and v
  - Commonly same basis functions and same degree in u and v

$$\underline{P}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) \underline{P}_{ij}$$

- **Interpretation**
  - Curve defined by curves

$$\underline{P}(u,v) = \sum_{i=0}^{m} B_i'(u) \underbrace{\sum_{j=0}^{n} B_j(v) \underline{P}_{ij}}_{P_i'(v)}$$

  - Symmetric in u and v

Matrix Representation

- **Similar to Curves**
  - Geometry now in a „tensor“ (m x n x 3)

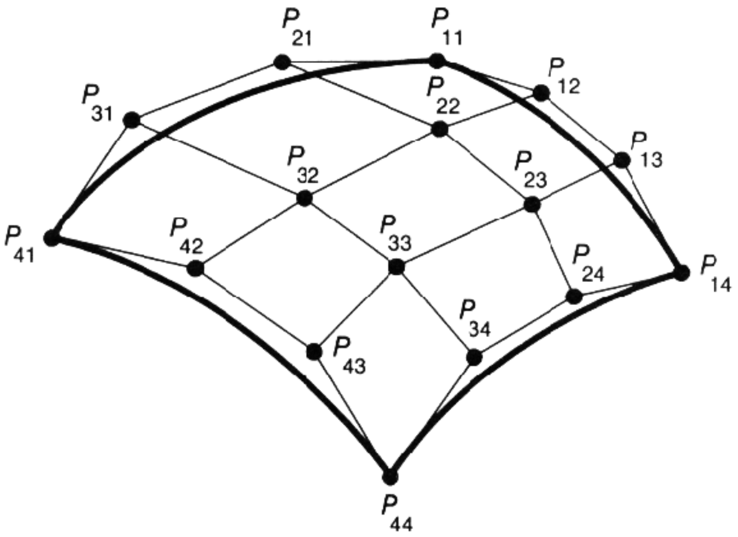
$$\underline{P}(u,v) = U\mathbf{G}_{monom}V^T = \begin{pmatrix} u^m & \cdots & u & 1 \end{pmatrix} \begin{pmatrix} G_{nn} & \cdots & G_{n0} \\ \vdots & \ddots & \vdots \\ G_{0n} & \cdots & G_{00} \end{pmatrix} \begin{pmatrix} v^n \\ \vdots \\ v \\ 1 \end{pmatrix} =$$

$$U\mathbf{B}'_U \mathbf{G}_{UV} \mathbf{B}_V^T V^T$$

  - Degree
    - u: m
    - v: n
    - Along the diagonal (u=v): m+n
      - Not nice → „Triangular Splines“

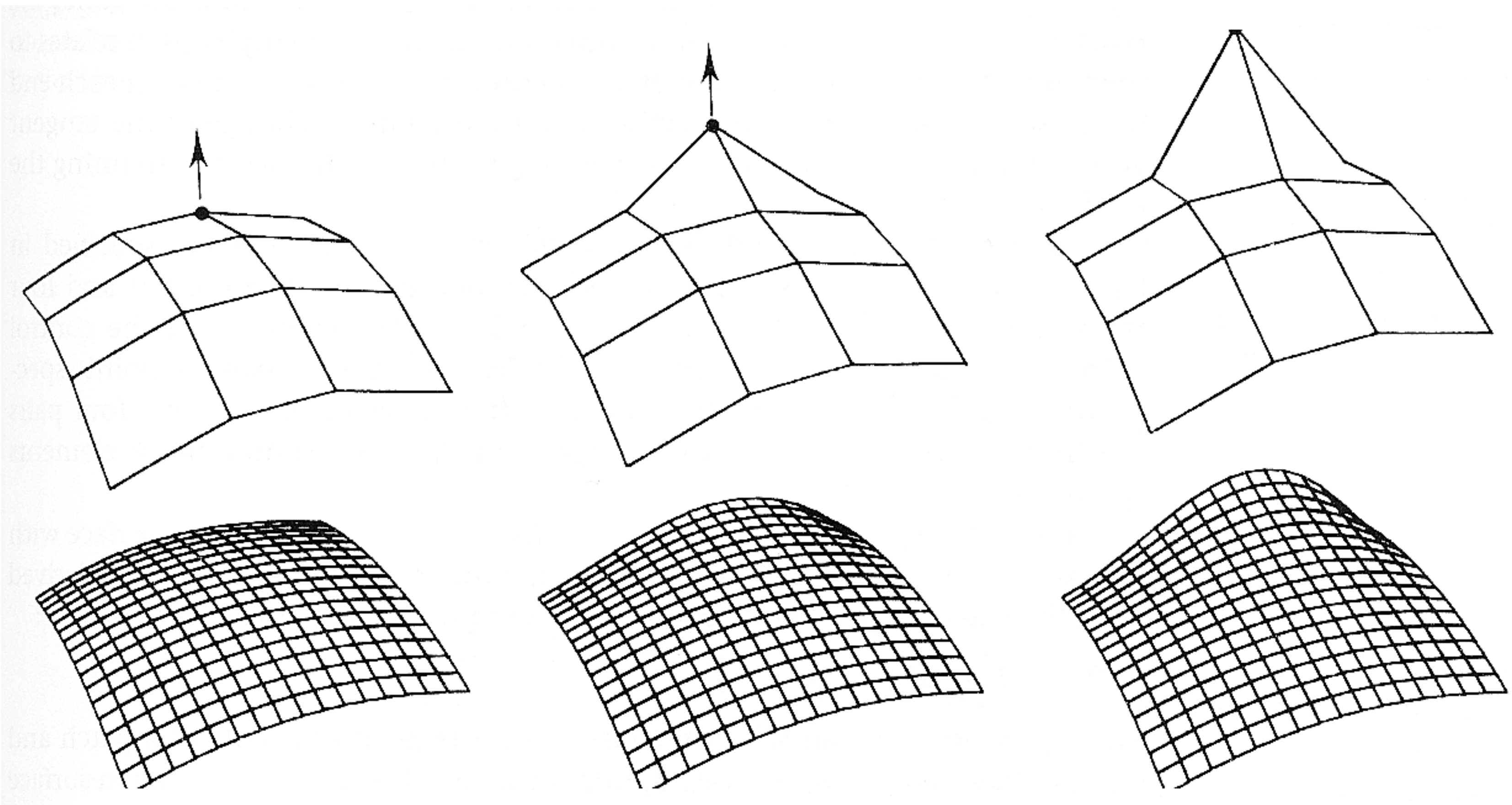
Tensor Product Surfaces

- **Properties Derived Directly From Curves**
- **Bézier Surface:**
  - Surface interpolates corner vertices of mesh
  - Vertices at edges of mesh define boundary curves
  - Convex hull property holds
  - Simple computation of derivatives
  - Direct neighbors of corners vertices define tangent plane
- **Similar for Other Basis Functions**



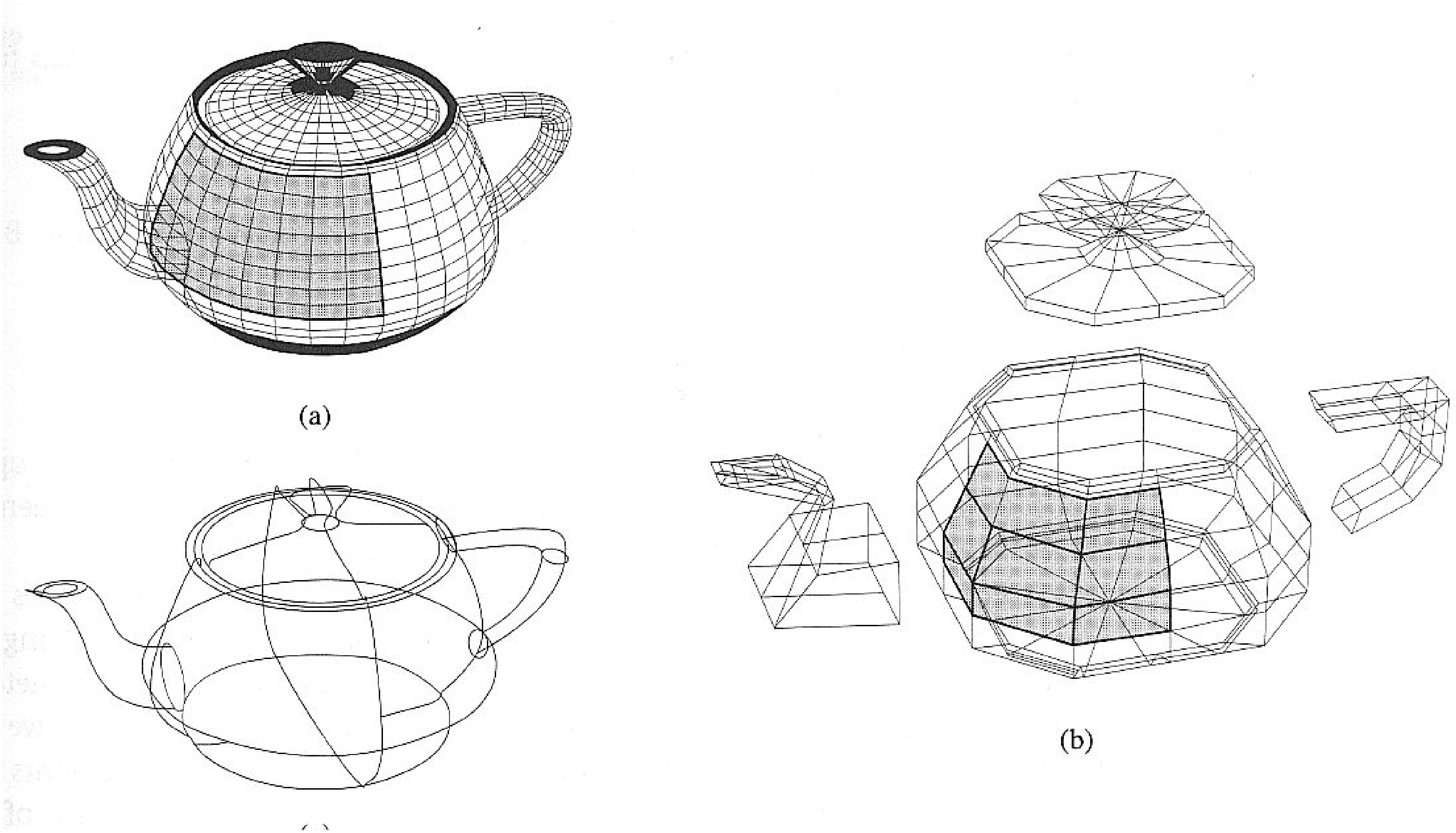
Tensor Product Surfaces

- **Modifying a Bézier Surface**



Tensor Product Surfaces

- **Representing the Utah Teapot as a set continuous Bézier patches**
  - http://www.holmes3d.net/graphics/teapot/

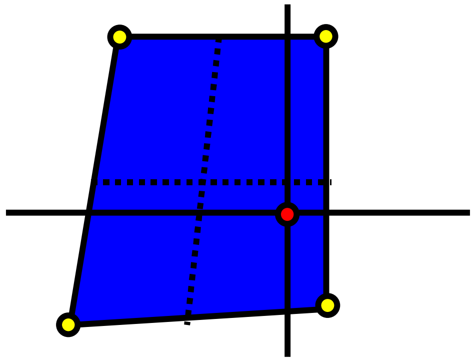


Operations on Surfaces

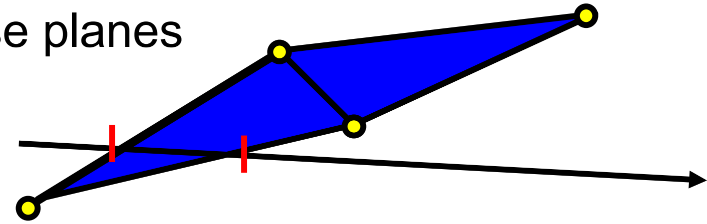
- **deCausteljau/deBoor Algorithm**
  - Once for u in each column
  - Once for v in the resulting row
  - Due to symmetry also in other order
- **Similarly we can derive the related algorithms**
  - Subdivision
  - Extrapolation
  - Display
  - ...

Ray Tracing of Spline Surfaces

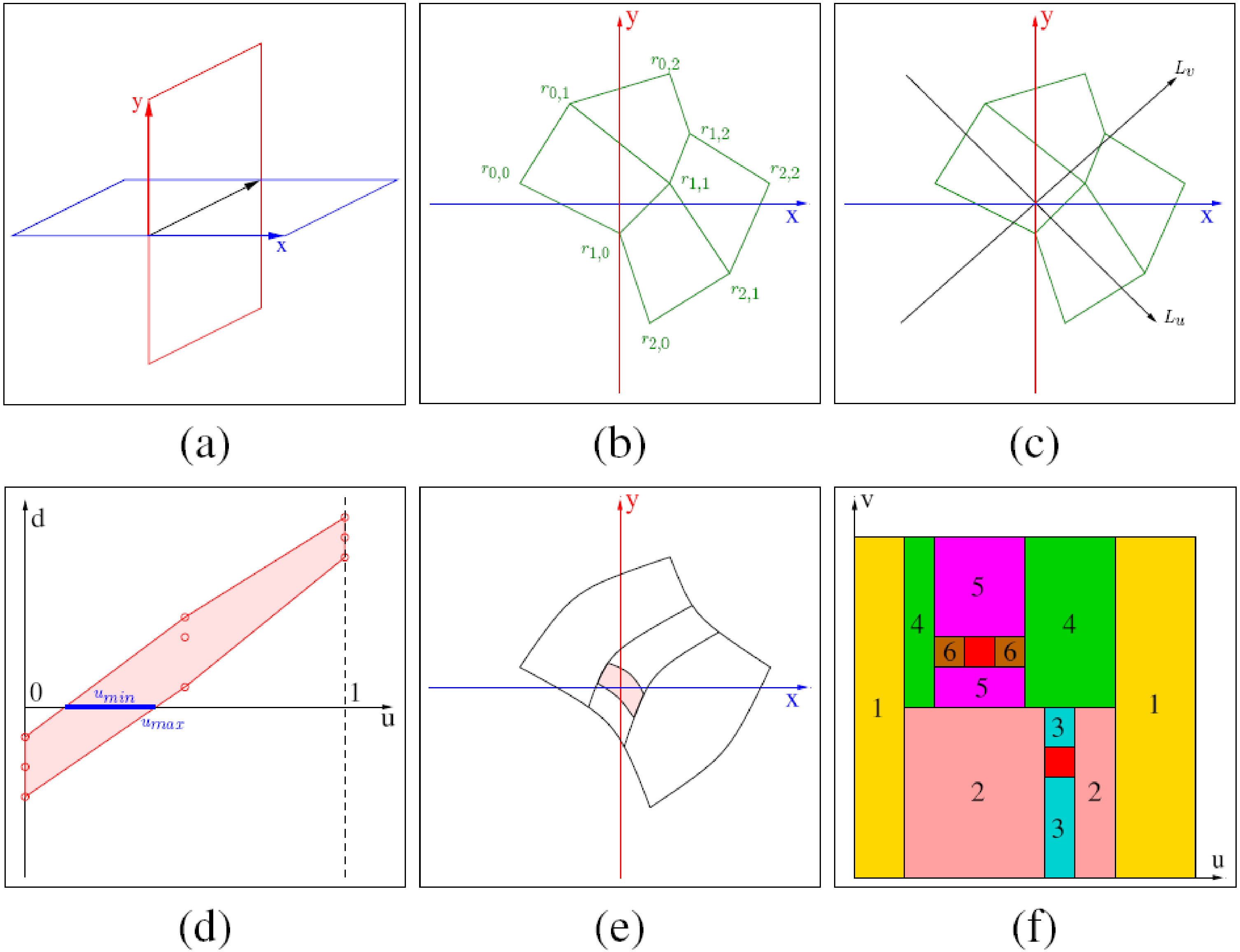
- **Several approaches**
  - Tessellate into many triangles (using deCasteljau or deBoor)
    - Often the fasted method
    - May need enormous amounts of memory
  - Recursive subdivision
    - Simply subdivide patch recursively
    - Delete parts that do not intersect ray (Pruning)
    - Fixed depth ensures crack-free surface



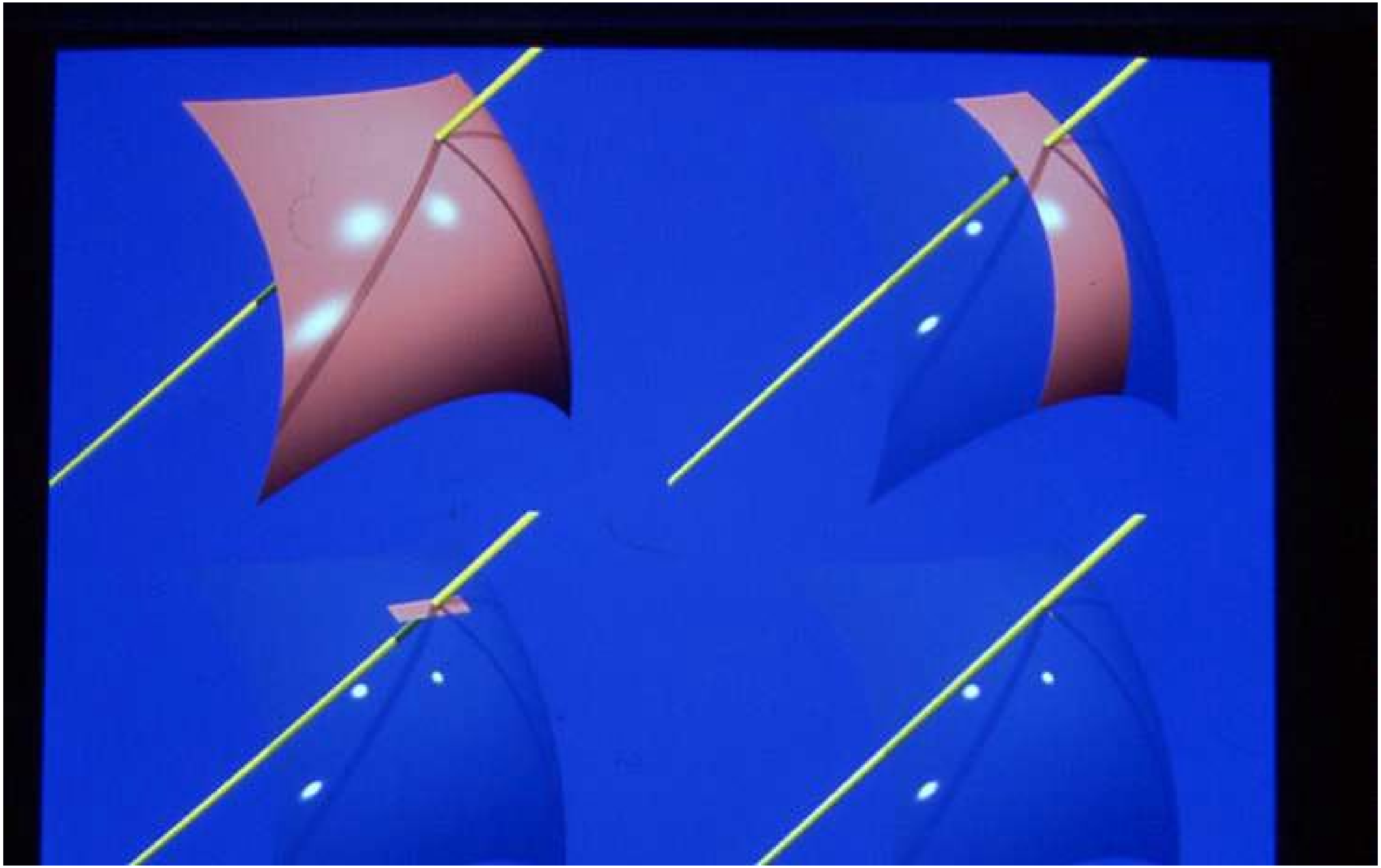
  - Bézier Clipping [Sederberg et al.]
    - Find two orthogonal planes that intersect in the ray
    - Project the surface control points into these planes
    - Intersection must have distance zero
      - ➔ Root finding
      - ➔ Can eliminate parts of the surface where convex hull does not intersect ray
    - Must deal with many special cases – rather slow



Bézier Clipping

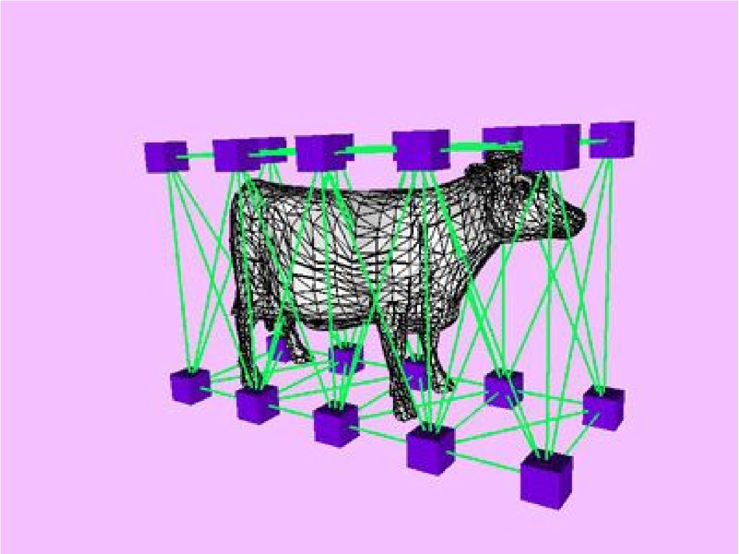


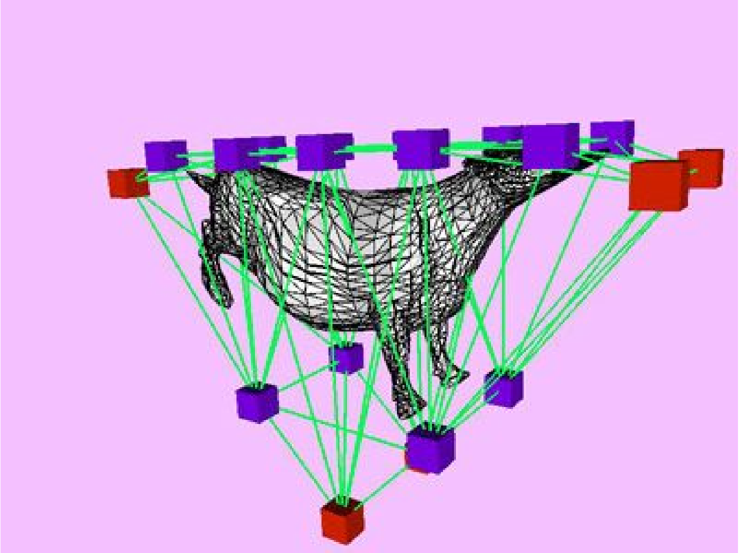
Bézier Clipping



Higher Dimensions

- **Volumes**
  - Spline: $R^3 \rightarrow R$
    - Volume density
    - Rarely used
  - Spline: $R^3 \rightarrow R^3$
    - Modifications of points in 3D
    - Displacement mapping
    - Free Form Deformations (FFD)





FFD