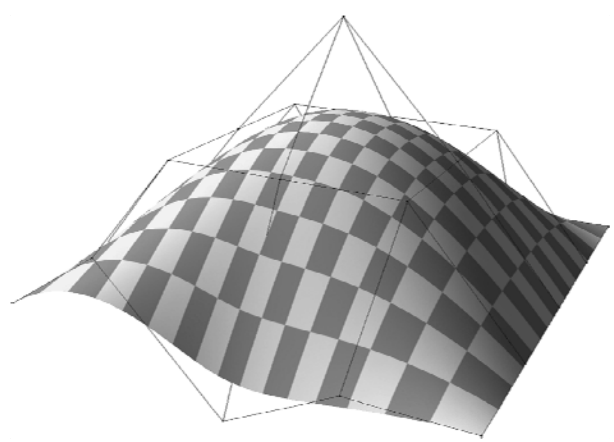
G22.3033-008, Spring 2010

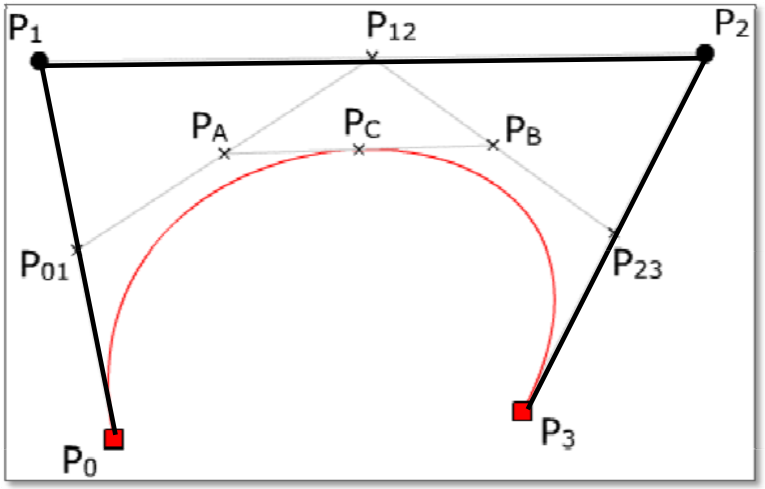
Geometric Modeling

Splines, tensor product surfaces



Bezier Curves

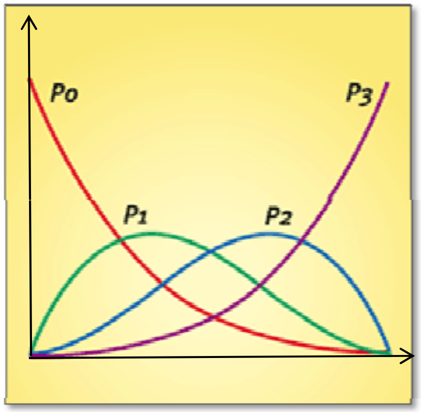
$$S(t) = \sum_{i=0}^{n} \mathbf{p}_i \, B_i^n(t)$$



Curve and
control polygon

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Basis functions

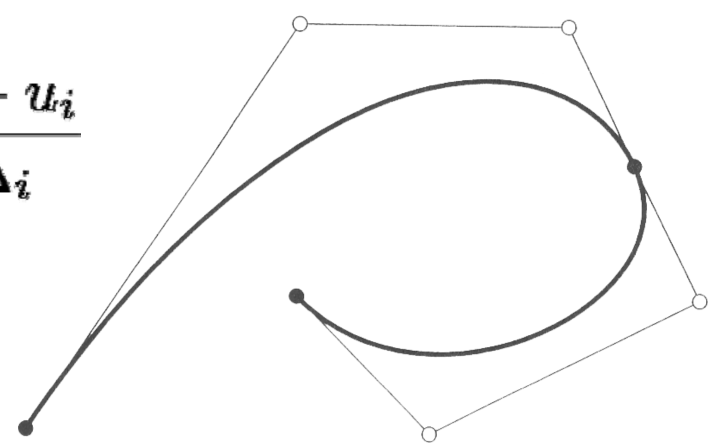


Disadvantages of Bezier Curves

- More control points
  - Higher degree
- Global support of basis functions
  - No local control
- Today
  - Piecewise Bezier Curves
  - B-Splines
  - Rational curves
  - Tensor Product Surfaces and NURBS

Piecewise Bezier Curves

- Smoothly connected curve segments
  - Segments p_0 to p_k
  - Each segment p_i is polynomial of degree n
  - Whole curve defined over parameter interval $[0, k+1]$ with a global parameter u
  - Each segment defined over interval $[u_i, u_{i+1}] = [i, i+1]$
    - Segment boundaries called knots
  - Local parameter $t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\Delta_i}$
  - Overall curve $s(u) = s_i(t)$

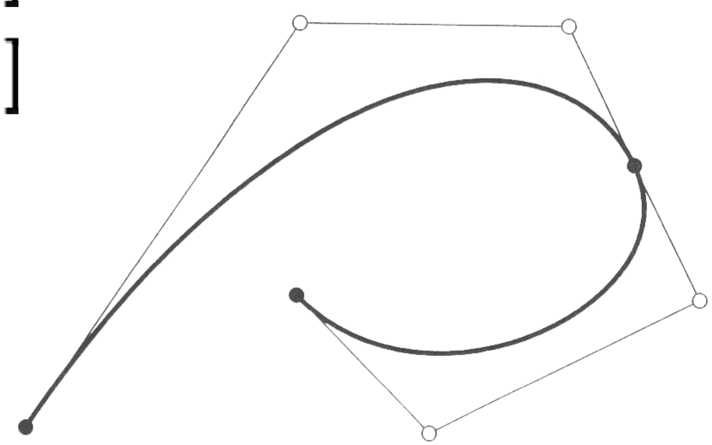


Piecewise Bezier Curves

- Spline curve
  - Maximally smooth connections between segments
  - C^{n-1} continuity

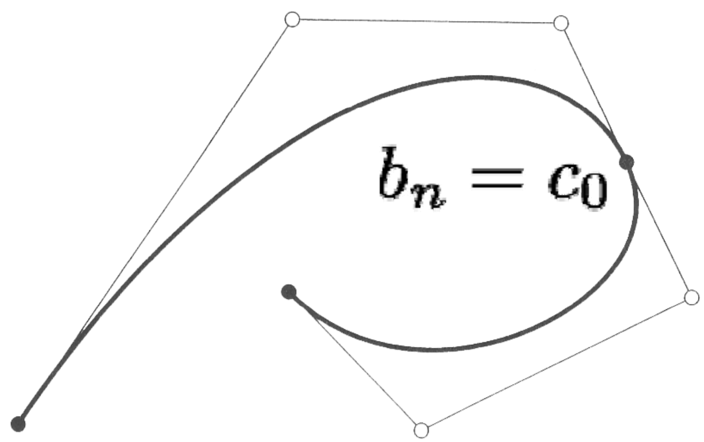
$$\forall l \in \{0, \ldots, n-1\} : \frac{\partial^l}{\partial t^l} p_{i-1}(i) = \frac{\partial^l}{\partial t^l} p_i(i), i \in \{1, \ldots, k\}$$

- Curve in $[u_0, u_2]$ from two Bezier segments
  - Control points $b_0, \ldots, b_n$ in $[u_0, u_1]$
  - Control points $c_0, \ldots, c_n$ in $[u_1, u_2]$
- C^r smoothness affects r+1 control points of each curve



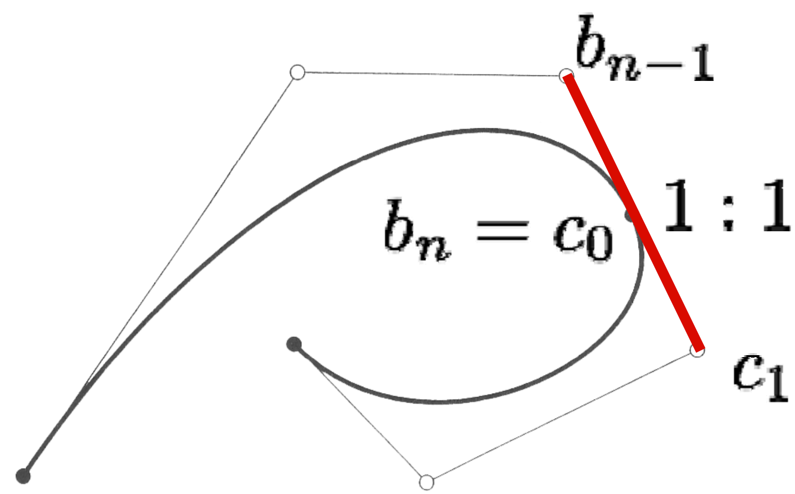
Piecewise Bezier Curves

- Example: cubic Bezier curves
- C^0 continuity
  - 1 control point of each curve affected
  - Endpoints have to match



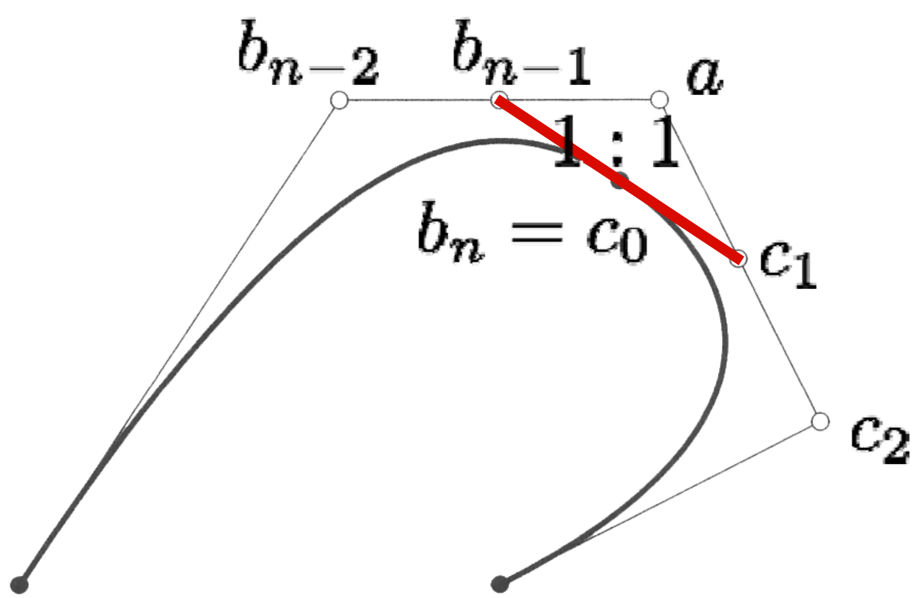
Piecewise Bezier Curves

- Example: cubic Bezier curves
- C^1 continuity
  - 2 control points of each curve affected
  - Co-linear, $b_n = \dfrac{b_{n-1} + c_1}{2}$



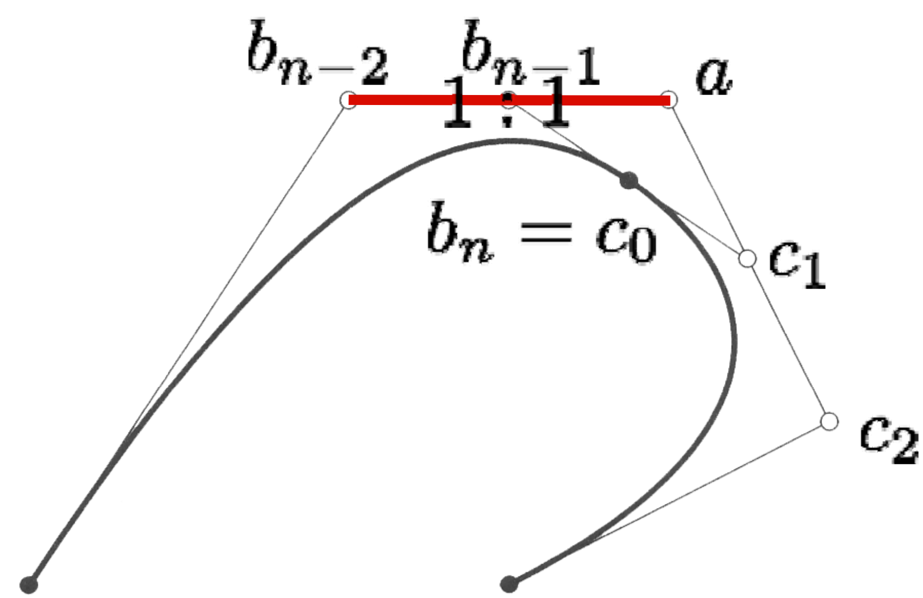
Piecewise Bezier Curves

- Example: cubic Bezier curves
- C^2 continuity
  - 3 control points of each curve affected
  - A-frame



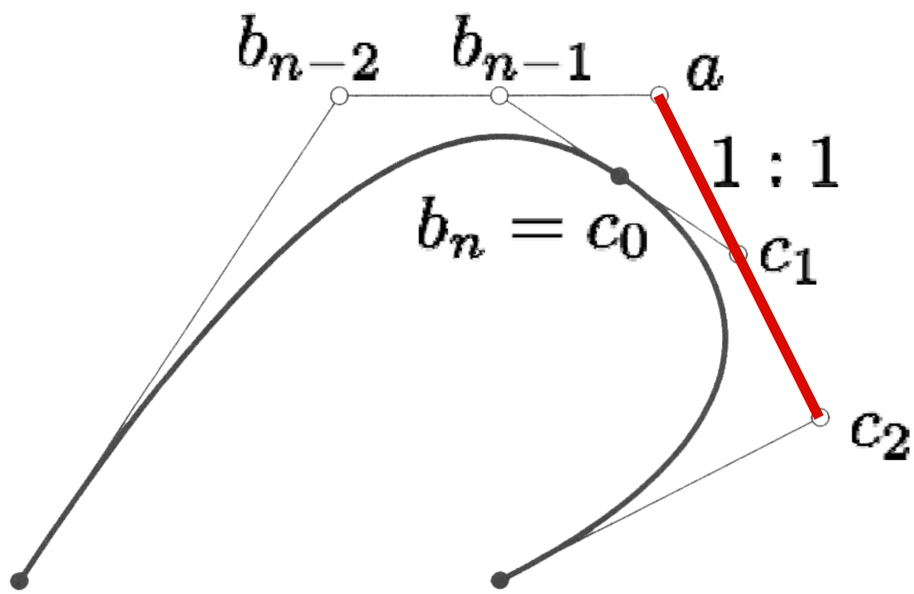
Piecewise Bezier Curves

- Example: cubic Bezier curves
- C^2 continuity
  - 3 control points of each curve affected
  - A-frame



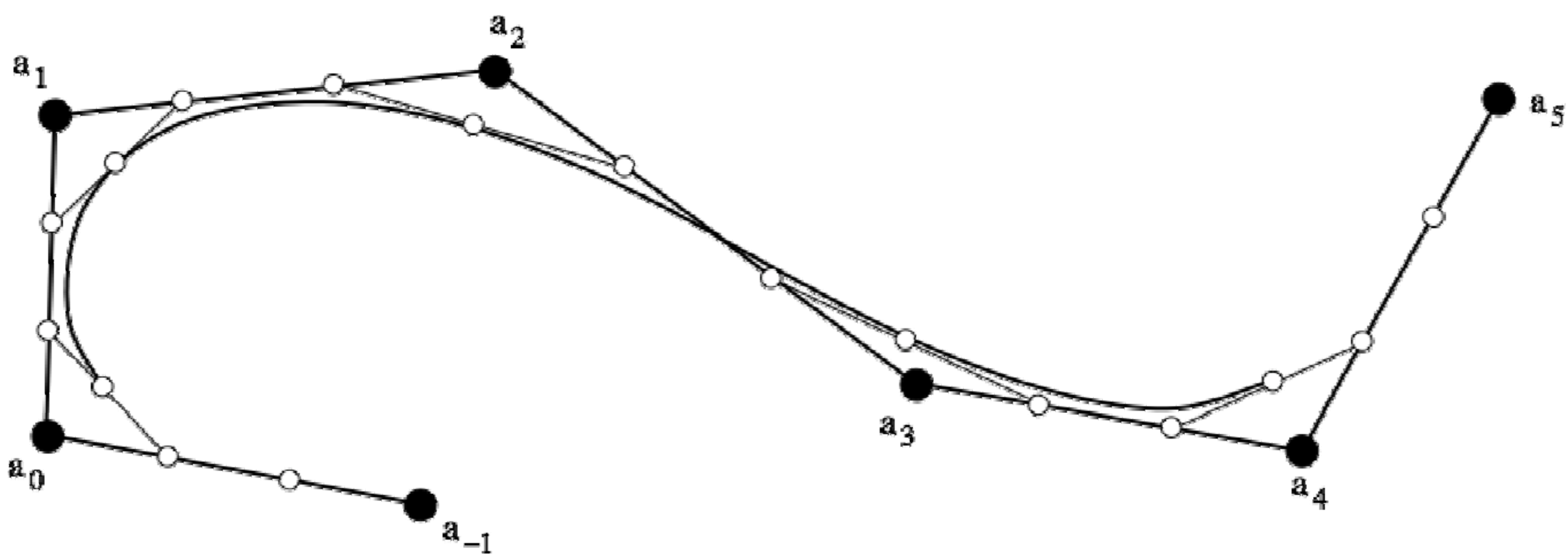
Piecewise Bezier Curves

- Example: cubic Bezier curves
- C^2 continuity
  - 3 control points of each curve affected
  - A-frame



Piecewise Bezier Curves

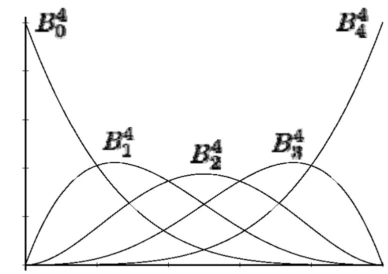
- Connect cubic segments to C^2 spline
- Control points defined by continuity conditions
- Can be specified by helper points alone



Piecewise Bezier Curves

- Properties
  - Bezier points of segments defined by helper points
  - Affine invariance
  - Convex hull
  - Maximal smoothness (here: C^2)
  - Local control?

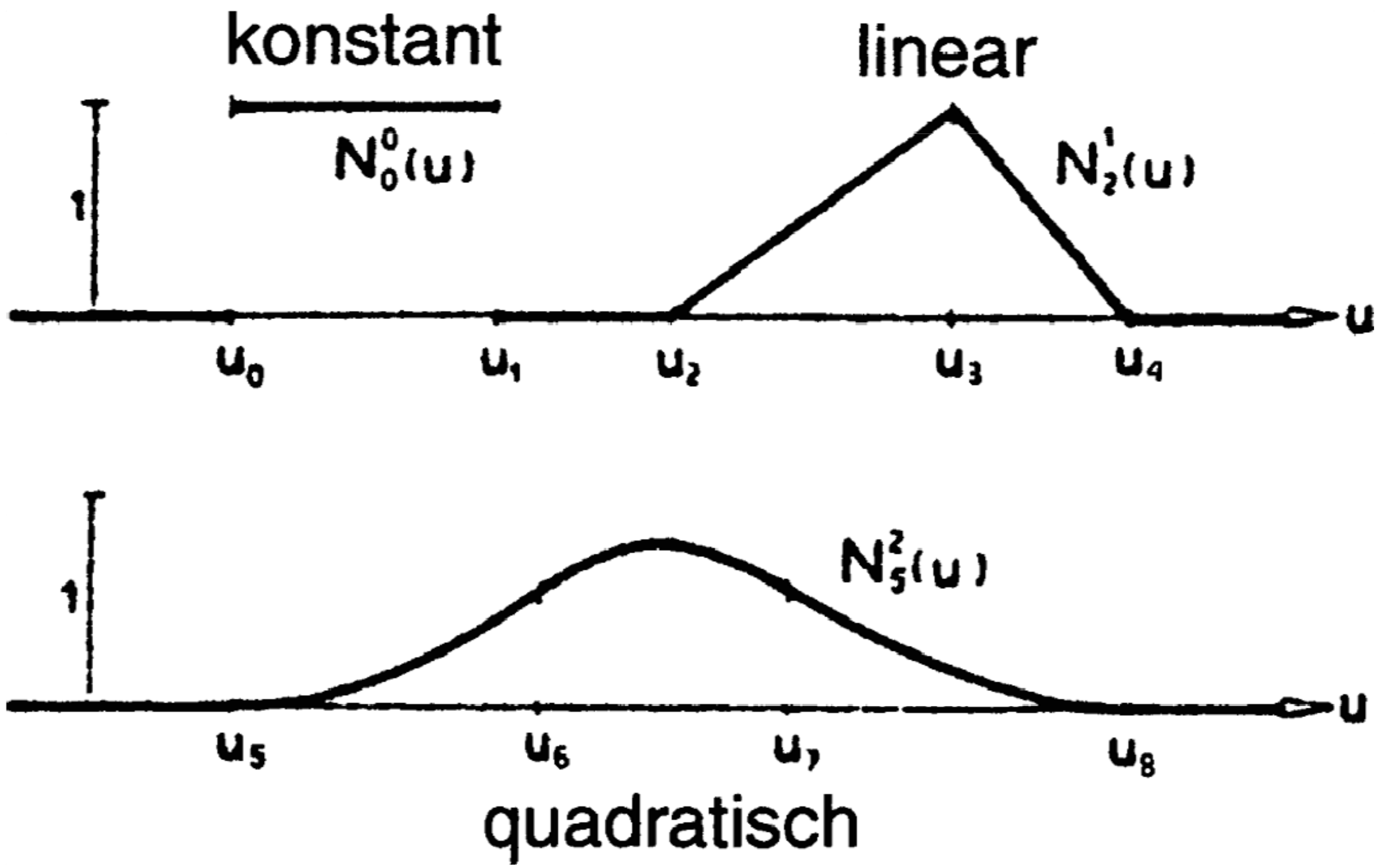
Piecewise Bezier Curves



- Disadvantages
  - Global support of basis functions
  - Insertion of control points increases degree
  - C^r continuity between segments restricts control polygon
- B(asis)-Spline bases overcome these problems
  - Local support
  - Continuity control
  - Each basis function has arbitrary support interval (knot vector)

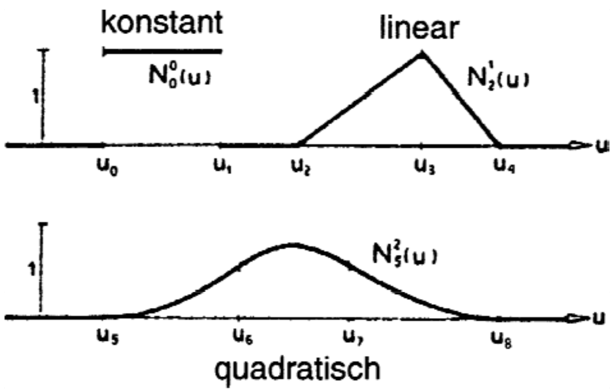
B-Spline Bases

- B-Spline bases of different degree



B-Spline Bases

- B-Spline bases of different degree



- Recurrence relation

$$N_i^0(u) = \begin{cases} 1 & u \in [u_i, u_{i+1}] \\ 0 & \text{else} \end{cases}$$

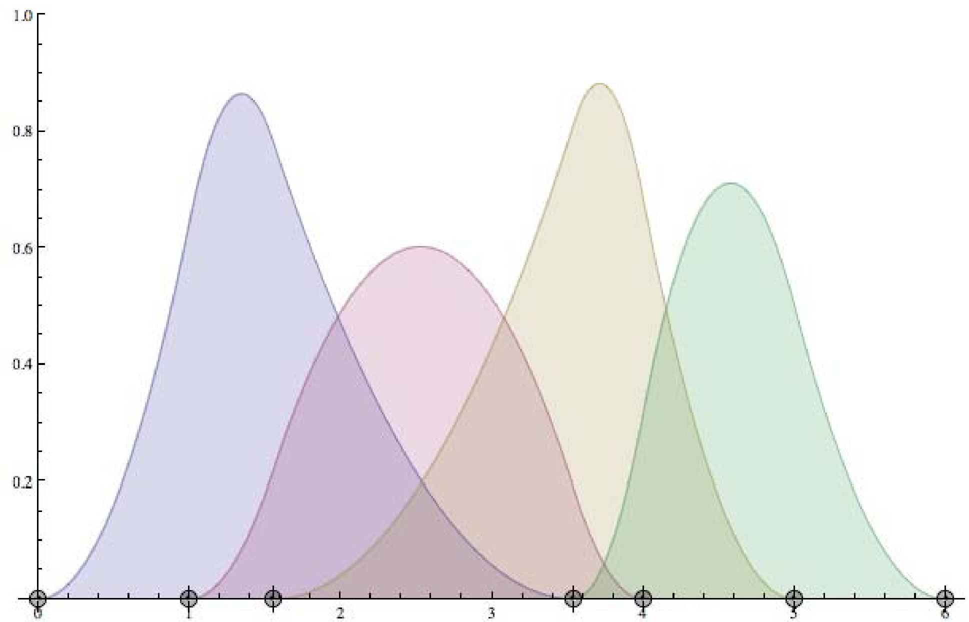
$$N_i^n(u) = (u - u_i)\frac{N_i^{n-1}(u)}{u_{i+n} - u_i} + (u_{i+n+1} - u)\frac{N_{i+1}^{n-1}(u)}{u_{i+n+1} - u_{i+1}}$$

- B-Spline basis of degree n has support over n+1 intervals of the knot vector

B-Spline Bases

DEMO

- Properties
  - Partition of unity
    $$\sum_i N_i^n(u) = 1$$
  - Positivity
    $$N_i^n(u) \geq 0$$
  - Compact support
    $$N_i^n(u) = 0,\ \forall u \notin [u_i, u_i + n + 1]$$
  - Continuity
    $N_i^n(u)$ is (n-1) continuously differentiable

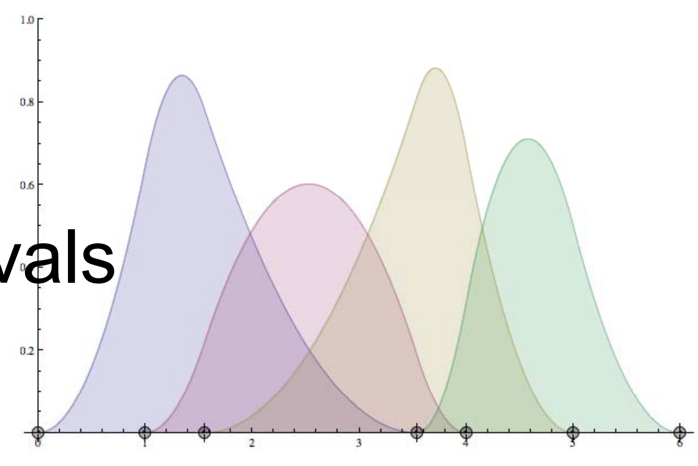


B-Spline Curve

- B-Spline curve is build from piecewise polynomial bases

$$s(u) = \sum_{i=0}^{k} d_i N_i^n(u)$$

- Coefficients d_i are called deBoor points
- Bases are piecewise, recursively defined polynomials
  - Sequence of knots $u_0 < u_1 < \ldots$
  - Endpoints of basis function intervals
  - Knot vector $\mathbf{u} = [u_0, \ldots, u_{k+n+1}]$



deBoor Algorithm

- Generalization of de Casteljau algorithm
- Evaluate curve $s(u)$ at parameter value u
  - control point in k-th step

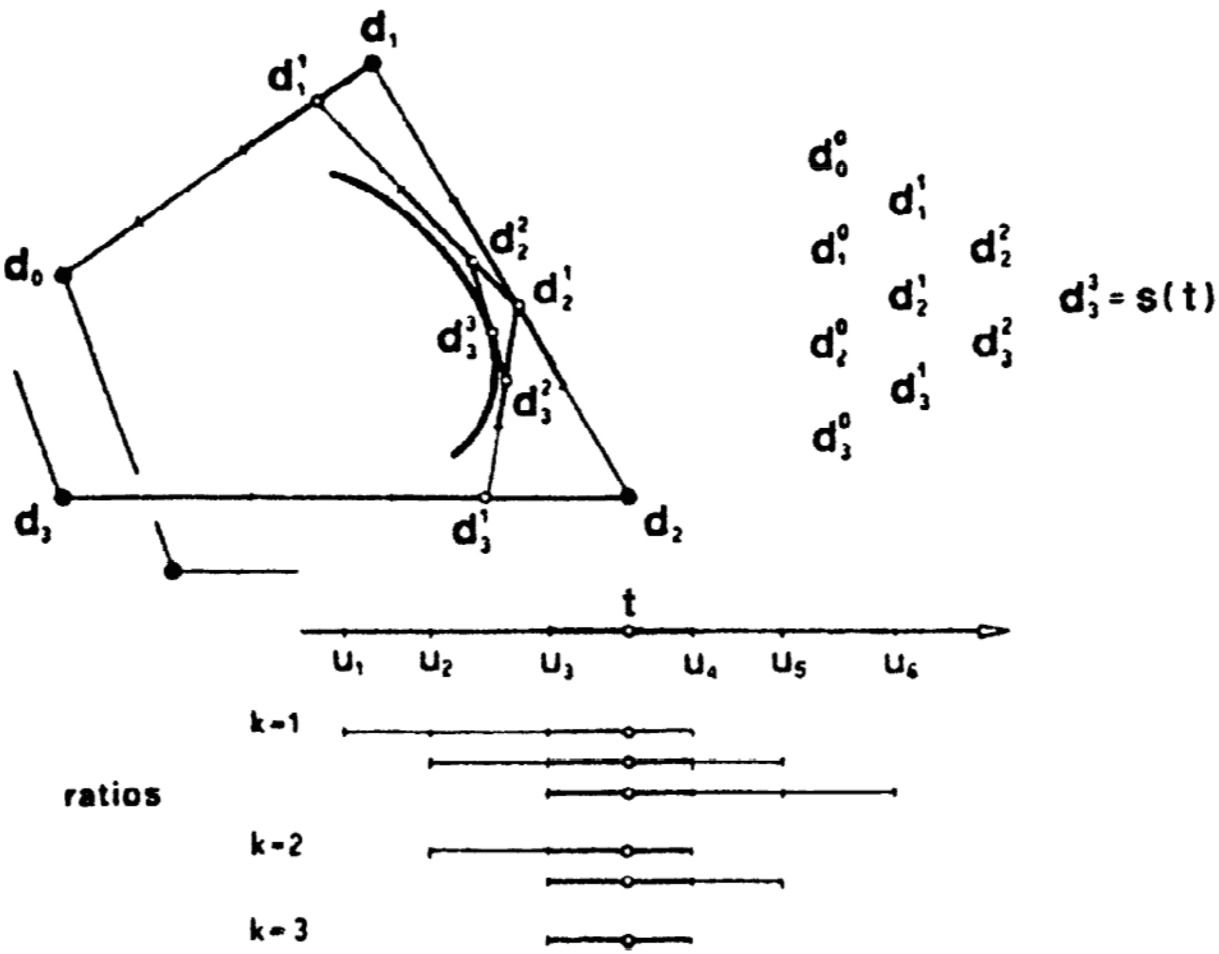
$$d_i^k = (1-\alpha_i^k)d_{i-1}^{k-1} + \alpha_i^k d_i^{k-1} \qquad \alpha_i^k = \frac{u-u_i}{u_{i+n+1-k}-u_i}$$

$$d_i^0 = d_i,\ d_n^n = s(u)$$

deBoor Algorithm

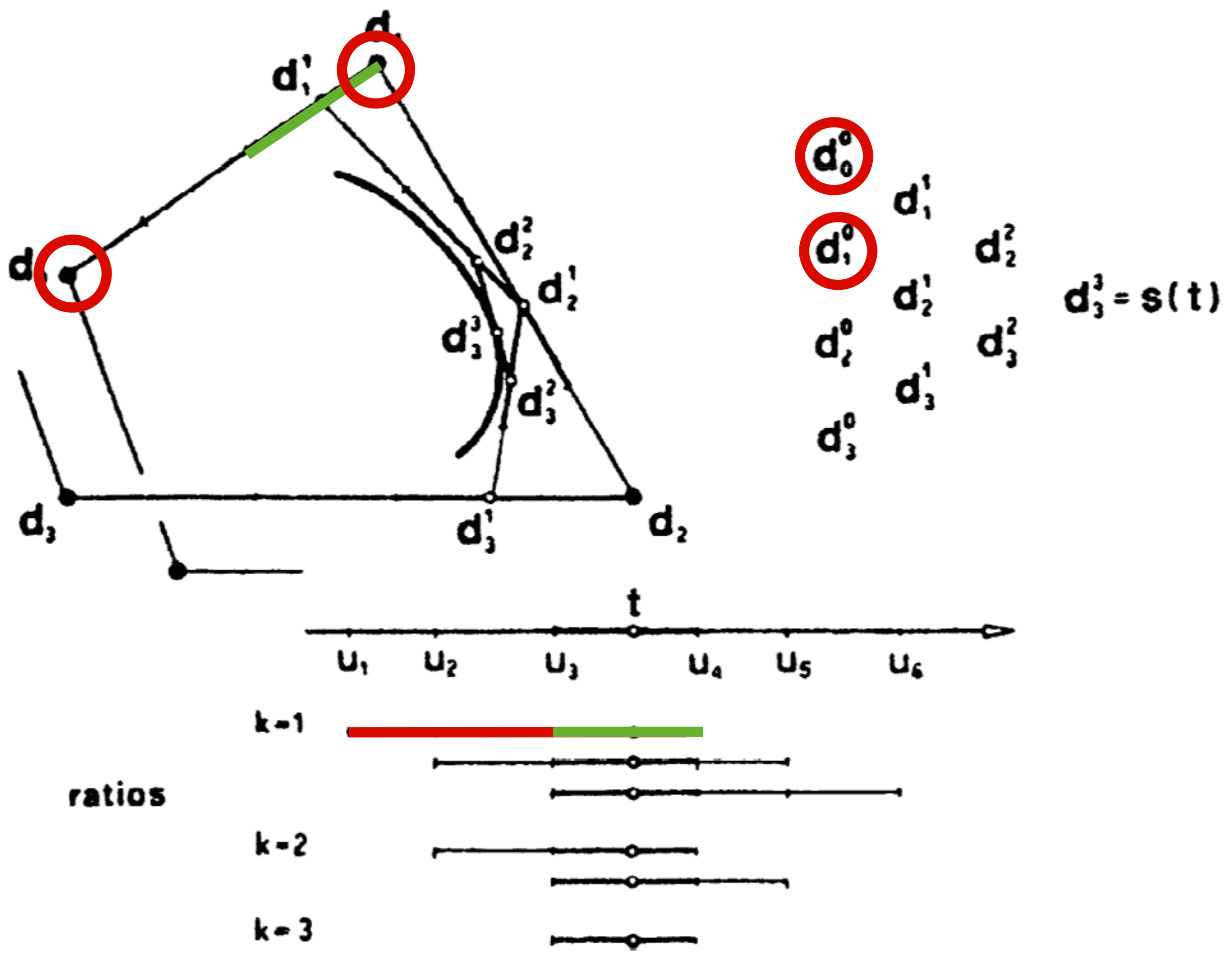
DEMO

- Cubic Basis



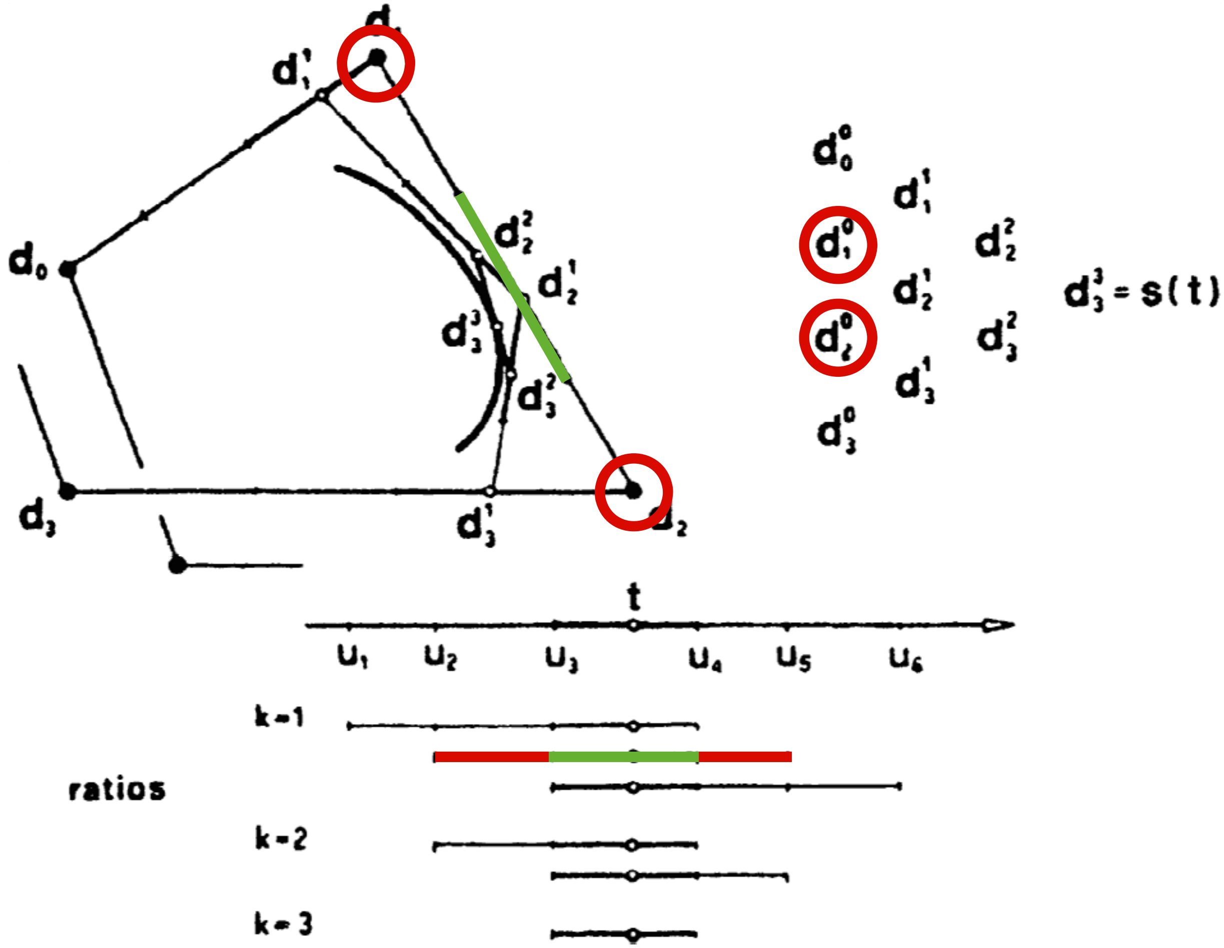
deBoor Algorithm

- Cubic Basis



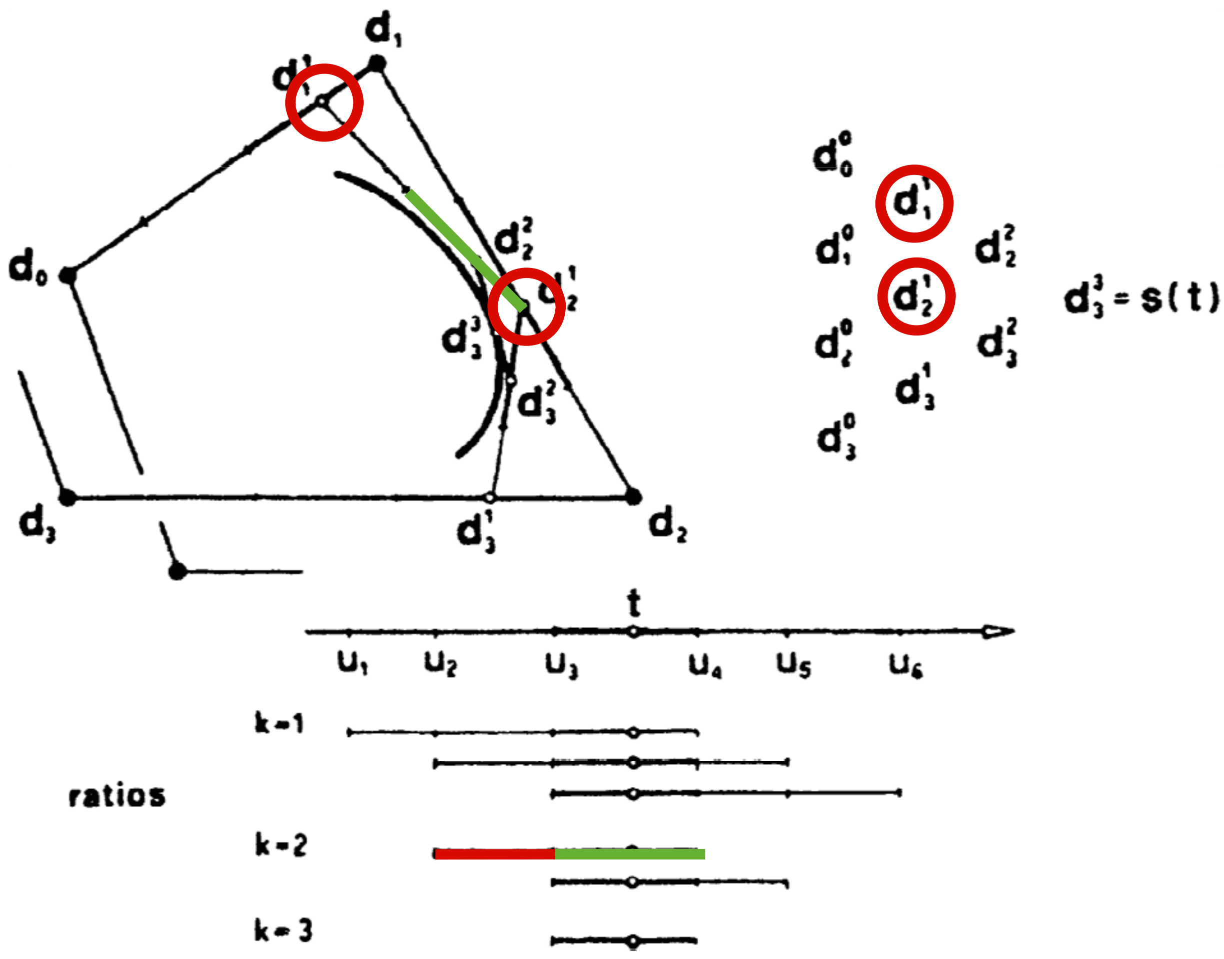
deBoor Algorithm

- Cubic Basis



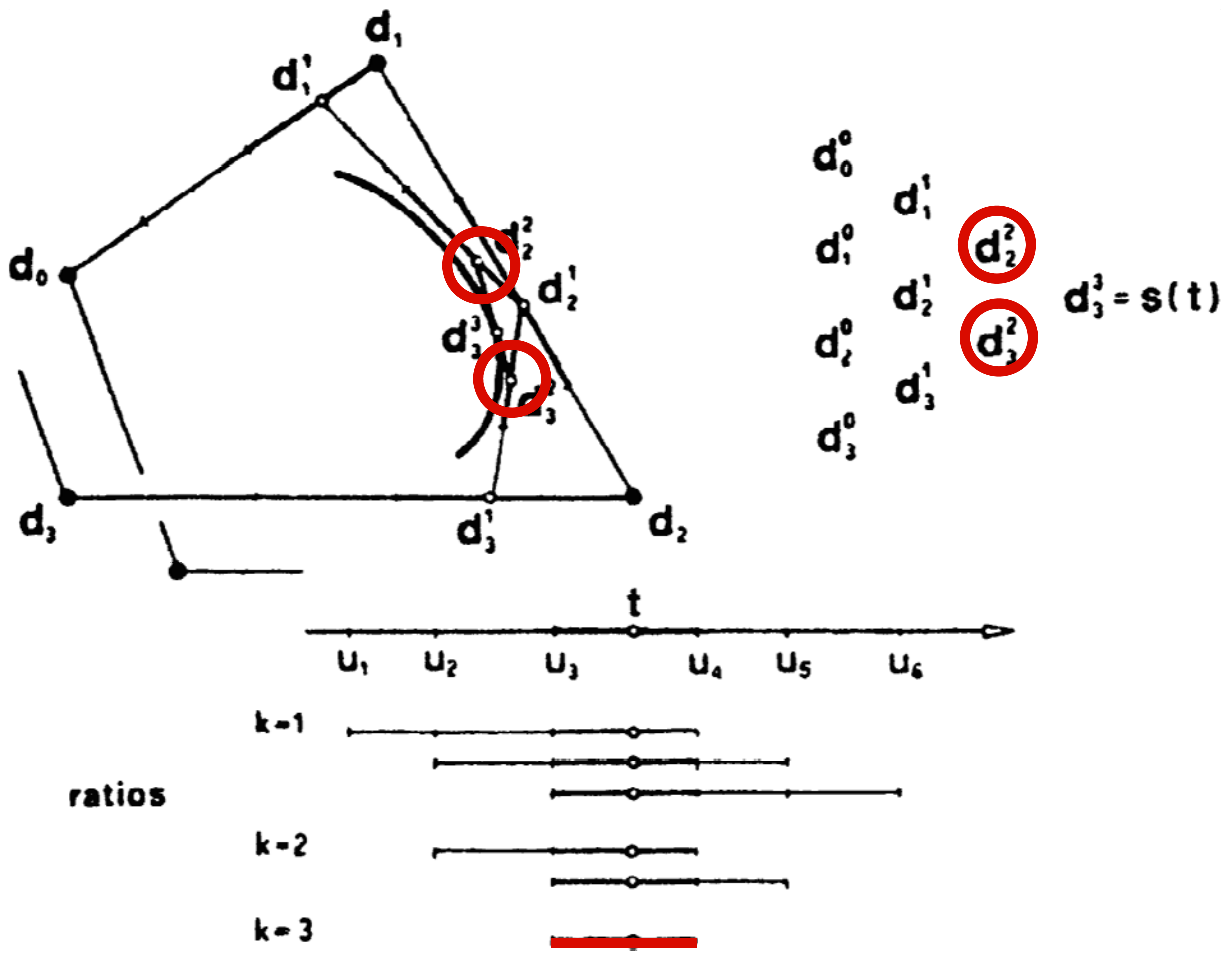
deBoor Algorithm

- Cubic Basis



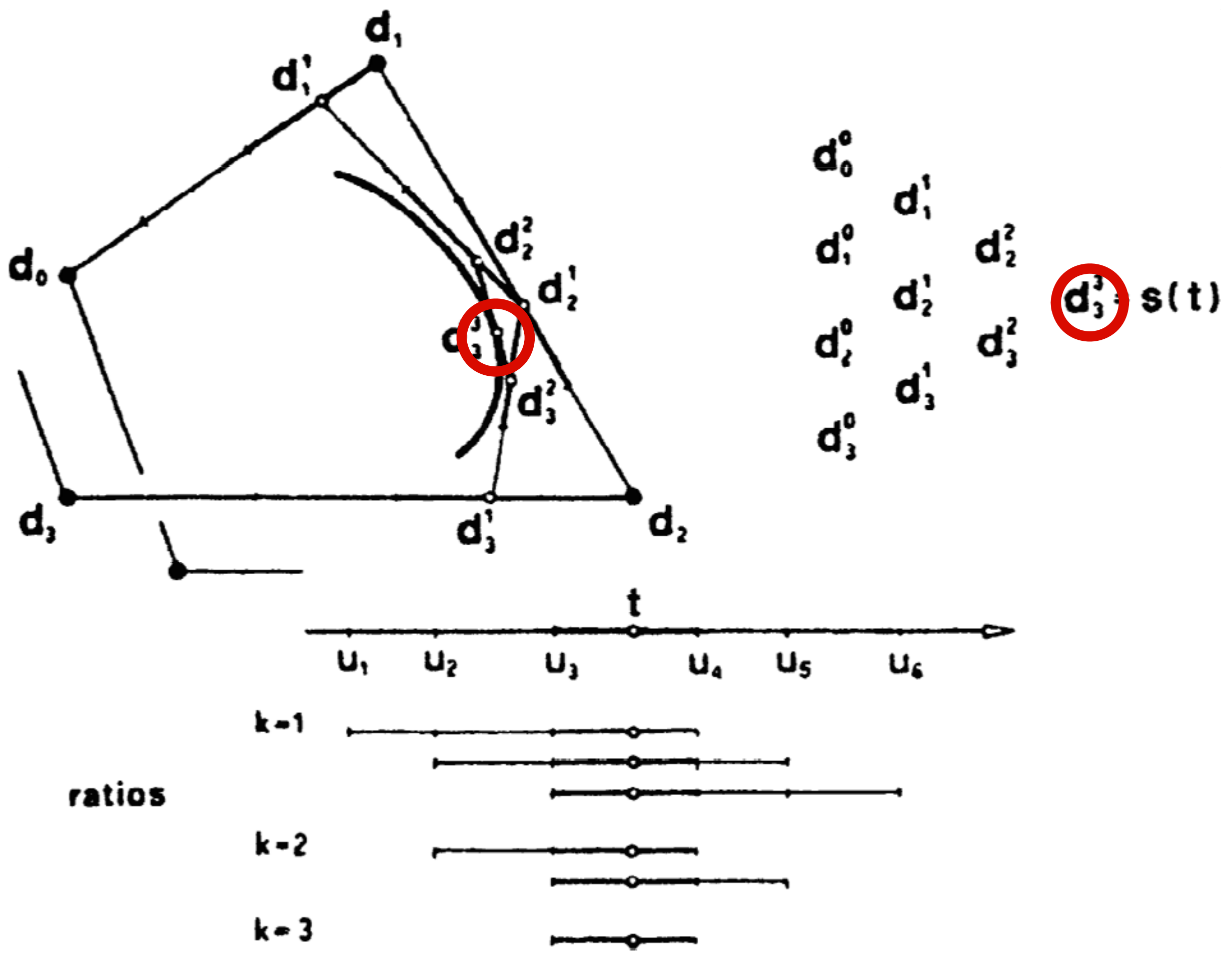
deBoor Algorithm

- Cubic Basis



deBoor Algorithm

- Cubic Basis



deBoor Algorithm

DEMO

- De Casteljau is a special case of deBoor:
  - First and last knot have multiplicity of n+1

    $$0 = u_0 = \cdots = u_n < u_{n+1} = \cdots = u_{2n+1}$$

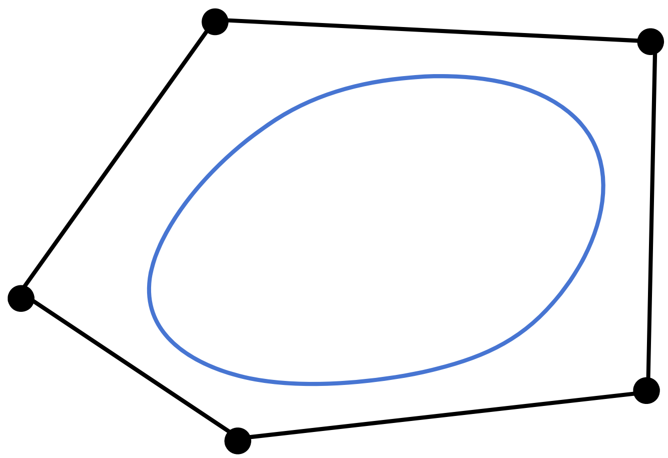
  - with $u_{n+k} = 1 \;\; \forall k \in [1, \ldots, n+1]$
  - Basis functions have global support

$$d_i^k(u) = u d_i^{k-1}(u) + (1-u) d_{i+1}^{k-1}(u)$$

de Casteljau Algorithm

End Conditions

- Closed curves
  - Periodic repetition of de Boor points and knots



B-Spline Curves

- Knot vector
  - Insertion of new knots does not change degree but number of segments/basis functions $\mathbf{u} = [u_0, \ldots, u_{k+n+1}]$
  - Knots can have multiplicity, i.e., $u_j = \ldots = u_{j+p-1}$
- Continuity
  - Curve is globally C^{n-1} continuous
  - At points of multiplicity p, continuity drops to C^{n-p}
- Properties
  - Variation diminishing & local convex hull

B-Spline Curves

- Control the curve by
  - moving control points
  - moving knots
- How intuitive is it to move knots? ...

Rational Bezier Curves

- Generalization of conic sections

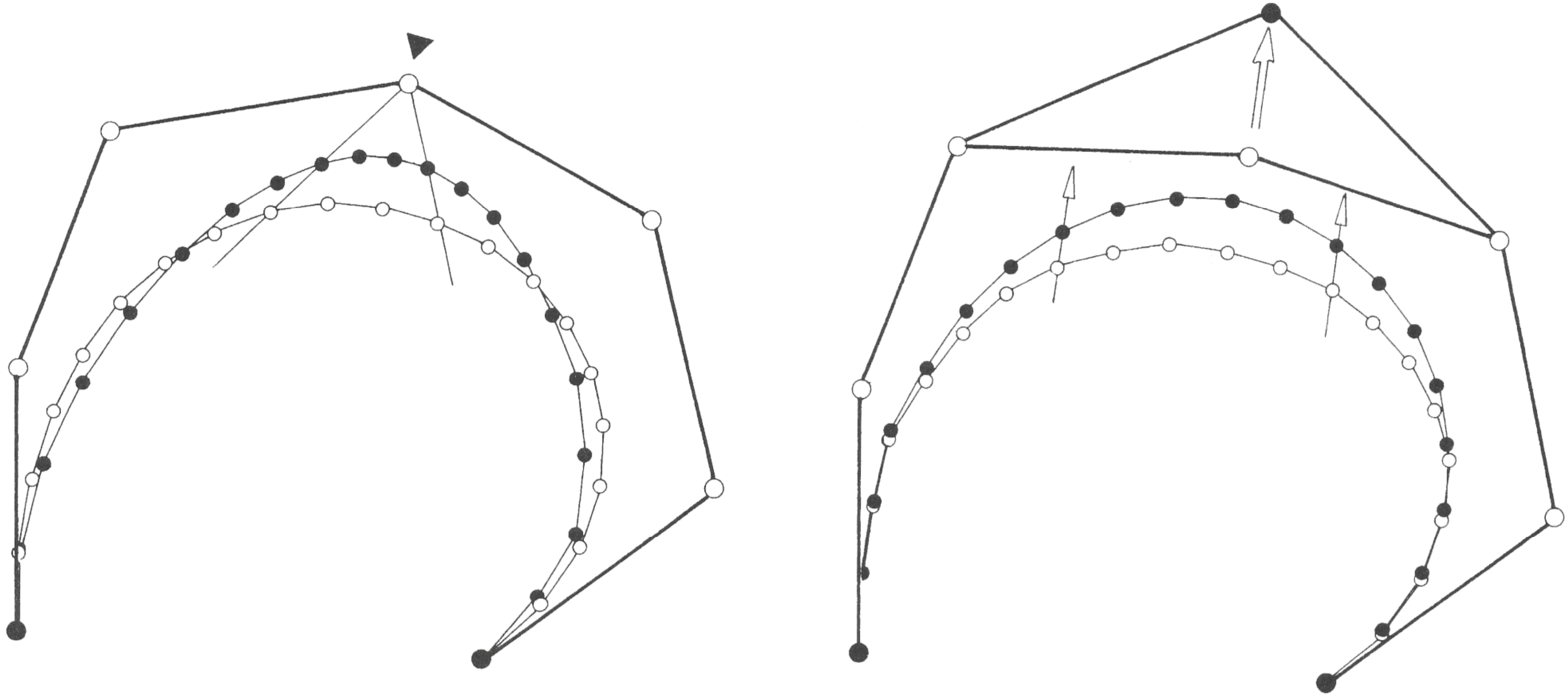
$$\mathbf{x}(t) = \frac{w_0 \mathbf{b}_0 B_0^n(t) + \cdots + w_n \mathbf{b}_n B_n^n(t)}{w_0 B_0^n(t) + \cdots + w_n B_n^n(t)} \qquad \mathbf{x}(t), \mathbf{b}_i \in \mathbb{R}^3$$

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{b}_i \frac{\omega_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)}$$

basis functions

Rational Bezier Curves

- Changing weights vs. moving control points
  - w_i shape parameters



Rational Bezier Curves

- deCasteljau algorithm: two alternatives
  - evaluate numerator and denominator separately
    - fast
    - unstable for large variations of weights
  - project intermediate points to hyperplane $w=1$

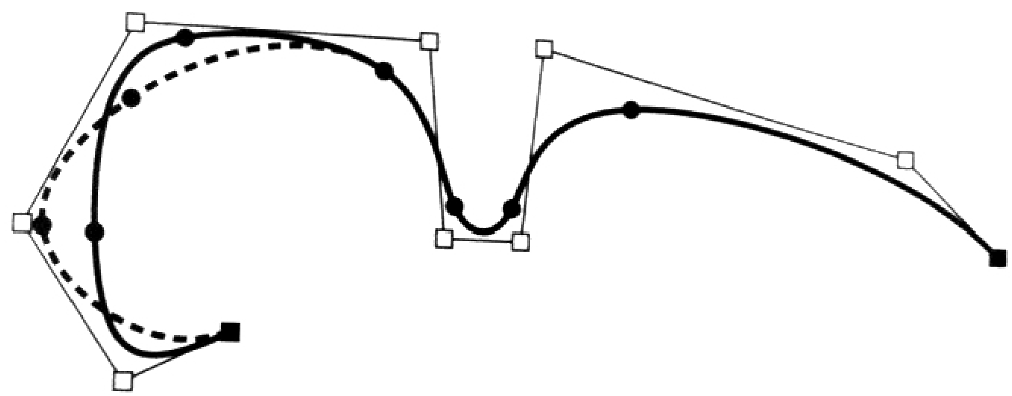
$$\mathbf{b}_i^r(t) = \frac{\sum_{j=0}^{r} w_{i+j}\mathbf{b}_{i+j}B_j^r(t)}{\sum_{j=0}^{r} w_{i+j}B_j^r(t)}$$

    - slower, but more stable

Rational B-Spline Curves

- **NURBS (N**on-**U**niform **R**ational **B-S**plines)
- Defined by
  - Knot sequence
  - 2D/3D control polygon
  - Weight sequence

$$\mathbf{x}(u) = \frac{\sum_{i=0}^{L} w_i \mathbf{d}_i N_i^n(u)}{\sum_{i=0}^{L} w_i N_i^n(u)}$$

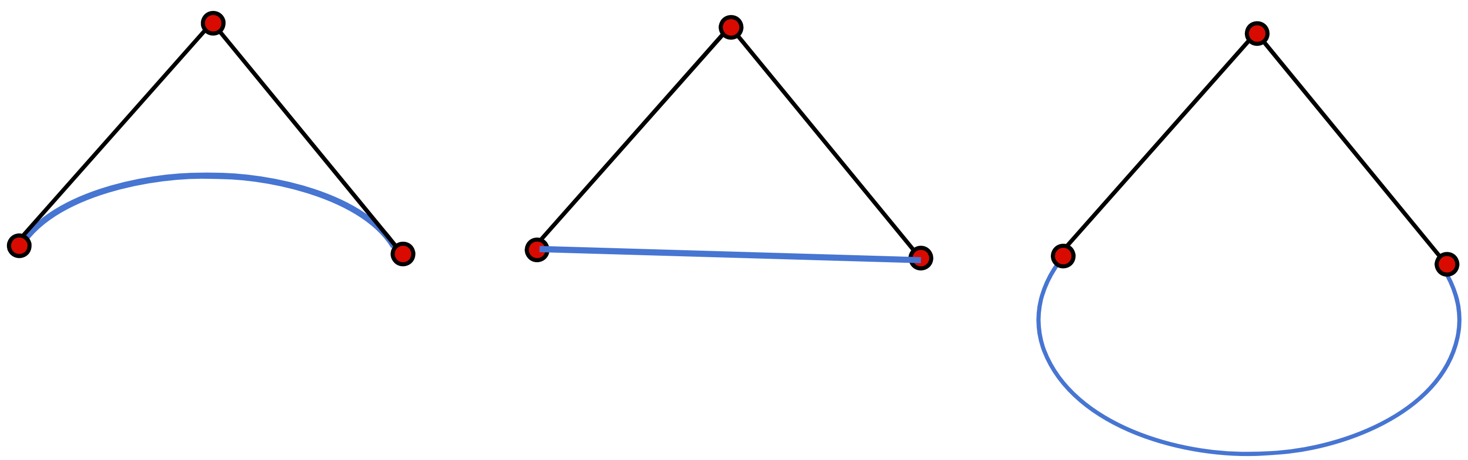


- Extension of deBoor algorithm
  - analogous to rational deCasteljau

Rational B-Spline Curves

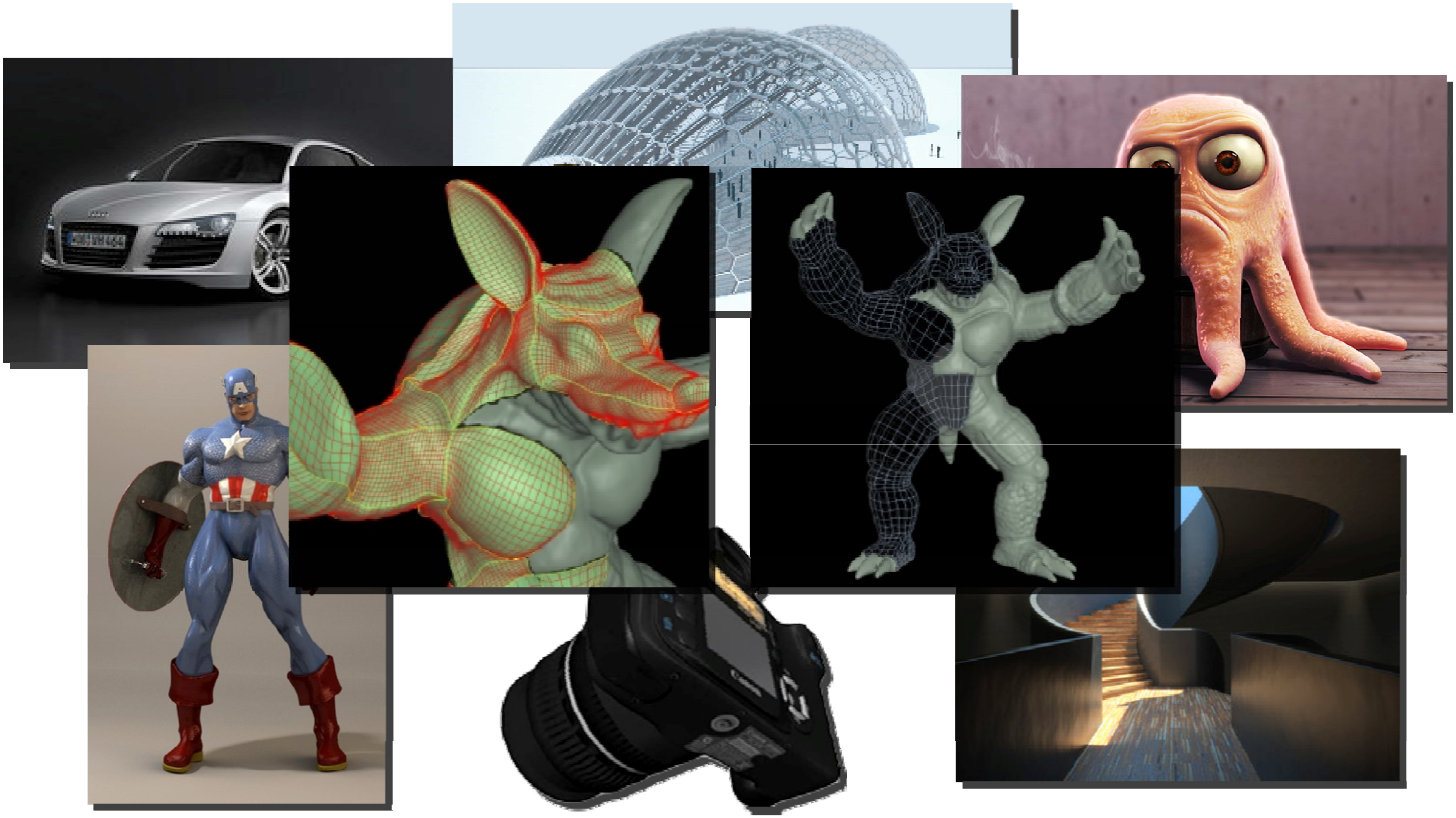
- Properties
  - local control
  - convex hull?
  - variation diminishing?

only if weights are non-negative!



Freeform Surfaces

- Extend curves to surfaces

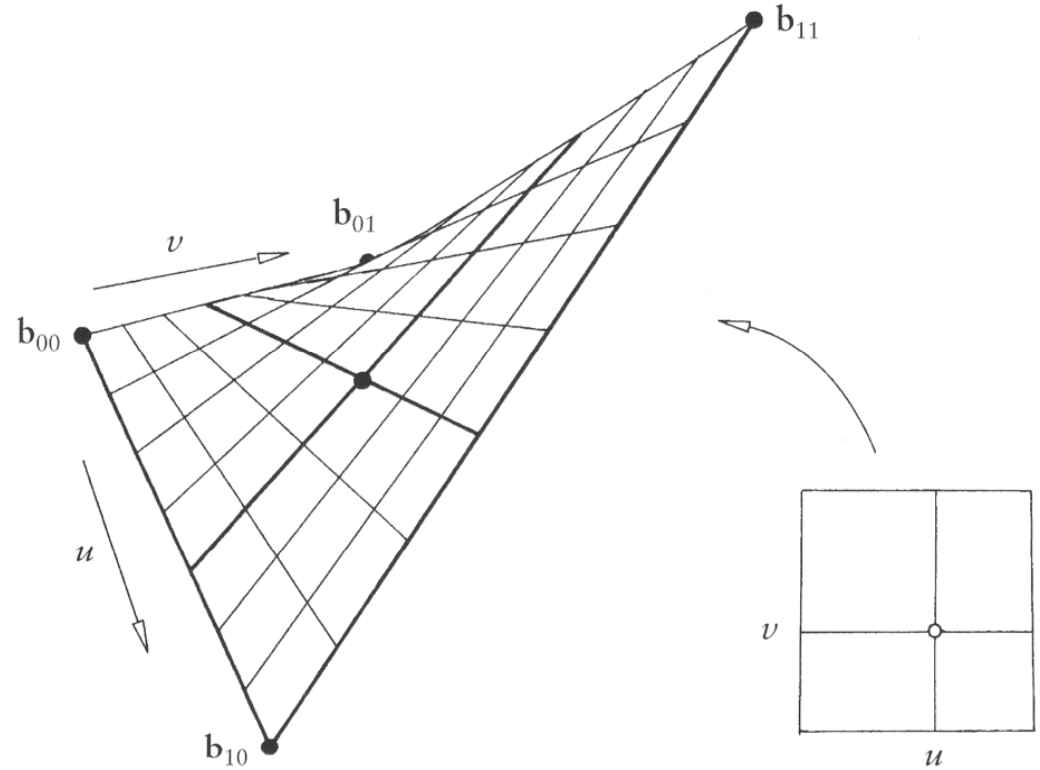


Bilinear Interpolation

- Hyperbolic paraboloid

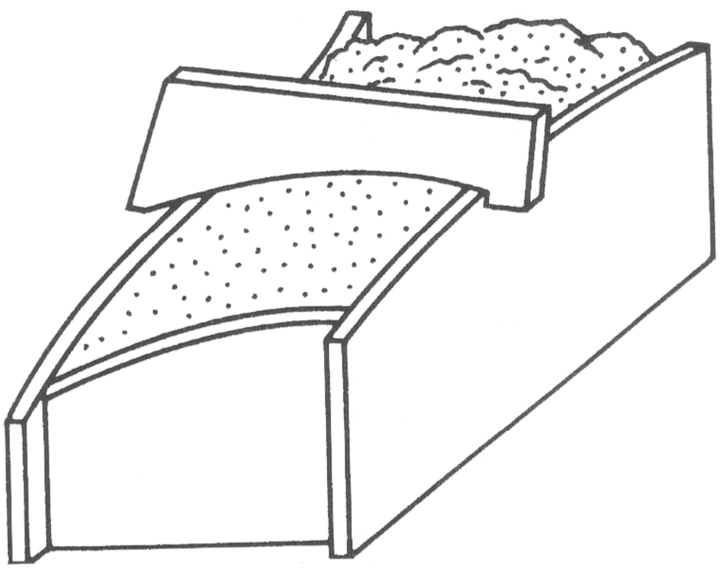
$$\mathbf{x}(u,v) = [1-u \quad u]\begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{bmatrix}\begin{bmatrix} 1-v \\ v \end{bmatrix}$$

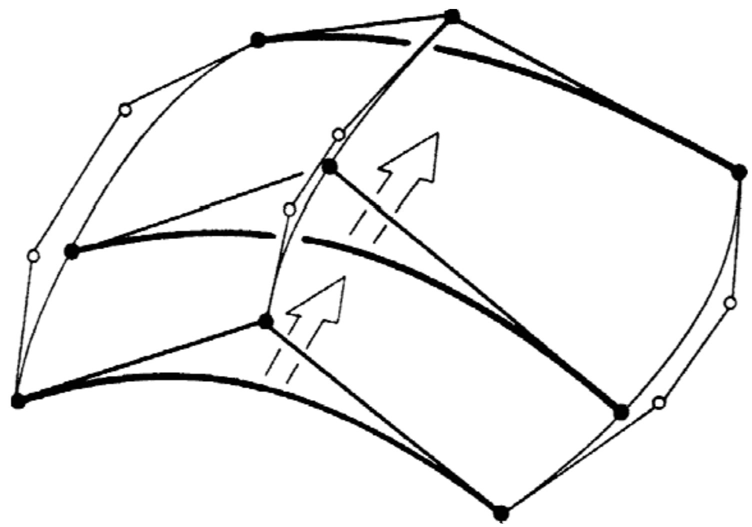
- Isoparametric curve?



Tensor Product Surfaces

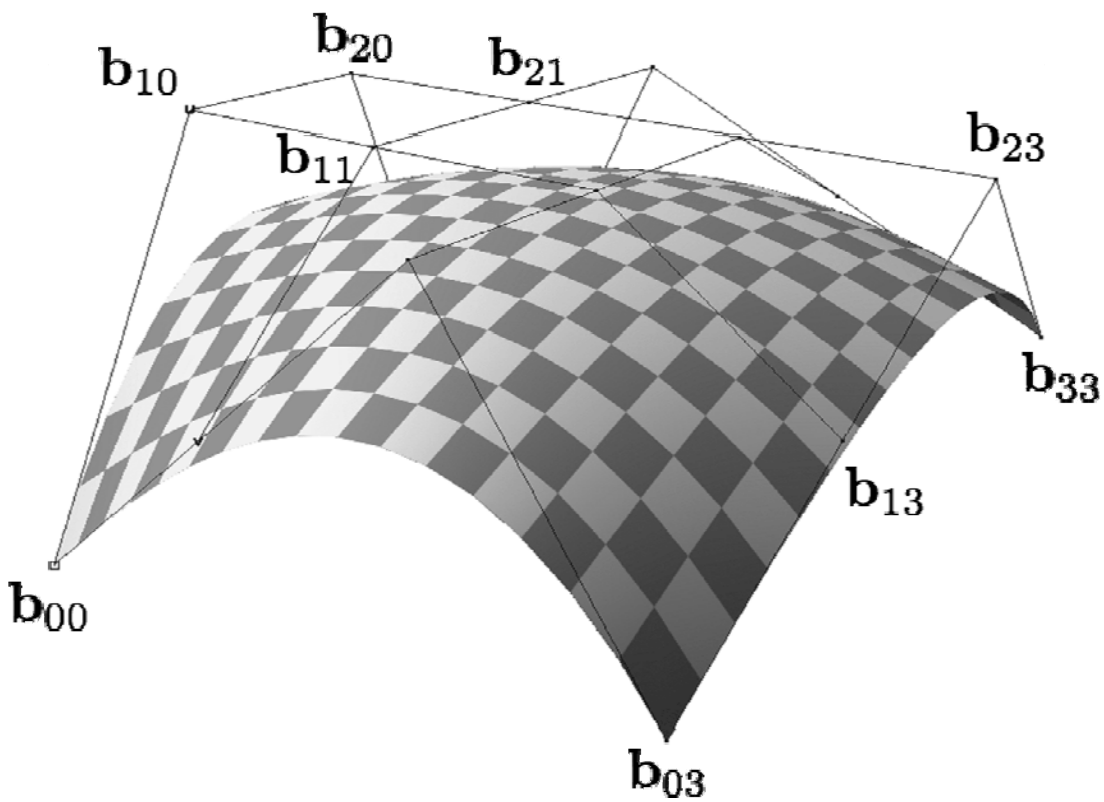
- Surface defined by curve moving through space
  - Curve may deform as it moves





Tensor Product Surfaces

- Surface defined by curve moving through space
  - Curve may deform as it moves
- Example: Bicubic Bezier Patch



Tensor Product Surfaces

- Surface defined by curve moving through space
  - Curve may deform as it moves
- Bezier curve

$$f(u) = \sum_{i=0}^{m} b_i B_i^m(u)$$

- Move control points on curves

$$f(u,v) = \sum_{i=0}^{m} b_i(v) B_i^m(u) \qquad b_i(v) = \sum_{j=0}^{n} b_{i,j} B_j^n(v)$$

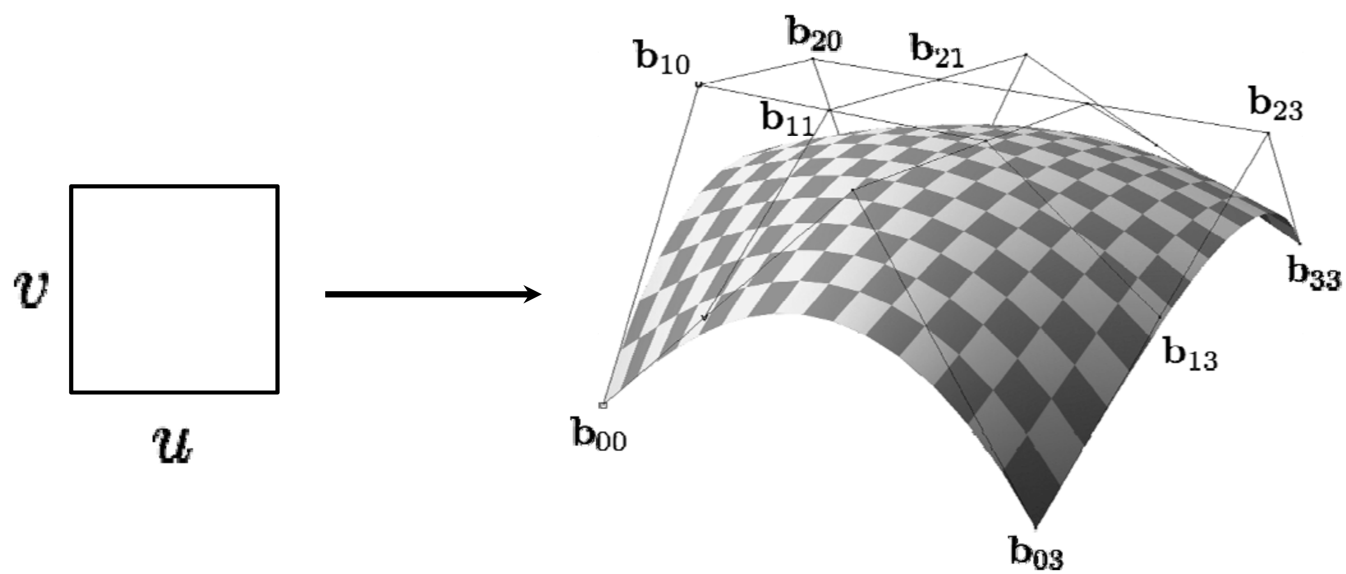
$$f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{i,j} B_i^m(u) B_j^n(v)$$

Tensor Product Surfaces

- Bezier patch: $f : [0,1]^2 \rightarrow \mathbb{R}^3$

$$f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{i,j} B_i^m(u) B_j^n(v)$$

$(m+1) \times (n+1)$ control points



Tensor Product Surfaces

- Bezier patch properties
  - Affine invariance
  - Convex hull
  - Boundary curves $f(u,0), f(u,1), f(0,v), f(1,v)$
  - Corner interpolation and normals
  - Smooth junctions
  - Derivatives

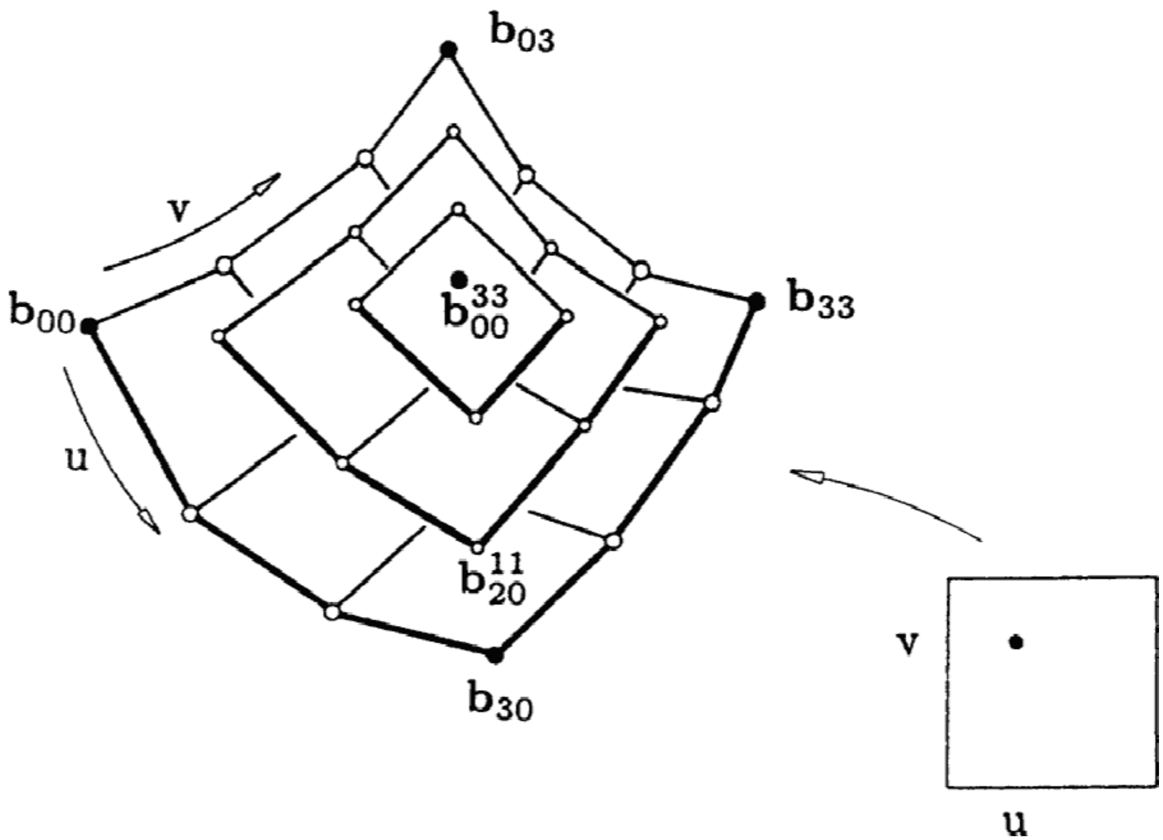
$$\frac{\partial}{\partial u} f(u,v) = m \sum_{i=0}^{m-1} \sum_{j=0}^{n} (b_{i+1,j} - b_{i,j}) B_i^{m-1}(u) B_j^n(v)$$

$$\frac{\partial}{\partial v} f(u,v) = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} (b_{i,j+1} - b_{i,j}) B_i^{m}(u) B_j^{n-1}(v)$$

$$\frac{\partial}{\partial u} f(u,v) \times \frac{\partial}{\partial v} f(u,v)$$

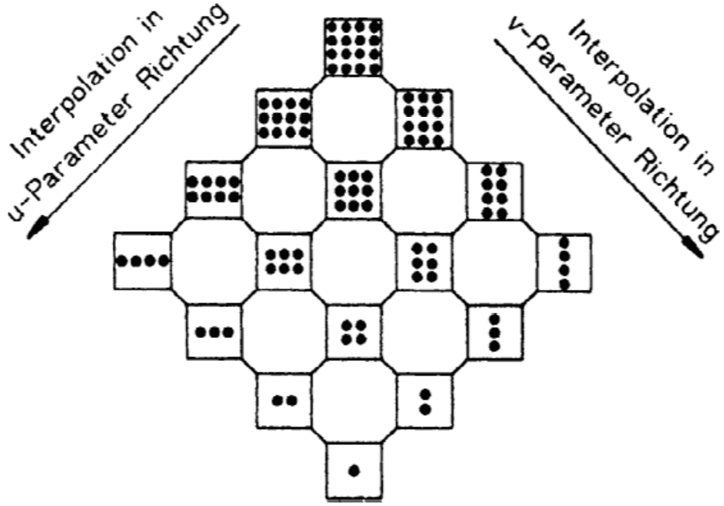
2D De Casteljau Algorithm

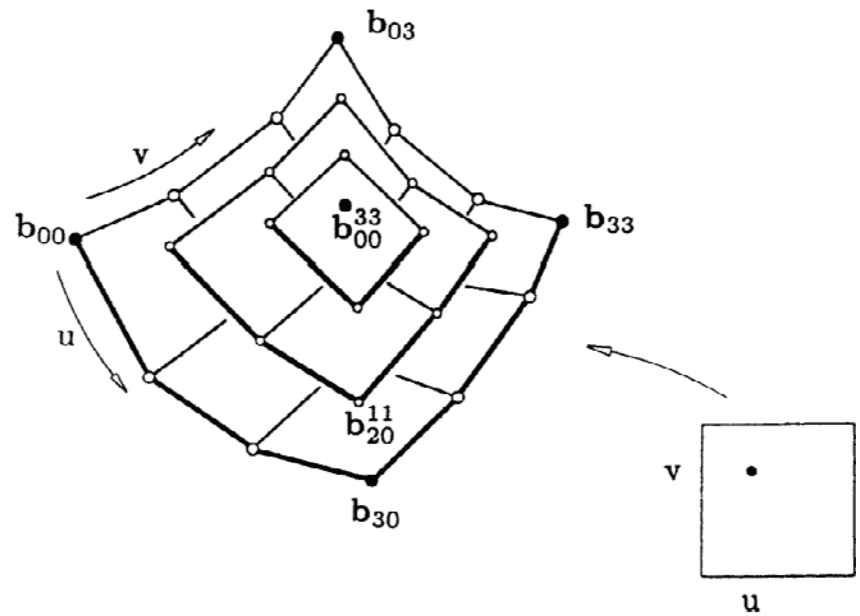
- Repeated bilinear interpolation



2D De Casteljau Algorithm

- Repeated bilinear interpolation





2D De Casteljau Algorithm

- 2D array of control points $\mathbf{b}_{i,j} = \mathbf{b}_{i,j}^{0,0},\ 0 \le i, j \le n$
- Parameter (u,v)
- Recursive interpolation

$$\mathbf{b}_{i,j}^{r,r} = [1-u \quad u] \begin{bmatrix} \mathbf{b}_{i,j}^{r-1,r-1} & \mathbf{b}_{i,j+1}^{r-1,r-1} \\ \mathbf{b}_{i+1,j}^{r-1,r-1} & \mathbf{b}_{i+1,j+1}^{r-1,r-1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

$$r = 1, \ldots, n$$

$$i, j = 0, \ldots, n-r$$

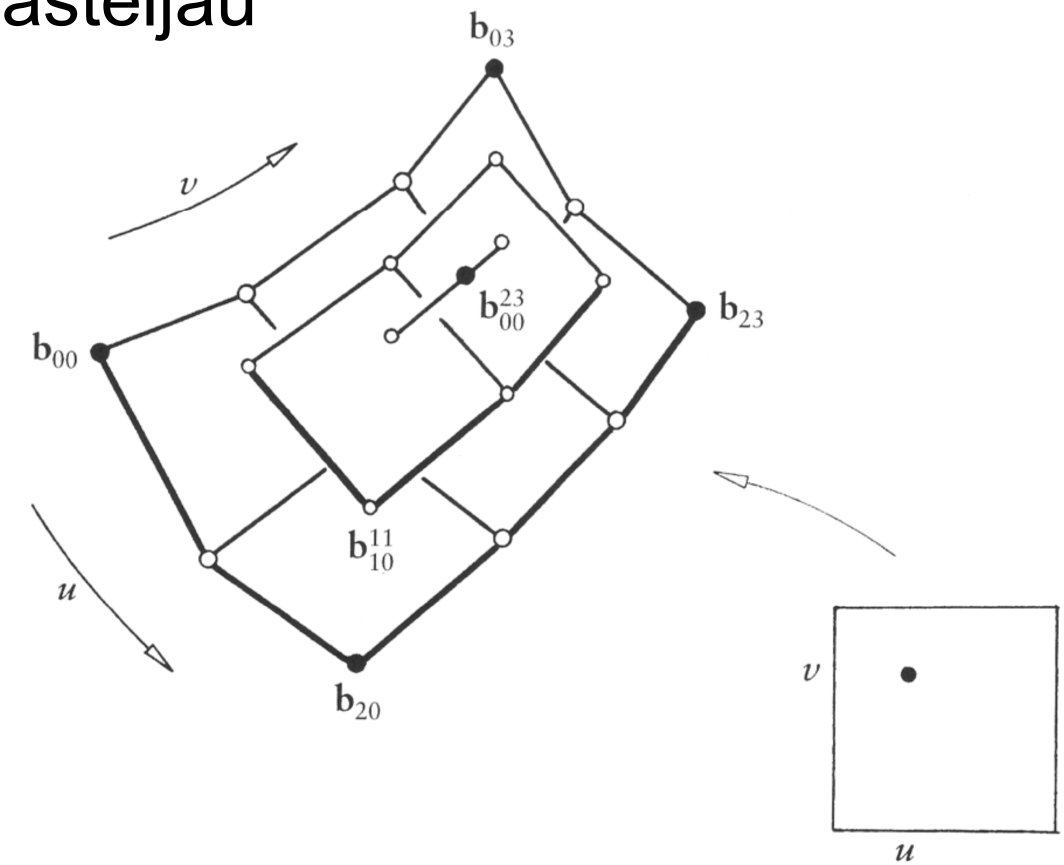
2D De Casteljau Algorithm

- Example (u,v)=(0.5, 0.5)

$$
\begin{matrix}
\begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline 0 \\ 2 \\ 0 \\ \hline 0 \\ 4 \\ 0 \end{bmatrix}
\begin{bmatrix} 2 \\ 0 \\ 0 \\ \hline 2 \\ 2 \\ 0 \\ \hline 2 \\ 4 \\ 4 \end{bmatrix}
\begin{bmatrix} 4 \\ 0 \\ 0 \\ \hline 4 \\ 2 \\ 2 \\ \hline 4 \\ 4 \\ 4 \end{bmatrix}
&
\longrightarrow
&
\begin{bmatrix} 1 \\ 1 \\ 0 \\ \hline 1 \\ 3 \\ 1 \end{bmatrix}
\begin{bmatrix} 3 \\ 1 \\ 0.5 \\ \hline 3 \\ 3 \\ 2.5 \end{bmatrix}
&
\longrightarrow
&
\begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}
\\
r = 1 & & r = 2 & & r = 3
\end{matrix}
$$

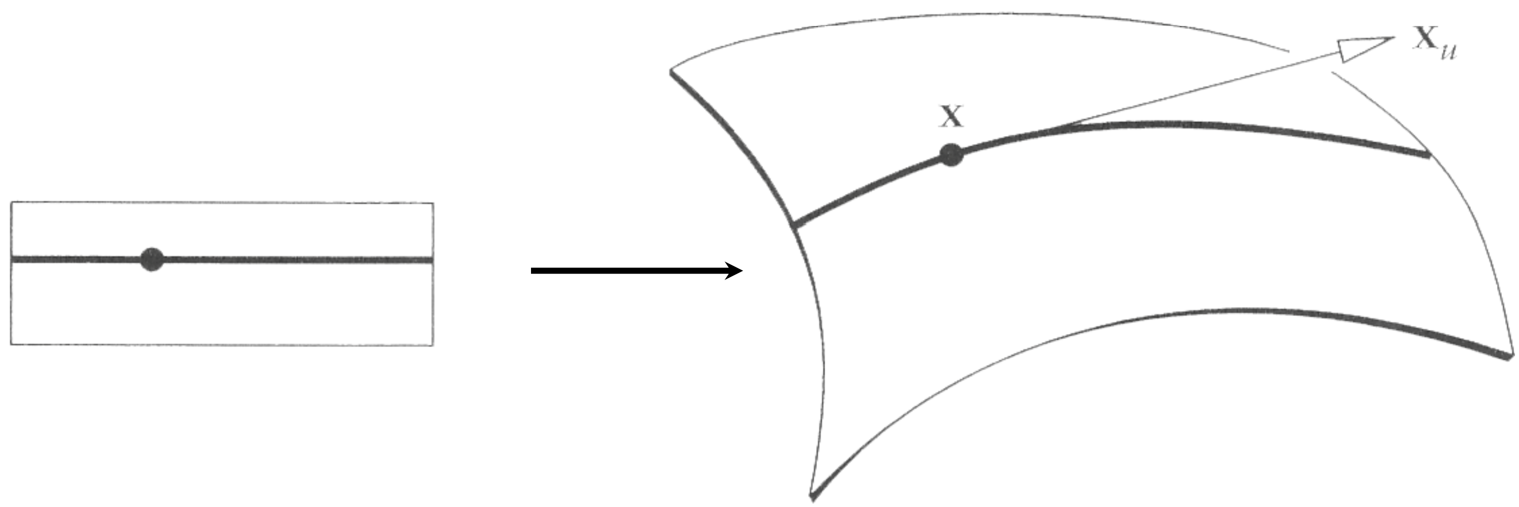
2D De Casteljau Algorithm

- If #control points differs in u- and v- direction
  1. compute $k = \min(m,n)$ 2D interpolation steps
  2. proceed with 1D deCasteljau



Derivatives

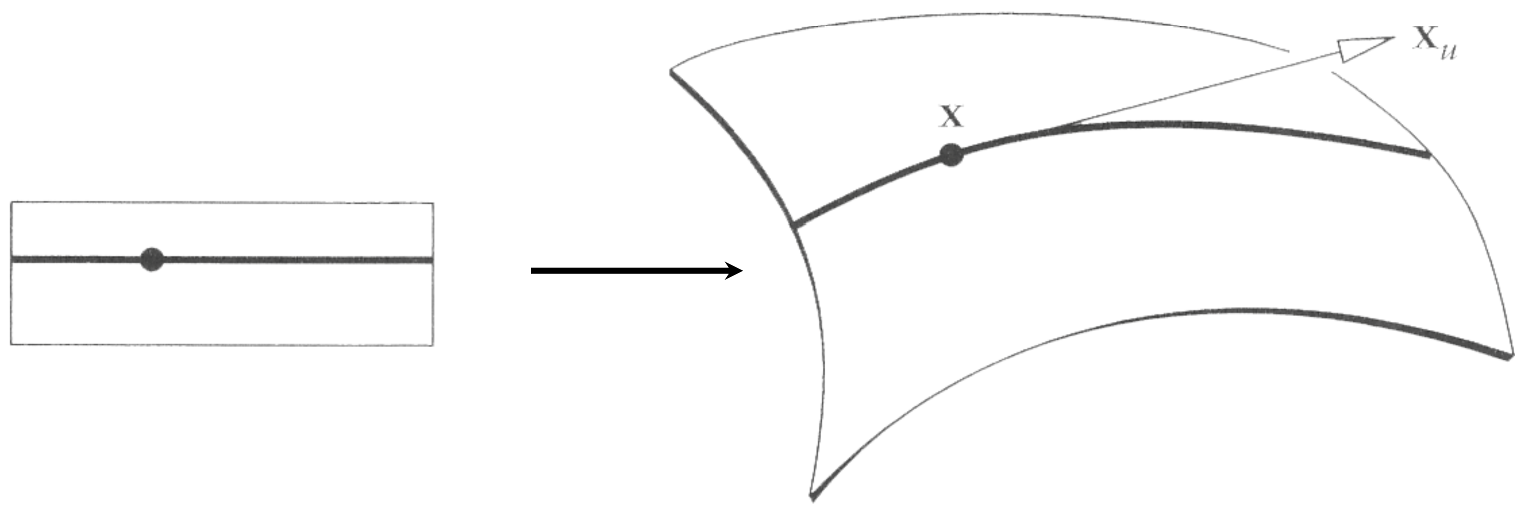
- Analogous to curve setting → partial derivatives



Derivatives

- Analogous to curve setting → partial derivatives

$$\frac{\partial}{\partial u} b^{m,n}(u,v) = \sum_{j=0}^{n} \left[\frac{\partial}{\partial u} \sum_{i=0}^{m} b_{i,j} B_i^m(u) \right] B_j^n(v)$$

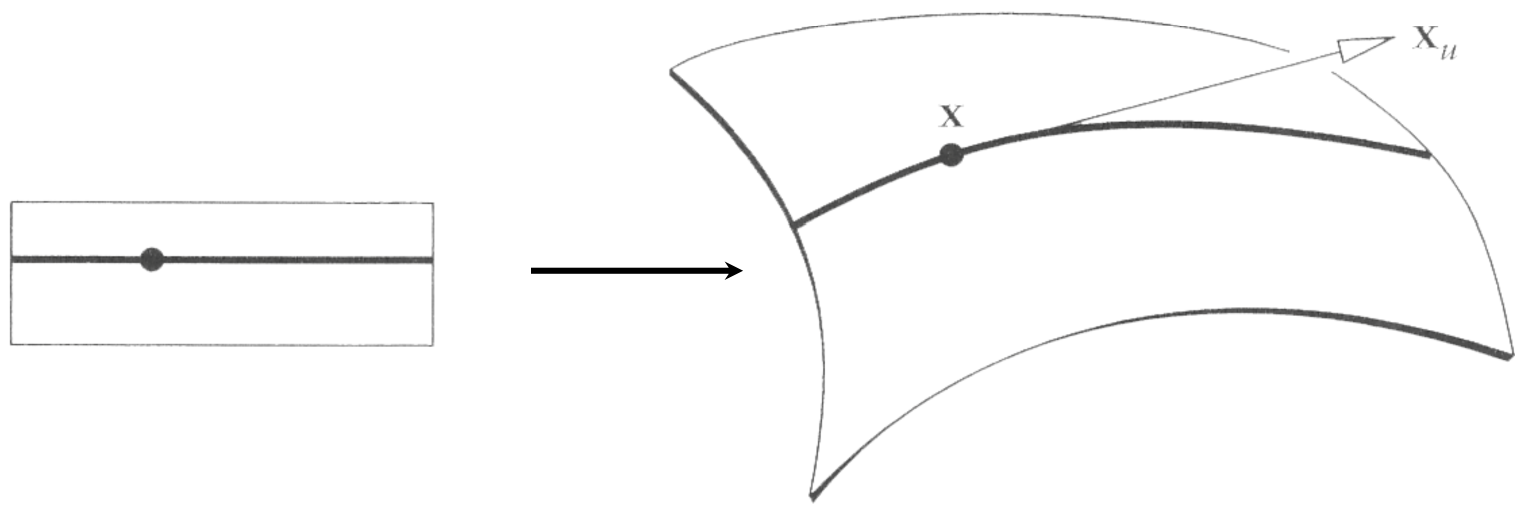


Derivatives

- Analogous to curve setting → partial derivatives

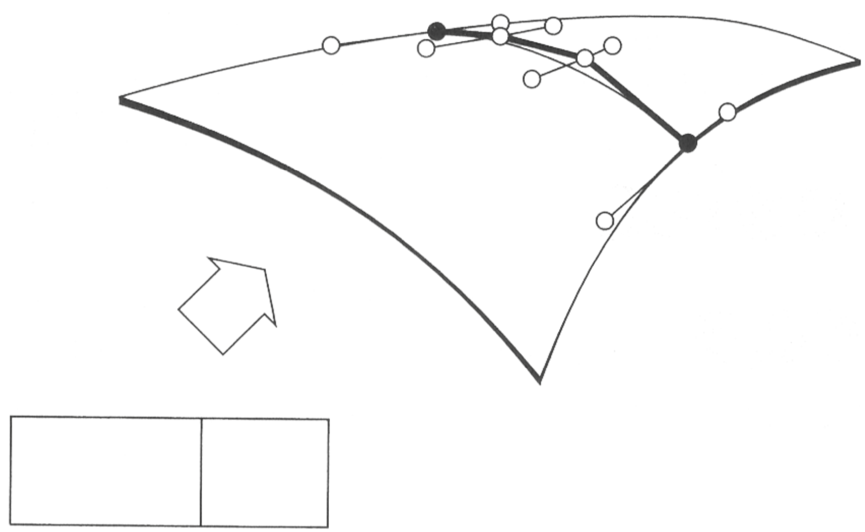
$$\frac{\partial}{\partial u} b^{m,n}(u,v) = \sum_{j=0}^{n} \left[\frac{\partial}{\partial u} \sum_{i=0}^{m} b_{i,j} B_i^m(u) \right] B_j^n(v)$$

$$= \sum_{j=0}^{n} m \sum_{i=0}^{m-1} (b_{i+1,j} - b_{i,j}) B_i^{m-1}(u) B_j^n(v)$$



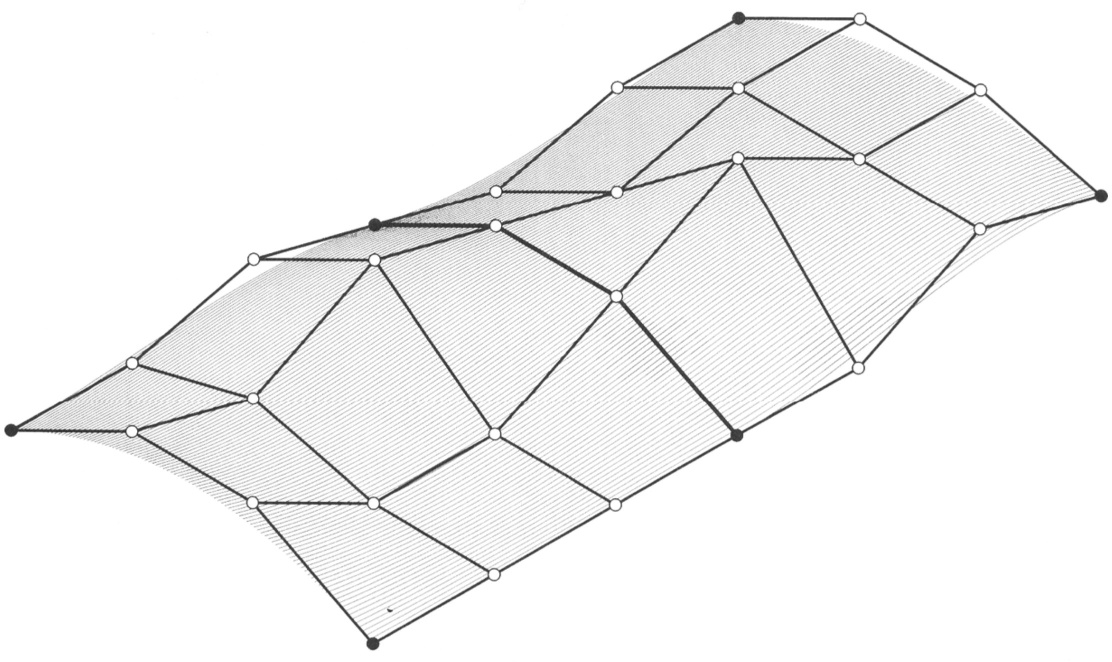
Composite Surfaces

- C^1 continuous Bezier patch
  - control points must be collinear
  - same ratio



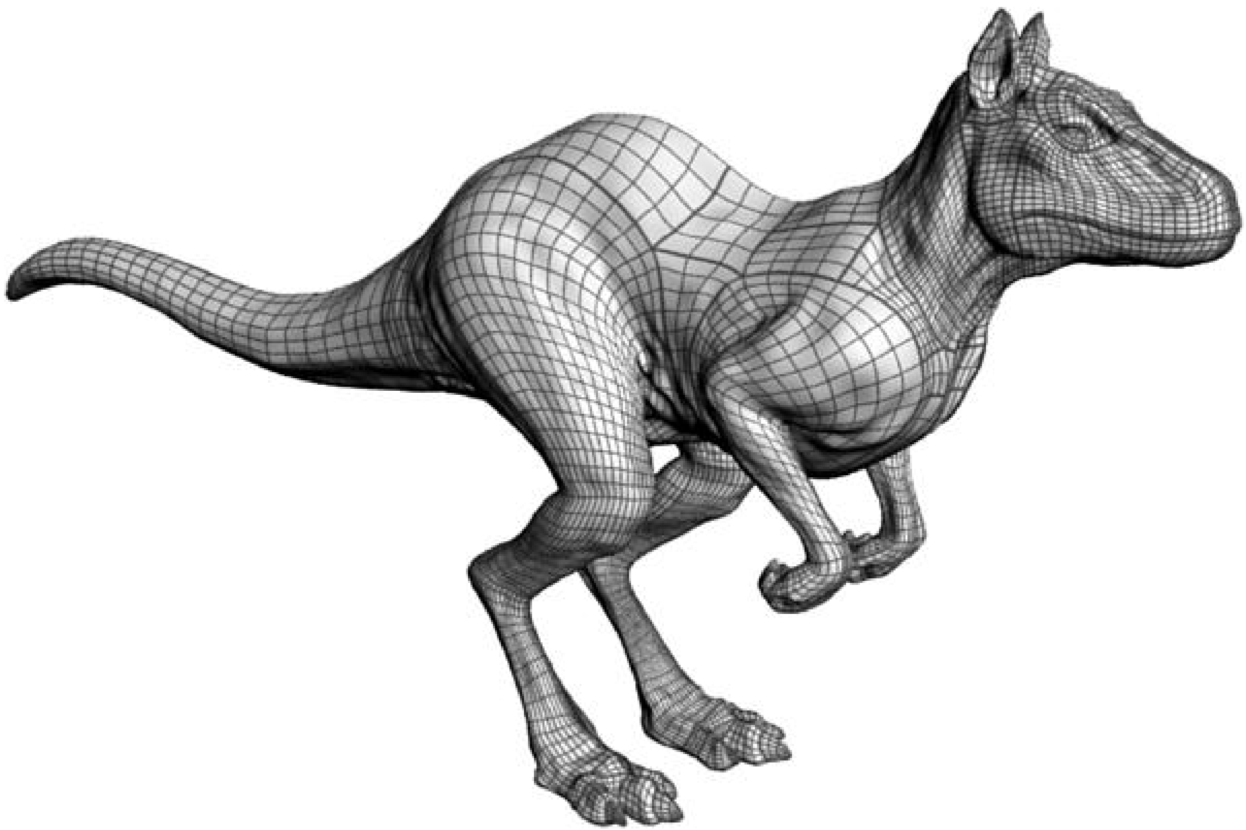
Composite Surfaces

- C^1 continuous Bezier patch
  - control points must be collinear
  - same ratio



NURBS Surfaces

- Standard in most advanced modeling systems



NURBS Surfaces

- Standard in most advanced modeling systems

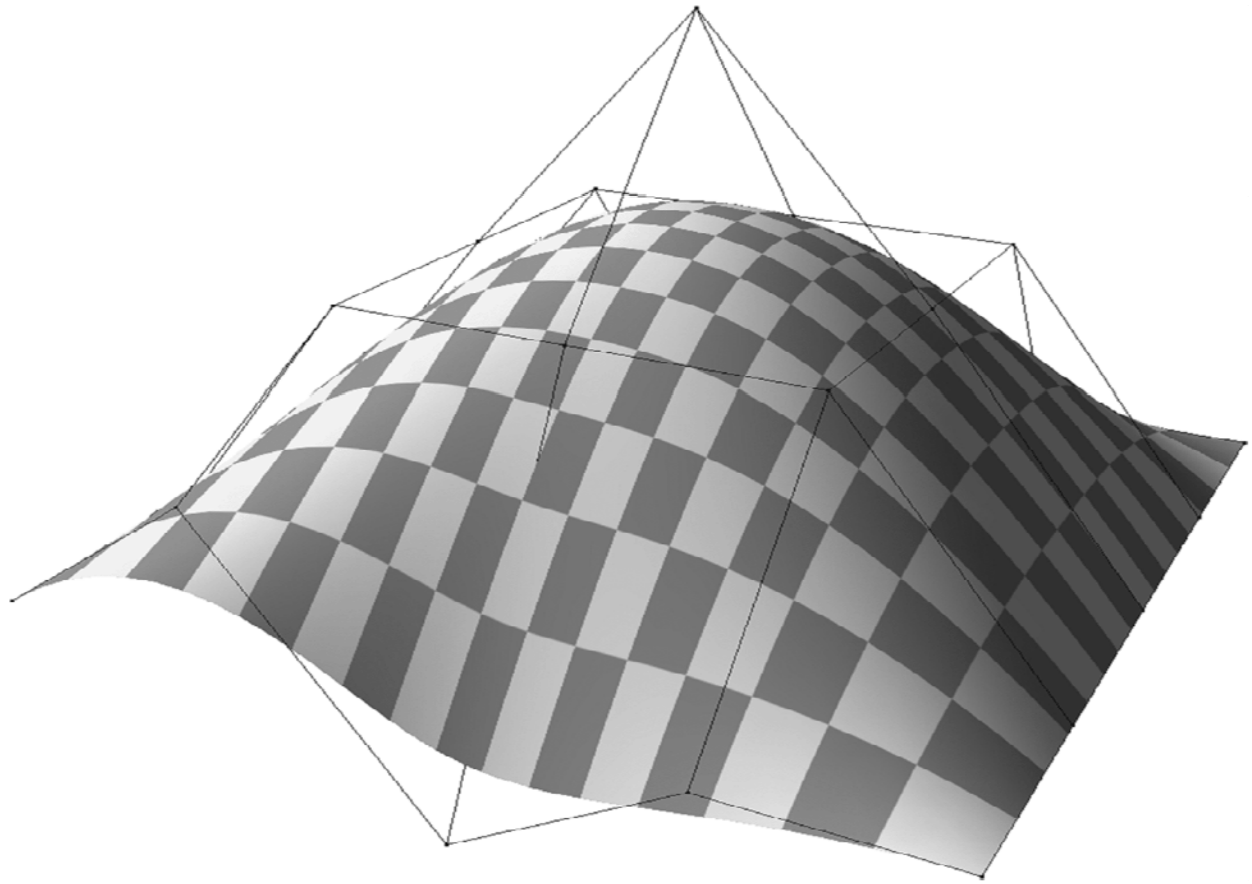
$$\mathbf{x}(u,v) = \frac{\sum_i \sum_j w_{i,j} \mathbf{d}_{i,j} N_i^m(u) N_j^n(v)}{\sum_i \sum_j w_{i,j} N_i^m(u) N_j^n(v)}$$

projection of tensor product patches ≠ tensor product surface! (basis is not separable)

NURBS Surfaces

- Influence of weights

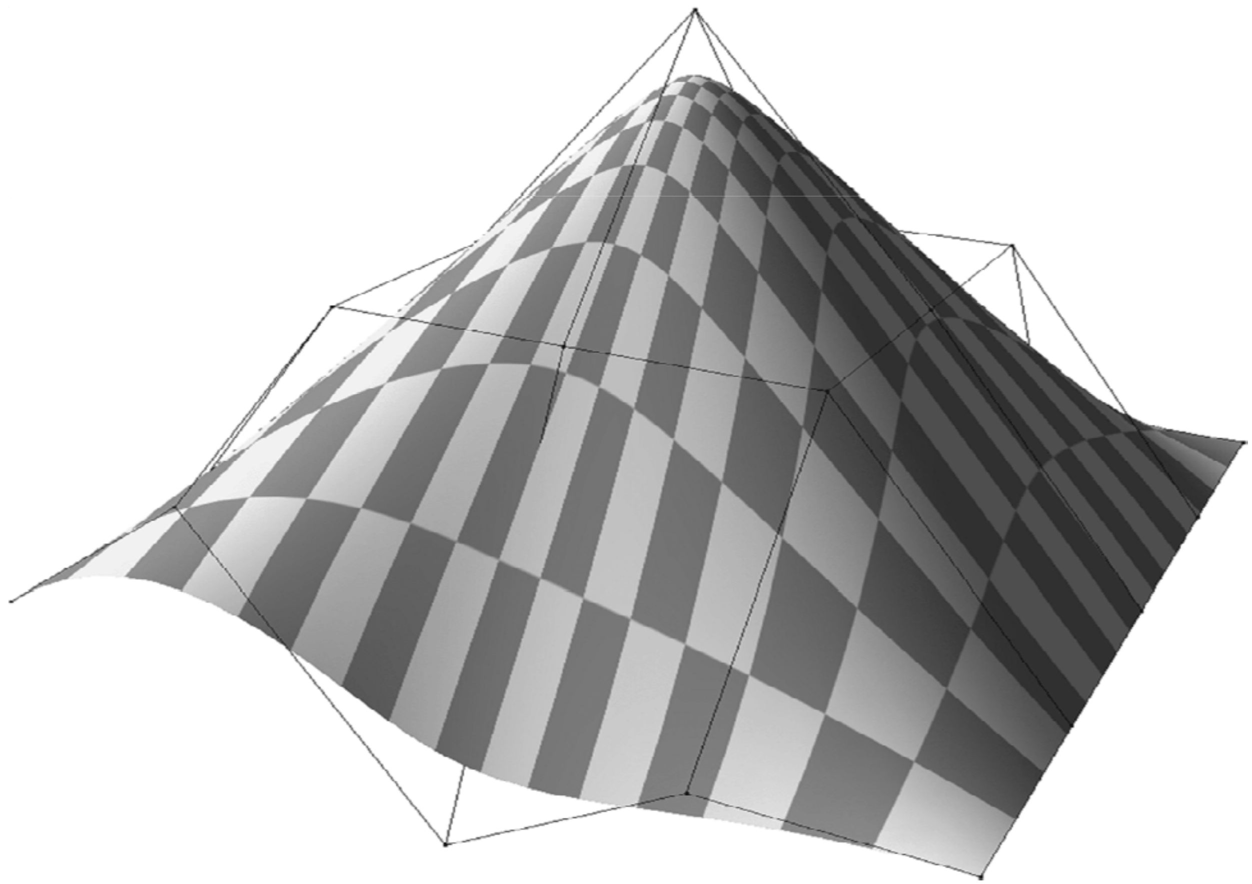
$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



NURBS Surfaces

- Influence of weights

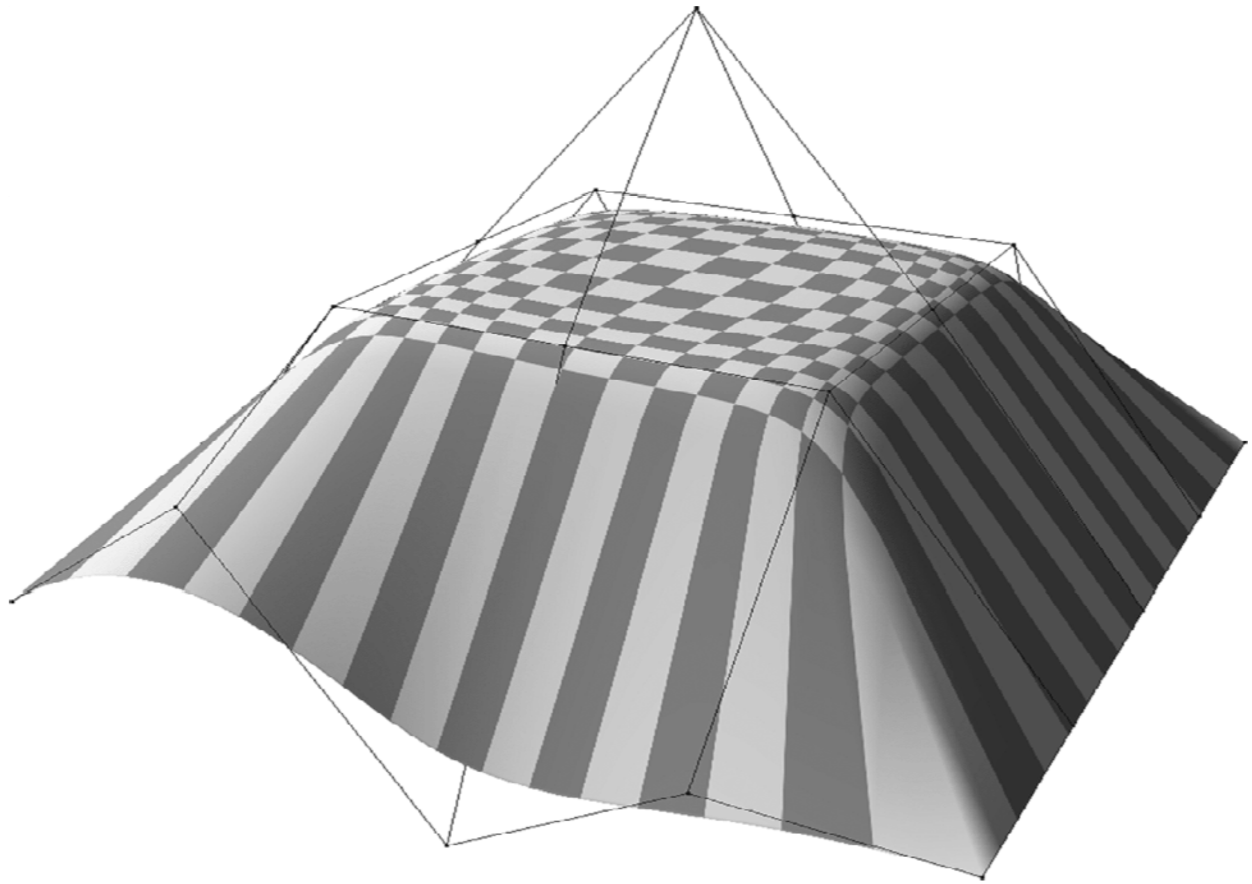
$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 10 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



NURBS Surfaces

- Influence of weights

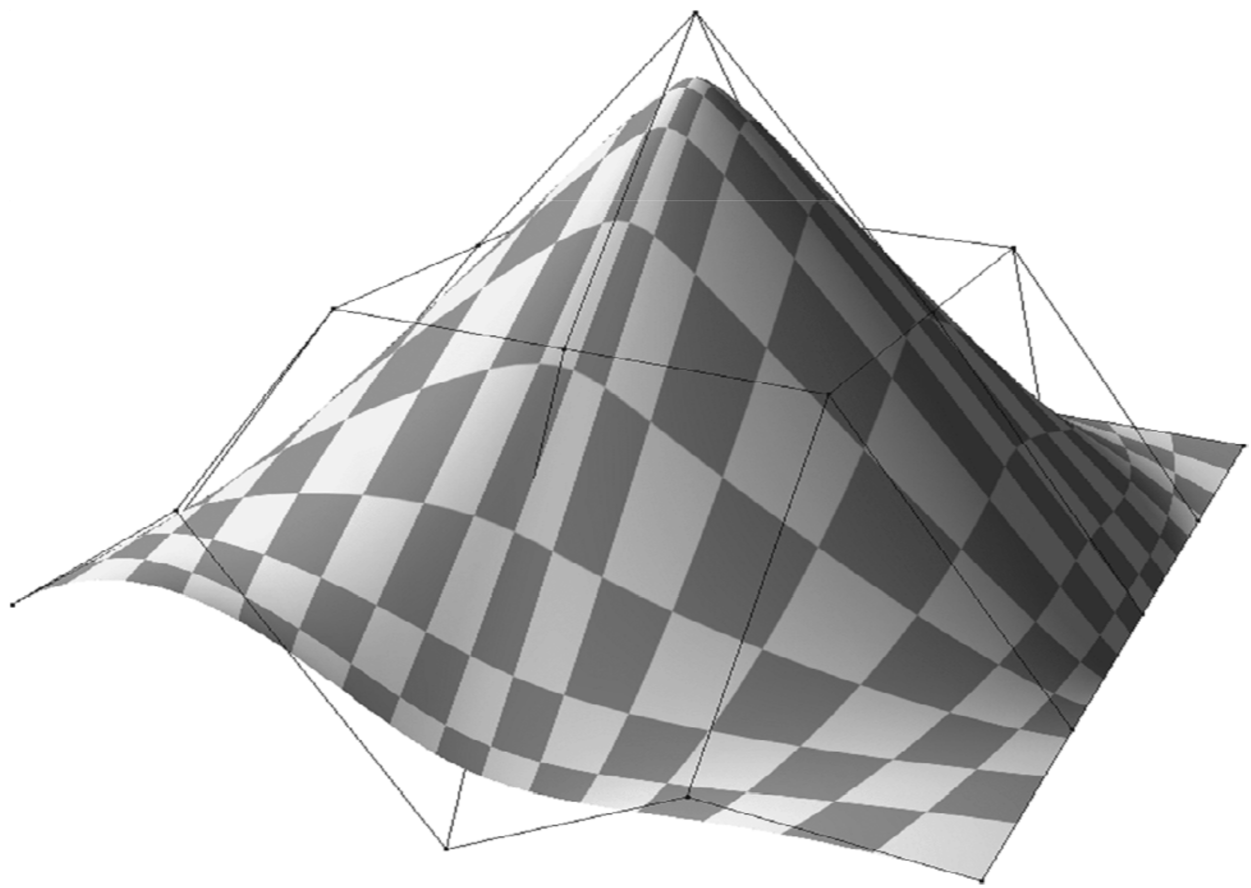
$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 30 & 30 & 1 \\ 1 & 30 & 1 & 30 & 1 \\ 1 & 30 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



NURBS Surfaces

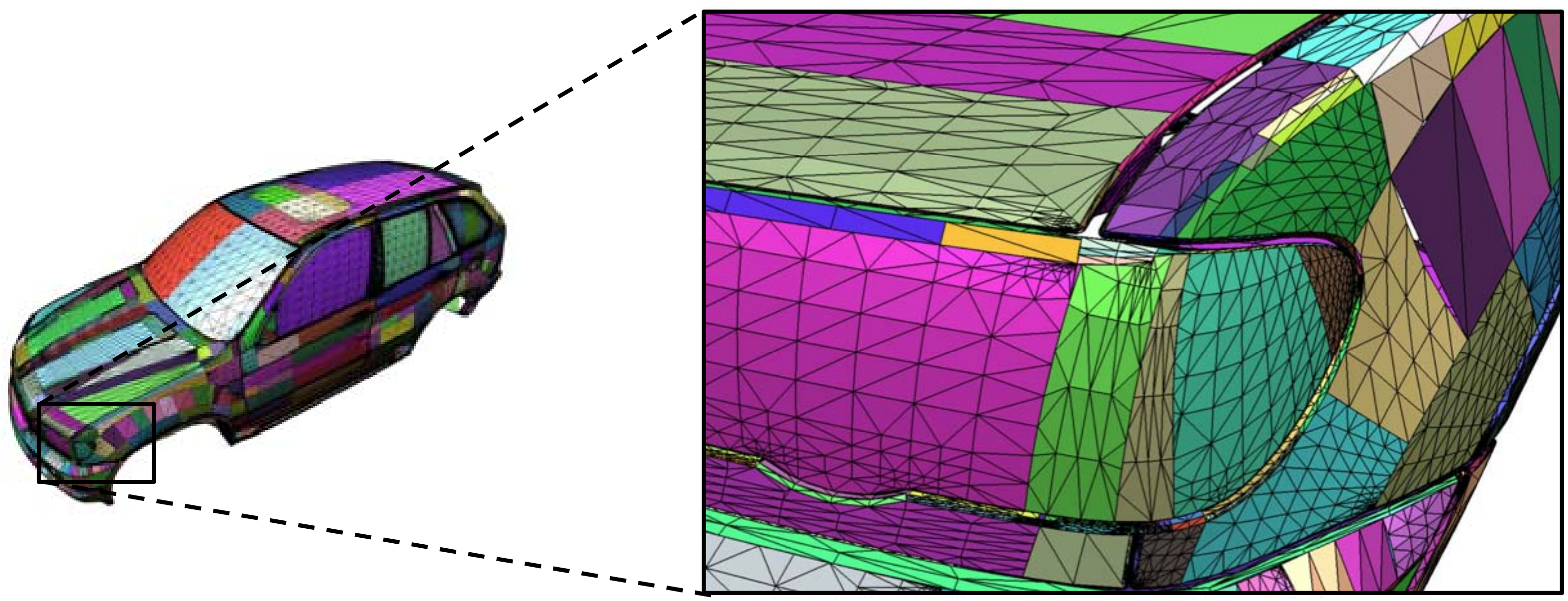
- Influence of weights

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0.1 & 0.1 & 0.1 & 1 \\ 1 & 0.1 & 1 & 0.1 & 1 \\ 1 & 0.1 & 0.1 & 0.1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



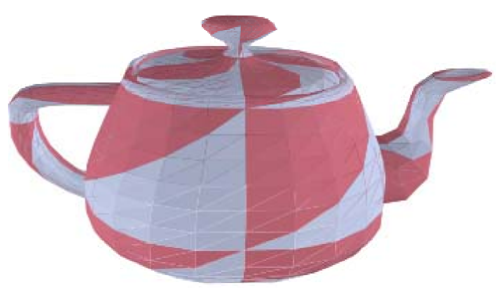
Raytracing Spline Surfaces

- Tessellate into triangles (deCasteljau, deBoor)
  - probably the fastest (with advanced data structures!)
  - large memory overhead



Raytracing Spline Surfaces

- Bezier clipping
  - reduces to root finding
  - exploits convex hull property for acceleration
  - many special cases, slow



control nets

Bezier surface

Roth et al., EG 2001

Geometric Modeling

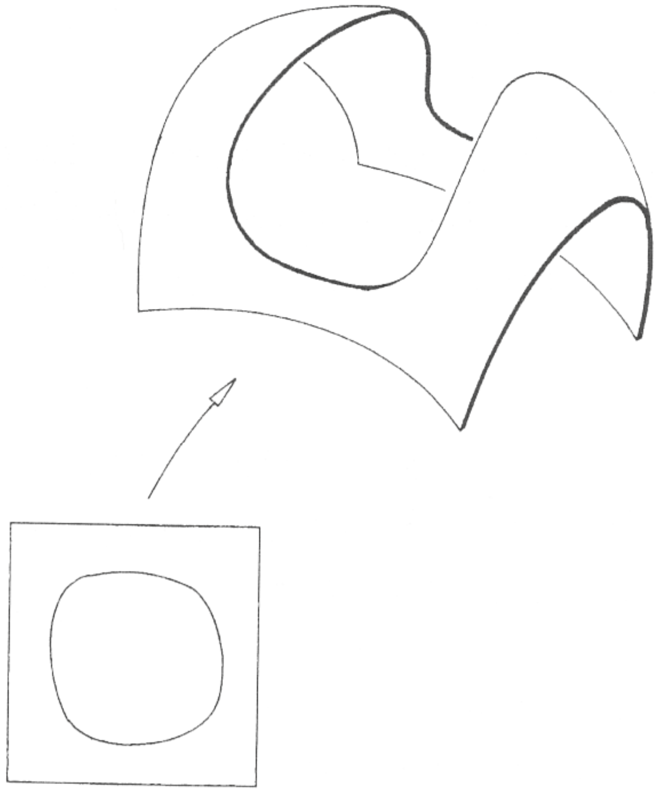
- Requirements on a surface representation
  - modeling flexibility
  - approximation power
  - ease of implementation
  - compact representation
  - efficient evaluation of surface and derivatives
  - fast spatial queries
    - ray-surface intersections, collision detection, inside-outside tests, etc.

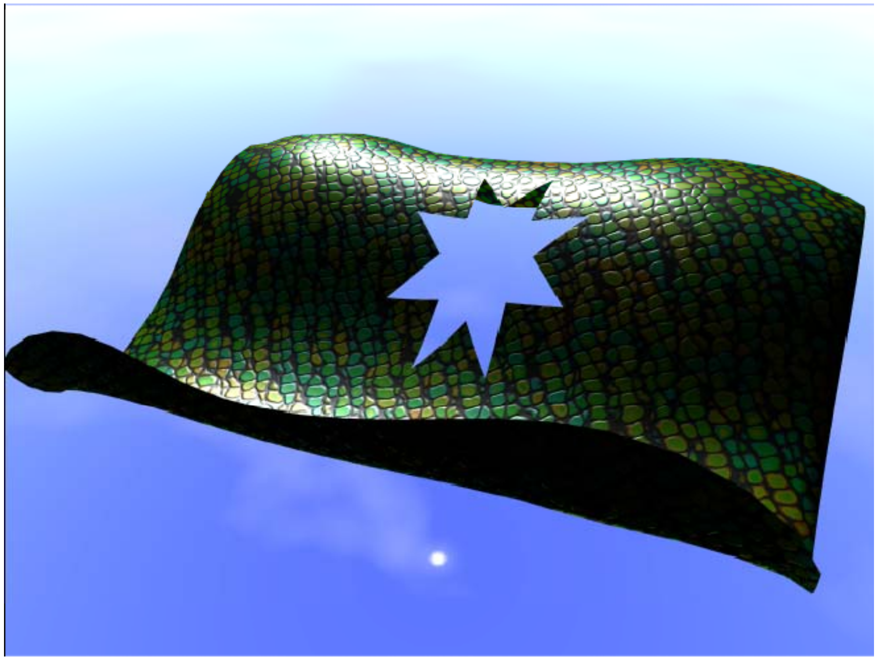
Bezier- and B-Spline Surfaces

- Simple functions: Polynomials
  – efficient evaluation (no exp, sin, sqrt, etc.)
  – simple derivatives
- Intuitive editing
  – surface deforms naturally when moving control points
- Boundary constraints
  – difficult to model (and modify!) complex geometries

Additional Topics

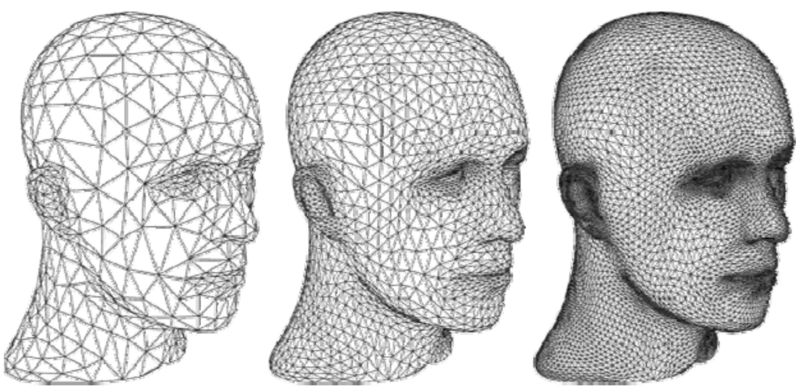
- Trimming

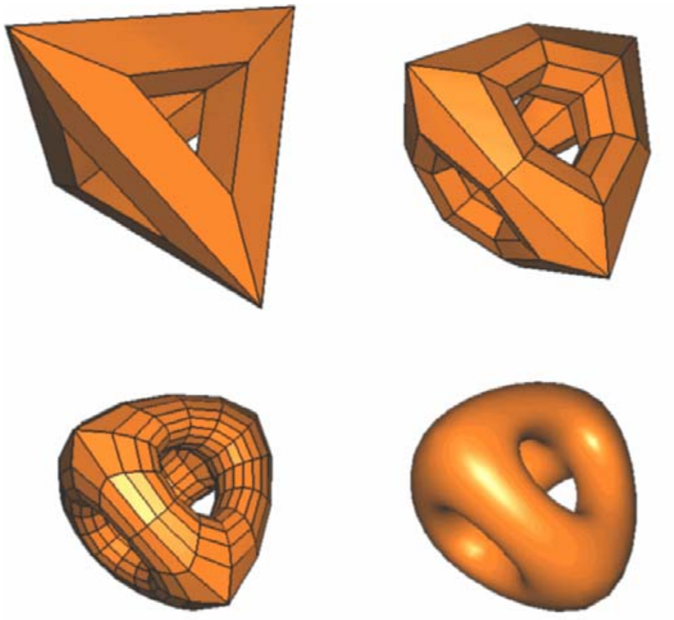




Additional Topics

- Subdivision surfaces
  - smooth surface as the limit of a sequence of successive refinements





Thank you!