CSE528 Computer Graphics: Theory, Algorithms, and Applications

Hong Qin

Department of Computer Science

State University of New York at Stony Brook (Stony Brook University)

Stony Brook, New York 11794--4400

Tel: (631)632-8450; Fax: (631)632-8334

qin@cs.sunysb.edu

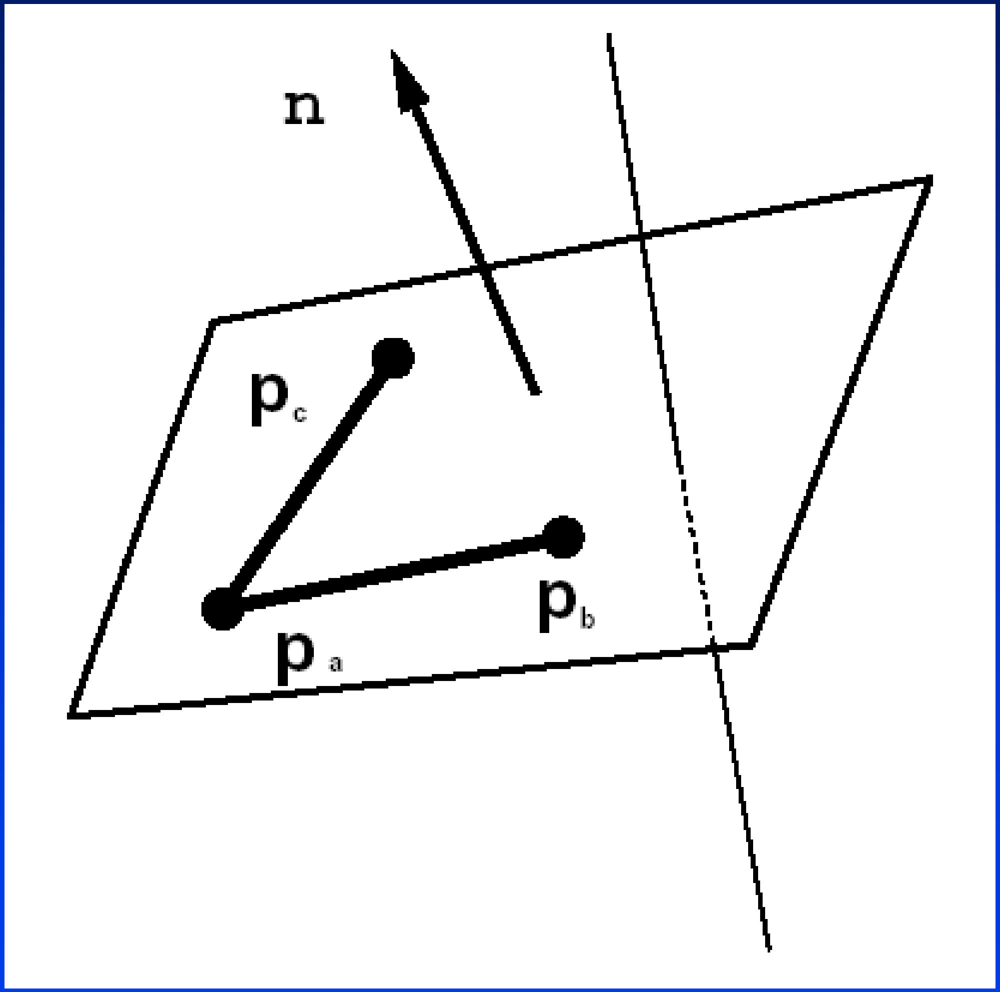
http://www.cs.sunysb.edu/~qin

Parametric Surfaces

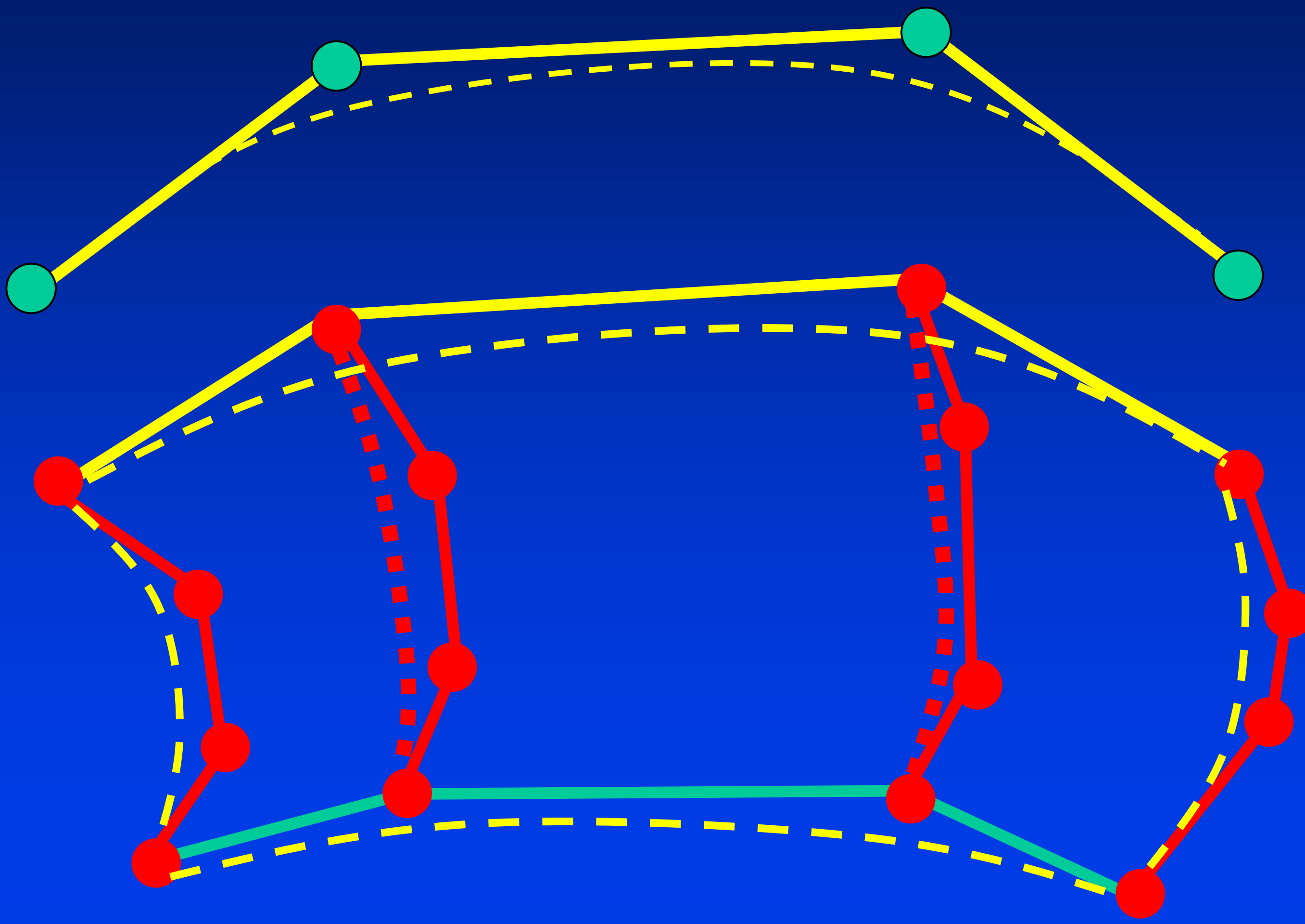
Geometric Modeling Motivation

- Why geometric modeling
- Fundamental for visual computing
  - Graphics, visualization
  - Computer aided design and manufacturing
  - Imaging
  - Entertainment, etc.
- Critical for virtual engineering
- Interaction
- Geometric information for decision making

Plane and Intersection



From Curve to Surface



Parametric Representations

- Hermit curves and surfaces (S.A.Coons[63] and J.C.Ferguson[64])
- Bézier curves and surfaces (P.Bézier[66] and P.de Casteljau[59])
- B-Splines (W.J.Gordon and R.F.Riesenfeld 70s)
- NURBS (Versprille 75)
- Mathematical foundations (M.G.Cox[72], C.de Boor[72], et al)

Parametric Representation

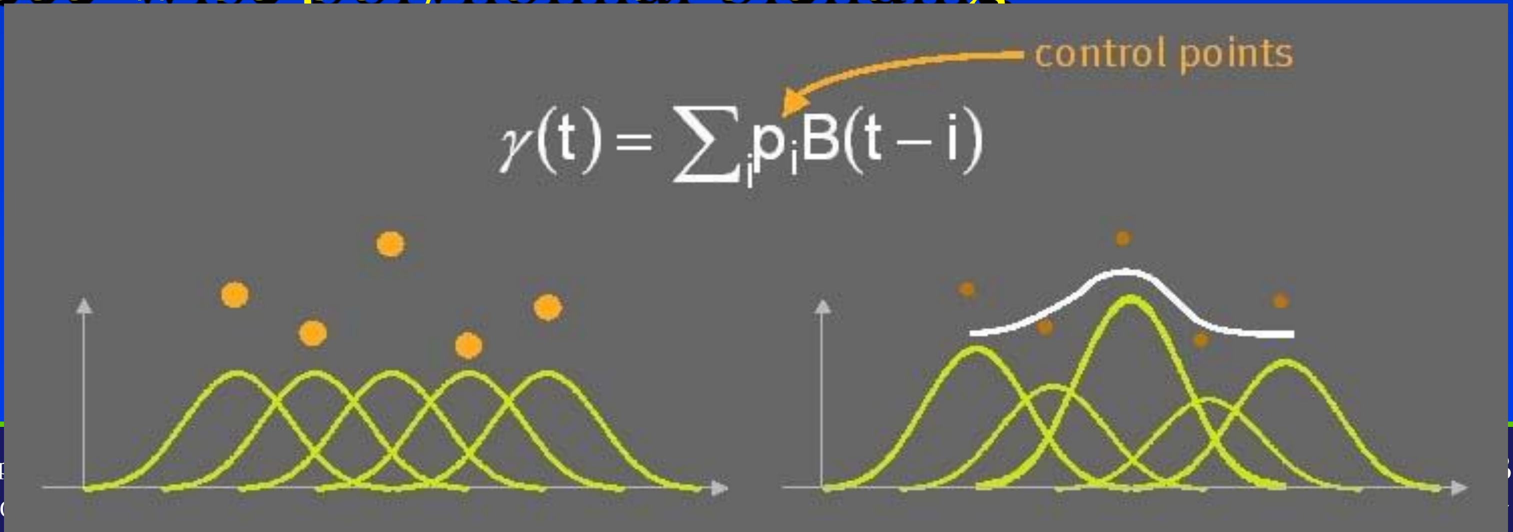
- Parametric curve functions

$$x = x(u), y = y(u), z = z(u)$$

- Parametric surface functions

$$x = x(u,v), y = y(u,v), z = z(u,v)$$

- Piece-wise polynomial blending



Surfaces

- From curves to surfaces
- A simple curve example (Bezier)

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i(u)$$
$$u \in [0,1]$$

- Consider each control point now becoming a Bezier curve

$$\mathbf{p}_i = \sum_{j=0}^{3} \mathbf{p}_{i,j} B_j(v)$$
$$v \in [0,1]$$

Surfaces

- Then, we have

$$\mathbf{s}(u,v)=\sum_{i=0}^{3}\left(\sum_{j=0}^{3}\mathbf{p}_{i,j}B_j(v)\right)B(u)=\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{p}_{i,j}B_i(u)B_j(v)$$

- Matrix form

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

$$=UMPM^TV^T$$

Surfaces

- Further generalize to degree of n and m along two parametric directions

$$\mathbf{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{p}_{i,j}B_i^n(u)B_j^m(v)$$

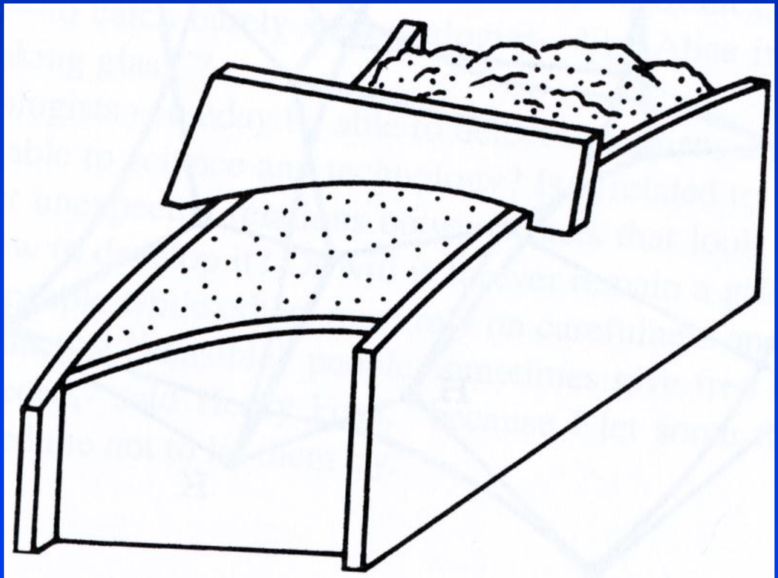
- Question: which control points are interpolated?
- How about B-spline surfaces???

Tensor-Product: Basic Concepts

- Direct generalization from two vectors:

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1b_1 & a_2b_1 & a_3b_1 \\ a_1b_2 & a_2b_2 & a_3b_2 \\ a_1b_3 & a_2b_3 & a_3b_3 \\ a_1b_4 & a_2b_4 & a_3b_4 \end{bmatrix}$$

- Similarly, we can define a surface as the tensor product of two curves....



Tensor Product Surfaces

- Where are they from?
- Monomial form

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{a}_{i,j} u^i v^j$$

- Bezier surface

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{p}_{i,j} B_i^m(u) B_j^n(v)$$

- B-spline surface

$$\mathbf{s}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j} B_{i,k}(u) B_{j,l}(v)$$

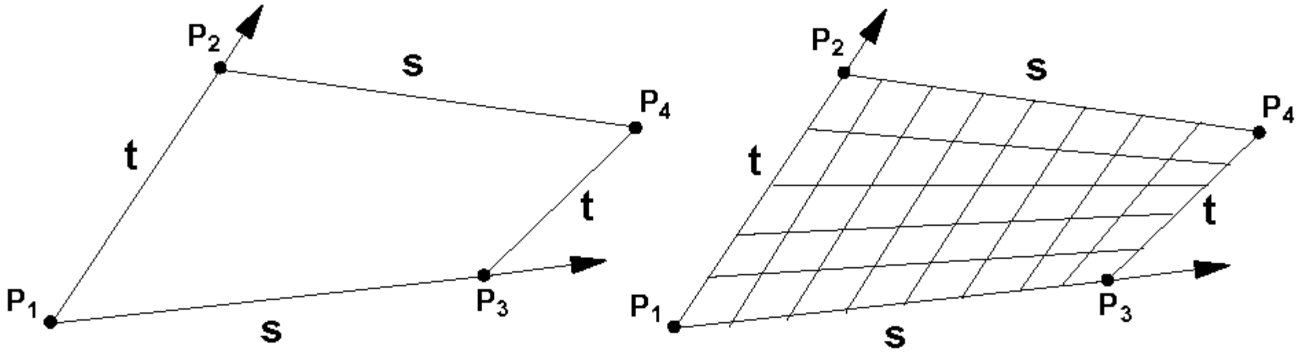
- General case

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{v}_{i,j} F_i(u) G_j(v)$$

Bilinear Patch

- Perhaps the easiest example is bilinear interpolation

Bi-lerp a (typically non-planar) quadrilateral

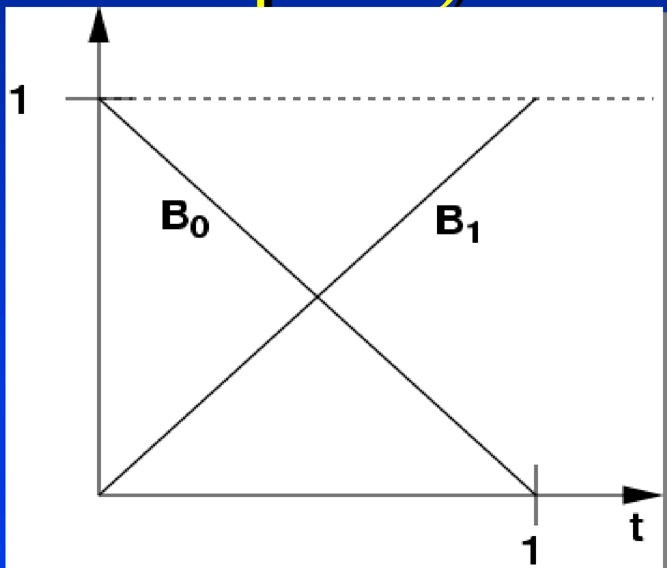


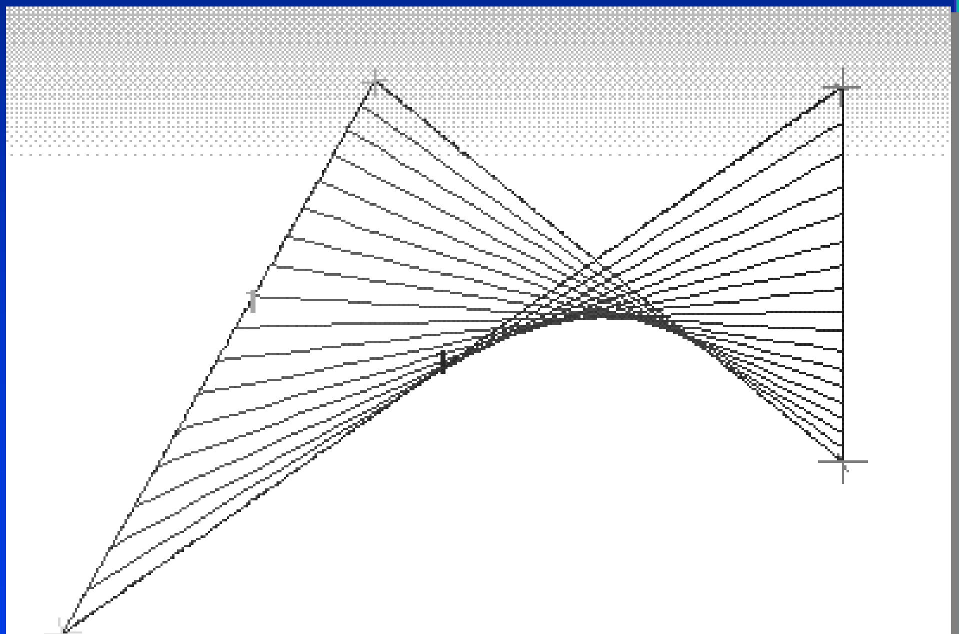
Notation: $\mathbf{L}(P_1, P_2, \alpha) \equiv (1-\alpha)P_1 + \alpha P_2$

$$Q(s,t) = \mathbf{L}(\mathbf{L}(P_1, P_2, t), L(P_3, P_4, t), s)$$

Bilinear Patch

- Smooth version of quadrilateral with non-planar vertices... (four points are NOT on the same plane)

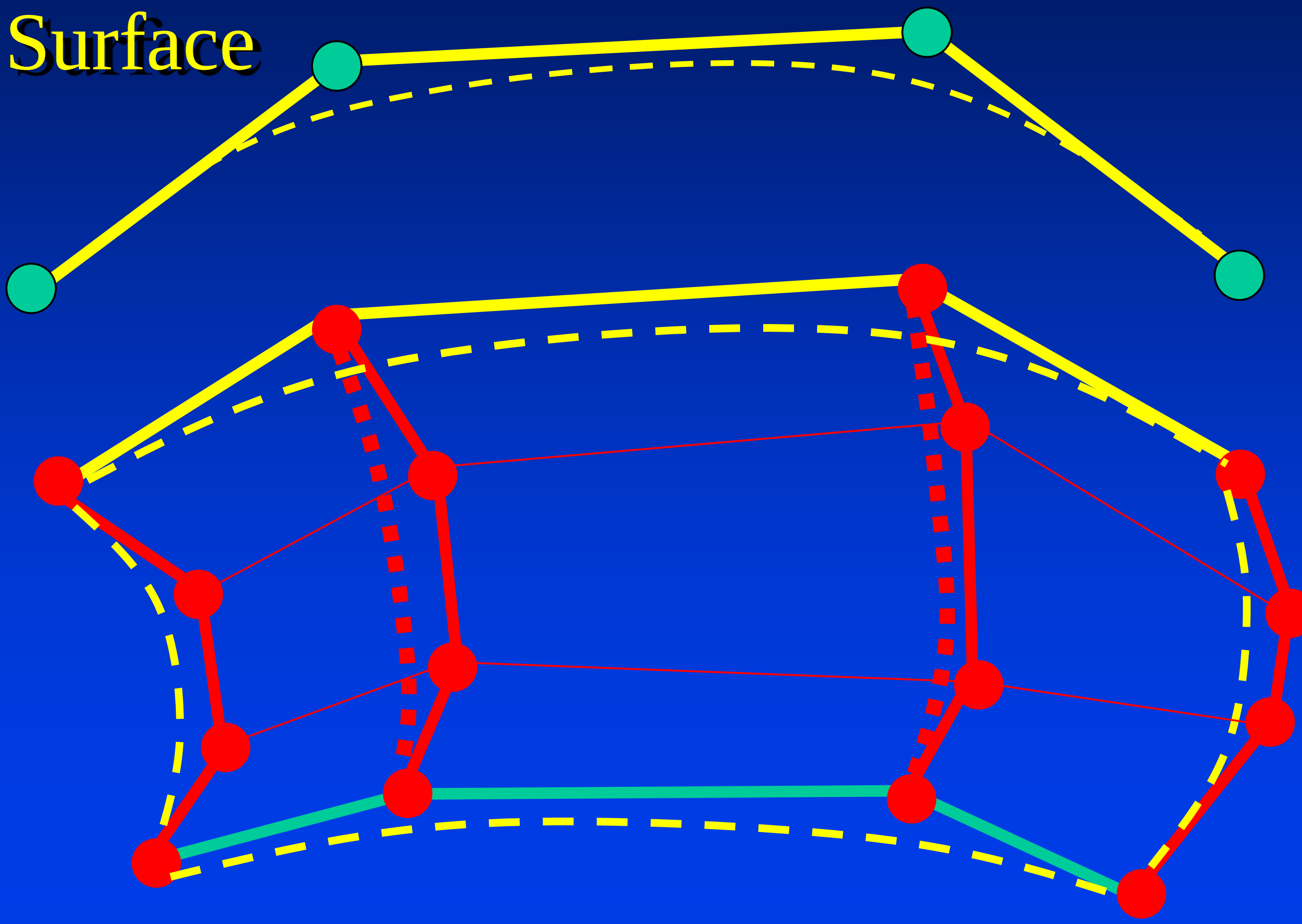




  - But will this help us model smooth surfaces?
  - Do we have control of the derivative at the edges?

Tensor Product Surface

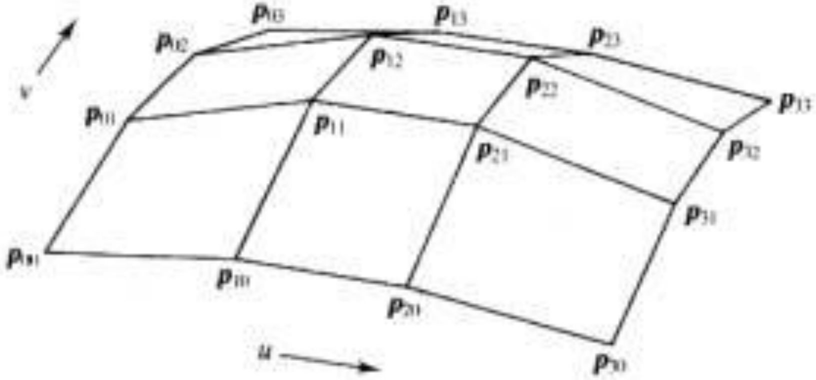
- Bezier Surface



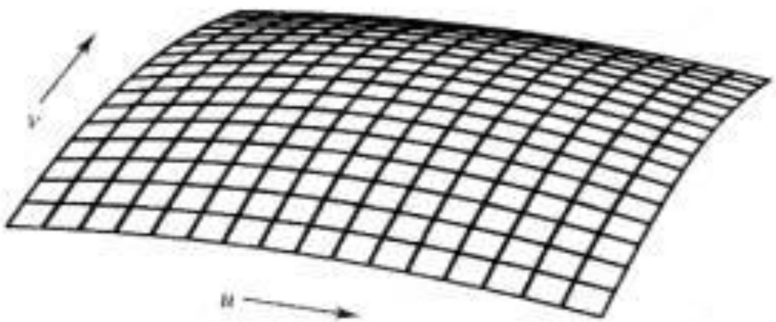
Bicubic Bezier Patch

- How do we define a tensor-product bicubic Bezier surface?

$$
\begin{aligned}
Q(s,t) = \mathbf{CB}(\quad & \mathbf{CB}(P_{00}, P_{01}, P_{02}, P_{03}, t), \\
& \mathbf{CB}(P_{10}, P_{11}, P_{12}, P_{13}, t), \\
& \mathbf{CB}(P_{20}, P_{21}, P_{22}, P_{23}, t), \\
& \mathbf{CB}(P_{30}, P_{31}, P_{32}, P_{33}, t), \\
& s)
\end{aligned}
$$

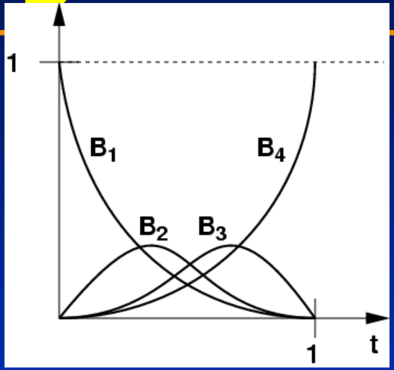


(a)

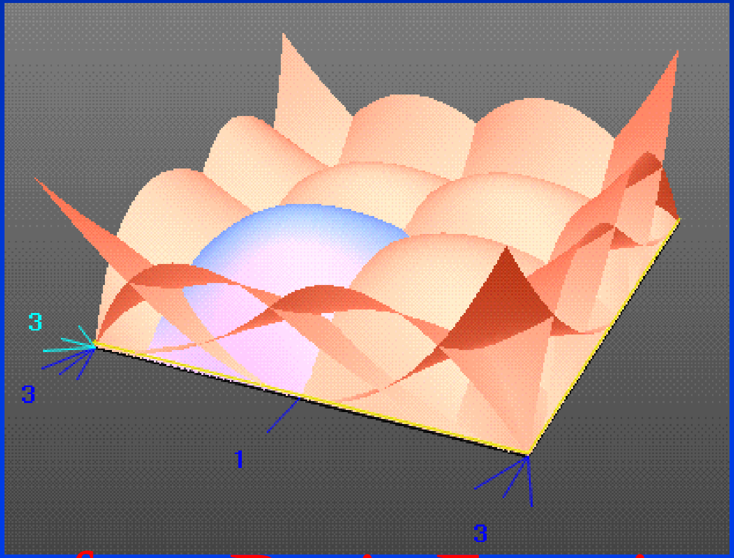


(b)

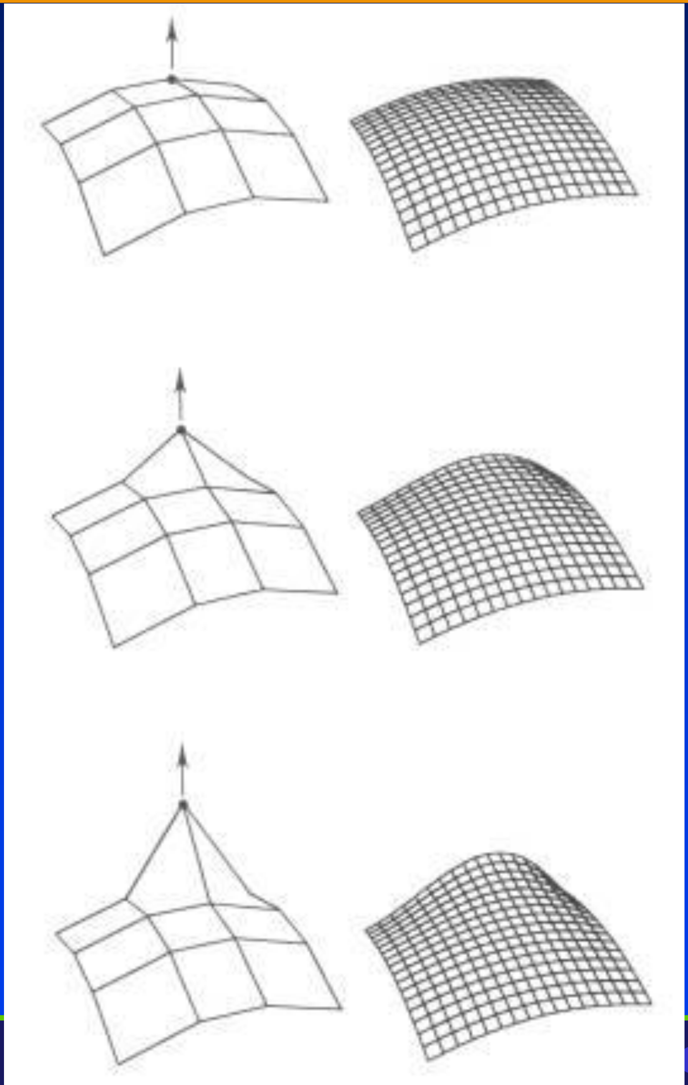
Editing Bicubic Bezier Patches



Curve Basis Functions



Surface Basis Functions



Bezier Surface Patch Corner Boundary Conditions

Four equations for each corner gives 16 total.

$$\mathbf{p}(0,0) = \mathbf{p}_{00}$$

Patch must interpolate the corner.

$$\frac{\partial \mathbf{p}}{\partial u}(0,0) = 3(\mathbf{p}_{10} - \mathbf{p}_{00})$$

Defines derivative in u direction.

$$\frac{\partial \mathbf{p}}{\partial v}(0,0) = 3(\mathbf{p}_{01} - \mathbf{p}_{00})$$

Defines derivative in v direction.

$$\frac{\partial^2 \mathbf{p}}{\partial u \partial v}(0,0) = 9(\mathbf{p}_{00} - \mathbf{p}_{01} + \mathbf{p}_{10} - \mathbf{p}_{11})$$

“Twist”

B-Splines

- B-spline curves $$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(u)$$
- Tensor product B-splines $$\mathbf{s}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j} B_{i,k}(u) B_{j,l}(v)$$
- Question again: which control points are interpolated???
- Another question: can we get NURBS surface this way???
- Answer: NO!!! NURBS are not tensor-product surfaces
- Another question: can we have NURBS surface?
- YES!!!

NURBS Curves

$$c(u) = \frac{\sum_{i=1}^{n} p_i w_i B_{i,k}(u)}{\sum_{i=1}^{n} w_i B_{i,k}(u)}$$

$$\begin{bmatrix} c_x / c_w \\ c_y / c_w \\ c_z / c_w \end{bmatrix} \Leftarrow \begin{bmatrix} c_x(u) \\ c_y(u) \\ c_z(u) \\ c_w(u) \end{bmatrix} = \sum_{i=1}^{n} B_{i,k}(u) \begin{bmatrix} w_i x_i \\ w_i y_i \\ w_i z_i \\ w_i \end{bmatrix}$$

NURBS Surface

- NURBS surface mathematics

$$\mathbf{s}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{p}_{i,j}w_{i,j}B_{i,k}(u)B_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{i,j}B_{i,k}(u)B_{j,l}(v)}$$

- Understand this geometric construction
- Question: why is it not the tensor-product formulation??? Compare it with Bezier and B-spline construction

NURBS Surfaces

$$s(u) = \frac{\sum_{i,j=1}^{n} p_{ij} w_{ij} B_{i,k}(u) B_{j,l}(v)}{\sum_{i,j=1}^{n} w_{ij} B_{i,k}(u) B_{j,l}(v)}$$

$$\begin{bmatrix} s_x / s_w \\ s_y / s_w \\ s_z / s_w \end{bmatrix} \Leftarrow \begin{bmatrix} s_x(u) \\ s_y(u) \\ s_z(u) \\ s_w(u) \end{bmatrix} = \sum_{i,j=1}^{n} B_{i,k}(u) B_{j,l}(v) \begin{bmatrix} w_{ij} x_{ij} \\ w_{ij} y_{ij} \\ w_{ij} z_{ij} \\ w_{ij} \end{bmatrix}$$

NURBS Surface

- Parametric variables: u and v
- Control points and their associated weights: (m+1)(n+1)
- Degrees of basis functions: (k-1) and (l-1)
- Knot sequence:

$$u_0 <= u_1 <= \ldots\ldots <= u_{m+k}$$
$$v_0 <= v_1 <= \ldots\ldots <= v_{n+l}$$

- Parametric domain:

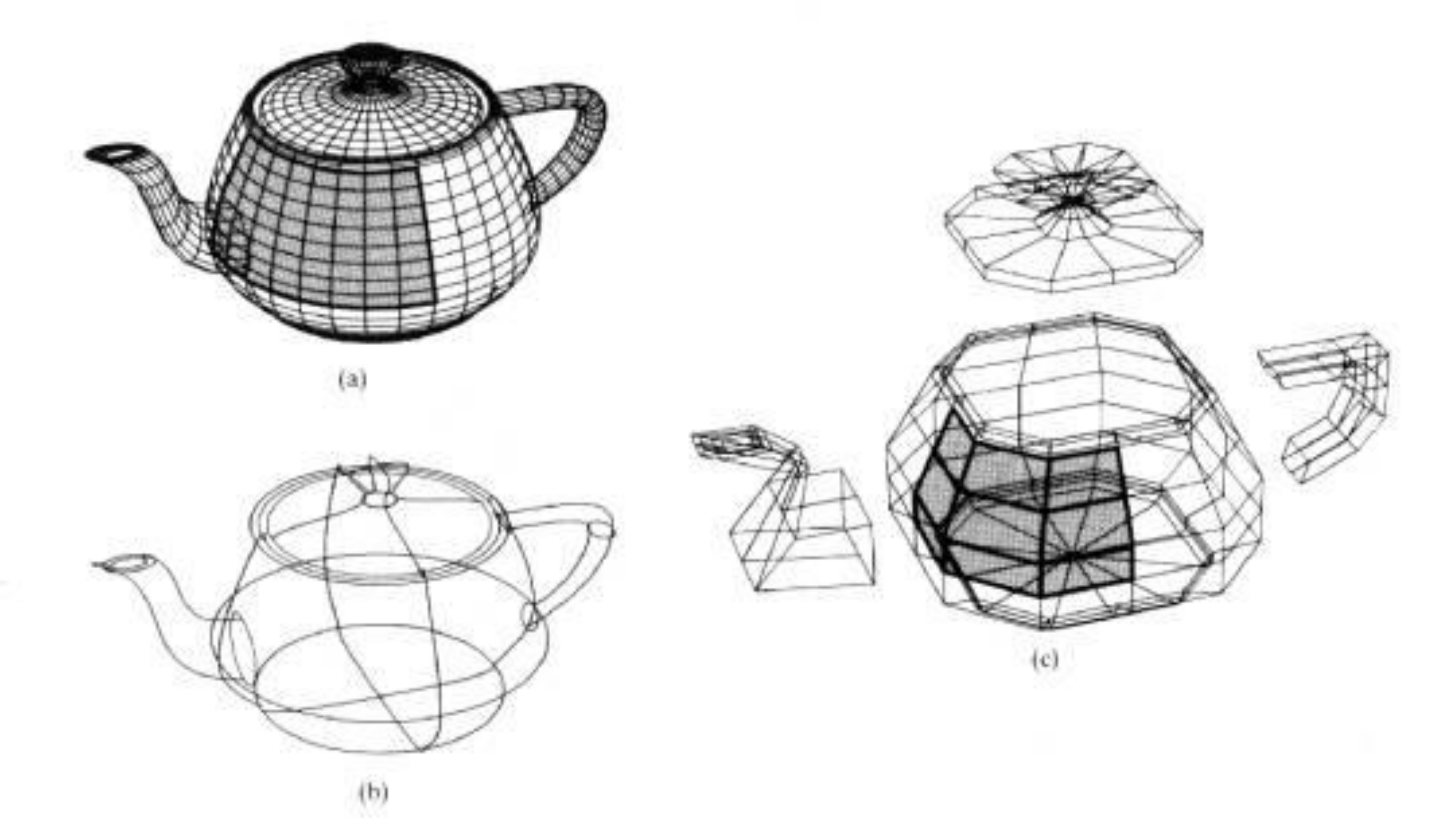
$$u_{k-1} <= u <= u_{m+1}$$
$$v_{l-1} <= v <= v_{n+1}$$

NURBS Surface

- The same principle to generate curves via projection
- Idea: associate weights with control points
- Generalization of B-spline surface

Modeling with Bicubic Bezier Patches

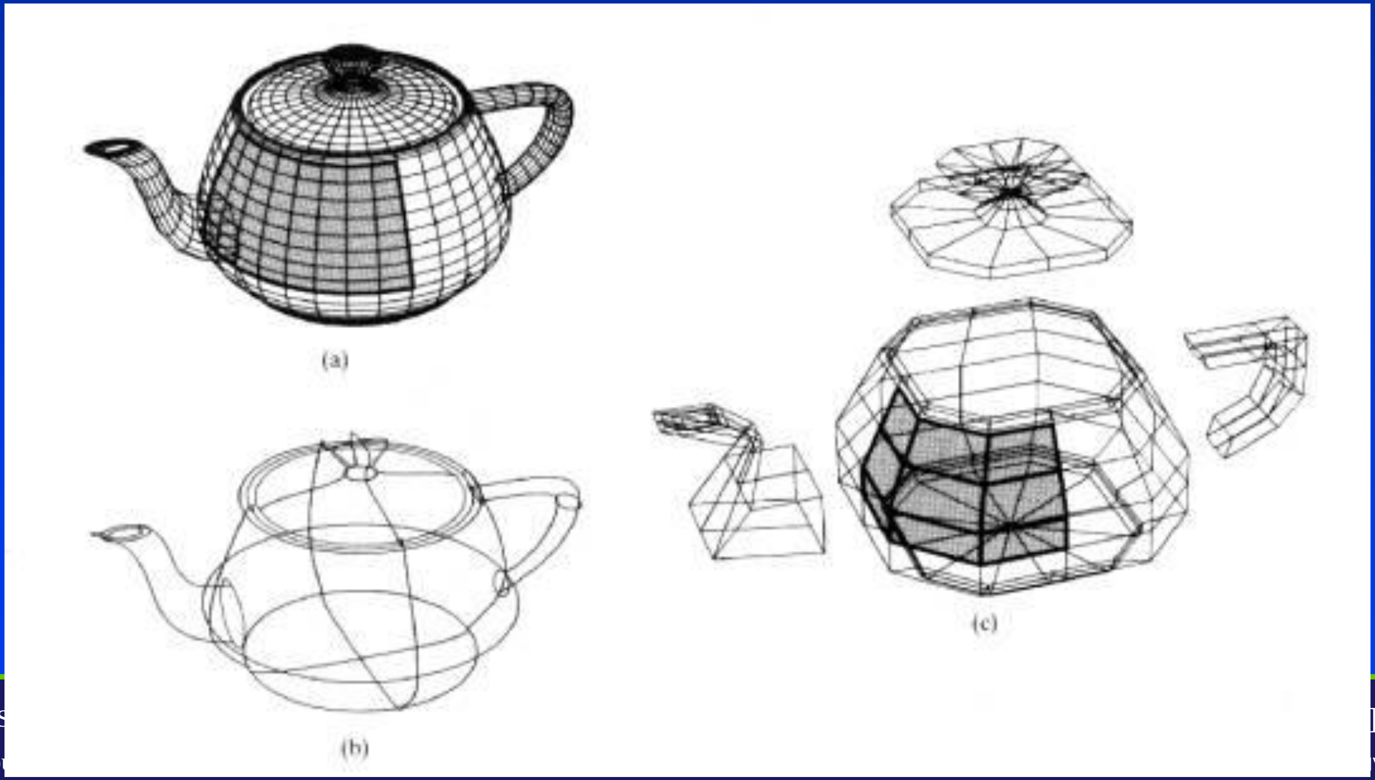
- Original Teapot specified with Bezier Patches



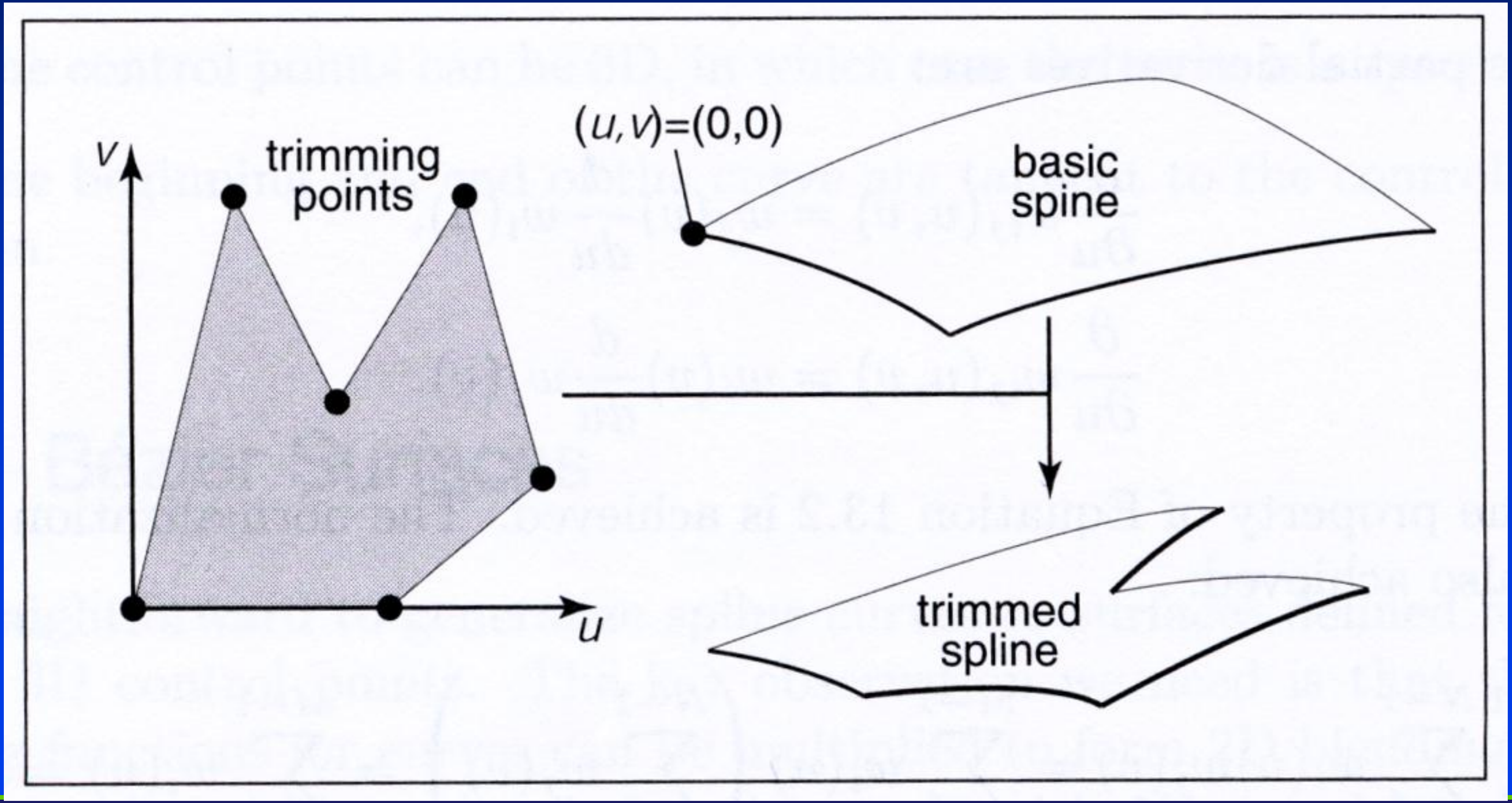
Modeling Difficulties

- Original Teapot model is not "watertight":

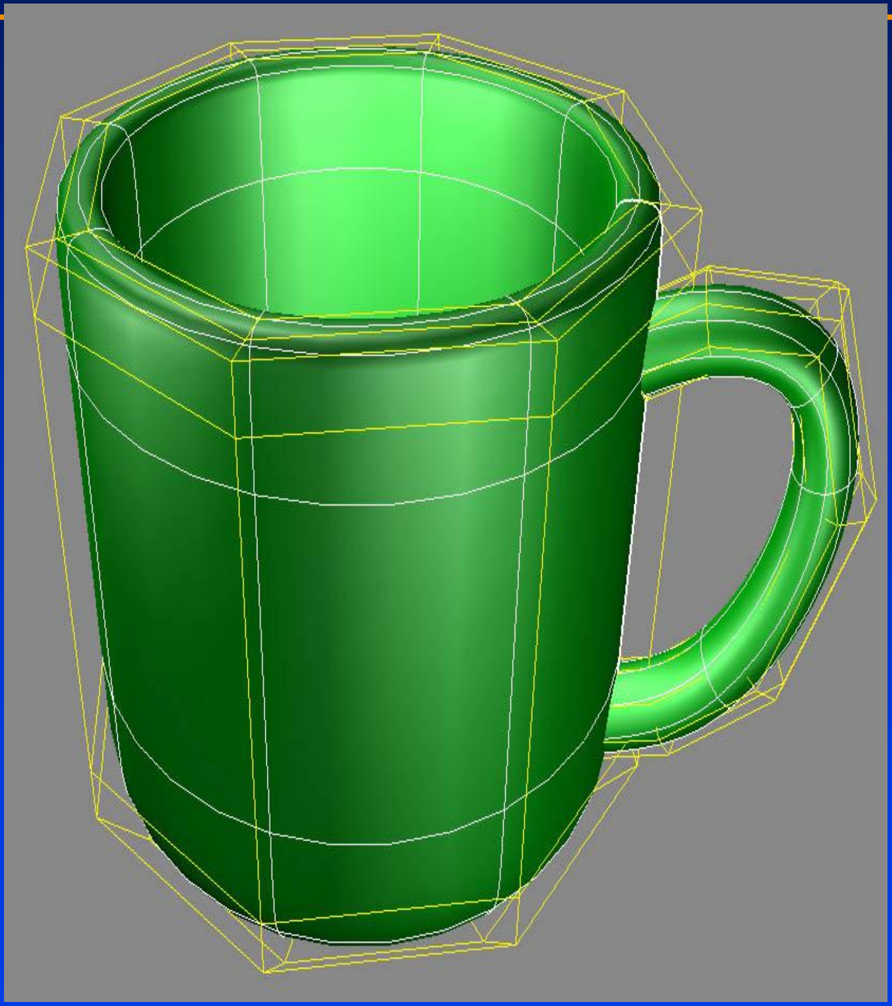
  intersecting surfaces at spout & handle, no bottom, a hole at the spout tip, a gap between lid & base

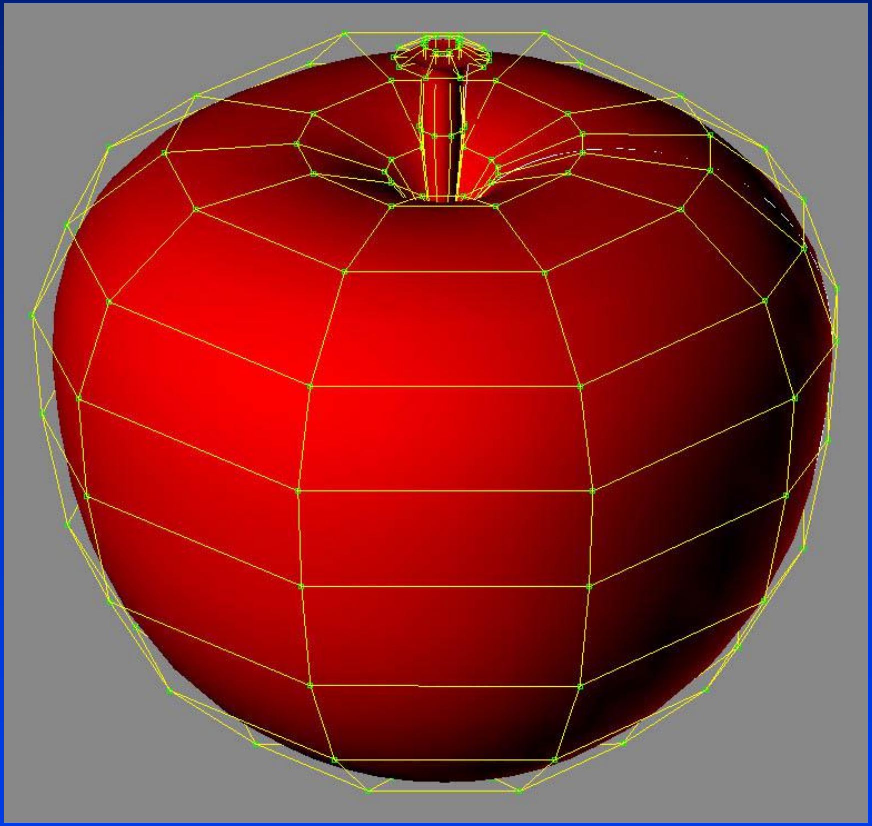


Trimming Curves for Patches



NURBS Surface Examples





NURBS Surfaces

- Good for
  - Mechanical, manufactured parts
  - Smooth free-form surface representation
- Bad for
  - Non-genus-0 surfaces
  - Interactive design of free-form surfaces

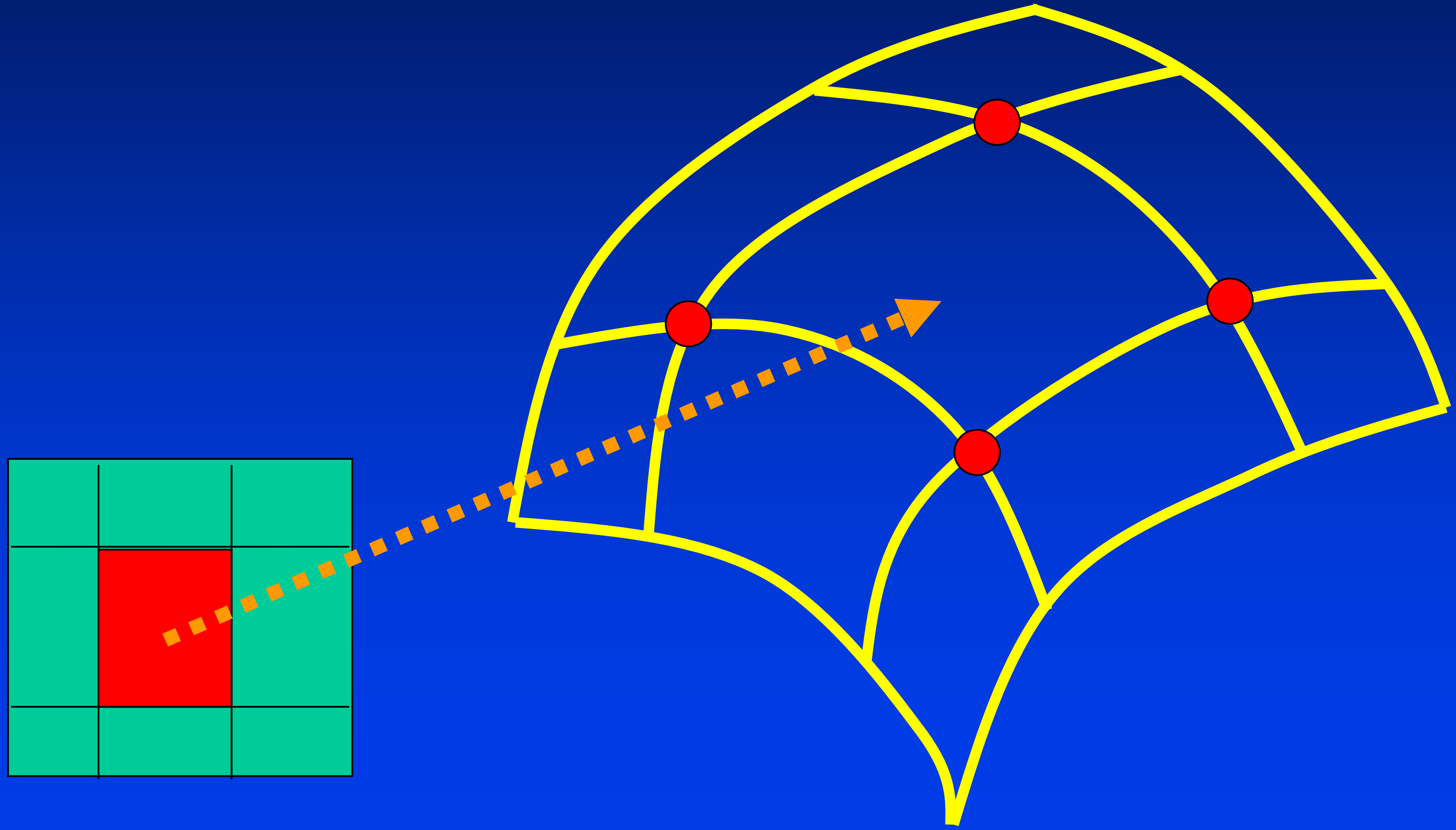
Why NURBS

- Support free-form curves/surfaces modeling.
- Support standard analytic shapes precisely.
- Local support.
- Strong convex hull property.
- Affine transformation invariant
- Strict analytic form for evaluation (important in CAD/CAM/CAE)

Why NOT NURBS

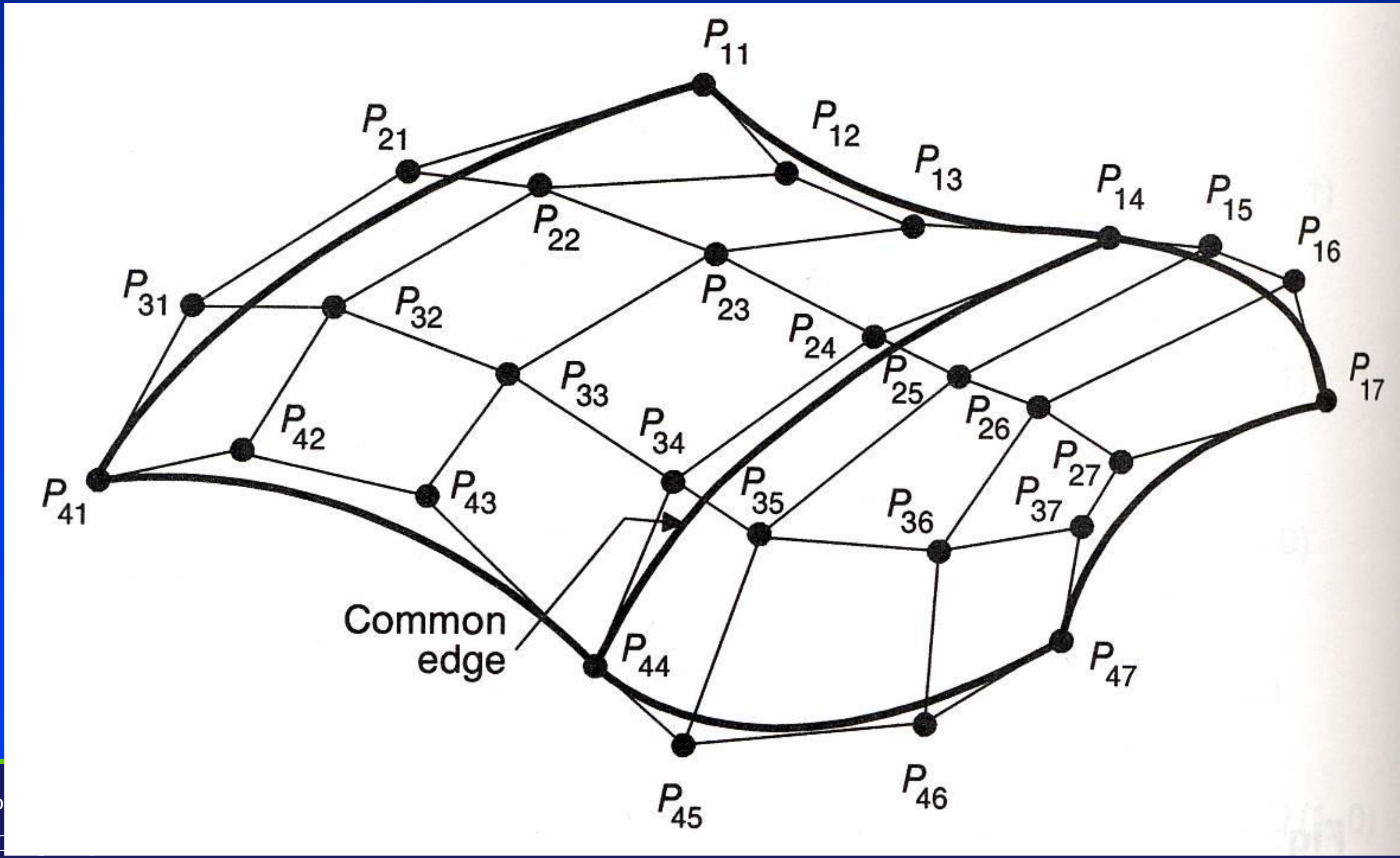
- Hard to model arbitrary topology.
- Regularity of tensor-product control polygon poses difficulty for level of detail.
- Numerical instable for geometric operations such as surface intersection.
- Weights and knots are less intuitive for shape control.

Rectangular Surface



Adjacent Bézier Patches

- Continuity conditions across the common, shared boundary



Hermite Surfaces

- How about Hermite surfaces???
- Hermite Curve

$$\mathbf{c}(u) = \begin{bmatrix} H_0(u) & H_1(u) & H_2(u) & H_3(u) \end{bmatrix} \begin{bmatrix} \mathbf{c}(0) \\ \mathbf{c}(1) \\ \mathbf{c}'(0) \\ \mathbf{c}'(1) \end{bmatrix}$$

- C(0) is not a curve s(0,v) which is also a Hermite Curve:

$$s(0,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix} \begin{bmatrix} \mathbf{s}(0,0) \\ \mathbf{s}(0,1) \\ \mathbf{s}_v(0,0) \\ \mathbf{s}_v(0,1) \end{bmatrix}$$

Hermite Surfaces

- Similarly, c(1) is now a curve s(1,v) which is also a Hermite curve:

$$\mathbf{s}(1,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix} \begin{bmatrix} \mathbf{s}(1,0) \\ \mathbf{s}(1,1) \\ \mathbf{s}_v(1,0) \\ \mathbf{s}_v(1,1) \end{bmatrix}$$

- The same are for c'(0) and c'(1):

$$\mathbf{s}_u(0,v) = H(v) \begin{bmatrix} \mathbf{s}_u(0,0) \\ \mathbf{s}_u(0,1) \\ \mathbf{s}_{uv}(0,0) \\ \mathbf{s}_{uv}(0,1) \end{bmatrix}$$

$$\mathbf{s}_u(1,v) = H(v) \begin{bmatrix} \mathbf{s}_u(1,0) \\ \mathbf{s}_u(1,1) \\ \mathbf{s}_{uv}(1,0) \\ \mathbf{s}_{uv}(1,1) \end{bmatrix}$$

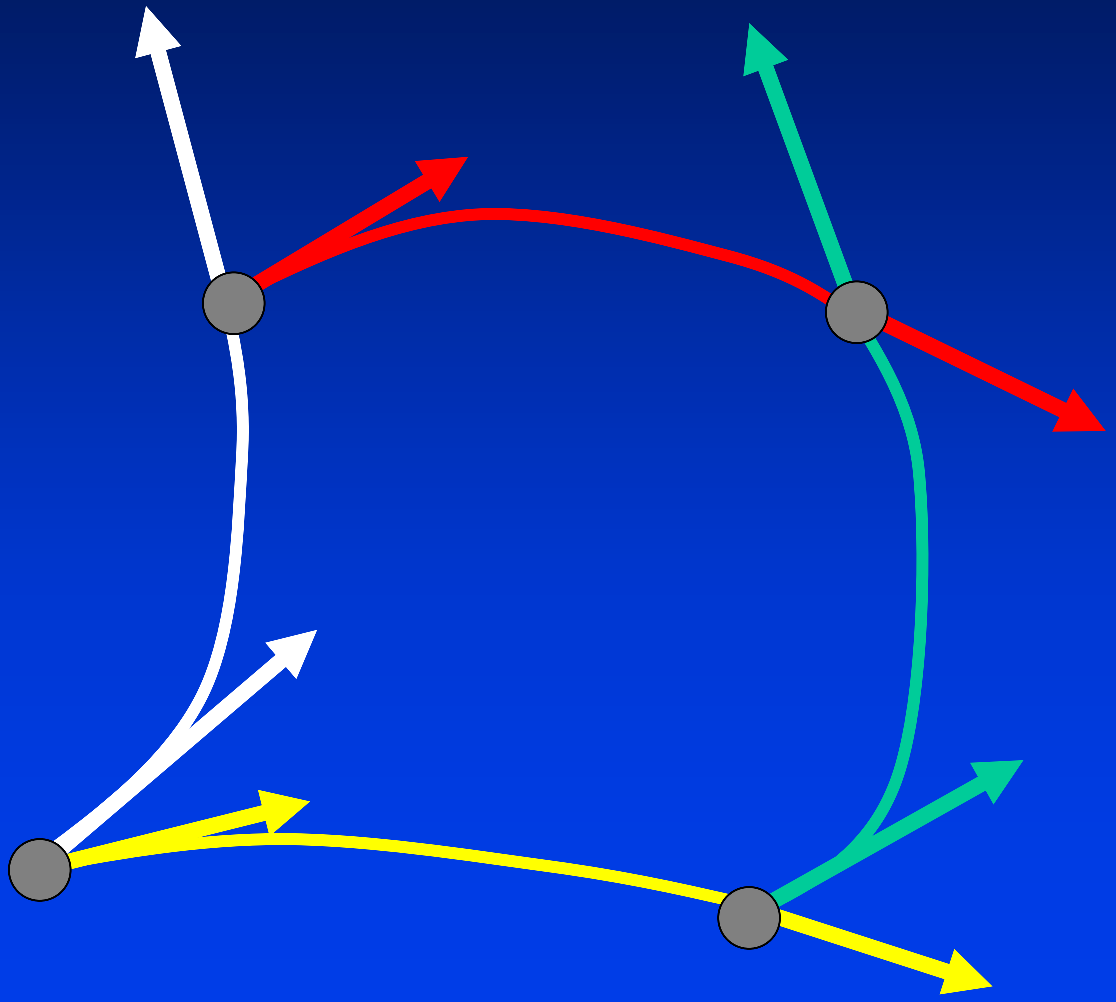
Hermite Surfaces

- It is time to put them together!

$$\mathbf{s}(u,v) = H(u)\begin{bmatrix} \mathbf{s}(0,0) & \mathbf{s}(0,1) & \mathbf{s}_v(0,0) & \mathbf{s}_v(0,1) \\ \mathbf{s}(1,0) & \mathbf{s}(1,1) & \mathbf{s}_v(1,0) & \mathbf{s}_v(1,1) \\ \mathbf{s}_u(0,0) & \mathbf{s}_u(0,1) & \mathbf{s}_{uv}(0,0) & \mathbf{s}_{uv}(0,1) \\ \mathbf{s}_u(1,0) & \mathbf{s}_u(1,1) & \mathbf{s}_{uv}(1,0) & \mathbf{s}_{uv}(1,1) \end{bmatrix} H(v)^T$$

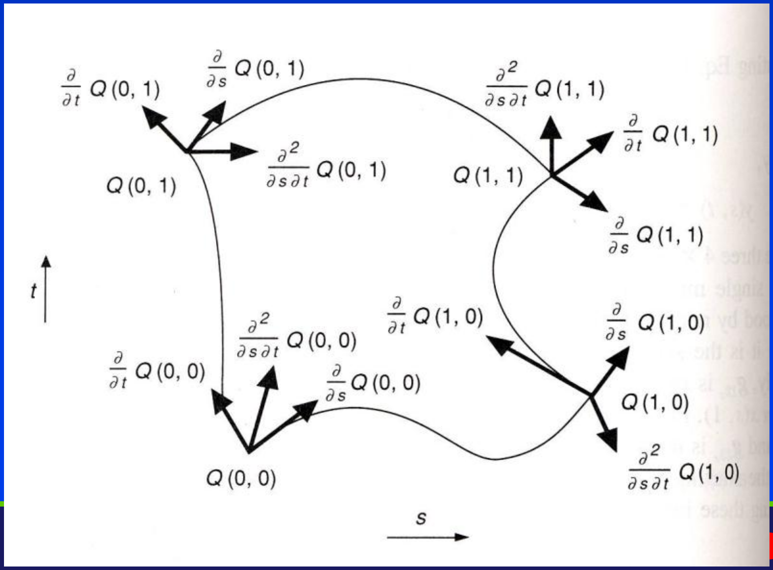
- Continuity conditions for surfaces
- Bezier surfaces, B-splines, NURBS, Hermite surfaces
- C1 and G1 continuity

Hermite Surfaces



Hermite Surfaces

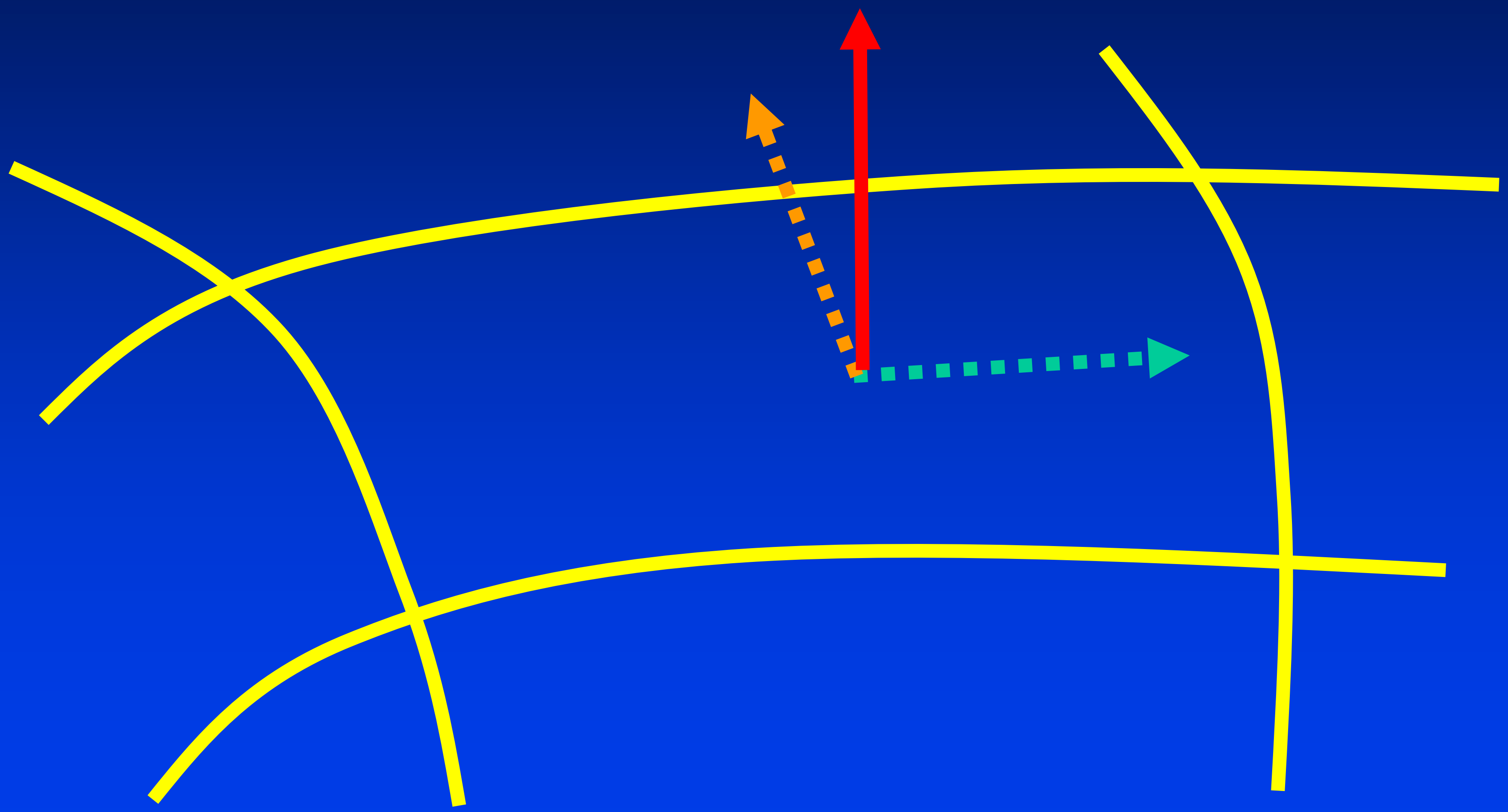
$$\mathbf{G}_{H_x} = \begin{bmatrix} x(0,0) & x(0,1) & \frac{\partial}{\partial t}x(0,0) & \frac{\partial}{\partial t}x(0,1) \\ x(1,0) & x(1,1) & \frac{\partial}{\partial t}x(1,0) & \frac{\partial}{\partial t}x(1,1) \\ \frac{\partial}{\partial s}x(0,0) & \frac{\partial}{\partial s}x(0,1) & \frac{\partial^2}{\partial s \partial t}x(0,0) & \frac{\partial^2}{\partial s \partial t}x(0,1) \\ \frac{\partial}{\partial s}x(1,0) & \frac{\partial}{\partial s}x(1,1) & \frac{\partial^2}{\partial s \partial t}x(1,0) & \frac{\partial^2}{\partial s \partial t}x(1,1) \end{bmatrix}.$$



Rendering Curves and Surfaces

- One way of rendering a curve/surface is to compute intersections with rays from the eye through each pixel.
  - – costly for real-time rendering
- Another approach is to evaluate the curve or surface at enough points to approximate it with standard flat objects (i.e. lines or polygons)
- Recursive subdivision techniques can also be used and are very efficient - good for adaptive rendering.

Surface Normal



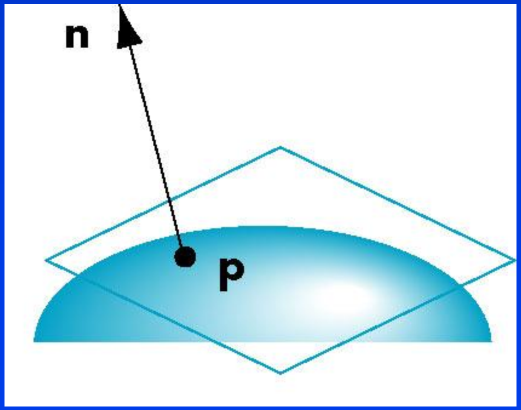
Normals

We can differentiate with respect to u and v to obtain the normal at any point **p**

$$\frac{\partial \mathbf{p}(u,v)}{\partial u} = \begin{bmatrix} \partial x(u,v)/\partial u \\ \partial y(u,v)/\partial u \\ \partial z(u,v)/\partial u \end{bmatrix}$$

$$\frac{\partial \mathbf{p}(u,v)}{\partial v} = \begin{bmatrix} \partial x(u,v)/\partial v \\ \partial y(u,v)/\partial v \\ \partial z(u,v)/\partial v \end{bmatrix}$$

$$\mathbf{n} = \frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$



Normals to Surfaces

$$\frac{\partial}{\partial s}Q(s,t) = T^{\mathrm{T}} \bullet M^{\mathrm{T}} \bullet \mathbf{G} \bullet M \bullet \frac{\partial}{\partial s}S$$

$$= T^{\mathrm{T}} \bullet M^{\mathrm{T}} \bullet \mathbf{G} \bullet M \bullet \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix}^{\mathrm{T}}$$

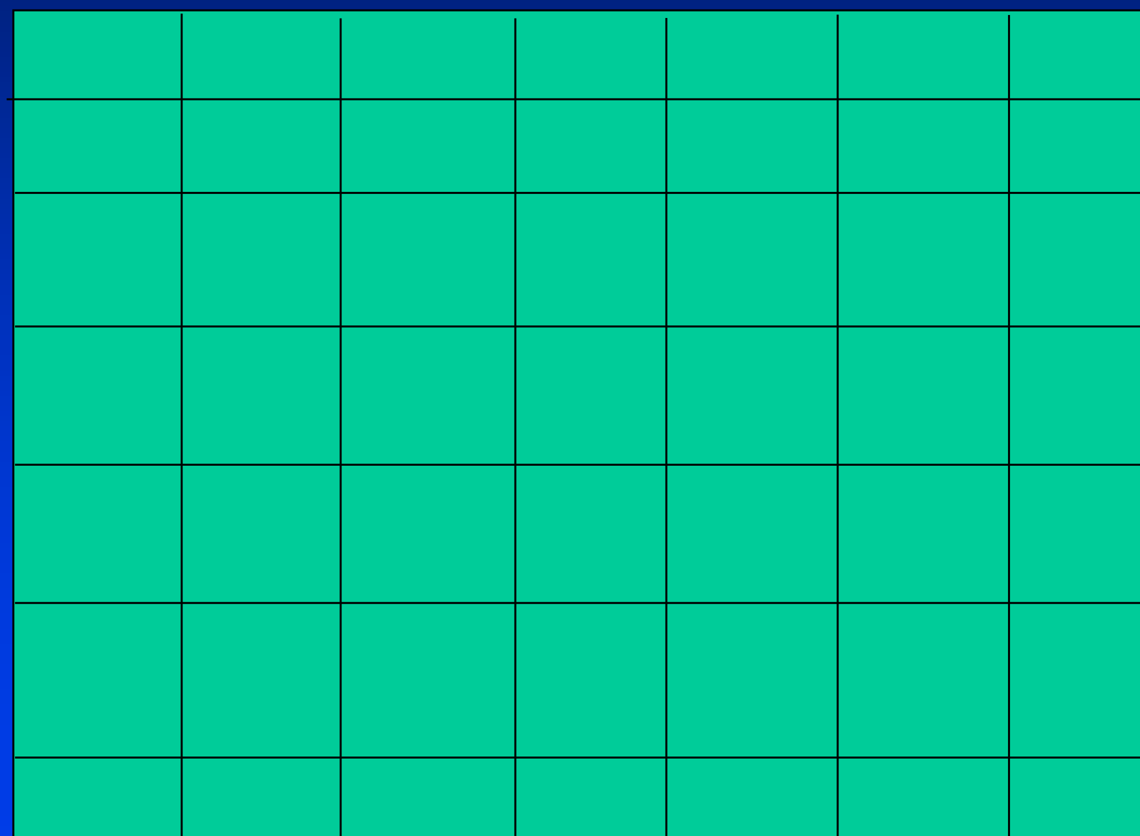
$$\frac{\partial}{\partial t}Q(s,t) = \frac{\partial}{\partial t}\left(T^{\mathrm{T}}\right) \bullet M^{\mathrm{T}} \bullet \mathbf{G} \bullet M \bullet S$$

$$= \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}^{\mathrm{T}} \bullet M^{\mathrm{T}} \bullet \mathbf{G} \bullet M \bullet S$$

$$\frac{\partial}{\partial s}Q(s,t) \times \frac{\partial}{\partial t}Q(s,t) \longleftarrow$$ **normal vector**

Surface Rendering

- Parametric grids ([0,1]X[0,1]) as a set of rectangles

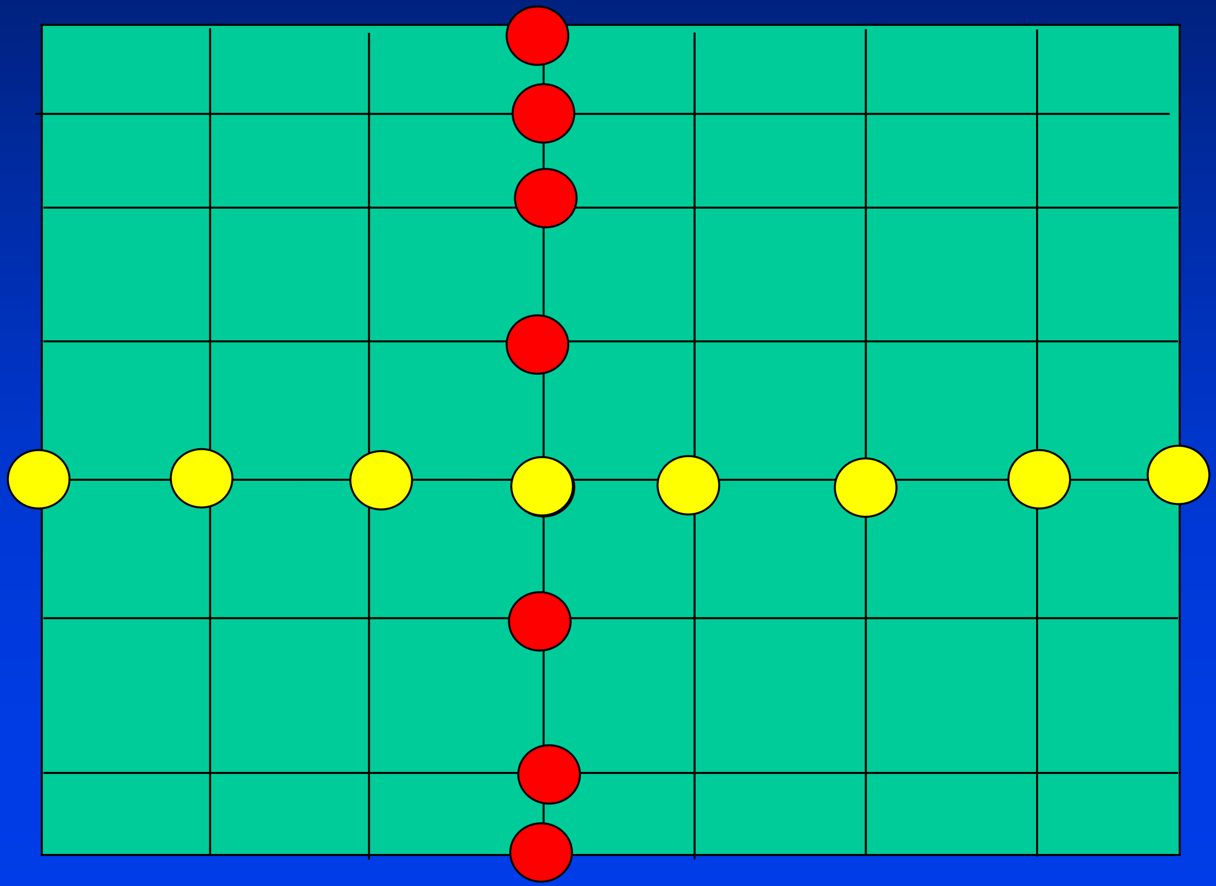


Surface (Patch) Rendering

- We use bicubic as an example
- The simplest (naïve): convert curved patches into primitives that we always know how to render
- From curved surfaces to polygon quadrilaterals (non-planar) and/or triangles (planar)
- Surface evaluation at grid points
- This is straight forward but inefficient, because it requires many times of evaluation of s(u,v)
- The total number is $3\frac{1}{\delta u}\frac{1}{\delta v}$

Surface Rendering

- Parametric grids ([0,1]X[0,1]) as a set of rectangles



Surface Rendering

- Better approach: precomputation

$$\mathbf{s}(u,v) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} M \begin{bmatrix} v^3 \\ v^2 \\ v^2 \\ 1 \end{bmatrix}$$

- M is constant throughout the entire patch. The followings are the same along isoparametric lines

$$\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$
$$\begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

- Use one dimensional array to compute and store (evaluation only once)

Surface Rendering

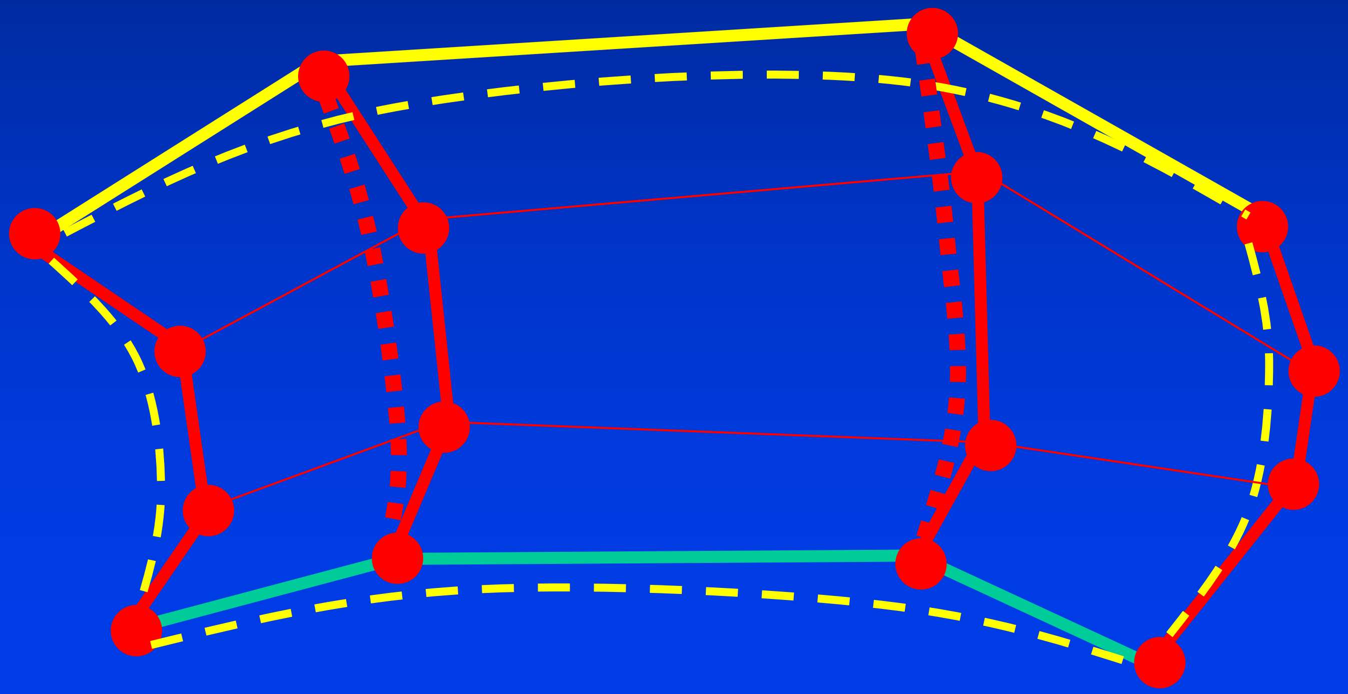
- How about many patches: the array is unchanged, its sampling rate is the same, this is more useful
- How about adaptive sampling based on curvature information!!!
- How to computer normal at any grid point (approximation)

$$\mathbf{s}_u(u,v)\times\mathbf{s}_v(u,v)$$

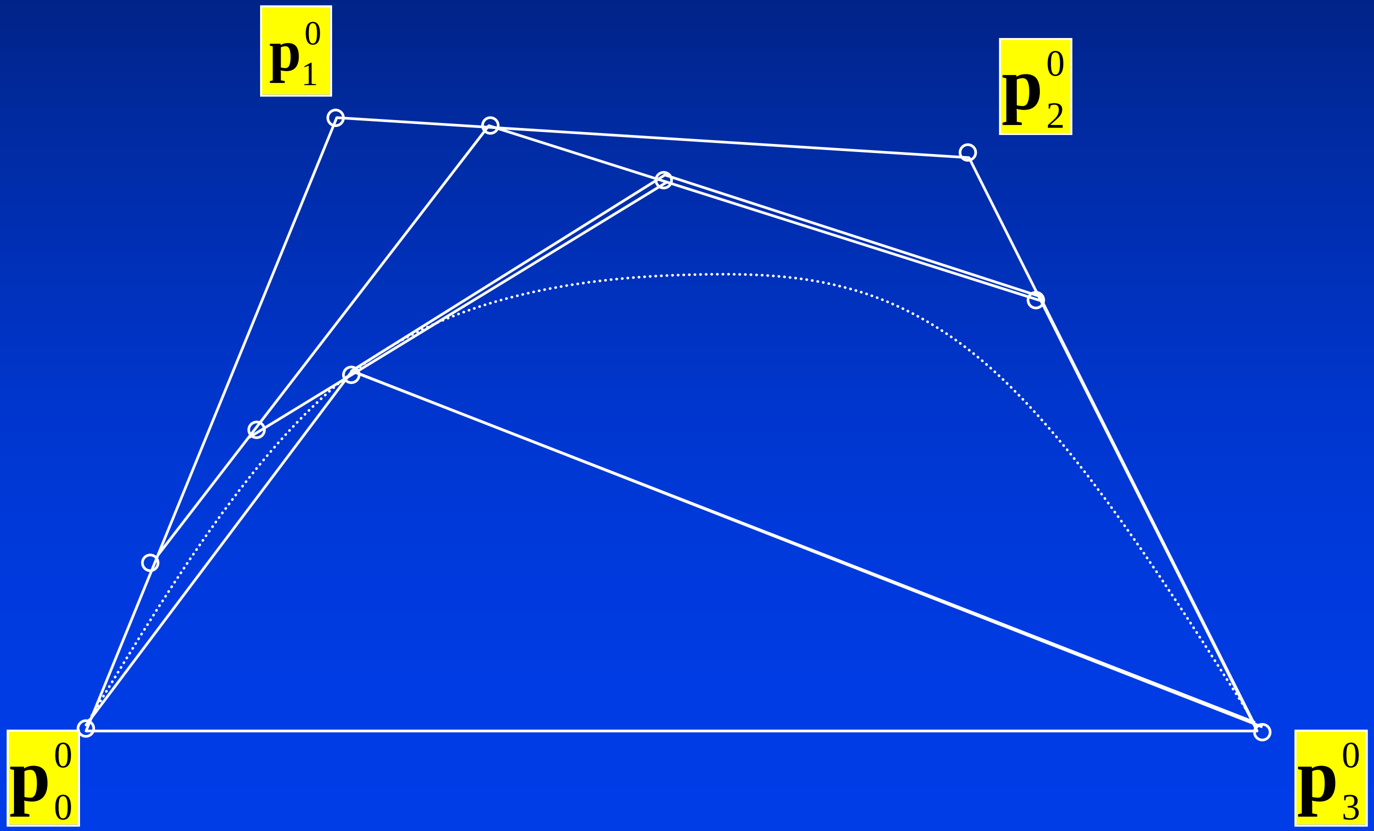
$$(\mathbf{s}(u+\delta u,v)-\mathbf{s}(u,v))\times(\mathbf{s}(u,v+\delta v)-\mathbf{s}(u,v))$$

Regular Surface

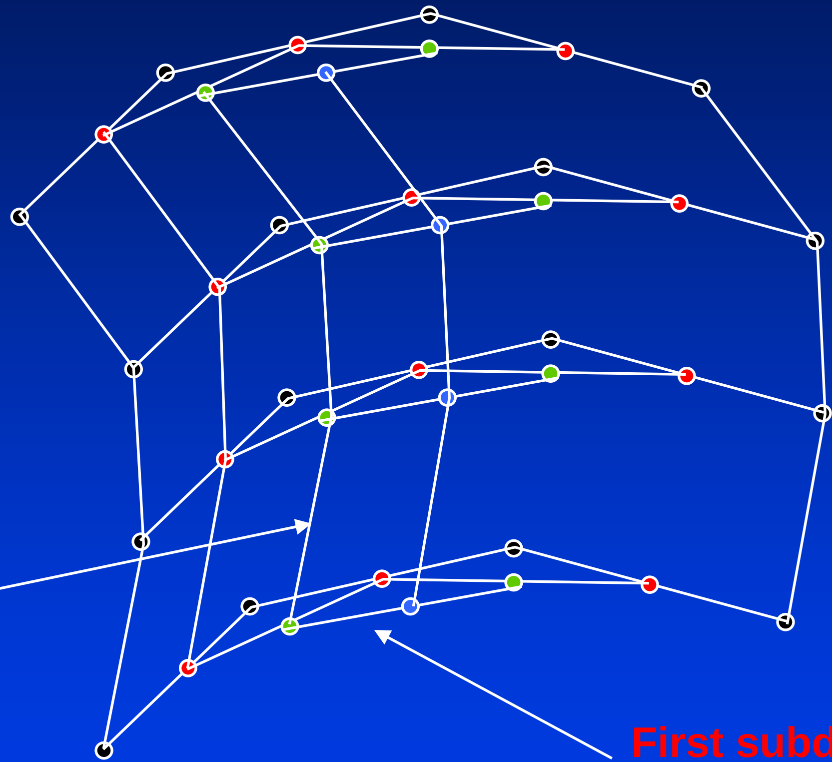
- Generated from a set of control points.



Recursive Subdivision of Bezier Curves



Rendering Bezier Patch by Recursive Subdivision

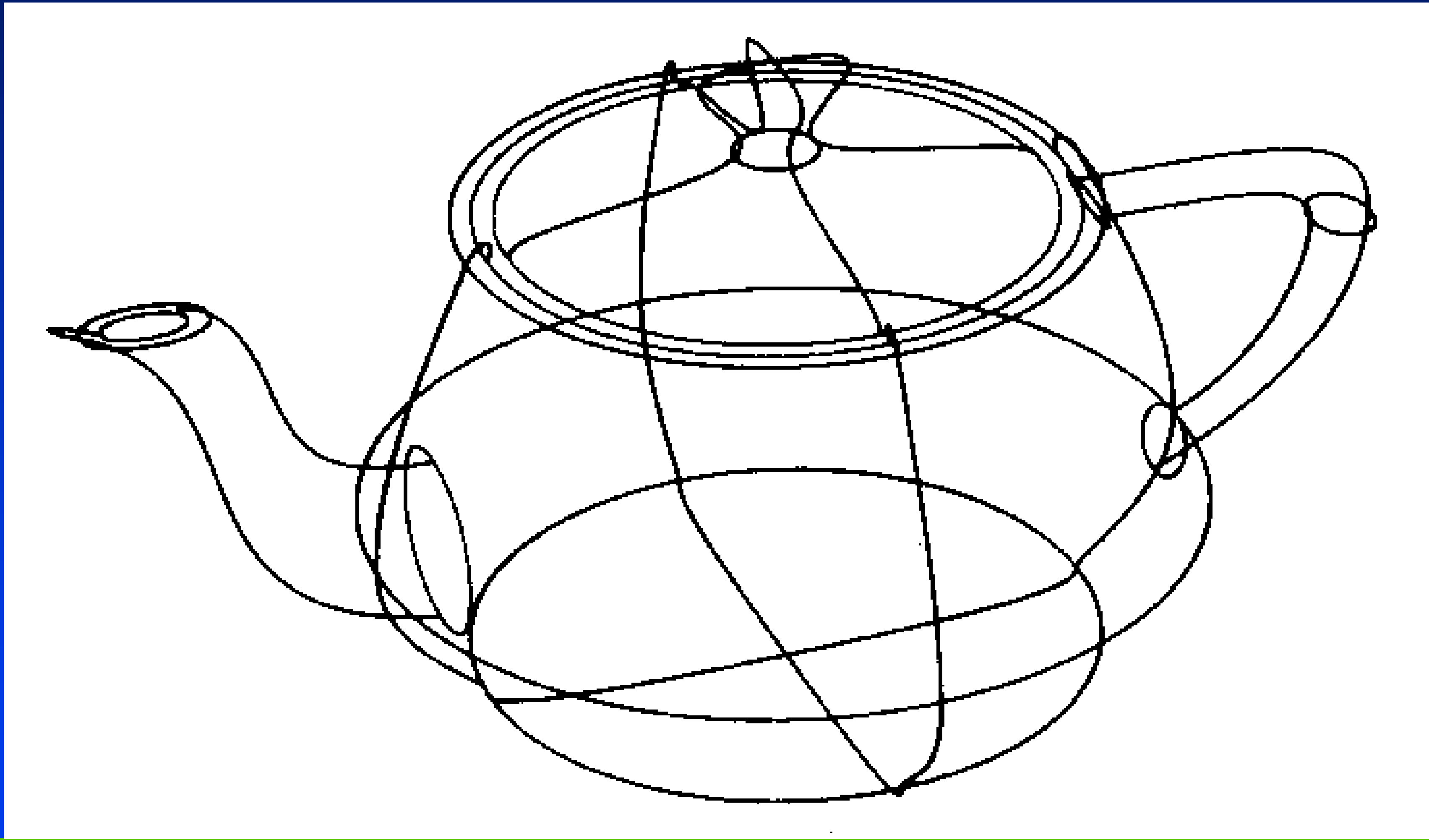


Connect new control points to form new curves.

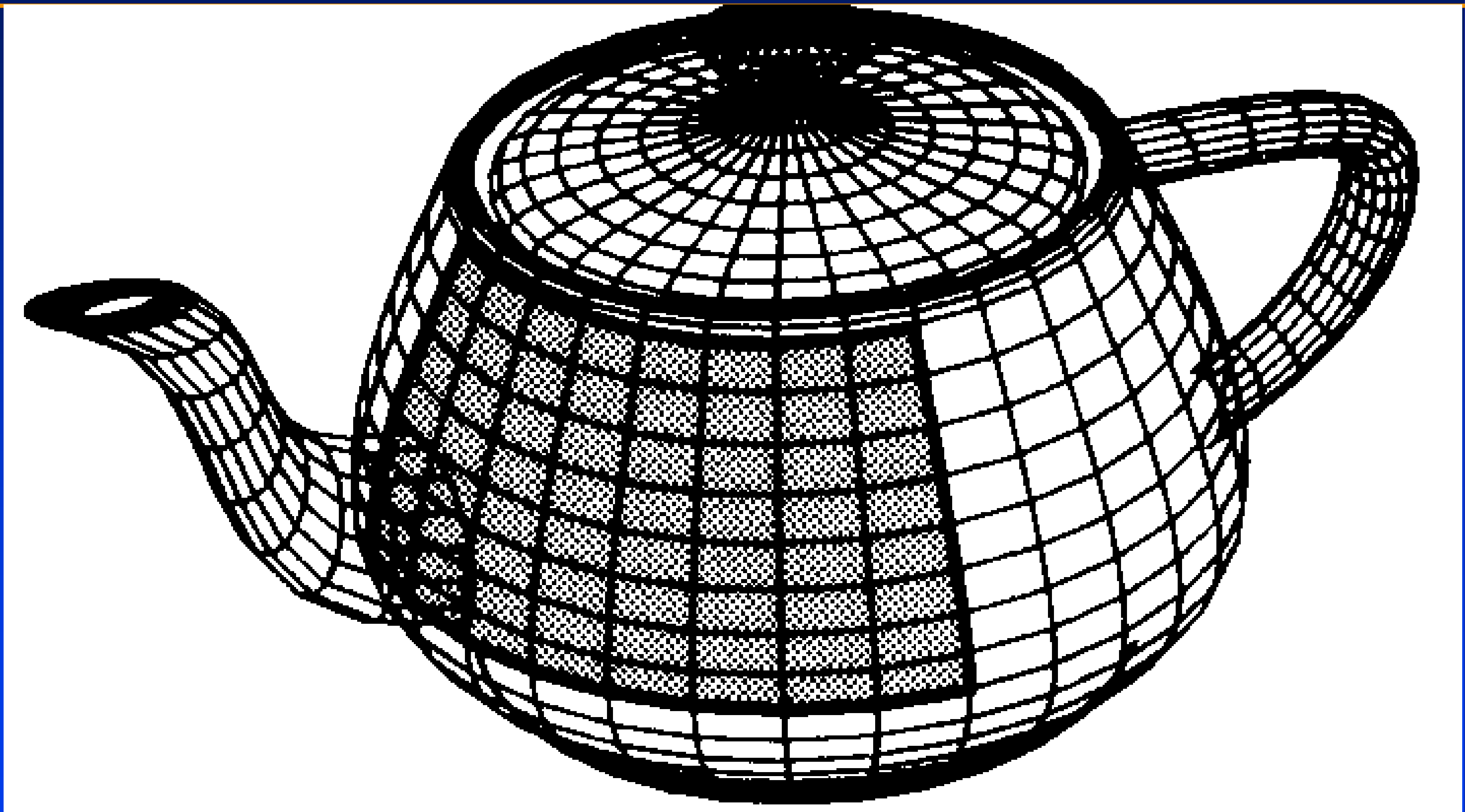
First subdivide curves of constant v.

Finally subdivide these curves to form 4 new patches.

The Utah Teapot: 32 Bezier Patches



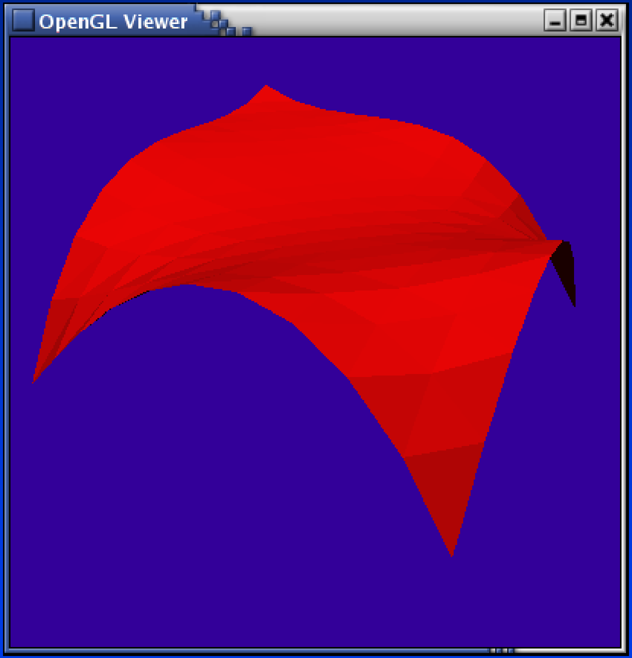
Utah Teapot: Polygon Representation



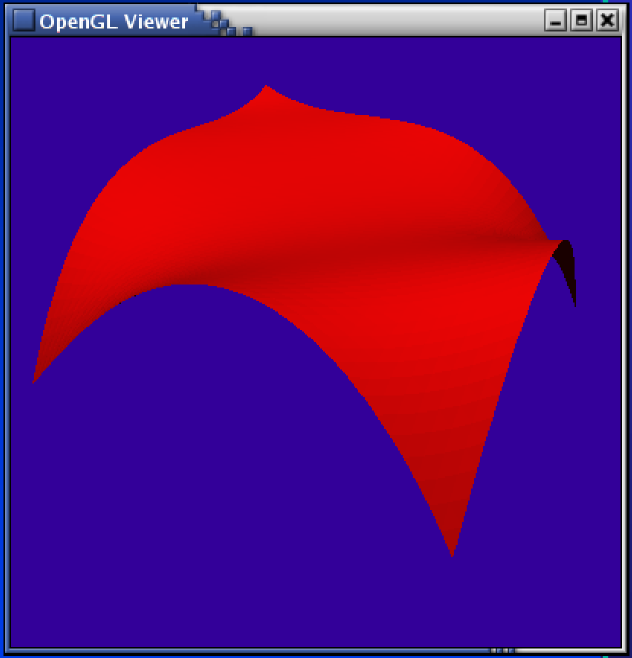
Displaying Bezier Patch

- Given 16 control points (Bicubic Bezier Patch) and a tessellation resolution, create a triangle mesh

resolution:
5x5 vertices



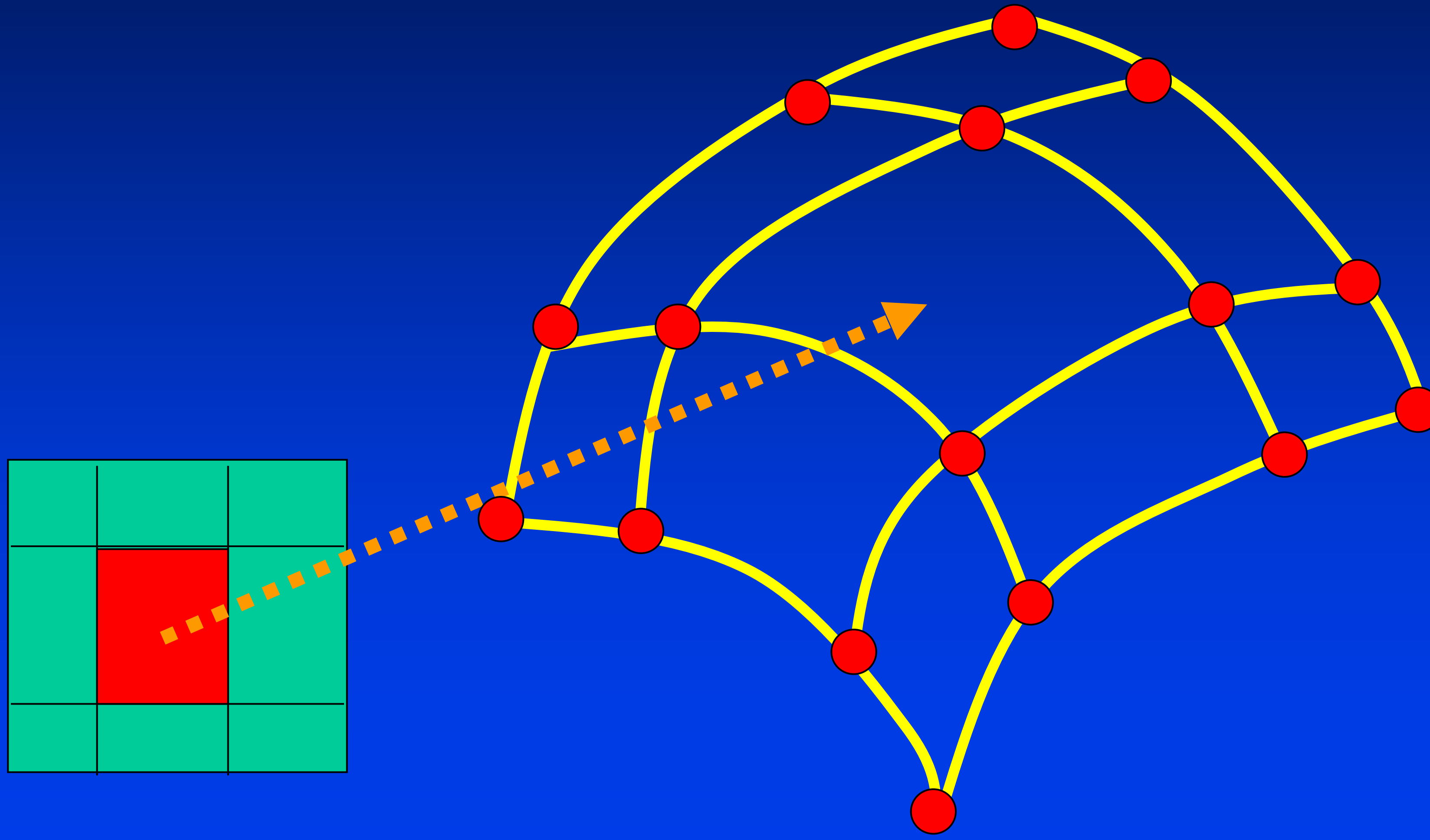
resolution:
11x11 vertices

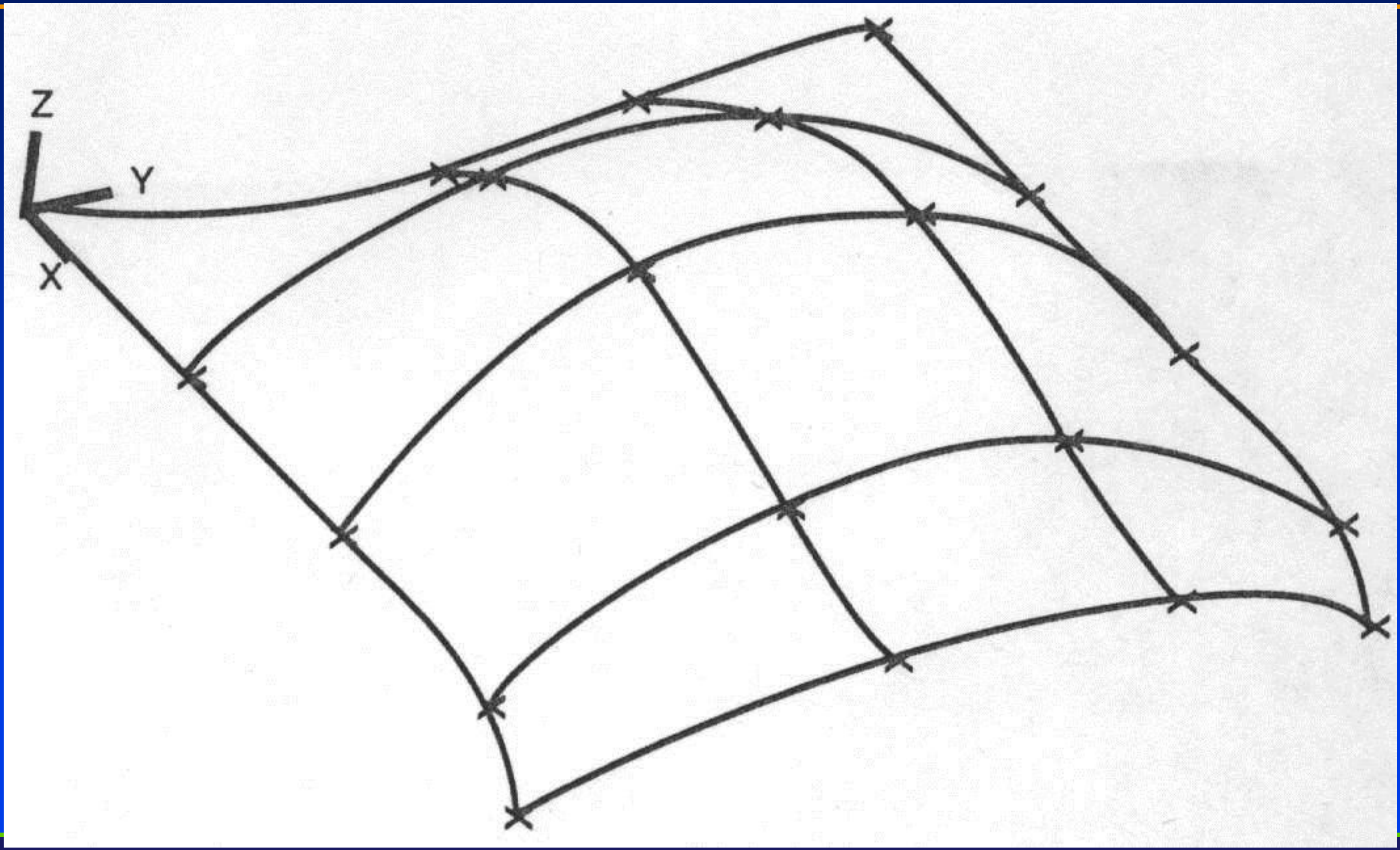


resolution:
41x41 vertices

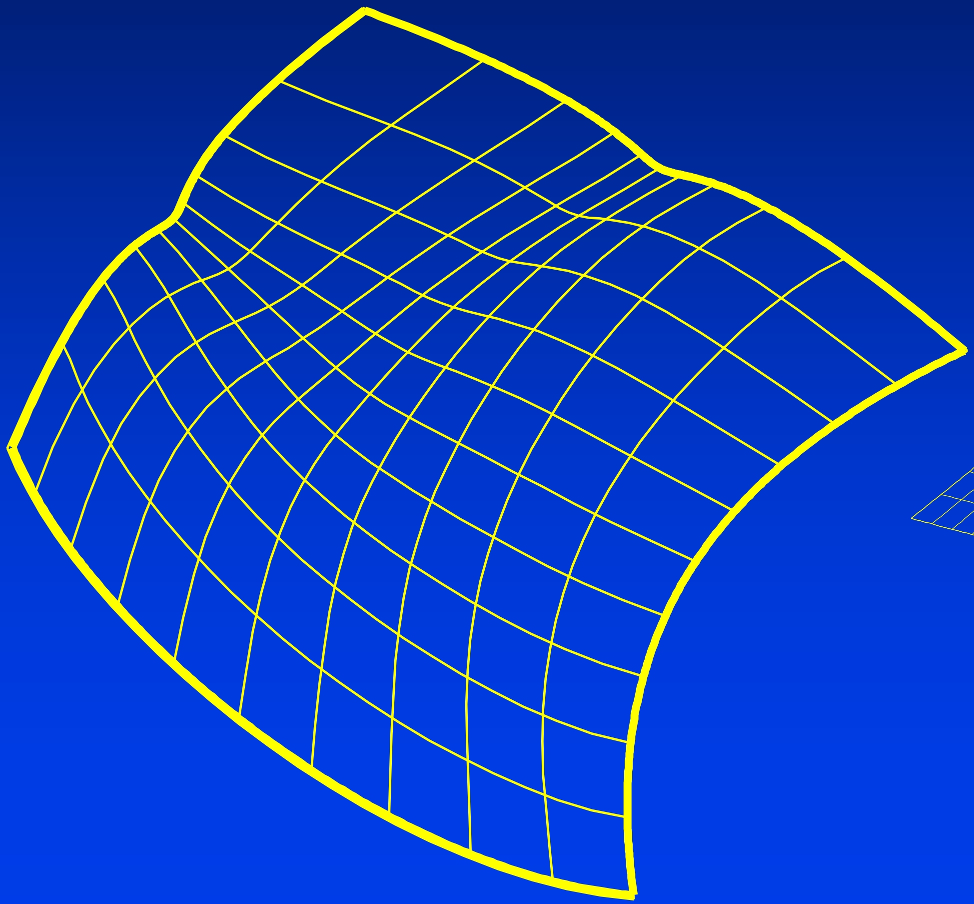
Rendering the Teapot

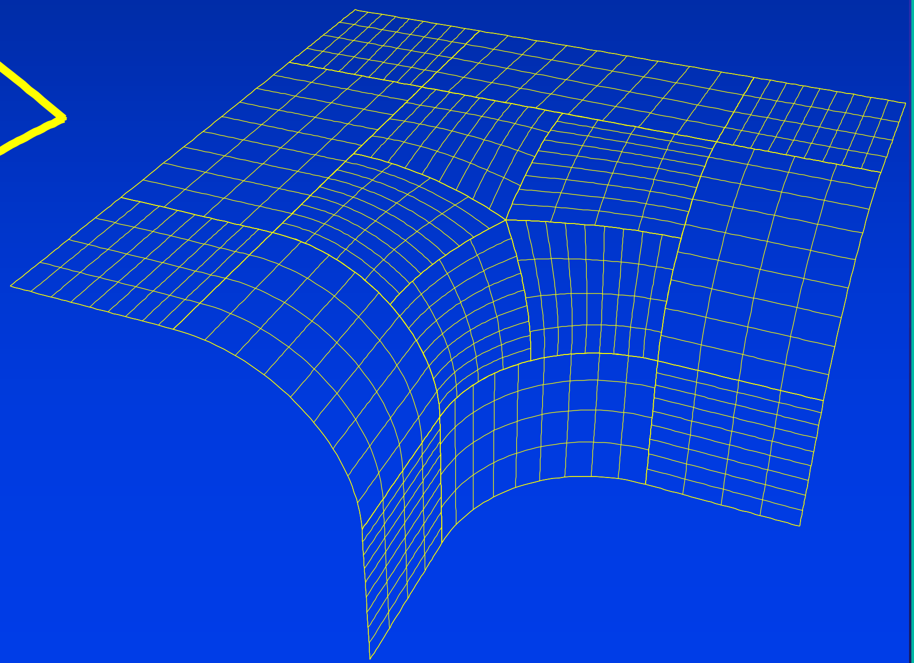
Curve Network



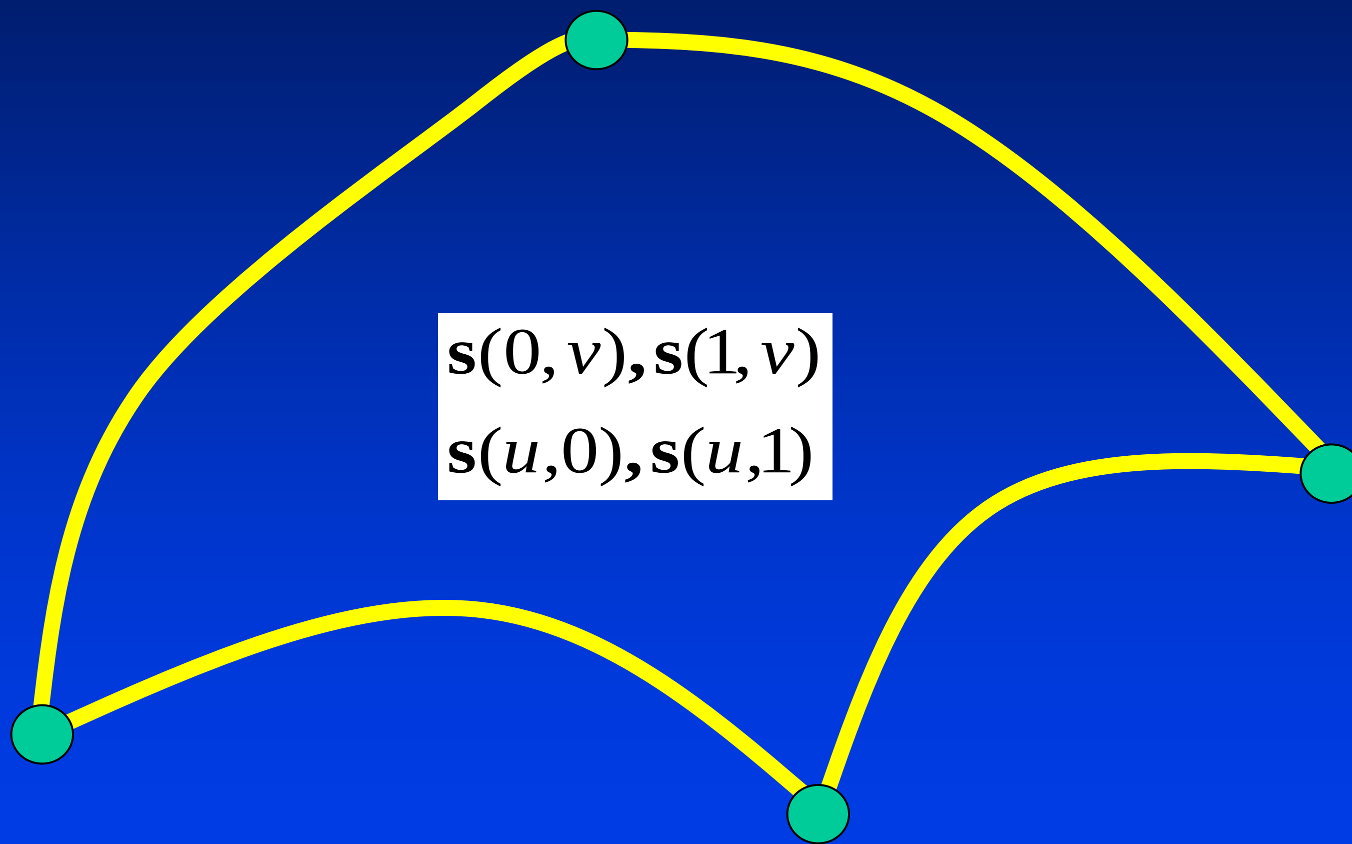

Z
Y
X

Transfinite Method and N-side Hole Filling





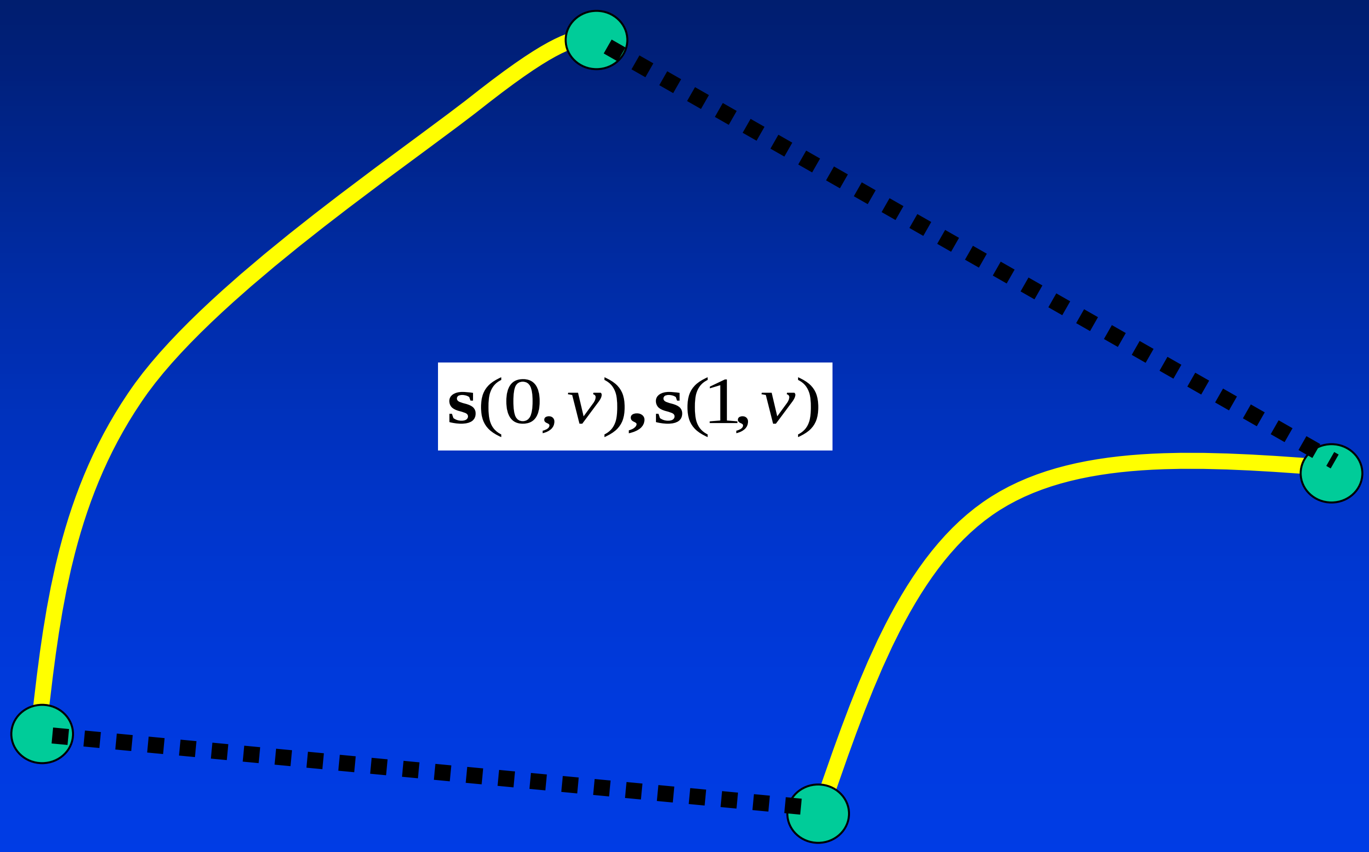
Coons Patch



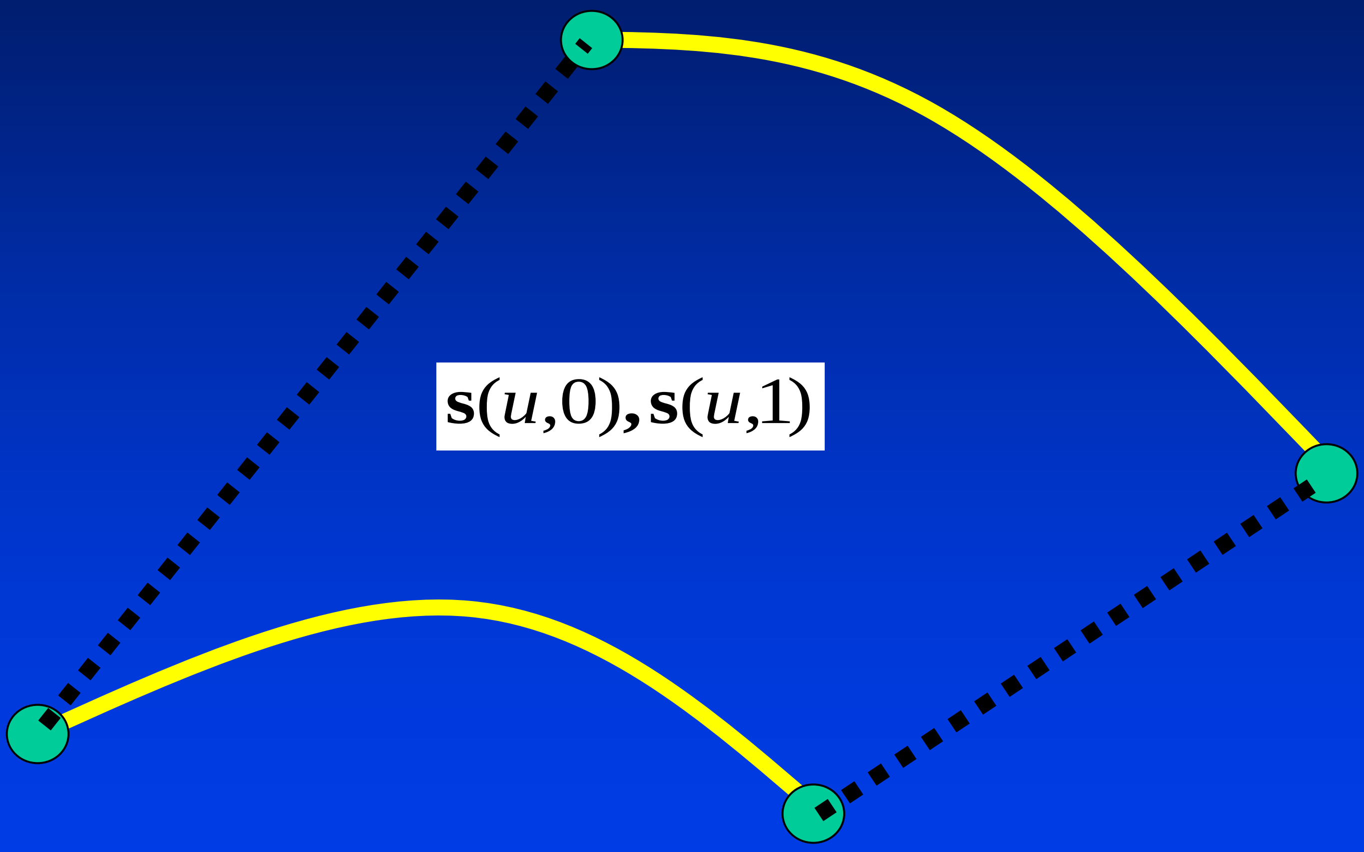
$$\mathbf{s}(0,v),\mathbf{s}(1,v)$$

$$\mathbf{s}(u,0),\mathbf{s}(u,1)$$

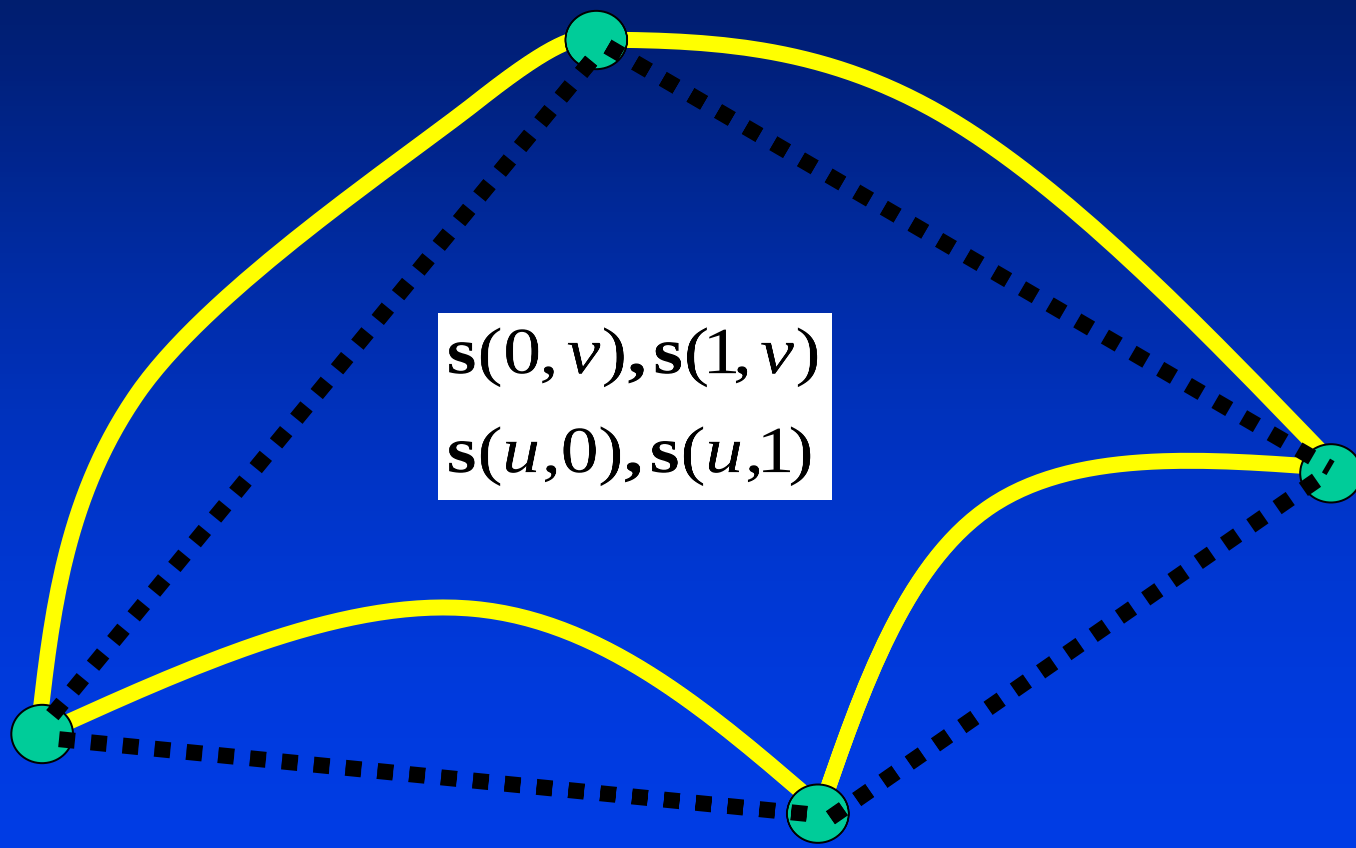
Coons Patch



Coons Patch



Coons Patch



Coons Patch

- Bilinearly blended Coons patch

$$(P)\mathbf{f} = (P_1 \oplus P_2)\mathbf{f} = (P_1 + P_2 - P_1P_2)\mathbf{f}$$
$$(P_1)\mathbf{f} = \mathbf{f}(0,v)L_0^1(u) + \mathbf{f}(1,v)L_1^1(u)$$
$$(P_2)\mathbf{f} = \mathbf{f}(u,0)L_0^1(v) + \mathbf{f}(u,1)L_1^1(v)$$

- Bicubically blended Coons patch

$$(P_1)\mathbf{f} = \mathbf{f}(0,v)H_0^3(u) + \mathbf{f}_u(0,v)H_1^3(u) + \mathbf{f}_u(1,v)H_2^3(u) + \mathbf{f}(1,v)H_3^3(u)$$
$$(P_2)\mathbf{f} = \mathbf{f}(u,0)H_0^3(v) + \mathbf{f}_v(u,0)H_1^3(v) + \mathbf{f}_v(u,1)H_2^3(v) + \mathbf{f}(u,1)H_3^3(v)$$

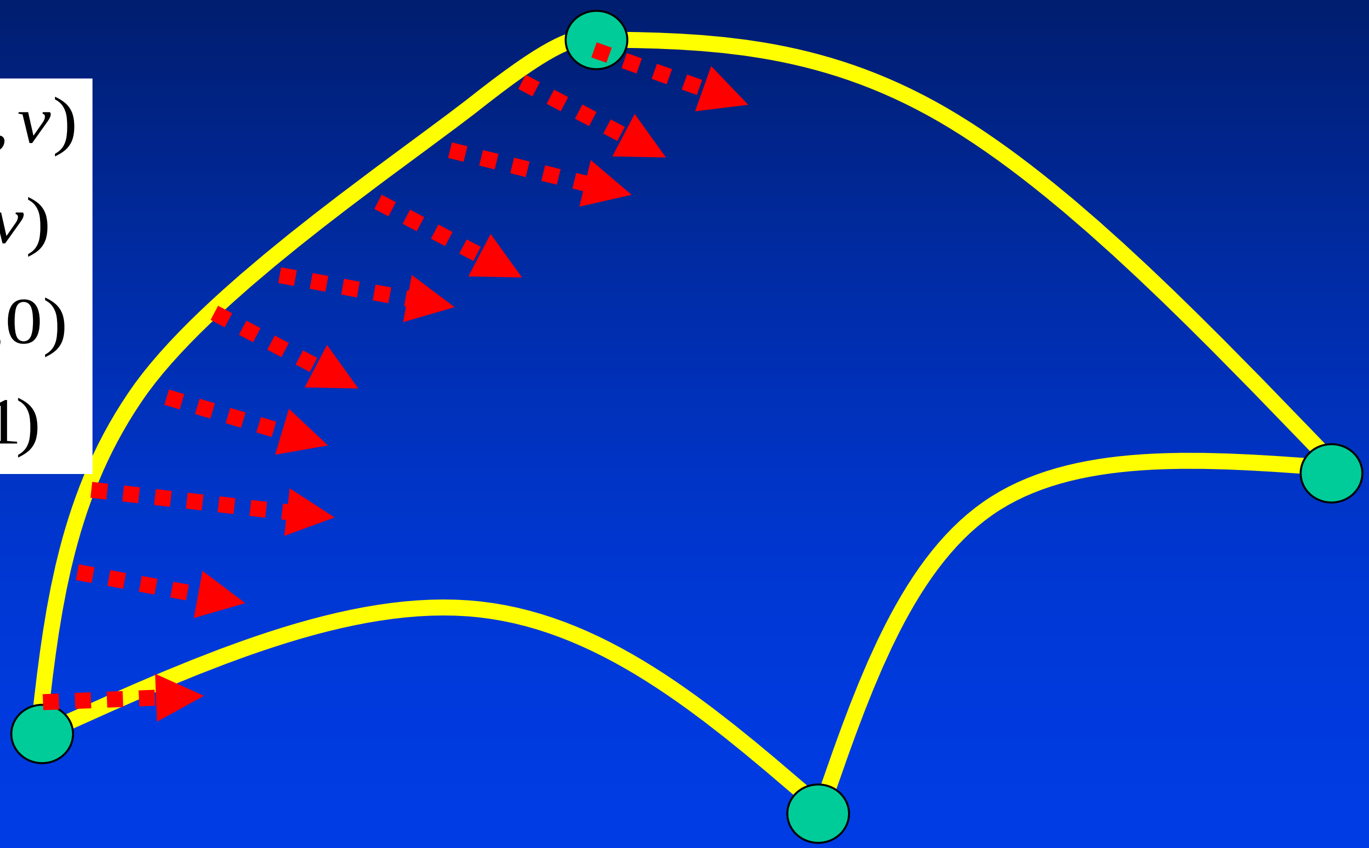
Coons Patch

$\mathbf{s}(0,v), \mathbf{s}_u(0,v)$

$\mathbf{s}(1,v), \mathbf{s}_u(1,v)$

$\mathbf{s}(u,0), \mathbf{s}_v(u,0)$

$\mathbf{s}(u,1), \mathbf{s}_v(u,1)$



Gordon Surfaces

- Generalization of Coons techniques
- A set of curves

$$\mathbf{f}(u_i, v), i = 0, \ldots\ldots, n$$

$$\mathbf{f}(u, v_j), j = 0, \ldots\ldots, m$$

- Boolean sum using Lagrange polynomials

$$(P_1)\mathbf{f} = \sum_{i=0}^{n} \mathbf{f}(u_i, v) L_i^n(u)$$

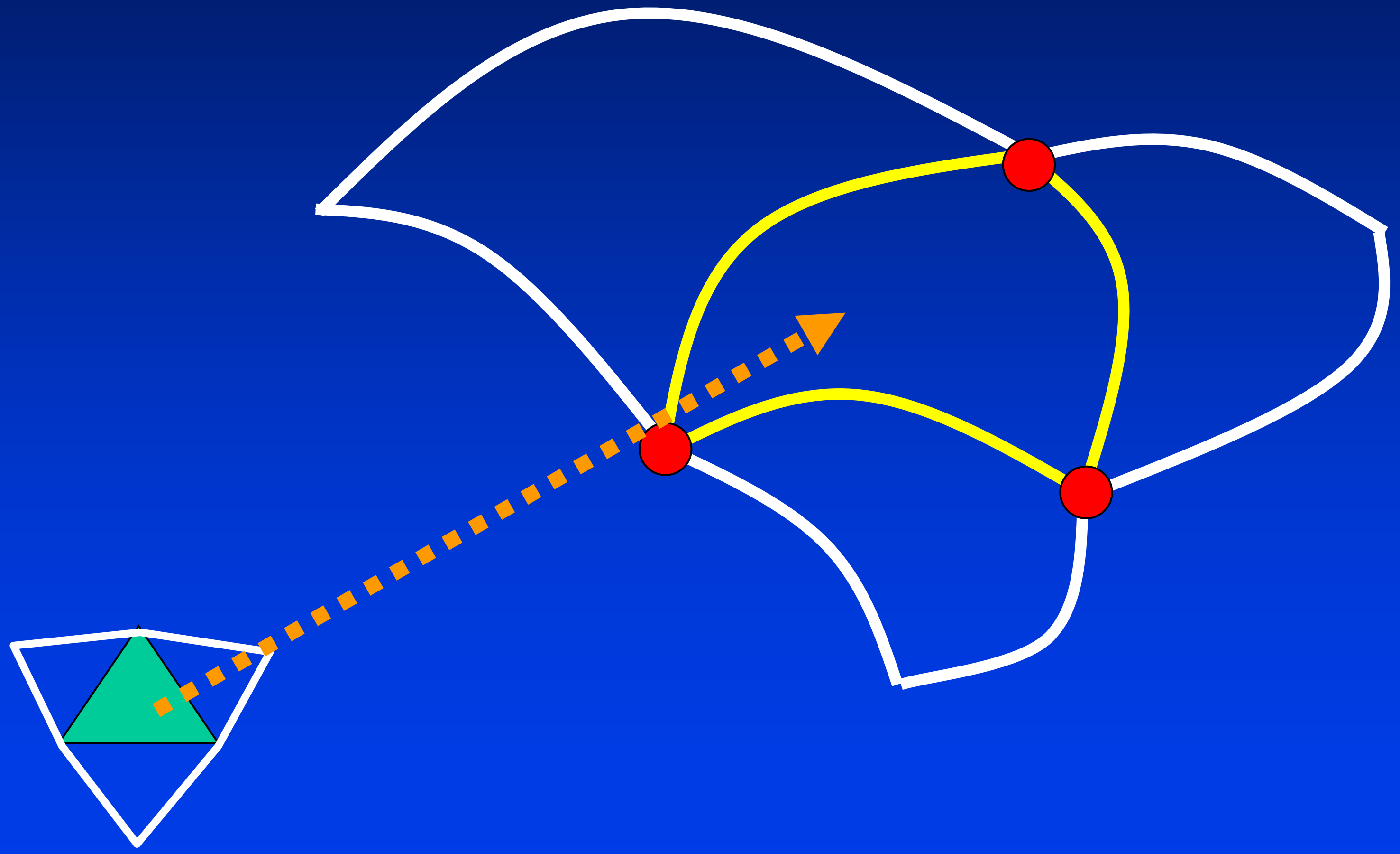
$$(P_2)\mathbf{f} = \sum_{j=0}^{m} \mathbf{f}(u, v_j) L_j^m(v)$$

$$(P)\mathbf{f} = (P_1 \oplus P_2)\mathbf{f} = (P_1 + P_2 - P_1 P_2)\mathbf{f}$$

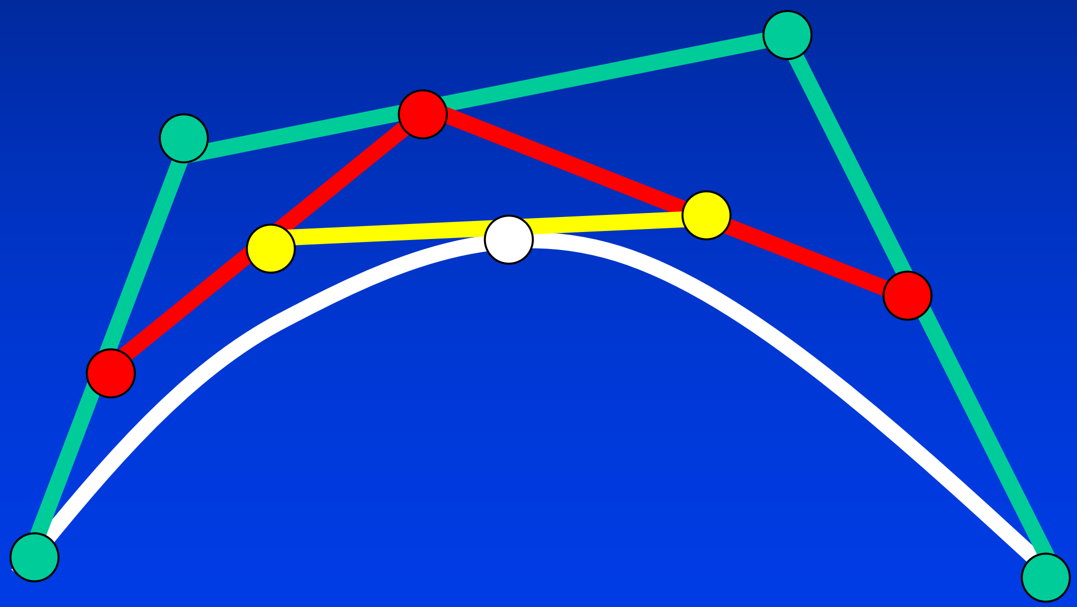
Transfinite Methods

- Bilinearly blended Coons patch
  – Interpolate four boundary curves
- Bicubically blended Coons patch
  – Interpolate curves and their derivatives
- Gordon surfaces
  – Interpolate a curve-network
- Triangular extension
  – Interpolate over triangles

Triangular Surfaces



Recursive Subdivision Algorithm



Curve Mathematics (Cubic)

- Bezier curve

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$$

- Control points and basis functions

$$B_0^3(u) = (1-u)^3$$
$$B_1^3(u) = 3u(1-u)^2$$
$$B_2^3(u) = 3u^2(1-u)$$
$$B_3^3(u) = u^3$$

- Image and properties of basis functions

Recursive Evaluation

- Recursive linear interpolation

$$(1-u) \quad (u)$$

$$\mathbf{p}_0^0 \quad \mathbf{p}_1^0 \quad \mathbf{p}_2^0 \quad \mathbf{p}_3^0$$

$$\mathbf{p}_0^1 \quad \mathbf{p}_1^1 \quad \mathbf{p}_2^1$$

$$\mathbf{p}_0^2 \quad \mathbf{p}_1^2$$

$$\mathbf{p}_0^3 = \mathbf{c}(u)$$

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

Derivatives

- Tangent vectors can easily evaluated at the end-points $\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0); \mathbf{c}'(1) = (\mathbf{p}_3 - \mathbf{p}_2)$
- Second derivatives at end-points can also be easily computed:

$$\mathbf{c}^{(2)}(0) = 2 \times 3((\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)) = 6(\mathbf{p}_2 - 2\mathbf{p}_1 + \mathbf{p}_0)$$

$$\mathbf{c}^{(2)}(1) = 2 \times 3((\mathbf{p}_3 - \mathbf{p}_2) - (\mathbf{p}_2 - \mathbf{p}_1)) = 6(\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1)$$

Derivative Curve

- The derivative of a cubic Bezier curve is a quadratic Bezier curve

$$\mathbf{c}'(u) = -3(1-u)^2\mathbf{p}_0 + 3((1-u)^2 - 2u(1-u))\mathbf{p}_1 + 3(2u(1-u) - u^2)\mathbf{p}_2 + 3u^2\mathbf{p}_3 =$$

$$3(\mathbf{p}_1 - \mathbf{p}_0)(1-u)^2 + 3(\mathbf{p}_2 - \mathbf{p}_1)2u(1-u) + 3(\mathbf{p}_3 - \mathbf{p}_2)u^2$$

More Properties (Cubic)

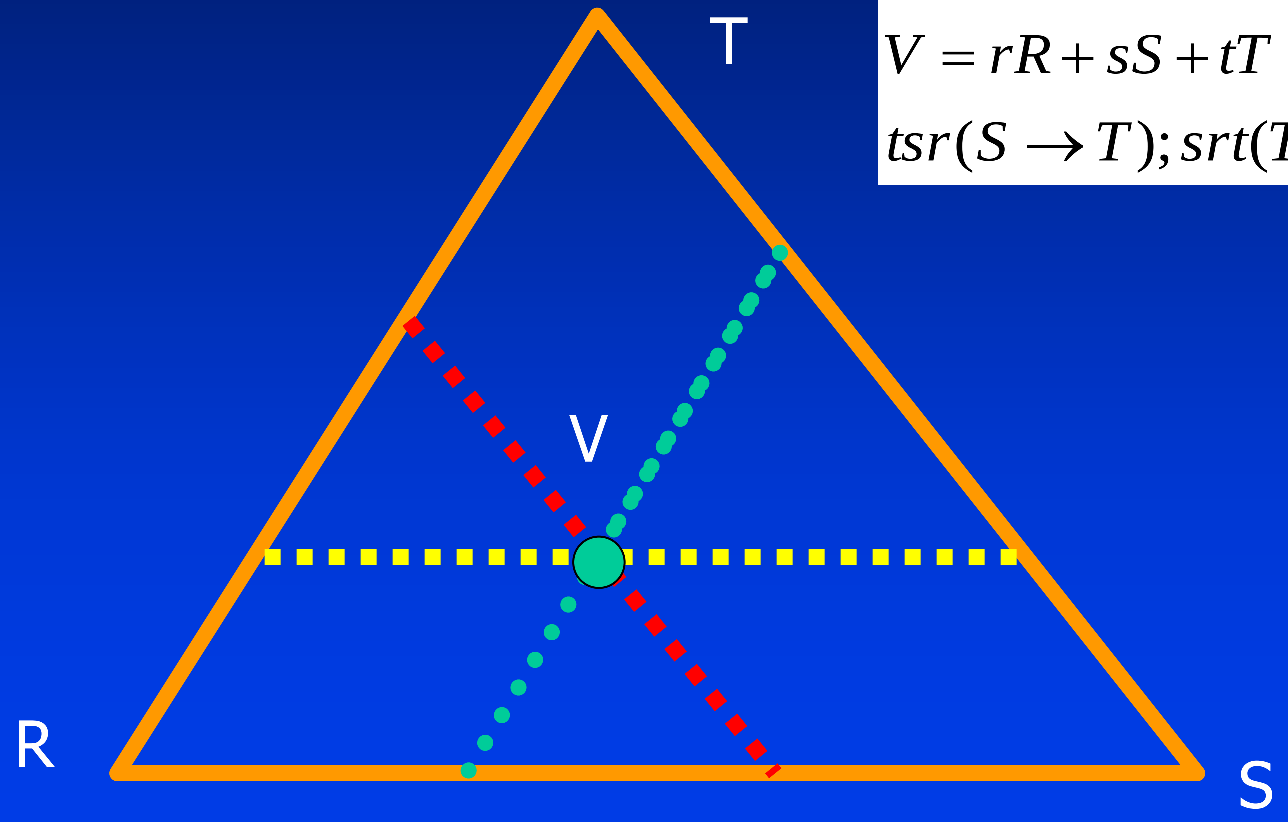
- Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

$$\mathbf{c}(v), v \in [0, u]$$
$$\mathbf{c}(v), v \in [u, 1]$$
$$\mathbf{c}_l(u), u \in [0,1]$$
$$\mathbf{c}_r(u), u \in [0,1]$$

- The control points for the left and the right are

$$\mathbf{P}_0^0, \mathbf{P}_0^1, \mathbf{P}_0^2, \mathbf{P}_0^3$$
$$\mathbf{P}_0^3, \mathbf{P}_1^2, \mathbf{P}_2^1, \mathbf{P}_3^0$$

Barycentric Coordinates



$$r + s + t = 1$$

$$V = rR + sS + tT$$

$$tsr(S \to T); srt(T \to S); rts(S \to R)$$

Triangular Bezier Patch

- Triangular Bezier surface

$$\mathbf{s}(u,v) = \sum_{i,j,k>=0}^{i+j+k=n} \mathbf{p}_{i,j,k} B^n_{i,j,k}(r,s,t)$$

- Where r+s+t=1, and they are local barycentric coordinates
- Basis functions are Bernstein polynomials of degree n

$$B^n_{i,j,k}(r,s,t) = \frac{n!}{i!\,j!\,k!} r^i s^j t^k$$

Triangular Bezier Patch

- How many control points and basis functions:

$$\frac{1}{2}(n+1)(n+2)$$

- Partition of unity

$$\sum_{i,j,k>=0} B^n_{i,j,k}(r,s,t) = 1$$

- Positivity

$$B^n_{i,j,k}(r,s,t) >= 0; r,s,t \in [0,1]$$

Recursive Evaluation

$$\mathbf{p}_{i,j,k}^{0} = \mathbf{p}_{i,j,k}$$

$$\mathbf{p}_{i,j,k}^{l} = r\mathbf{p}_{i+1,j,k}^{l-1} + s\mathbf{p}_{i,j+1,k}^{l-1} + t\mathbf{p}_{i,j,k+1}^{l-1}; i+j+k = n-l, i,j,k >= 0$$

$$\mathbf{s}(u,v) = \mathbf{p}_{0,0,0}^{n}$$

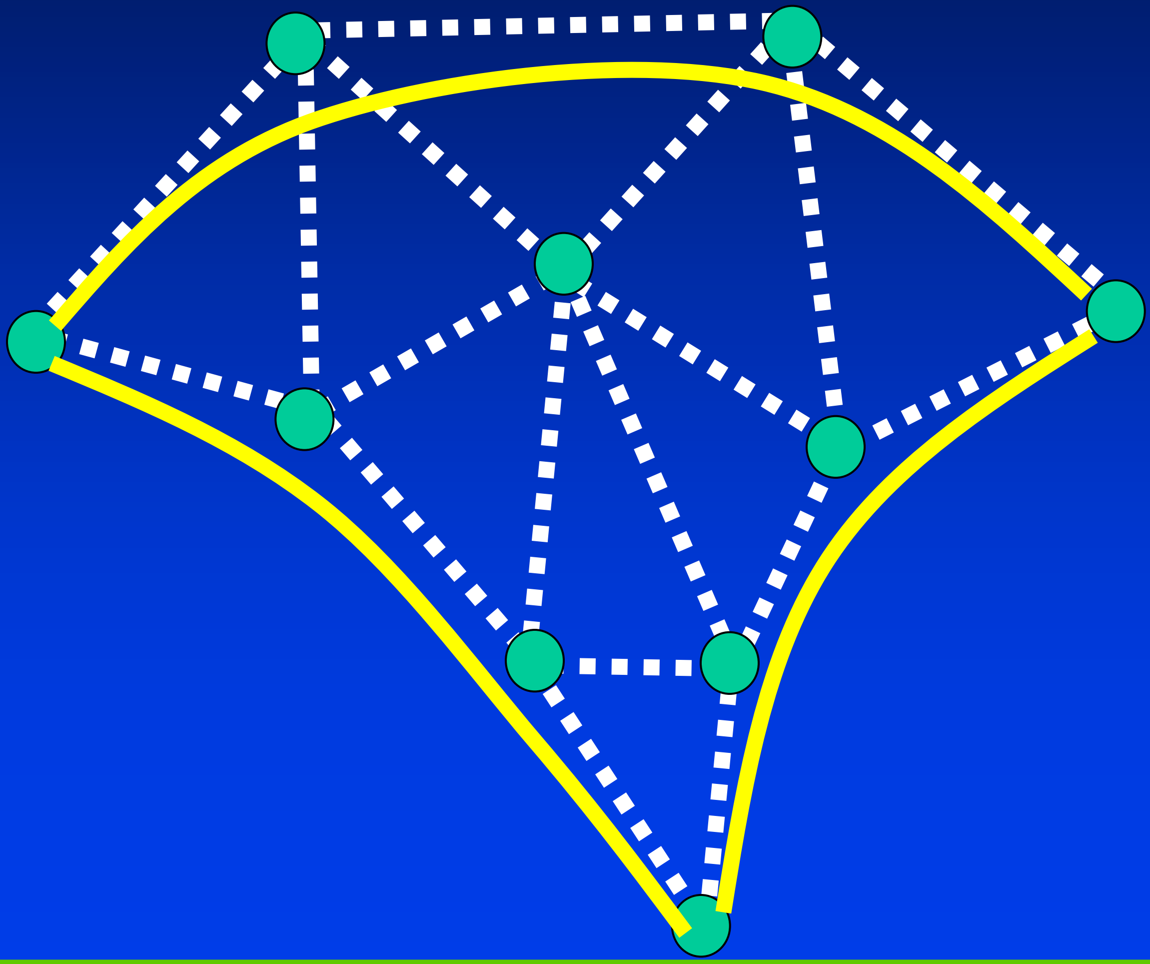
Properties

- Efficient algorithms
- Recursive evaluation
- Directional derivatives
- Degree elevation
- Subdivision
- Composite surfaces

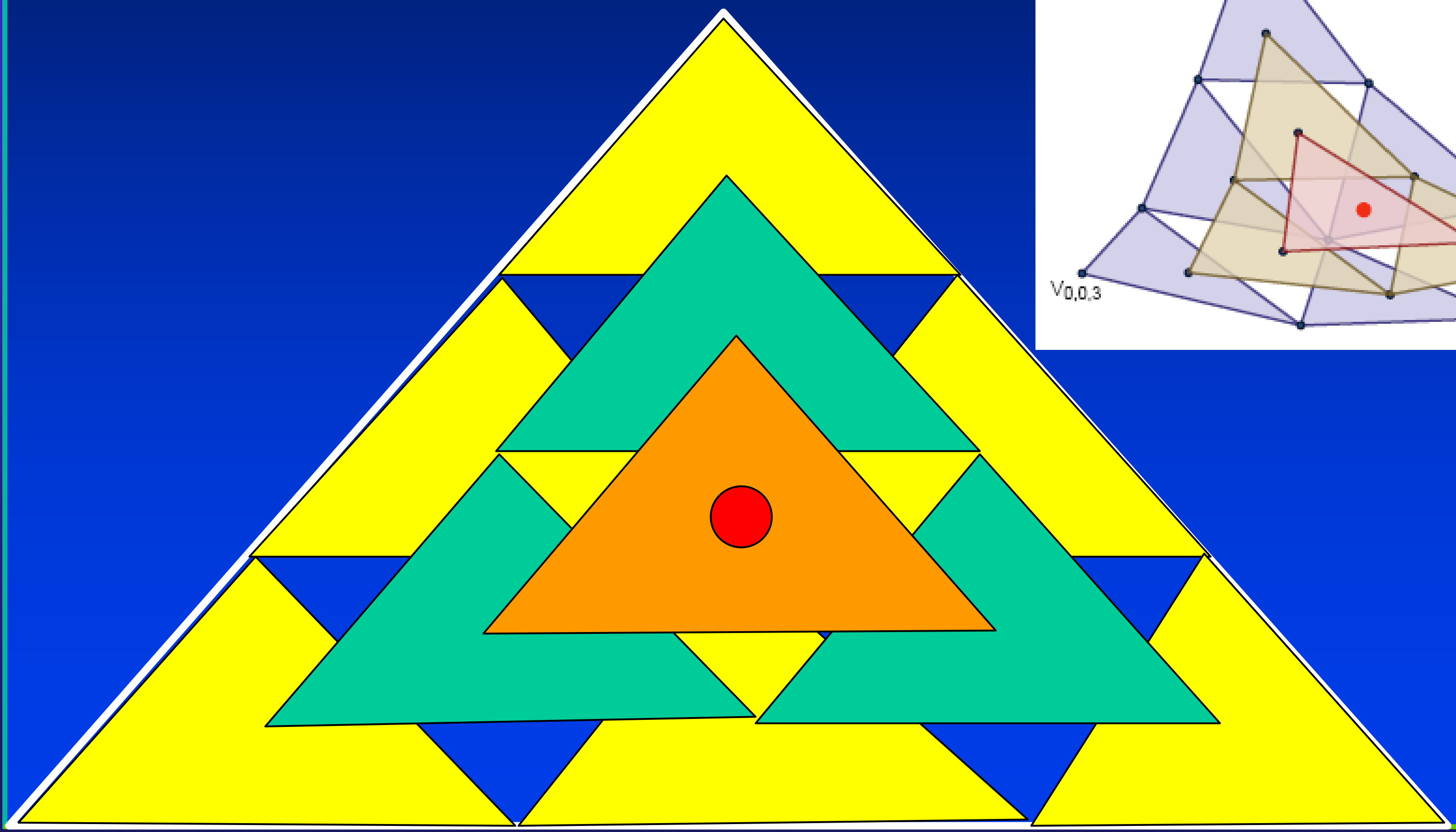
Research Issues

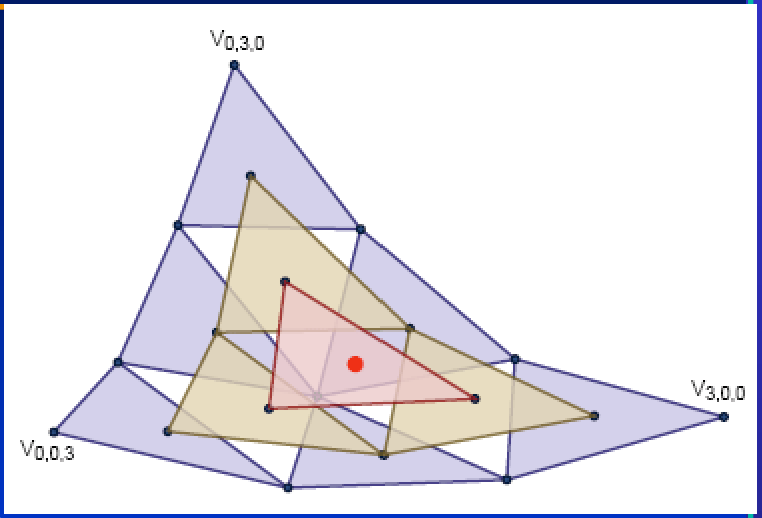
- Continuity across adjacent patches
- Integral computation
- Triangular splines over regular triangulation
- Transform triangular splines to a set of piecewise triangular Bezier patches
- Interpolation/approximation using triangular splines

Triangular Bezier Surface



Recursive Evaluation





Control points (Cubic)

$$\mathbf{p}_{0,3,0}$$

$$\mathbf{p}_{0,2,1} \quad \mathbf{p}_{1,2,0}$$

$$\mathbf{p}_{0,1,2} \quad \mathbf{p}_{1,1,1} \quad \mathbf{p}_{2,1,0}$$

$$\mathbf{p}_{0,0,3} \quad \mathbf{p}_{1,0,2} \quad \mathbf{p}_{2,0,1} \quad \mathbf{p}_{3,0,0}$$

Basis Functions (Cubic)

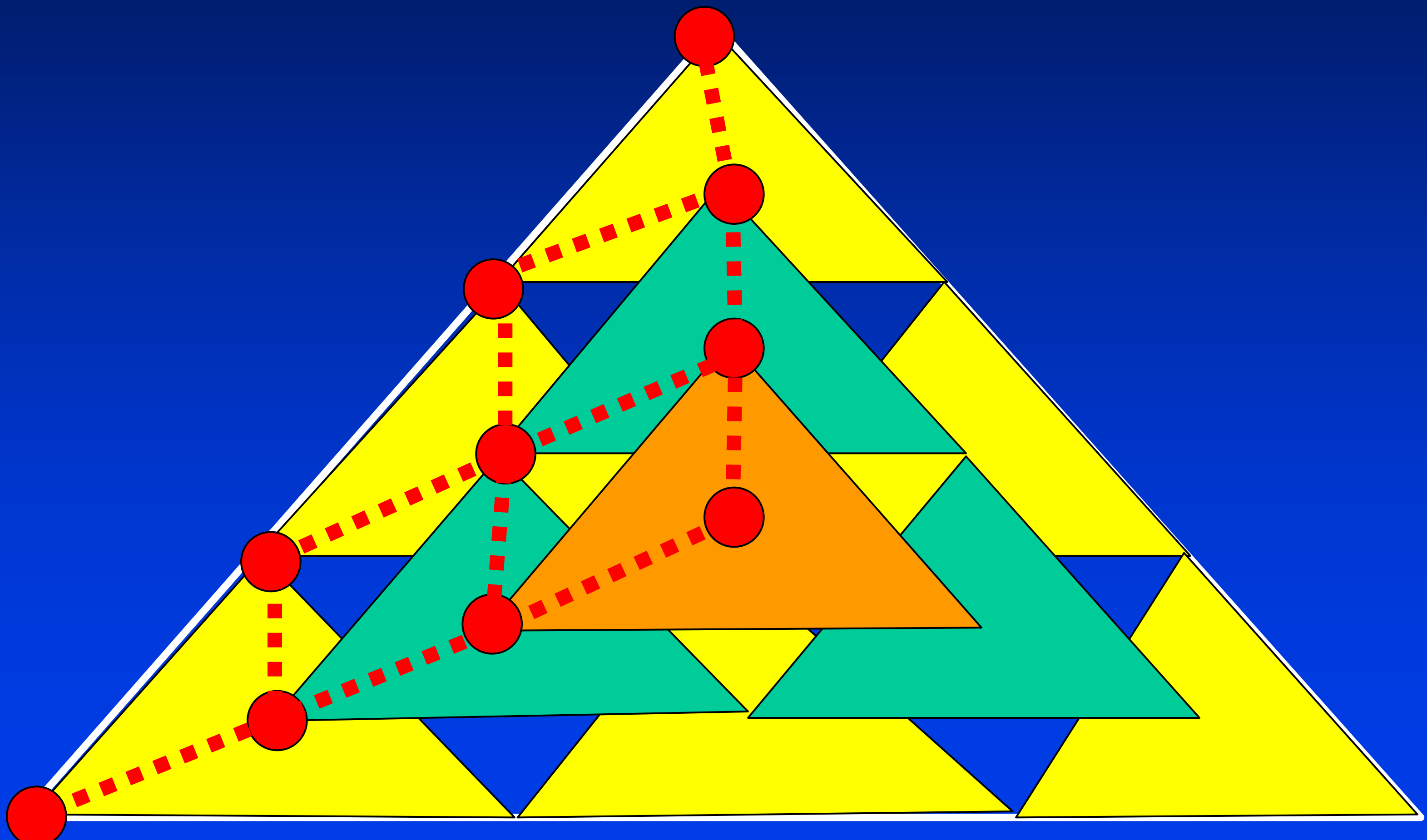
$$sss$$

$$3sst \quad 3rss$$

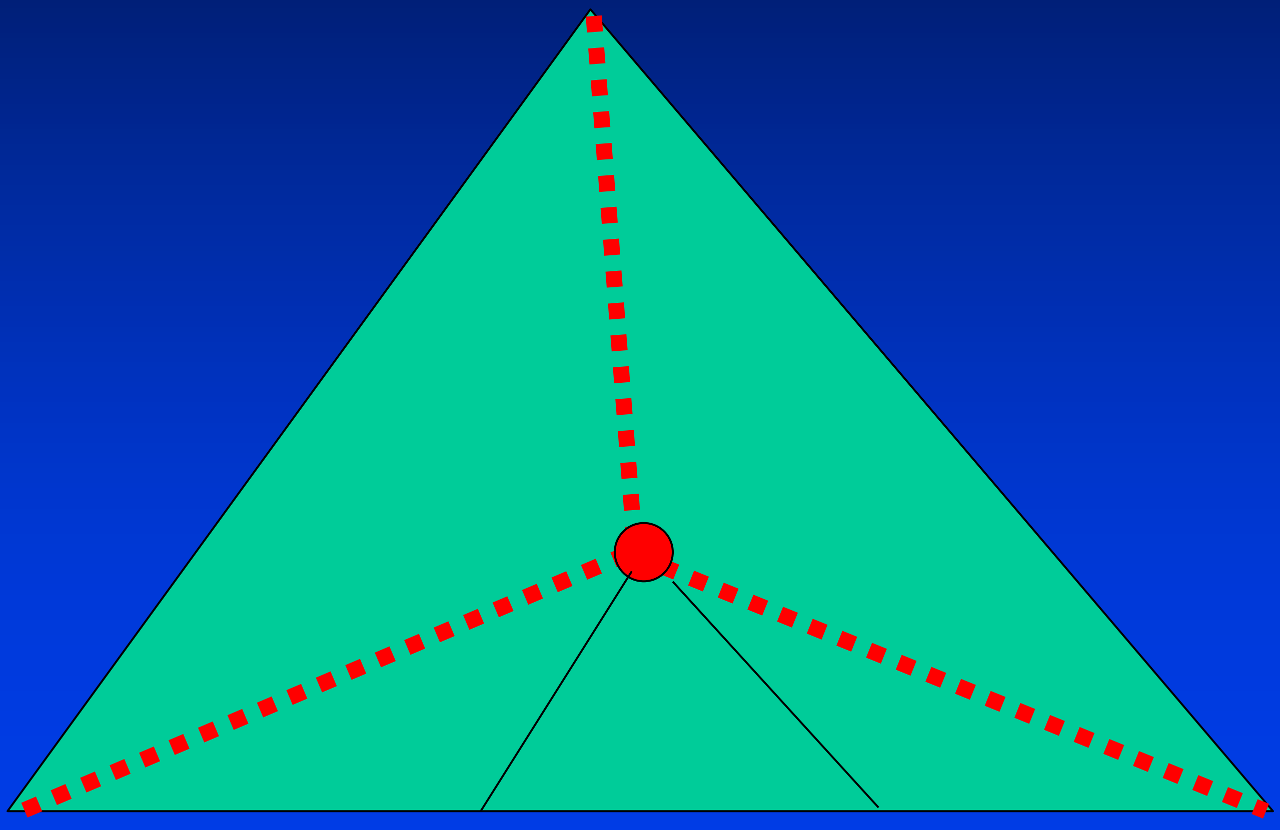
$$3stt \quad 6rst \quad 3rrs$$

$$ttt \quad 3rtt \quad 3rrt \quad rrr$$

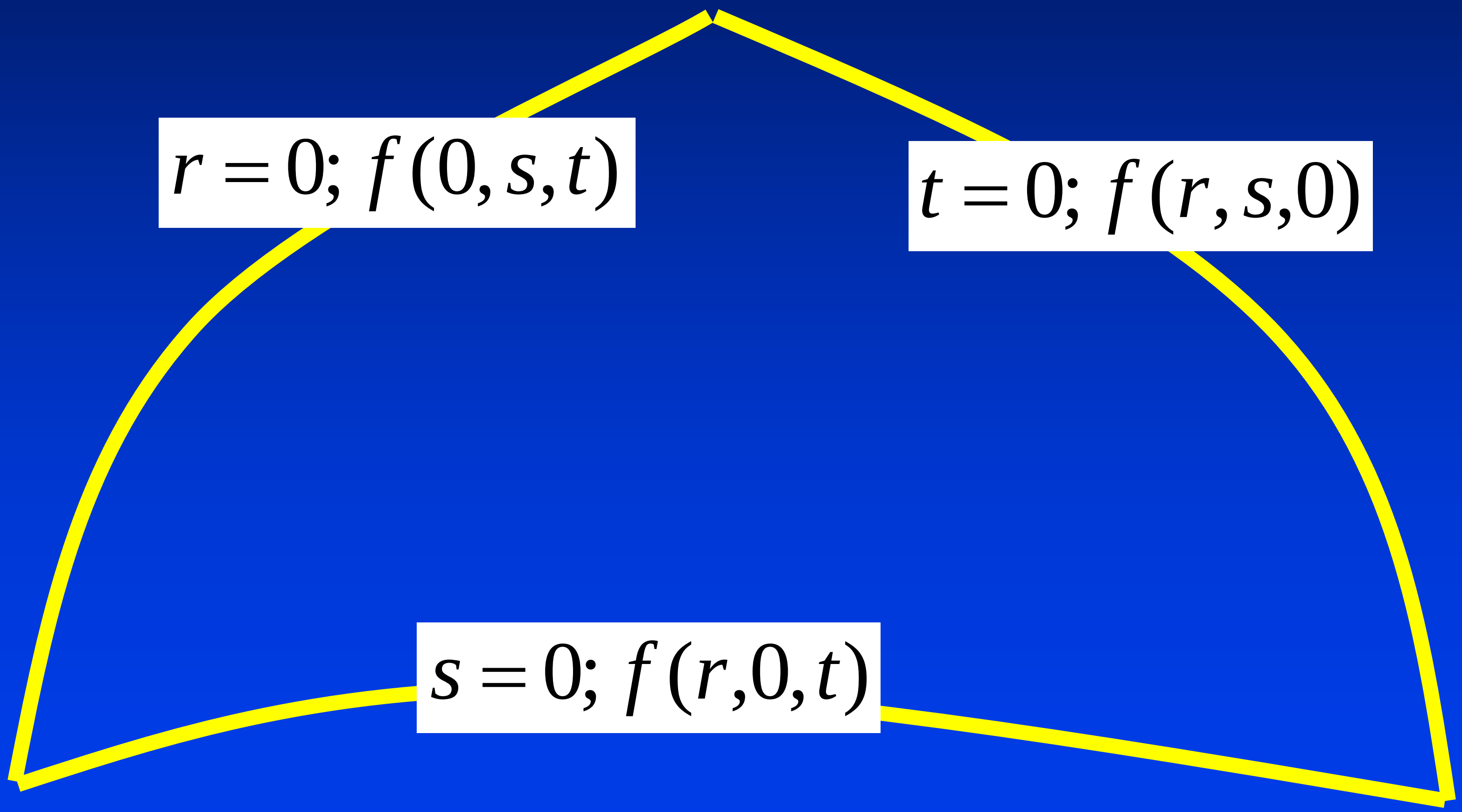
Triangular Patch Subdivision



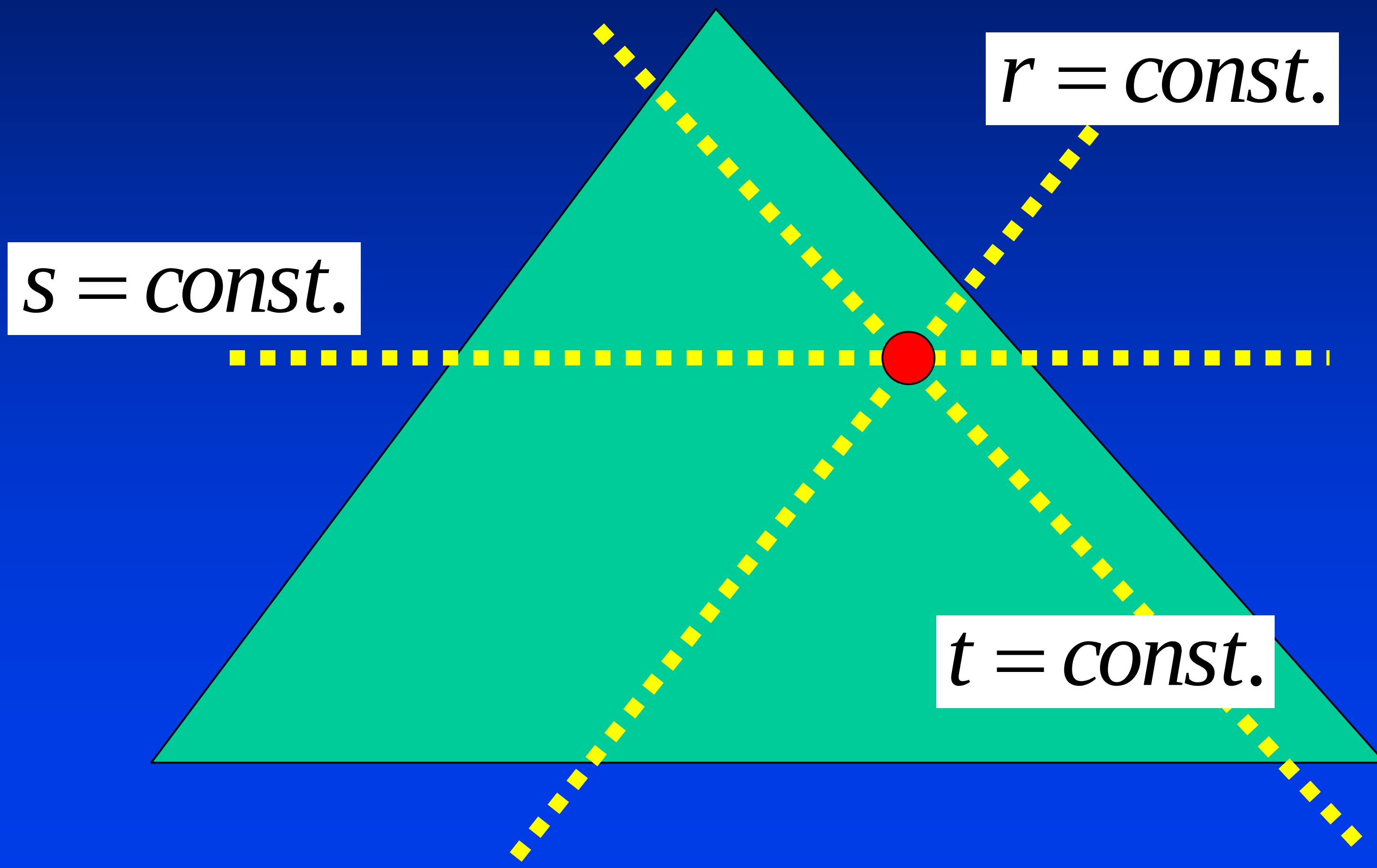
Triangular Domain



Triangular Coons-Gordon Surface



Triangular Coons-Gordon Surface



Triangular Interpolation

$$(P_1)\mathbf{f} = \mathbf{f}(r,0,t)L_0^1(\alpha) + \mathbf{f}(r,s,0)L_1^1(\alpha)$$

$$\alpha = \frac{s}{s+t}$$

$$(P_2)\mathbf{f} = \mathbf{f}(r,s,0)L_0^1(\beta) + \mathbf{f}(0,s,t)L_1^1(\beta)$$

$$\alpha = \frac{r}{r+t}$$

$$(P_3)\mathbf{f} = \mathbf{f}(0,s,t)L_0^1(\gamma) + \mathbf{f}(r,0,t)L_1^1(\gamma)$$

$$\alpha = \frac{r}{r+s}$$

Triangular Interpolation

- The Boolean sum of any two operators results the same!

$$(P_{12})\mathbf{f} = (P_1 \oplus P_2)\mathbf{f}$$
$$(P_{13})\mathbf{f} = (P_1 \oplus P_3)\mathbf{f}$$
$$(P_{23})\mathbf{f} = (P_2 \oplus P_3)\mathbf{f}$$

- Use cubic blending functions for C1 interpolation!

$$(Q_1)\mathbf{f} = \mathbf{f}(r,0,t)H_0^3(\alpha) + D_\alpha \mathbf{f}(r,0,t)H_1^3(\alpha) + D_\alpha \mathbf{f}(r,s,0)H_2^3(\alpha) + \mathbf{f}(r,s,0)H_3^3(\alpha)$$
$$(Q_2)\mathbf{f} = \ldots\ldots\ldots$$
$$(Q_3)\mathbf{f} = \ldots\ldots\ldots$$

Gregory's Method

- Convex combination

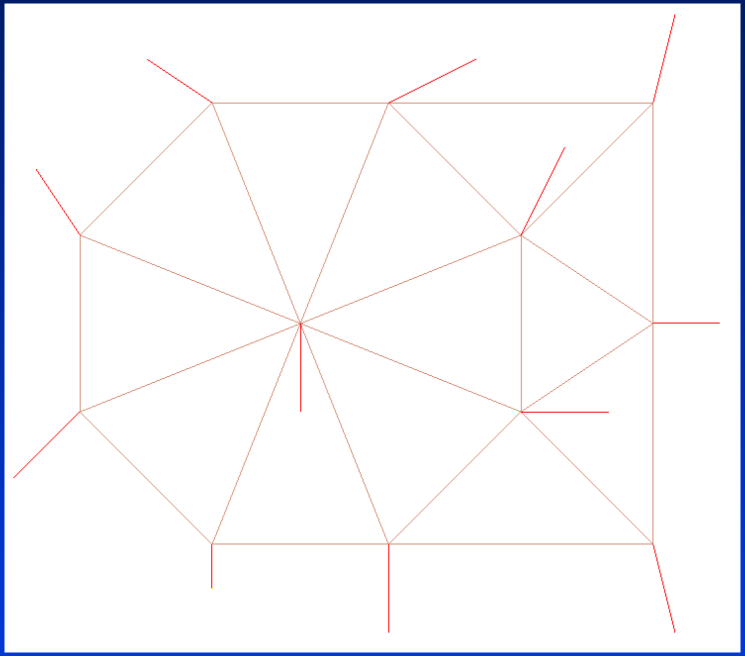
$$
\begin{aligned}
(T_1)\mathbf{f} &= \mathbf{f}(r,0,t) + \alpha D_\alpha \mathbf{f}(r,0,t) \\
(T_2)\mathbf{f} &= \ldots\ldots\ldots \\
(T_3)\mathbf{f} &= \ldots\ldots\ldots \\
(T_{12})\mathbf{f} &= (T_1 \oplus T_2)\mathbf{f} \\
(T_{13})\mathbf{f} &= (T_1 \oplus T_3)\mathbf{f} \\
(T_{23})\mathbf{f} &= (T_2 \oplus T_3)\mathbf{f} \\
(T)\mathbf{f} &= (a_1 T_{23} + a_2 T_{13} + a_3 T_{12})\mathbf{f} \\
a_1 &= \frac{s^2}{r^2 + s^2 + t^2} \\
a_2 &= \ldots\ldots\ldots \\
a_3 &= \ldots\ldots\ldots
\end{aligned}
$$

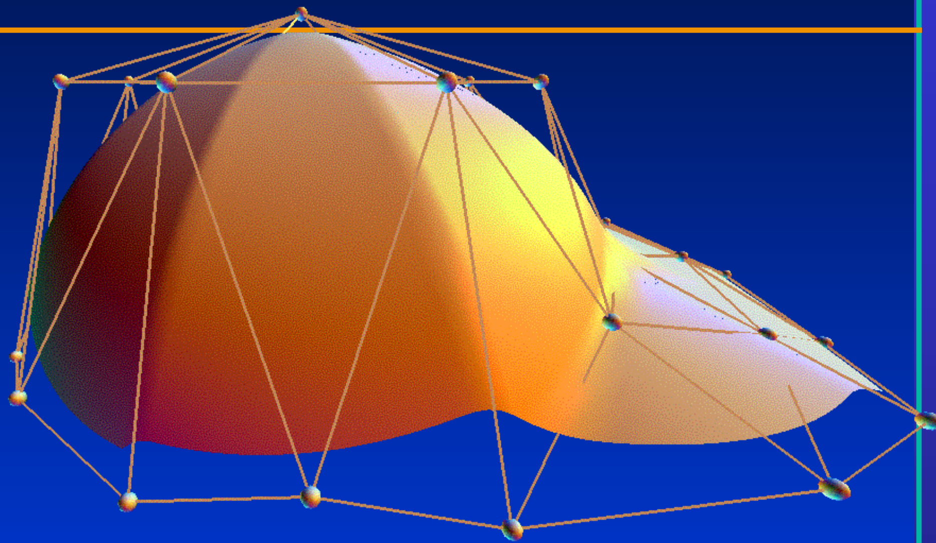
- Generalize to pentagonal patch!

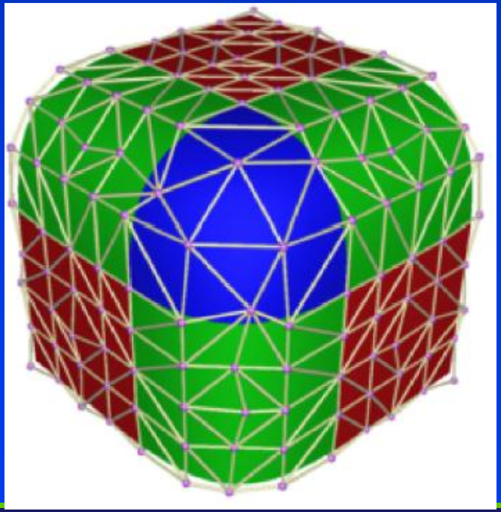
Surface Properties

- Inherit from their curve generators
- More!
- Efficient algorithms
- Continuity across boundaries
- Interpolation and approximation tools

Triangular B-splines



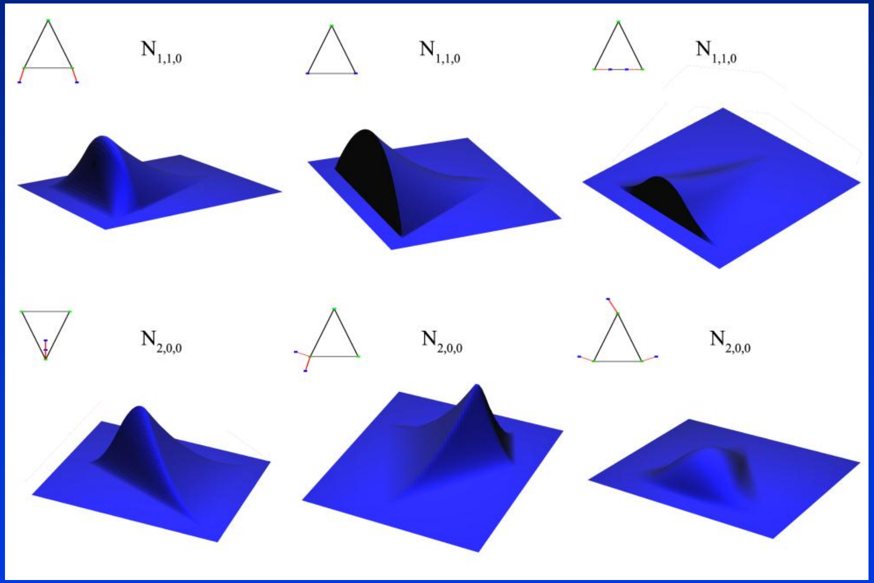




Simplex Spline Basis Functions

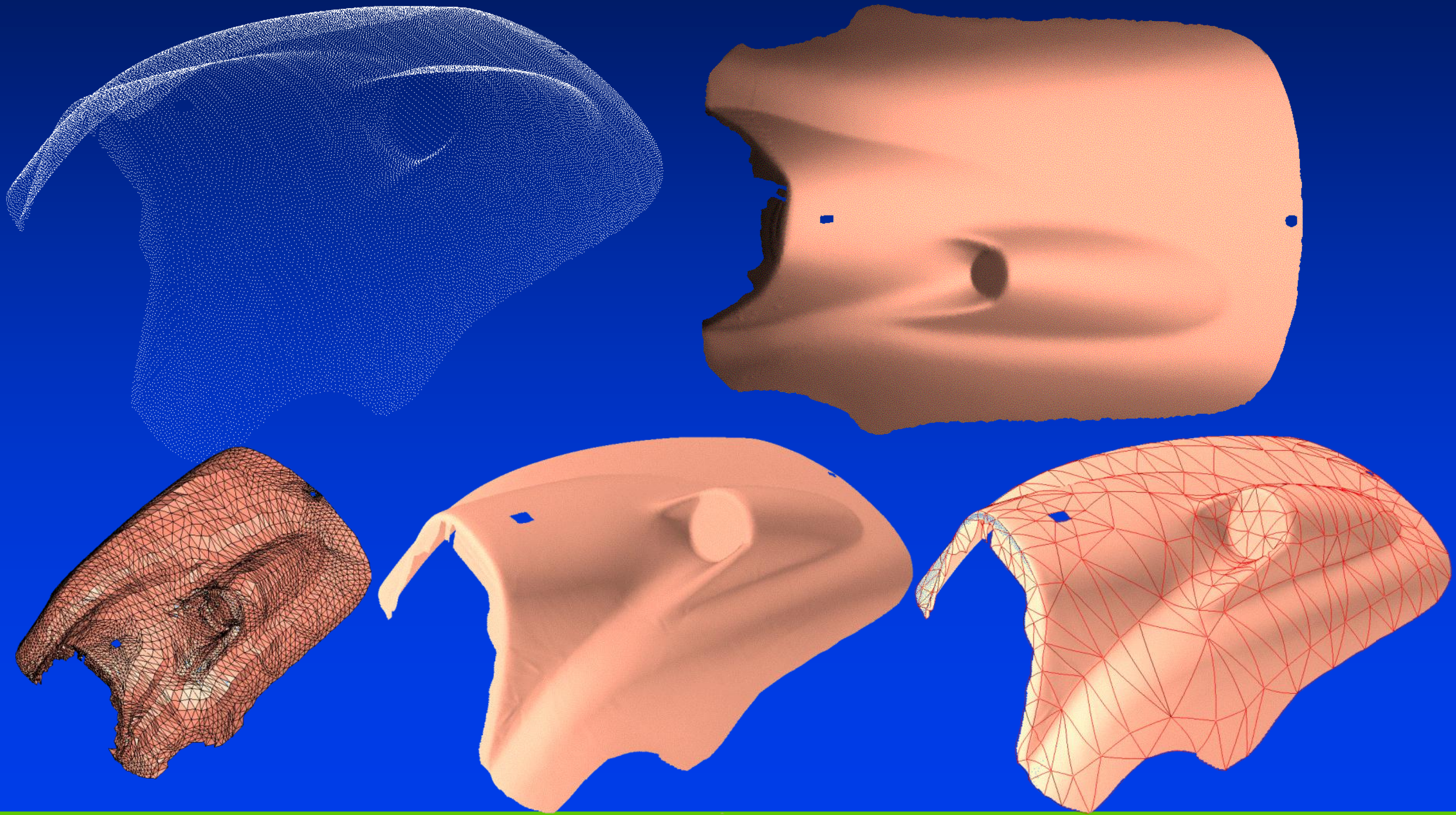
- Multivariate Simplex Splines
  - Defined by projection of a simplex (in dimension *n*) into lower dimension

$$N_S(\mathbf{w}) = \frac{(n-m)!}{n!}\frac{vol(\pi^{-1}(\mathbf{w}) \cap S)}{volU(\mathbf{w})}$$



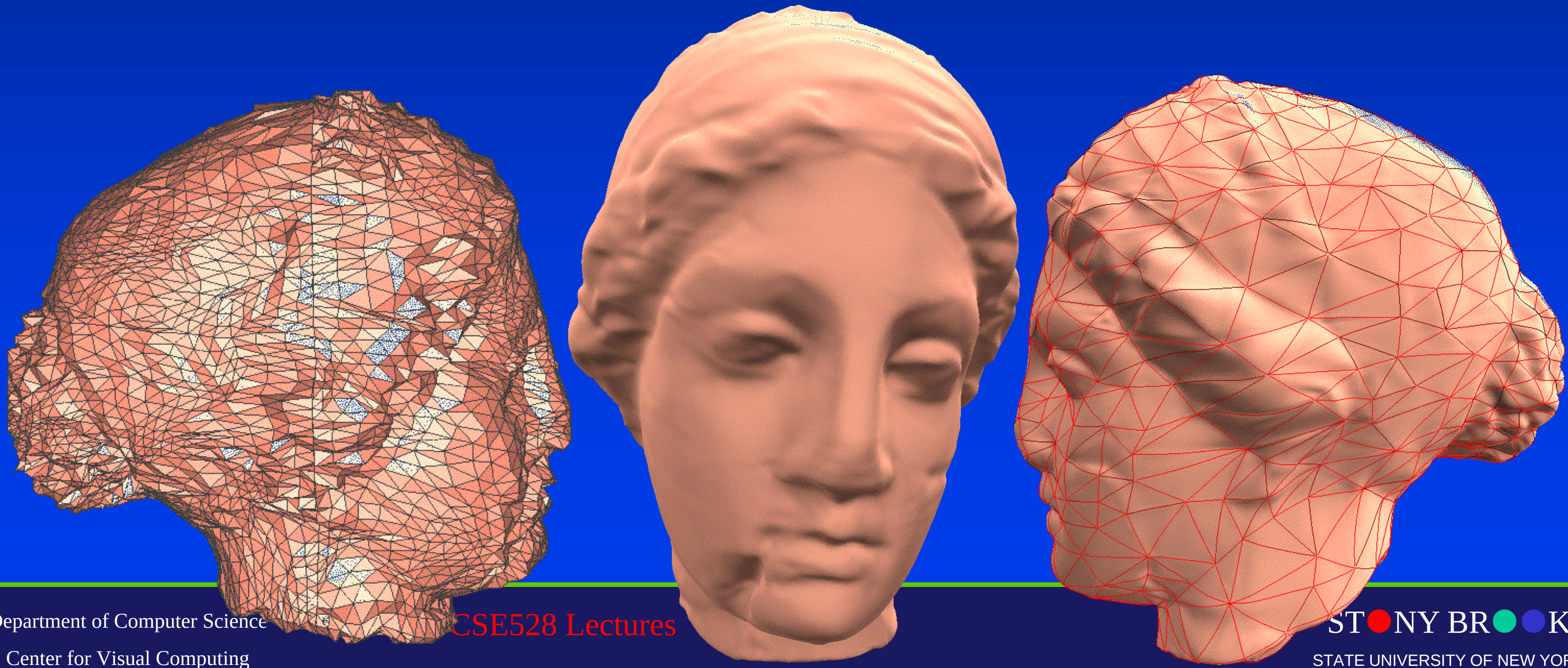
  - Recursive definition

Reverse Engineering (from Points to Splines)



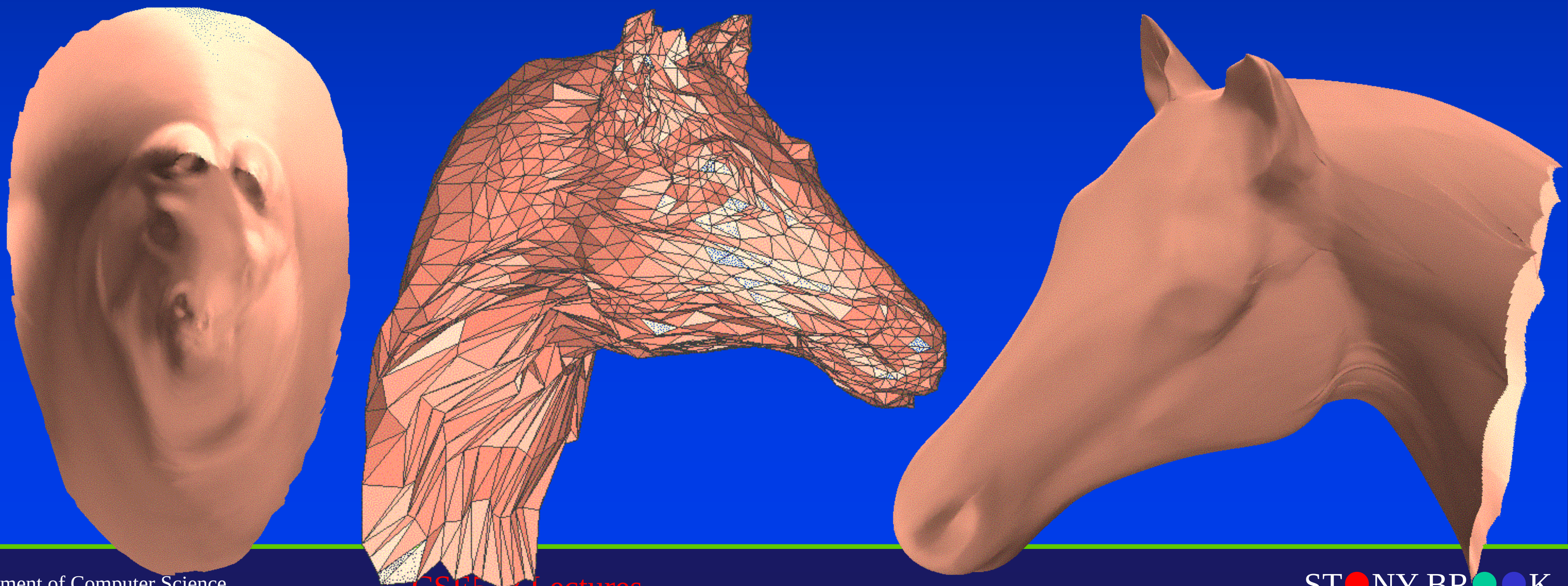
Another Example

- Venus model: 50,002 points (parameterization data courtesy of Hugues Hoppe)
- C^2 surface:
  - max error 0.64%, mean-square-root error 0.097%
  - 4,381 control points, 1,668 knots, 1,055 domain triangles

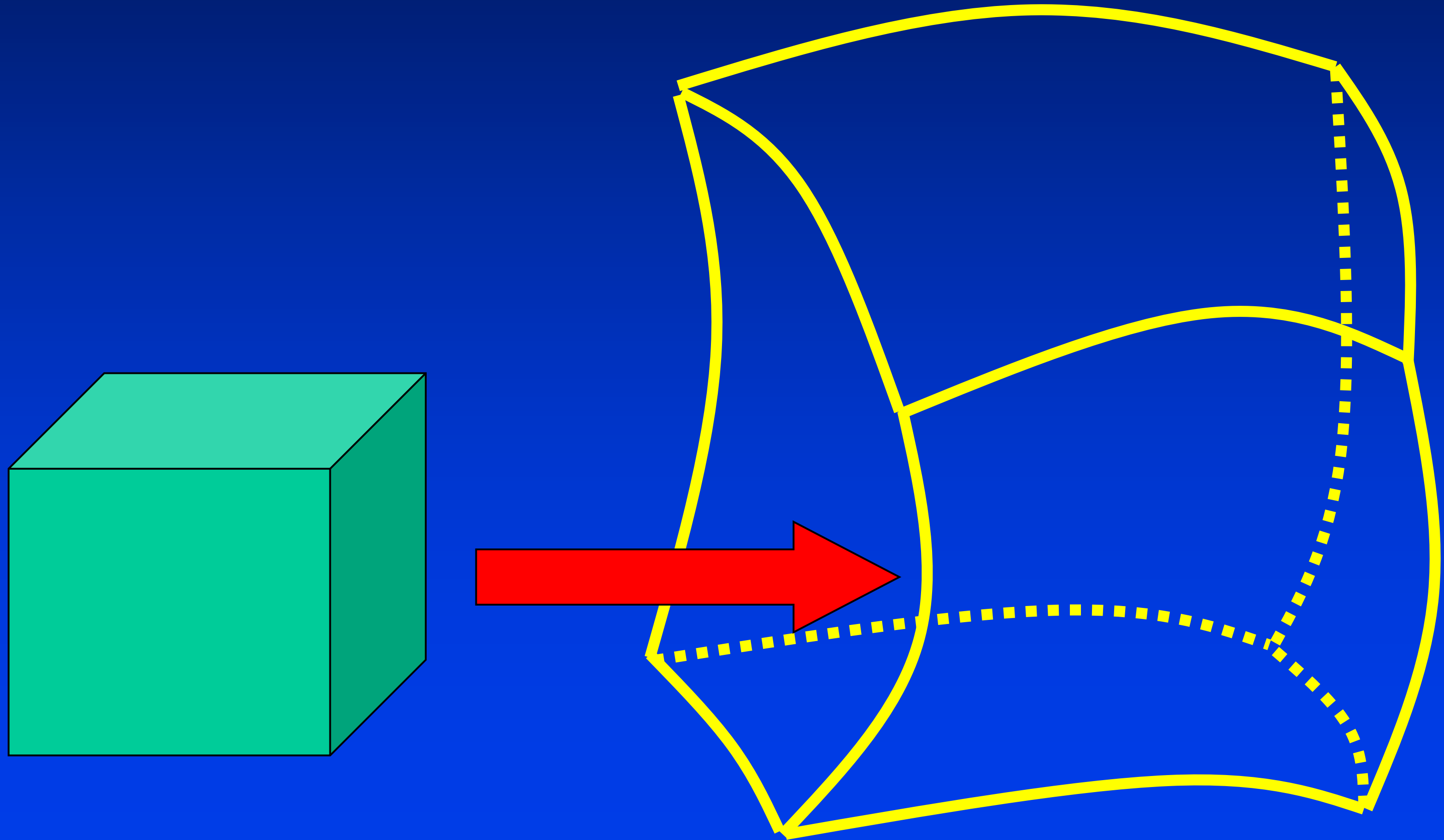


Horse Example

- Horse head model: 24,236 points after up-sampling (parameterization data courtesy of Hugues Hoppe)
- C^2 surface:
  - max error 1.04%, mean-square-root error 0.19%
  - 1,663 control points, 573 knots, 364 domain triangles



Solid



Parametric Solids

- Tricubic solid

$$\mathbf{p}(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\mathbf{a}_{ijk}u^i v^j w^k$$
$$u,v,w\in[0,1]$$

- Bezier solid

$$\mathbf{p}(u,v,w)=\sum_{i}\sum_{j}\sum_{k}\mathbf{p}_{ijk}B_i(u)B_j(v)B_k(w)$$

- B-spline solid

$$\mathbf{p}(u,v,w)=\sum_{i}\sum_{j}\sum_{k}\mathbf{p}_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)$$

- NURBS solid

$$\mathbf{p}(u,v,w)=\frac{\sum_{i}\sum_{j}\sum_{k}\mathbf{p}_{ijk}q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}{\sum_{i}\sum_{j}\sum_{k}q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}$$

Parametric Solids

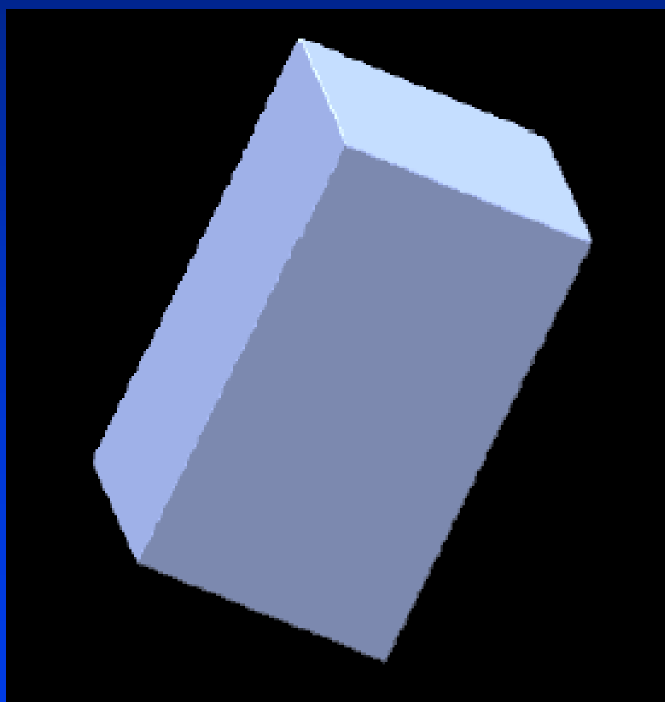
- Tricubic Hermite solid
- In general

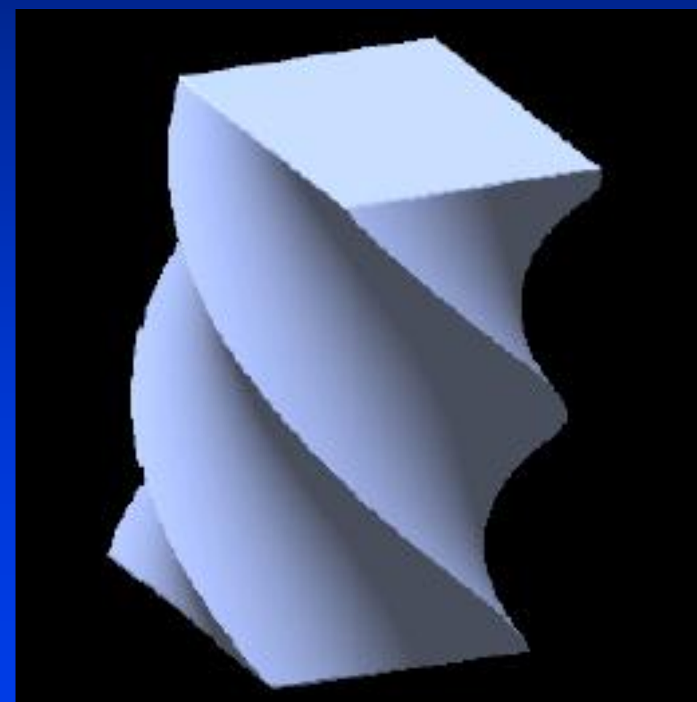
$$\mathbf{p}(u,v,w) = \begin{bmatrix} x(u,v,w) \\ y(u,v,w) \\ z(u,v,w) \end{bmatrix}$$

$$u,v,w \in [0,1]$$

- Also known as "hyperpatch"
- Parametric solids represent both exterior and interior
- Examples
  – A rectangular sold, a trilinear solid
- Boundary elements
  – 8 corner points, 12 curved edges, and 6 curved faces

Free-Form Deformation





Free-form Deformation

- Any geometric objects can be embedded into a space
- The surrounding space is represented by using commonly-used, popular splines
- Free-form deformation of the surrounding space
- All the embedded (geometric) objects are deformed accordingly, the quantitative measurement of deformation is obtained from the displacement vectors of the trivariate splines that define the surrounding space
- Essentially, the deformation is governed by the trivariate, volumetric splines
- Very popular in graphics and related fields

Surrounding Space represented by Parametric Solids

- Tricubic solid

$$\mathbf{p}(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\mathbf{a}_{ijk}u^i v^j w^k$$

$$u,v,w\in[0,1]$$

- Bezier solid

$$\mathbf{p}(u,v,w)=\sum_i\sum_j\sum_k \mathbf{p}_{ijk}B_i(u)B_j(v)B_k(w)$$

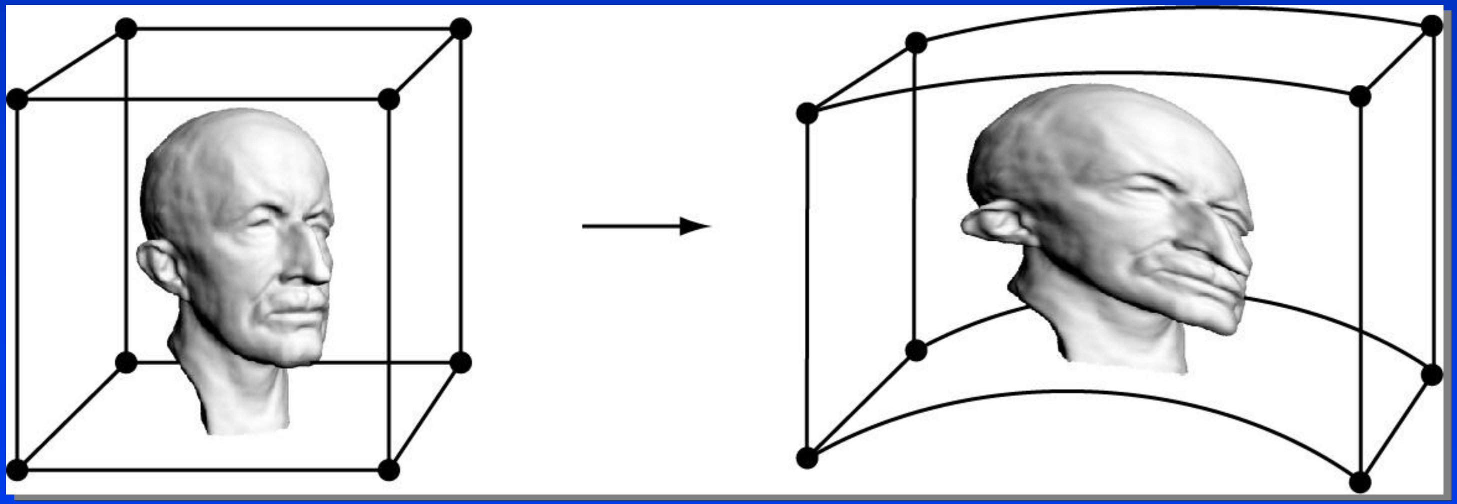
- B-spline solid

$$\mathbf{p}(u,v,w)=\sum_i\sum_j\sum_k \mathbf{p}_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)$$

- NURBS solid

$$\mathbf{p}(u,v,w)=\frac{\sum_i\sum_j\sum_k \mathbf{p}_{ijk}q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}{\sum_i\sum_j\sum_k q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}$$

Free-form Deformations



(courtesy of Pauly et al.)

Curves, Surfaces, and Solids

- Non-isoparametric curves for surfaces

$$\mathbf{s}(u,v)$$
$$\mathbf{c}(t)=\begin{bmatrix}u(t)\\v(t)\end{bmatrix}$$
$$\mathbf{s}(u(t),v(t))$$

- Non-isoparametric curves for solids

$$\mathbf{s}(u,v,w)$$
$$\mathbf{c}(t)=\begin{bmatrix}u(t)\\v(t)\\w(t)\end{bmatrix}$$
$$\mathbf{s}(u(t),v(t),w(t))$$

- Non-isoparametric surfaces for solids

$$\mathbf{s}(u,v,w)=\mathbf{s}(u(a,b),v(a,b),w(a,b))$$

Curves, Surfaces, and Solids

- Isoparametric curves for surfaces

$$\mathbf{s}(u,v), \mathbf{s}(u_i,v), \mathbf{s}(u,v_j)$$
$$u_i = const.; v_j = const.$$

- Isoparametric curves for solids

$$\mathbf{s}(u,v,w), \mathbf{s}(u_i,v_j,w), \mathbf{s}(u_i,v,w_k), \mathbf{s}(u,v_j,w_k)$$

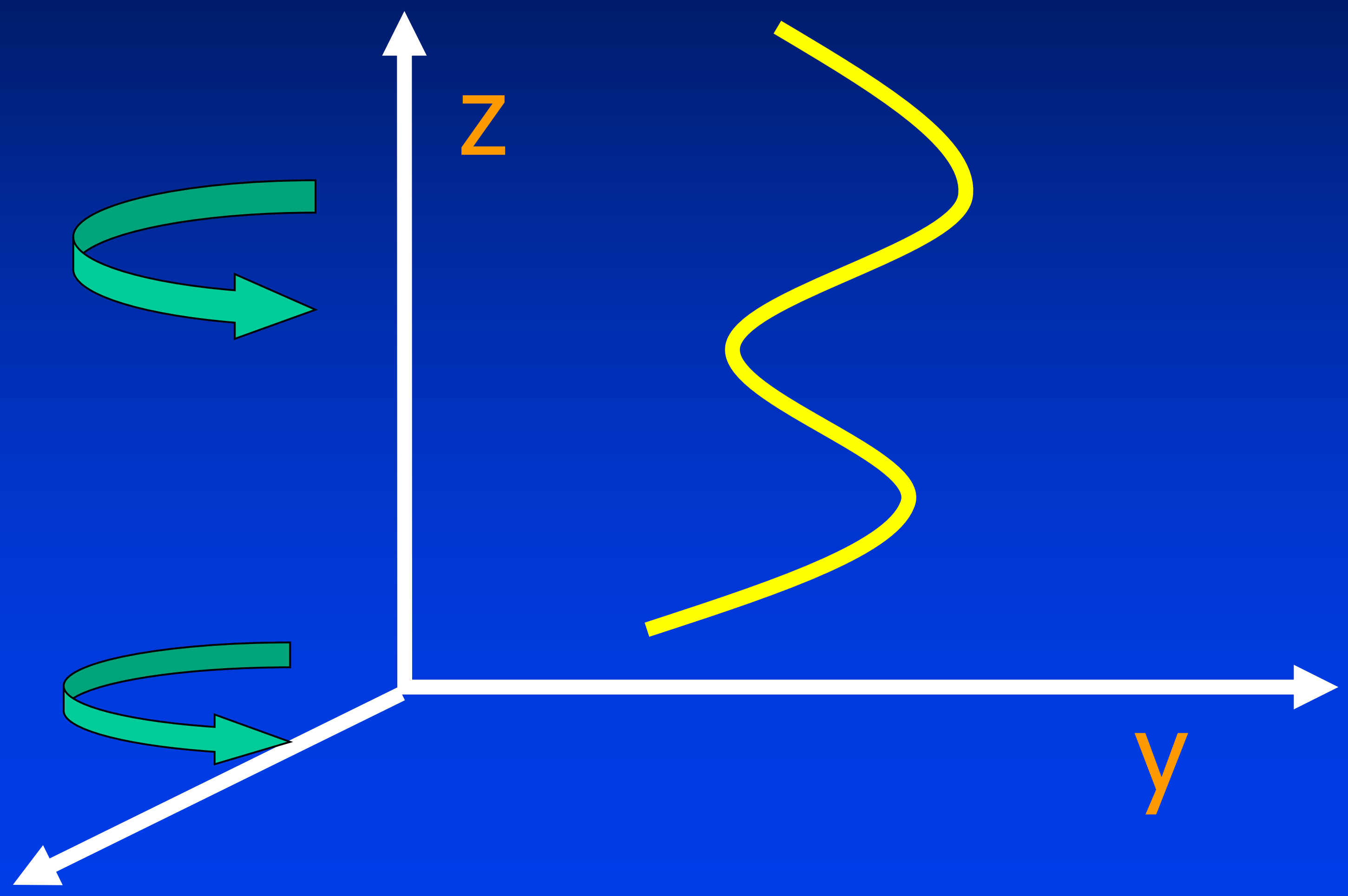
- Isoparametric surfaces for solids

$$\mathbf{s}(u,v,w), \mathbf{s}(u_i,v,w), \mathbf{s}(u,v_j,w), \mathbf{s}(u,v,w_k)$$

Solid Modeling

- Create unambiguous and complete geometric representation of object
  - B-reps (Boundary representations)
  - Spatial partition
  - Volumetric (Arie Kaufman)
  - CSG (Constructive Solid Geometry, popular in mechanics design)

Surface of Revolution



Surfaces of Revolution

- Geometric construction
  - Specify a planar curve profile on y-z plane
  - Rotate this profile with respect to z-axis
- Procedure-based model
- What kinds of shape can we model?
- Review: three dimensional rotation w.r.t. z-axis

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Surfaces of Revolution

- Mathematics: surfaces of revolution

$$\mathbf{c}(u) = \begin{bmatrix} 0 \\ y(u) \\ z(u) \end{bmatrix}$$

$$\mathbf{s}(u,v) = \begin{bmatrix} -y(u)\sin(v) \\ y(u)\cos(v) \\ z(u) \end{bmatrix}$$

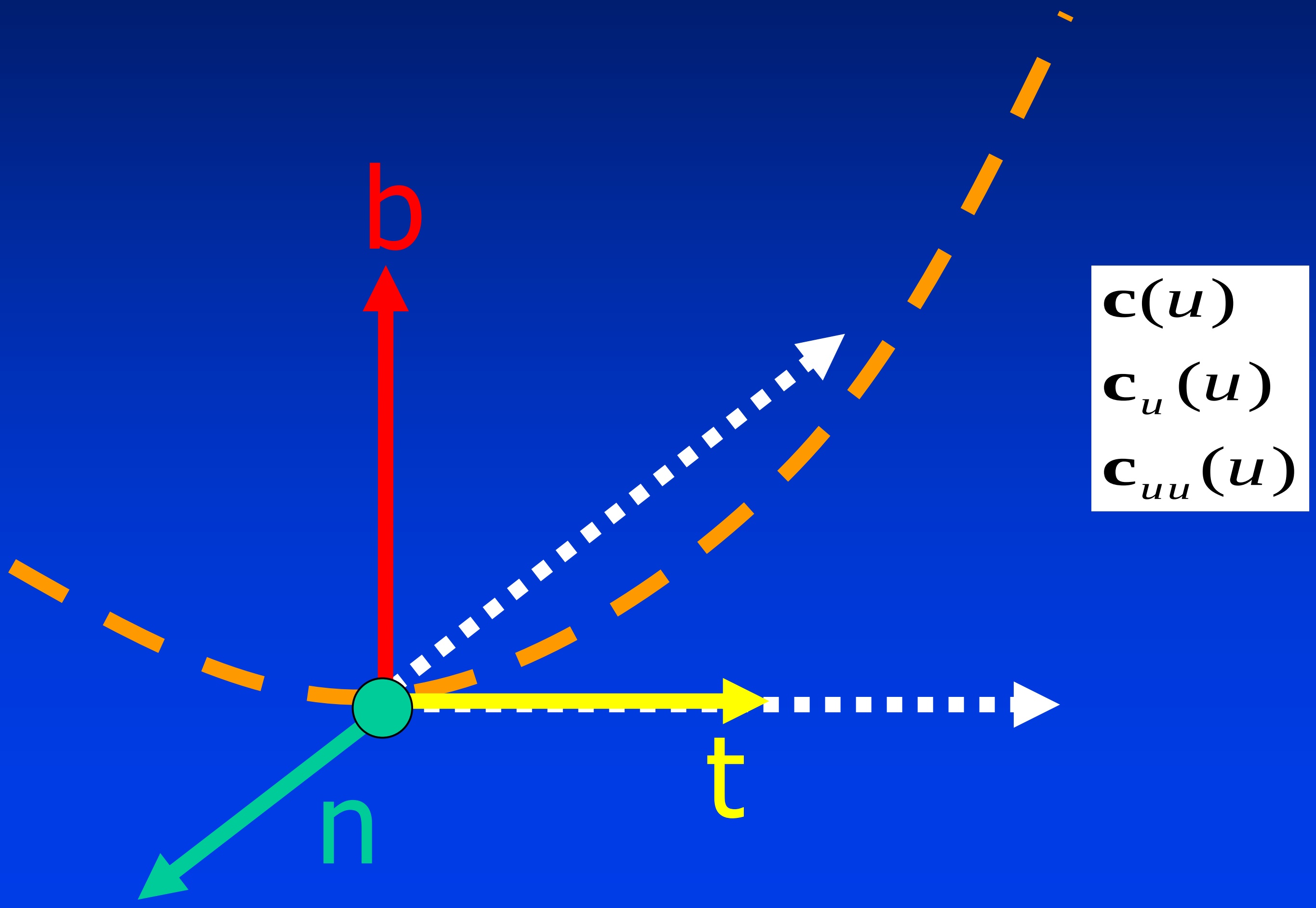
Frenet Frames

- Motivation: attach a smoothly-varying coordinate system to any location of a curve
- Three independent direction vectors for a 3D coordinate system: (1) tangent; (2) bi-normal; (3) normal

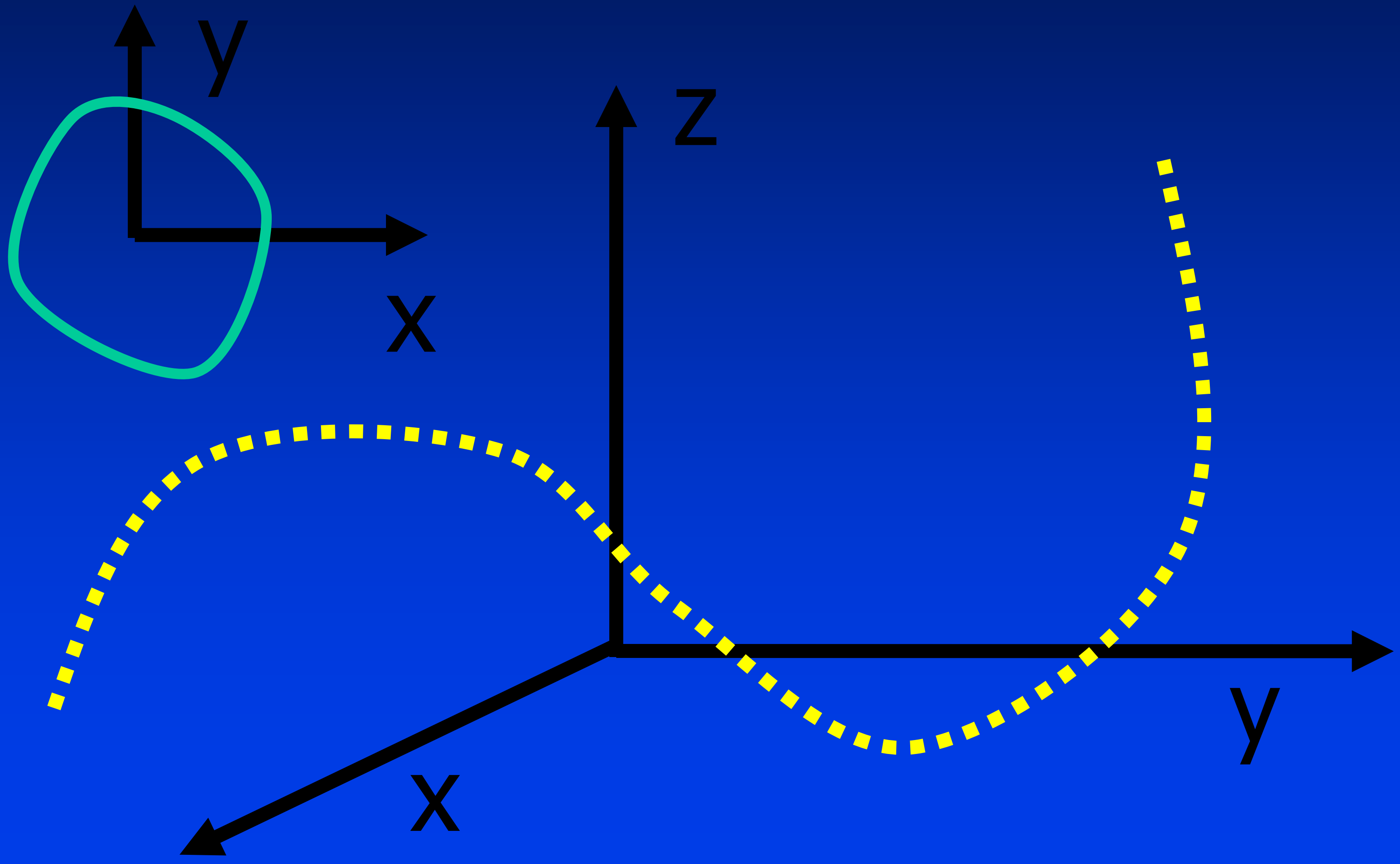
$$\mathbf{t}(u) = normalize(\mathbf{c}_u(u))$$
$$\mathbf{b}(u) = normalize(\mathbf{c}_u(u) \times \mathbf{c}_{uu}(u))$$
$$\mathbf{n}(u) = normalize(\mathbf{b}(u) \times \mathbf{t}(u))$$

- Frenet coordinate system (frame) (t,b,n) varies smoothly, as we move along the curve c(u)

Frenet Coordinate System



Sweeping Surface



General Sweeping Surfaces

- Surface of revolution is a special case of a sweeping surface
- Idea: a profile curve and a trajectory curve

$$\mathbf{c}_1(u)$$
$$\mathbf{c}_2(v)$$

- Move a profile curve along a trajectory curve to generate a sweeping surface
- Question: how to orient the profile curve as it moves along the trajectory curve?
- Answer: various options

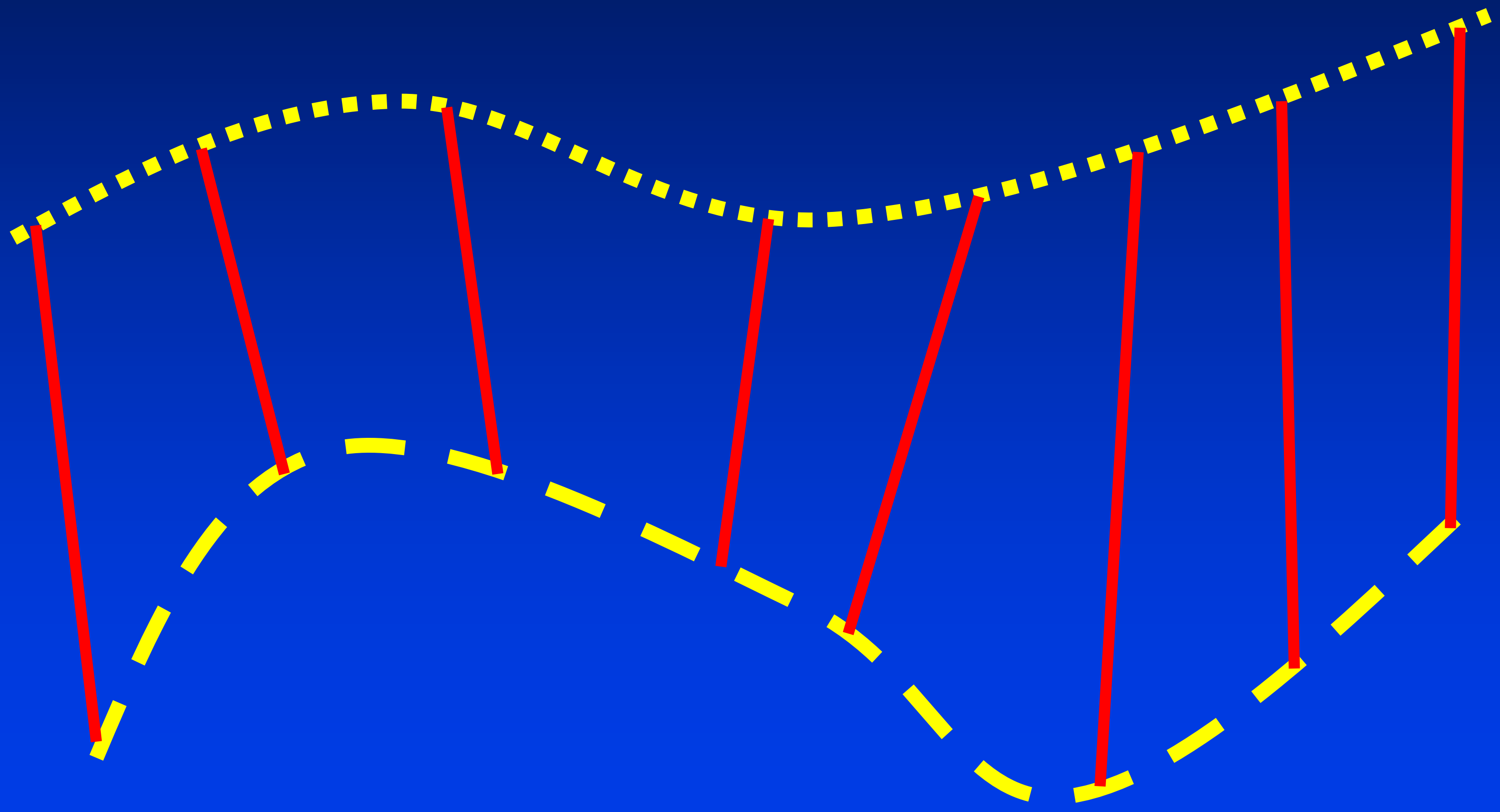
General Sweeping Surfaces

- Fixed orientation, simple translation of the coordinate system of the profile curve along the trajectory curve
- Rotation: if the trajectory curve is a circle
- Move using the “Frenet Frame” of the trajectory curve, smoothly varying orientation
- Example: surface of revolution
- Differential geometry fundamentals: Frenet frame

Frenet Swept Surfaces

- Orient the profile Curve (C1(u)) using the Frenet frame of C2(v)
  - Put C1(u) on the normal plane (n,b)
  - Place the original of C1(u) on C2(v)
  - Align the x-axis of C1(u) with –n
  - Align the y-axis of C1(u) with b
- Example: if C2(v) is a circle
- Variation (generalization)
- Scale C1(u) as it moves
- Morph C1(u) into C3(u) as it moves
- Use your own imagination!

Ruled surfaces

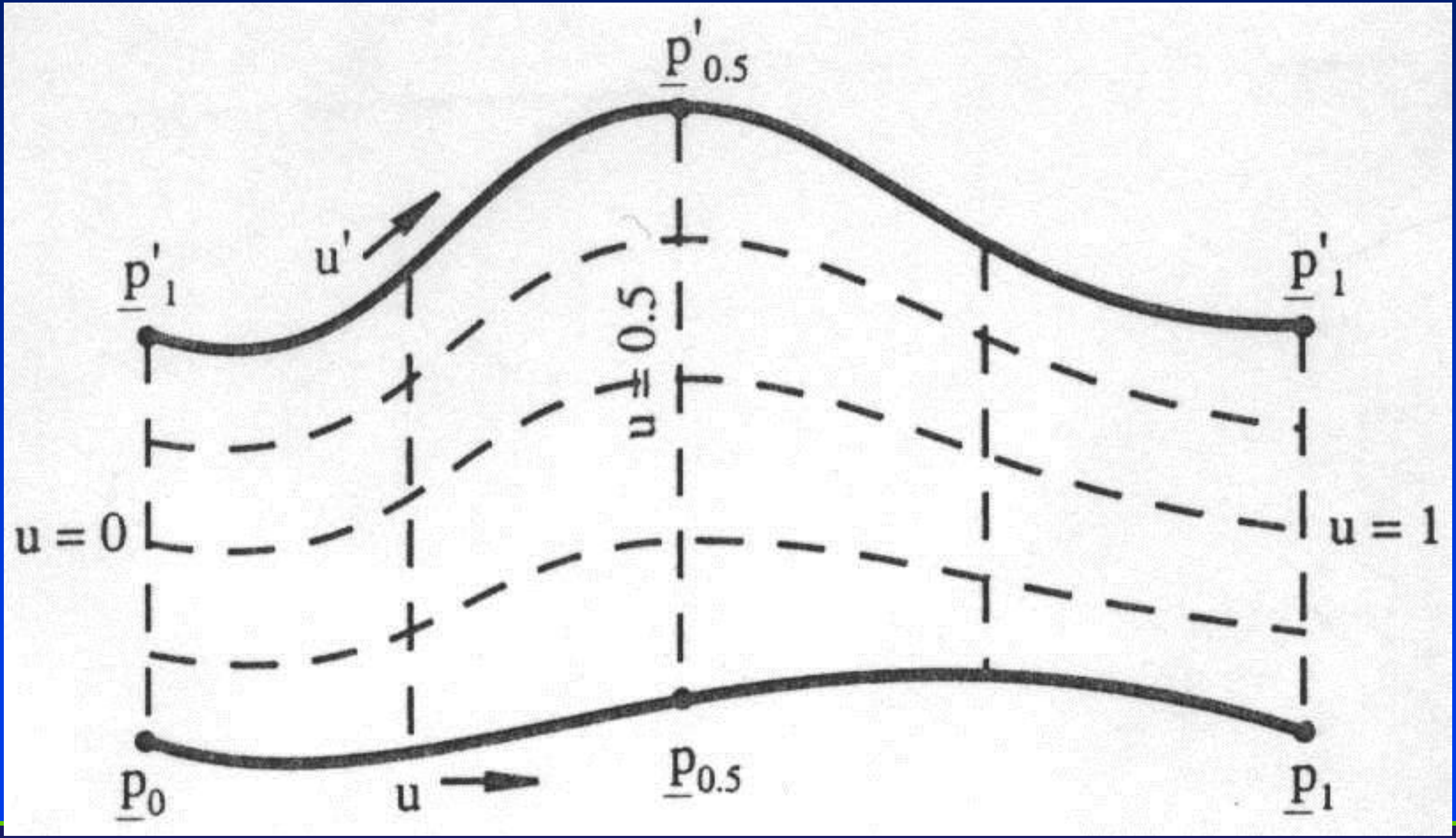


Ruled Surfaces

- Move one straight line along a curve, or join two parametric curves by straight lines
- Example: plane, cone, cylinder
- Cylindrical surface
- Surface equation

$$\mathbf{s}(u,v) = (1-v)\mathbf{a}(u) + v\mathbf{b}(u)$$
$$\mathbf{s}(u,v) = (1-v)\mathbf{s}(u,0) + v\mathbf{s}(u,1)$$
$$\mathbf{s}(u,v) = \mathbf{p}(u) + v\mathbf{q}(u)$$

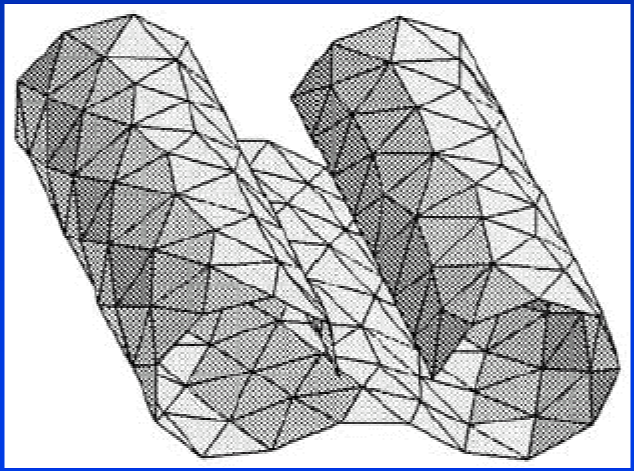
- Isoparametric lines
- Generalized cylinder
- Bending by roller

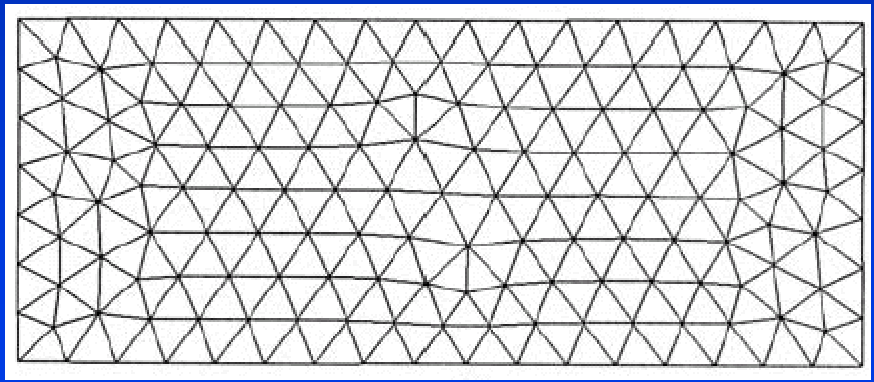

p'0.5
u'
p'1
p'1
u = 0.5
u = 0
u = 1
p0
u
p0.5
p1

Developable Surfaces

- Deform a surface to planar shape without length/area changes
- Unroll a surface to a plane without stretching/distorting
- Example: cone, cylinder
- Developable surfaces vs. Ruled surfaces
- More examples???

Developable Surface





Summary

- Parametric curves and surfaces
- Polynomials and rational polynomials
- Free-form curves and surfaces
- Other commonly-used geometric primitives (e.g., sphere, ellipsoid, torus, superquadrics, blobby, etc.)
- Motivation:
  – Fewer degrees of freedom
  – More geometric coverage

Surfaces

- Plane
- Quadratic surfaces
- Tensor product surfaces.
- Surfaces of revolution.
- Sweeping surfaces.
- Subdivision surfaces.