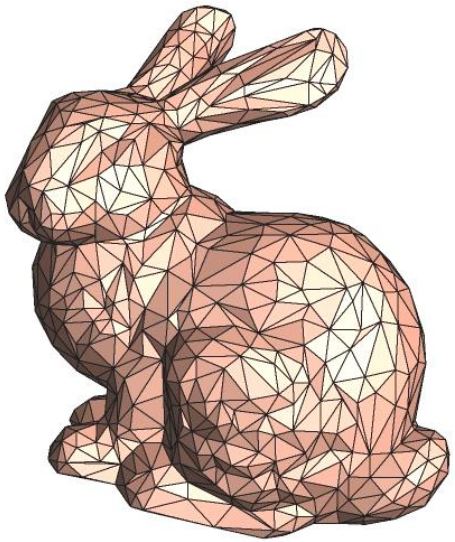
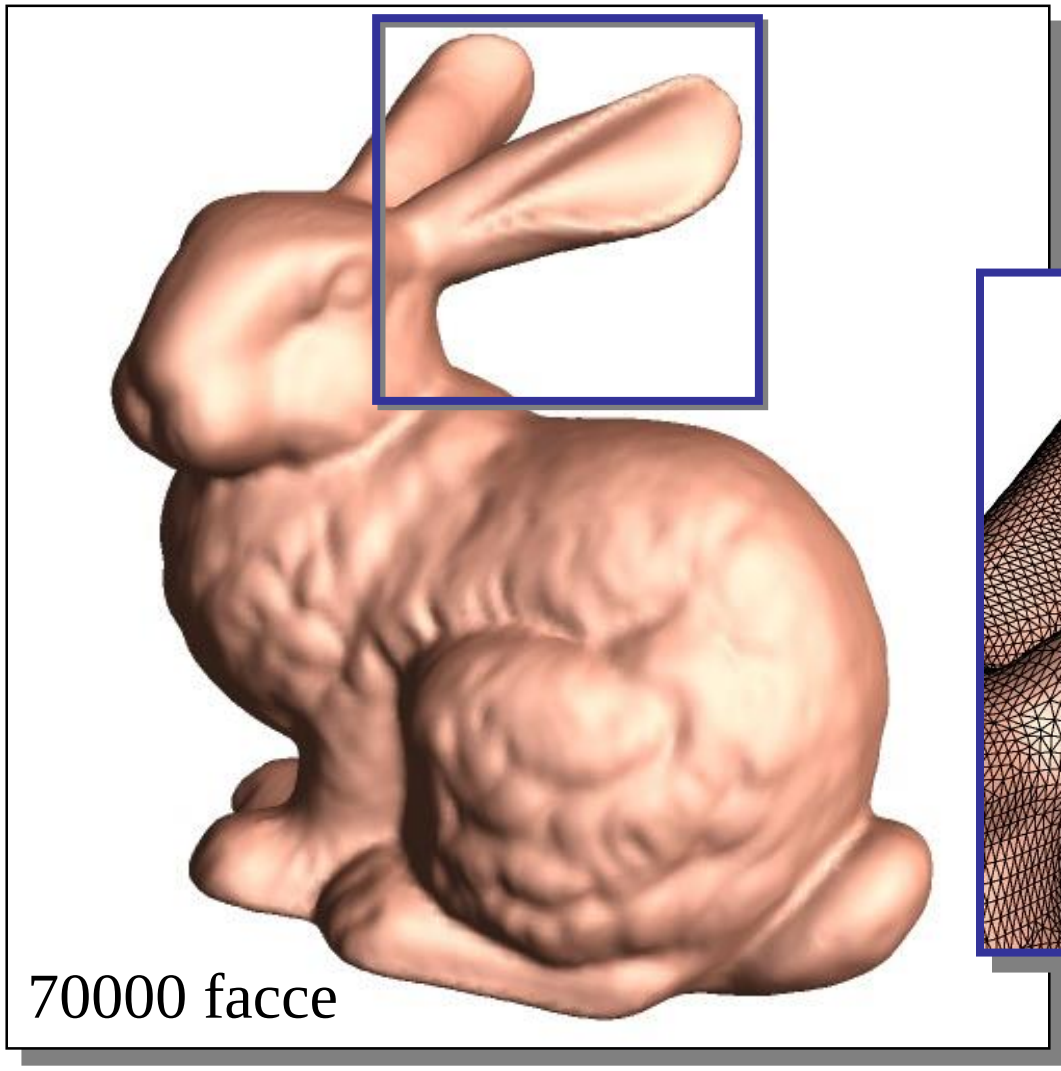
Geometric Modeling: surfaces

- **Patch:**
  - **patch di Bézier**
  - **NURBS surfaces**

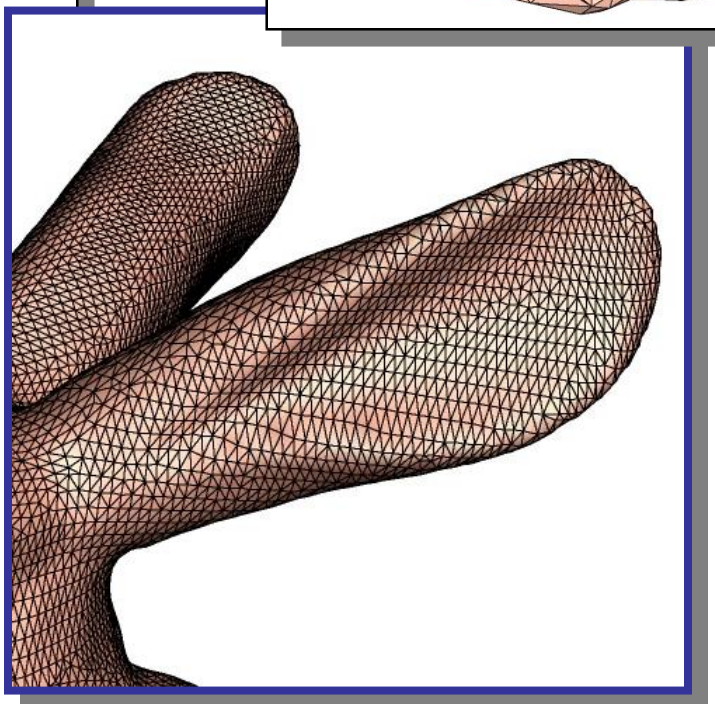
Polygonal Mesh:

piecewise planar surfaces

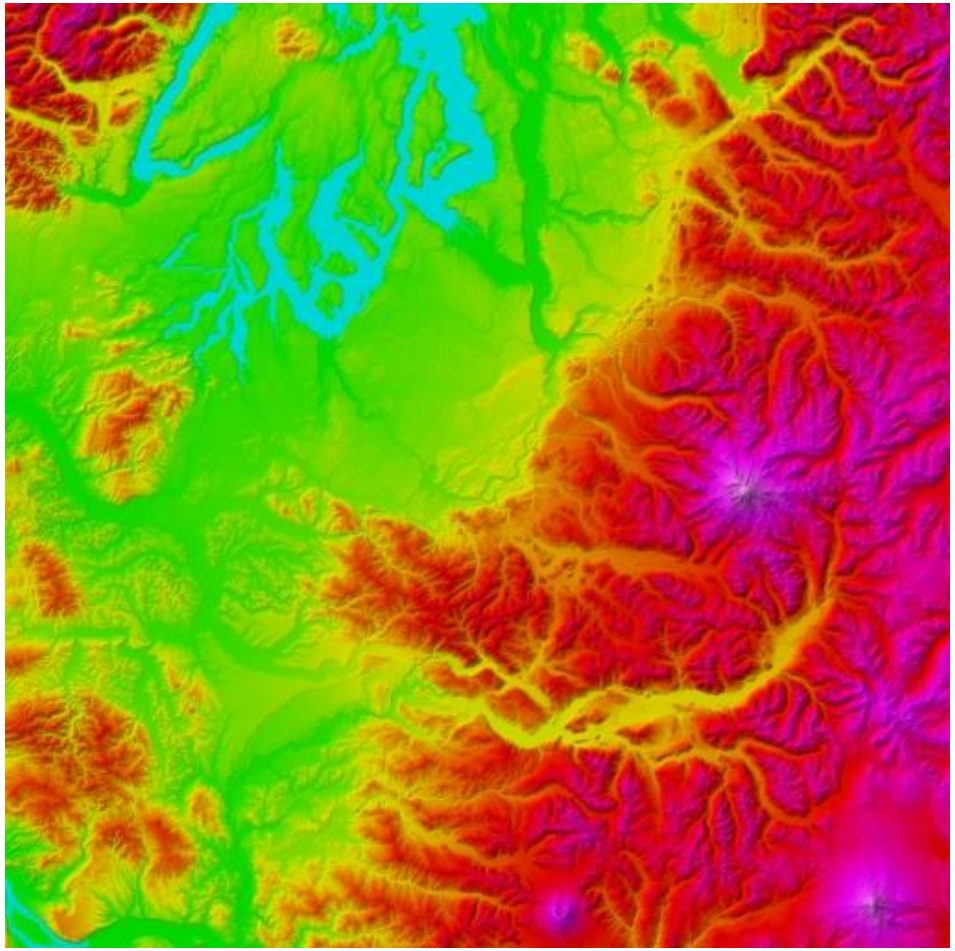


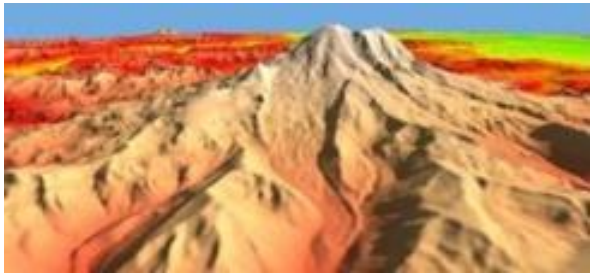


70000 facce



Polygonal Mesh:





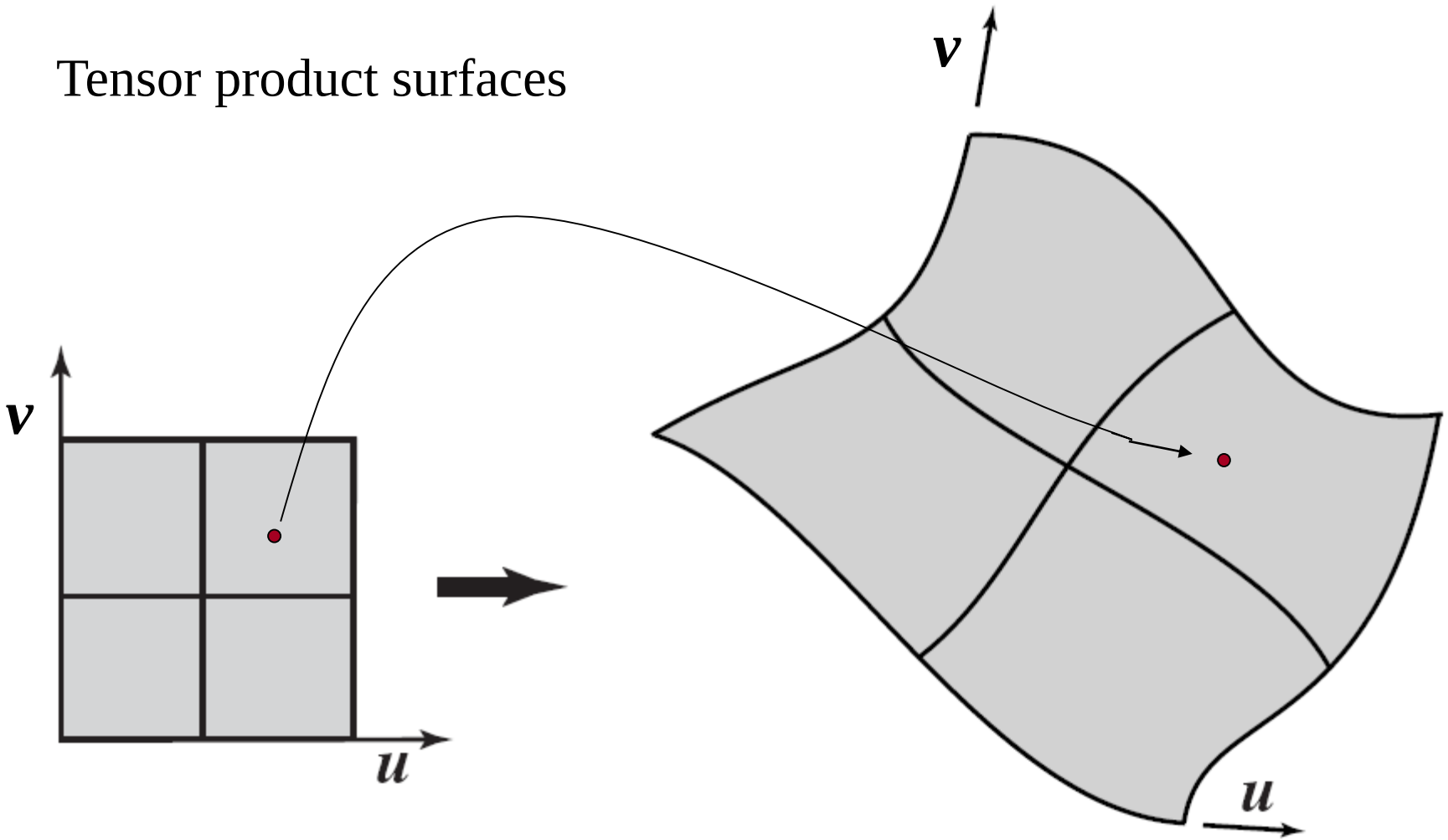
Territorial data

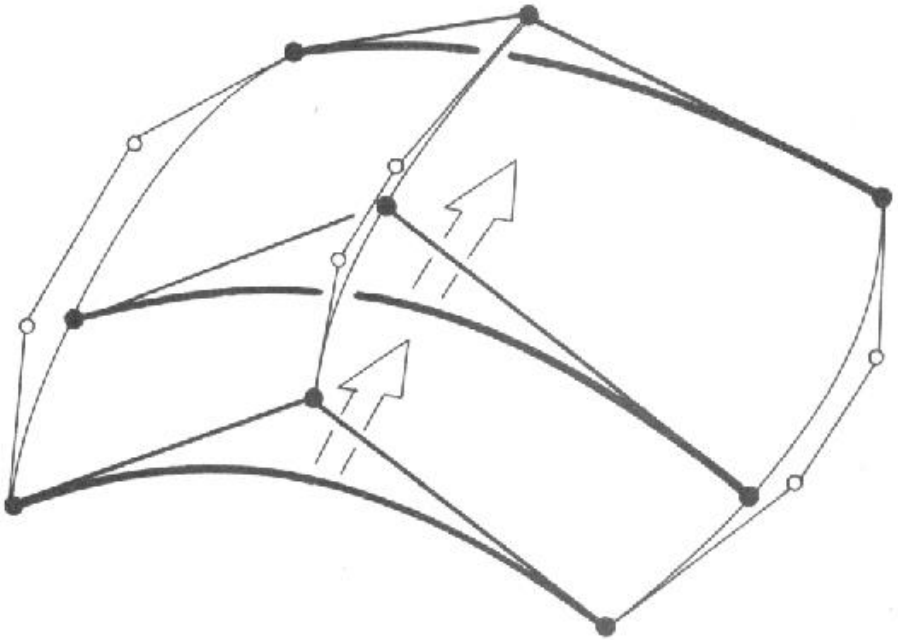
16K x 16K verteces

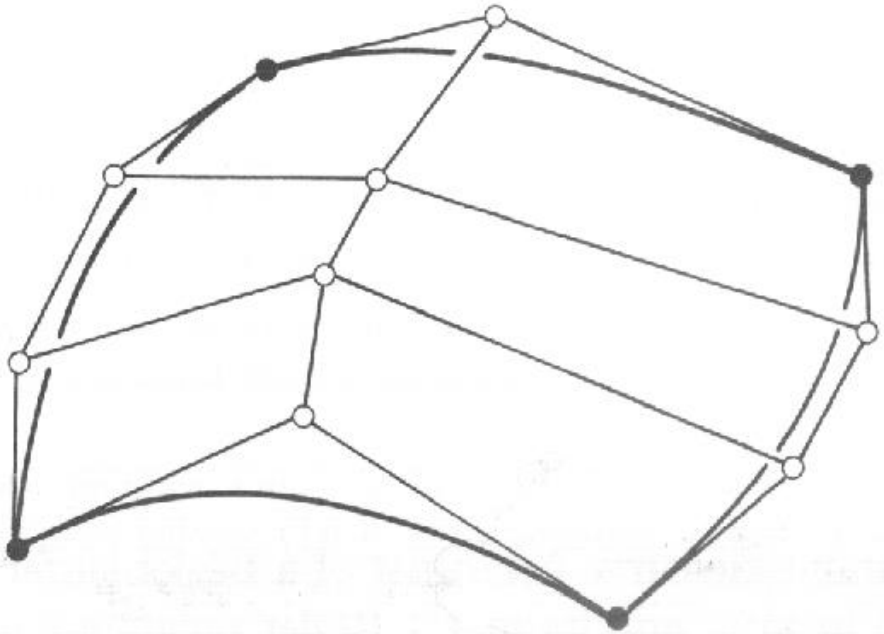
~537 milion of triangles

Parametric Surfaces

Tensor product surfaces





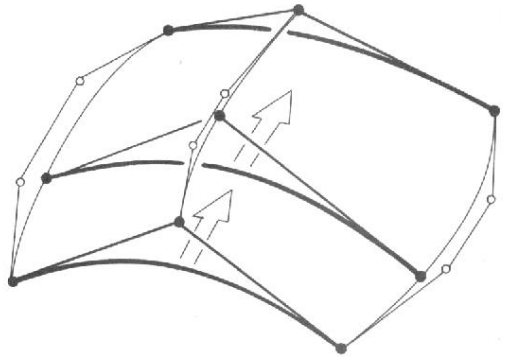


Bézier Patch

A surface is the locus of a curve that is moving through space and thereby changing its shape.

A surface is obtained by moving the control points of a Bézier curve along other Bézier curves.

Bézier Patch



Given an initial Bézier curve of degree m

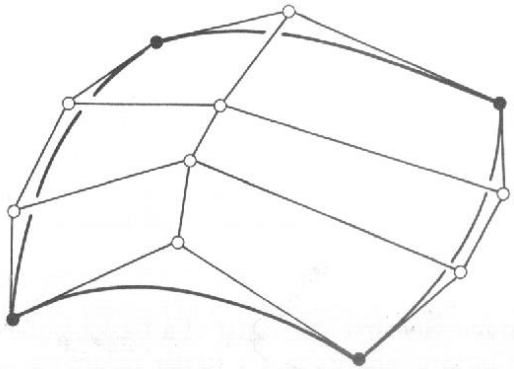
$$c(u) = \sum_{j=0}^{m} p_j B_j^m(u)$$

Let each control point b_j traverse a Bézier curve of degree n

$$p_j = c_j(v) = \sum_{i=0}^{n} b_{ij} B_i^n(v)$$

Combine these two eqs. and obtain the surface:

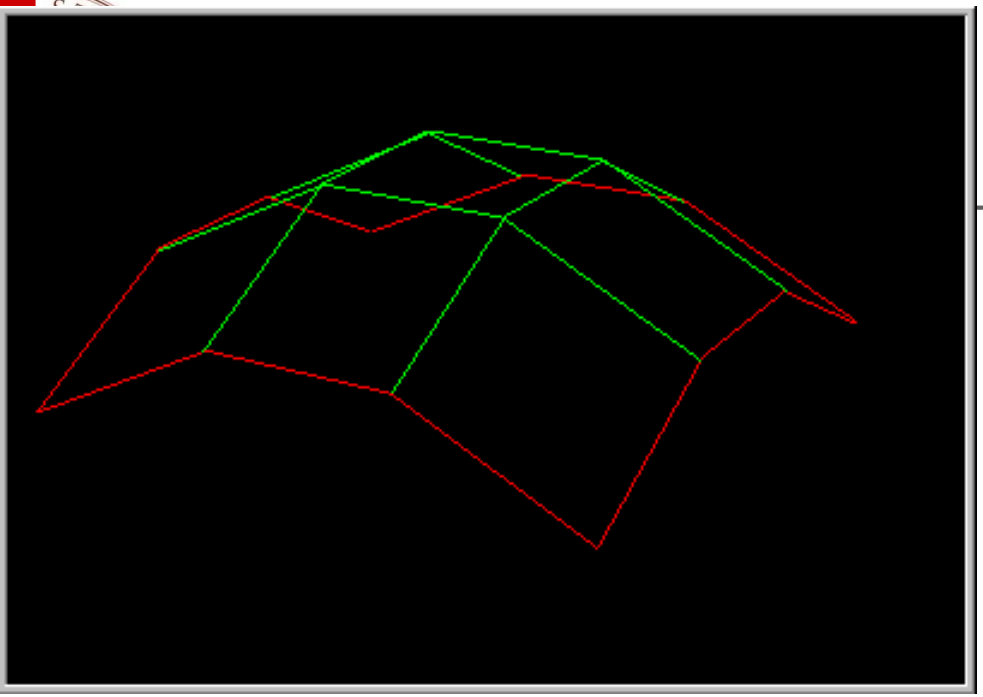
$$s(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n} b_{ij} B_i^n(v) B_j^m(u)$$

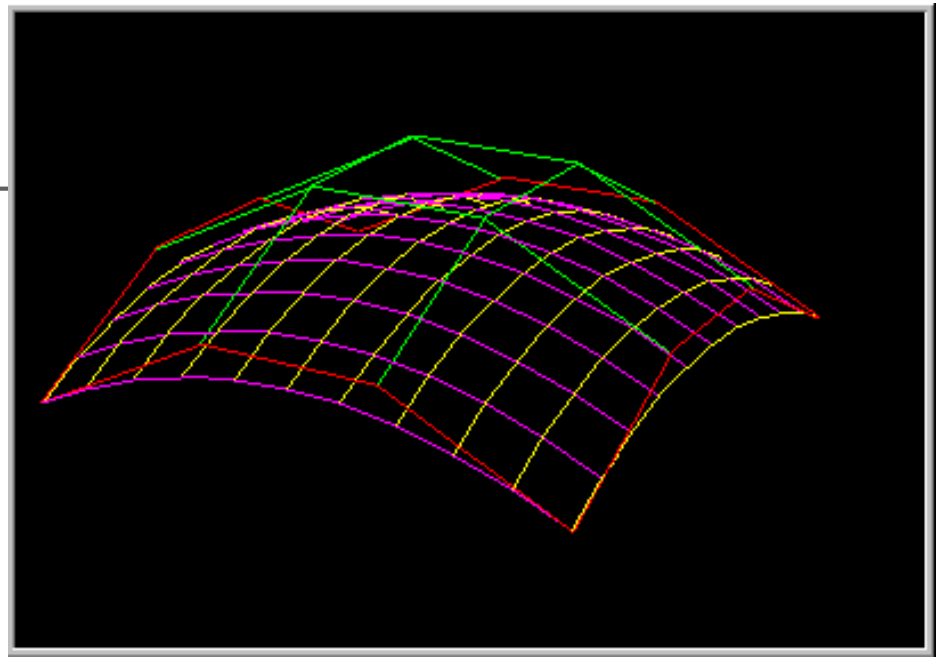


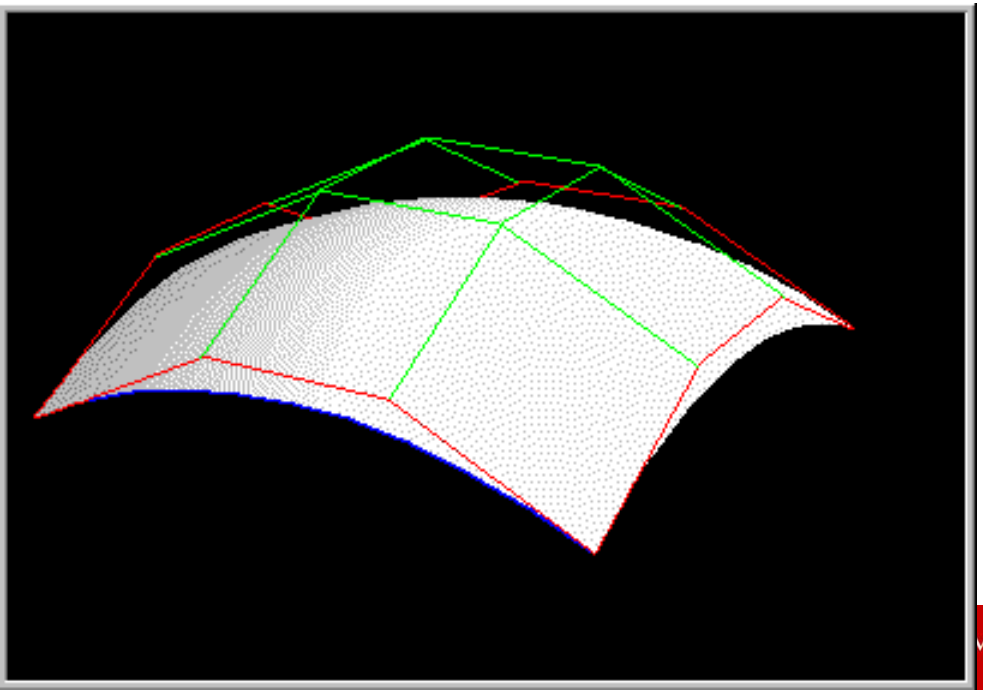
b_{ij} control points

Control mesh : (n+1)x(m+1)

Tensor-product surface defined on a rectangular parameter domain







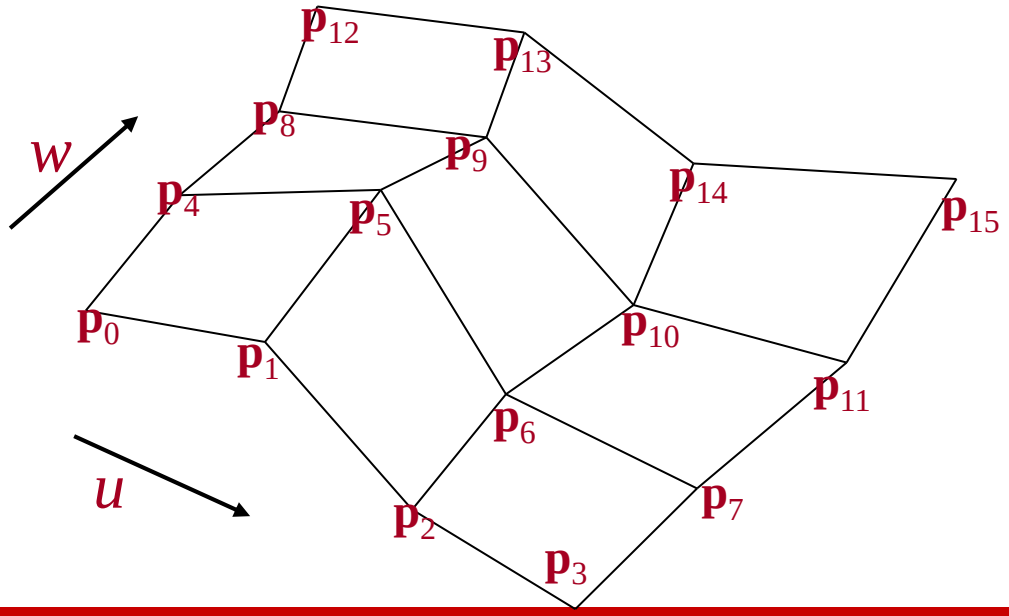
$$s(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_{ij} B_j^3(u) B_i^3(v)$$

$$b_{ij} \in \mathbb{R}^3$$

Example:
bicubic Bézier Patch

Control Mesh

- Consider a *bicubic* Bézier surface (bicubic means that it is a cubic function in both the *u* and *w* parameters)
- A cubic curve has 4 control points, and a bicubic surface has a grid of 4x4 control points, $\mathbf{p}_0$ through $\mathbf{p}_{15}$



Matrix form of a Bézier patch

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

$$s(u,v) = \begin{bmatrix} B_0^n(v) & \dots & B_n^n(v) \end{bmatrix} \begin{bmatrix} b_{00} & \cdots & b_{0m} \\ \vdots & & \vdots \\ b_{n0} & \cdots & b_{nm} \end{bmatrix} \begin{bmatrix} B_0^m(u) \\ \vdots \\ B_m^m(u) \end{bmatrix}$$

Write the Bernstein Poly in monomial form

$$s(u,v) = \begin{bmatrix} 1 & v & \dots & v^n \end{bmatrix} M \begin{bmatrix} b_{00} & \cdots & b_{0m} \\ \vdots & & \vdots \\ b_{n0} & \cdots & b_{nm} \end{bmatrix} N^T \begin{bmatrix} 1 \\ u \\ \vdots \\ u^m \end{bmatrix}$$

$$s(u,v) = VMGN^T U$$

M *is given by:* $m_{ij} = (-1)^{j-i} \binom{m}{j} \binom{j}{i}$

Bicubic Bézier patch in Matrix Form (m=n=3)

$$s(u,v) = \begin{bmatrix} V \cdot \mathbf{C}_x \cdot U^T \\ V \cdot \mathbf{C}_y \cdot U^T \\ V \cdot \mathbf{C}_z \cdot U^T \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$\mathbf{C}_x = M \cdot \mathbf{G}_x \cdot M^T$$

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

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

$$\mathbf{G}_x = \begin{bmatrix} p_{0x} & p_{4x} & p_{8x} & p_{12x} \\ p_{1x} & p_{5x} & p_{9x} & p_{13x} \\ p_{2x} & p_{6x} & p_{10x} & p_{14x} \\ p_{3x} & p_{7x} & p_{11x} & p_{15x} \end{bmatrix}$$

- The matrix form is a nice and compact notation and leads to an efficient method of computation
- It can also take advantage of 4x4 matrix support which is built into graphics hardware

Tangents

- To compute the u and v tangent vectors at some (*u*,v) location, we can use:

$$\frac{\partial s(u,v)}{\partial u} = \frac{\partial}{\partial u}\sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij} B_i^n(u) B_j^m(v) =$$

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

$$\frac{\partial s(u,v)}{\partial v} = \frac{\partial}{\partial v}\sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij} B_i^n(u) B_j^m(v) =$$

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

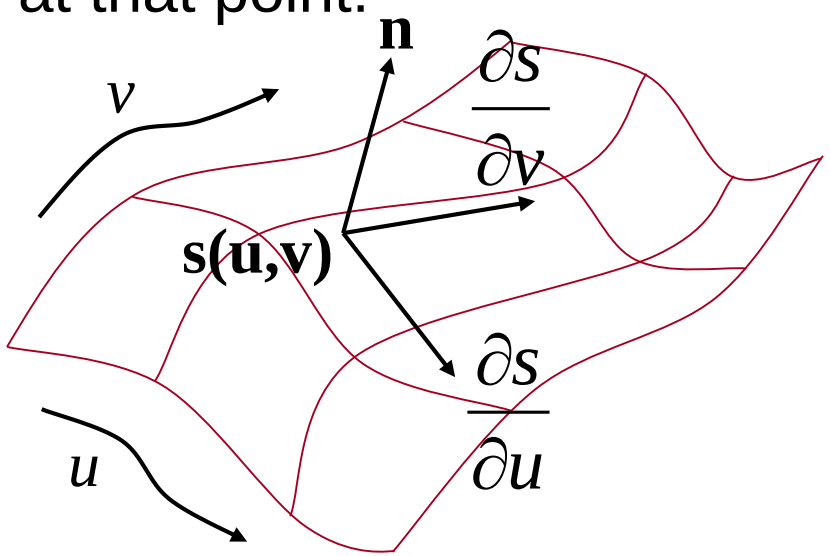
Normal Vectors

The normal vector **n** of a parametric surface is a normalized vector that is normal to the surface in a given point (u,v).

It is computed by the cross product of any two vectors that are tangent to the surface at that point:

$$\mathbf{n}^* = \frac{\partial s(u,v)}{\partial u} \times \frac{\partial s(u,v)}{\partial v}$$

$$\mathbf{n} = \frac{\mathbf{n}^*}{\|\mathbf{n}^*\|}$$

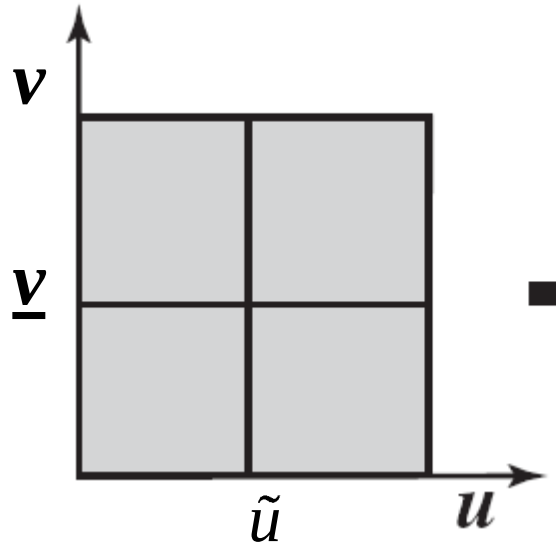


Isoparametric Bézier curves

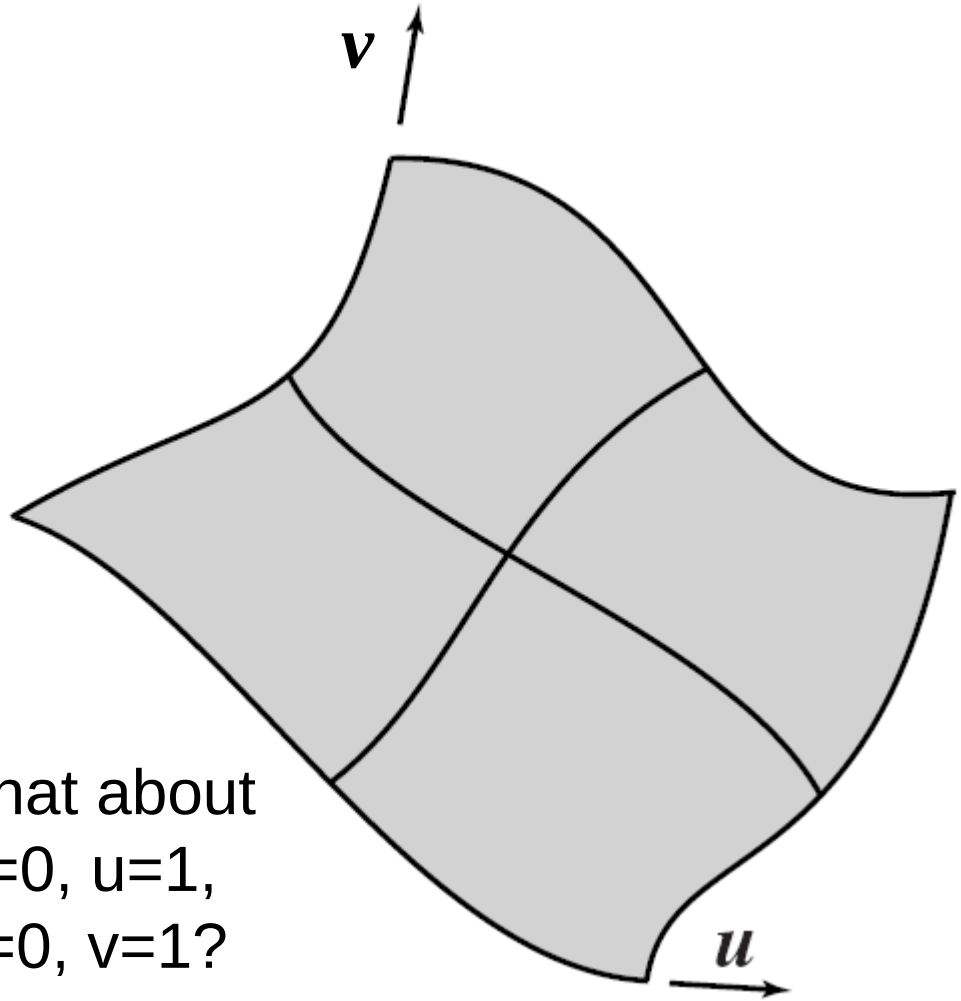
Fix $v = \underline{v}$ then s(u,v) reduces to a curve

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

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



What about
u=0, u=1,
v=0, v=1?



Properties

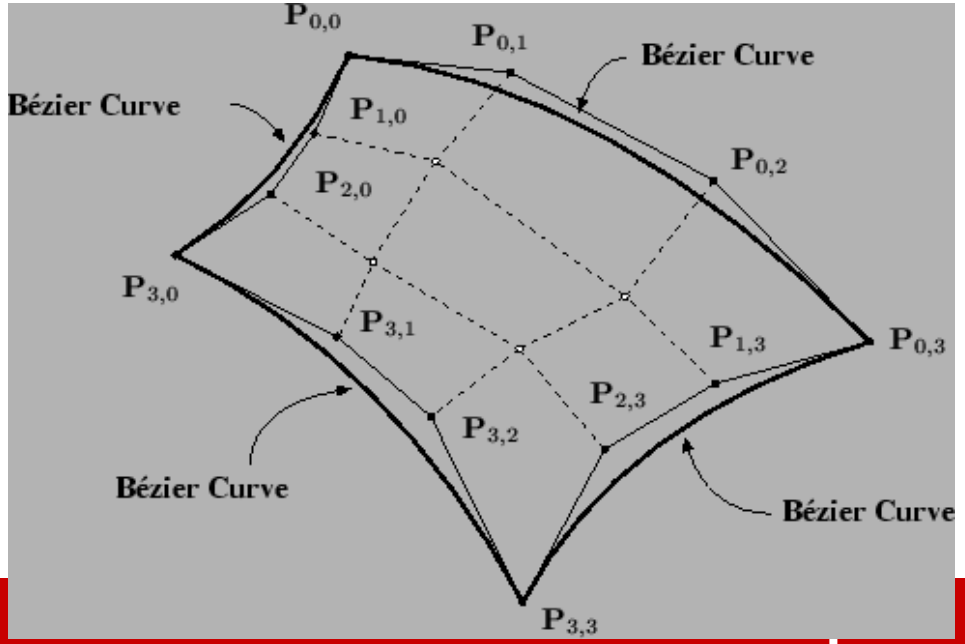
- The patch interpolates the four control points corners of the control mesh

$$s(0,0) = b_{00} \quad s(0,1) = b_{0n}, \quad s(1,0) = b_{n0}, \quad s(1,1) = b_{nn}$$

- Boundary curves: The 4 boundaries of the Bézier surface are just Bézier curves defined by the points on the edges of the surface.

$$s(u,0) = \sum_{i=0}^{n} b_{i0} B_i^n(u)$$

$$s(u,1), s(0,v), s(1,v)$$



Properties

- Partition of Unity $\sum_{i=0}^{n}\sum_{j=0}^{m} B_j^m(u)B_i^n(v) \equiv 1$

- Convex Hull property: for 0 ≤ u,v ≤1, the terms

  $B_i^n(v), B_j^m(u)$ are non-negative.

  Then, taking into account the partition of unity property,

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

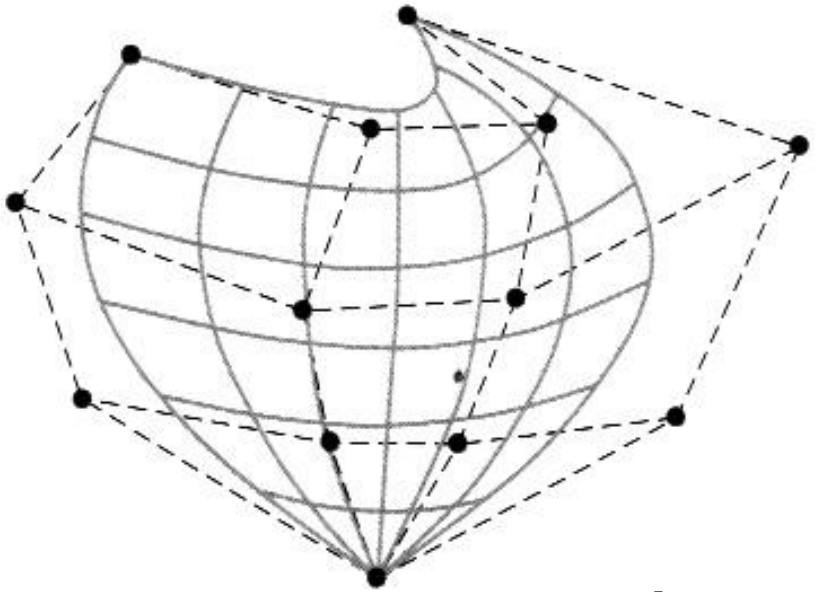
  is a convex combination. The Bézier patch will fall within the convex hull of the control points.

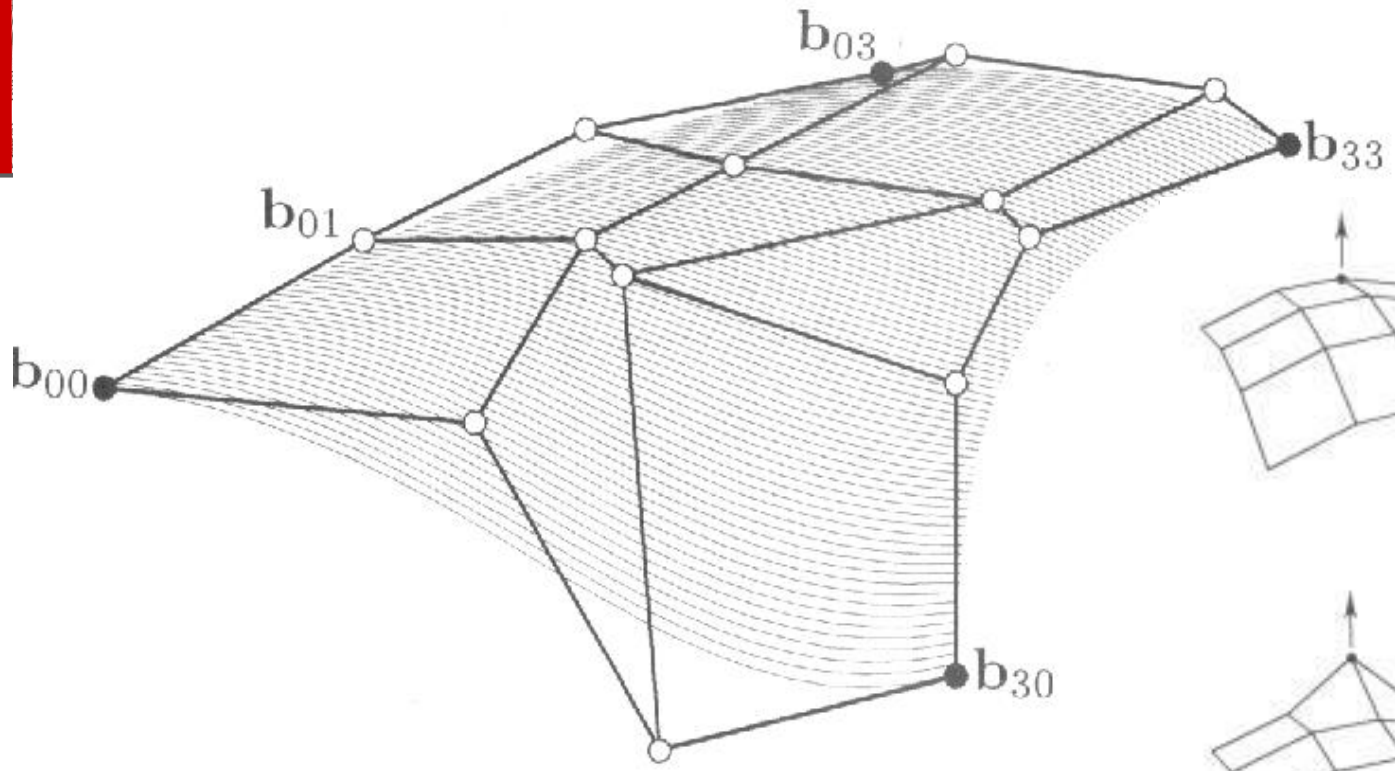
- Affine Invariance: each affine transformation of the control mesh defines a new Bézier patch which is the transformation of the original.

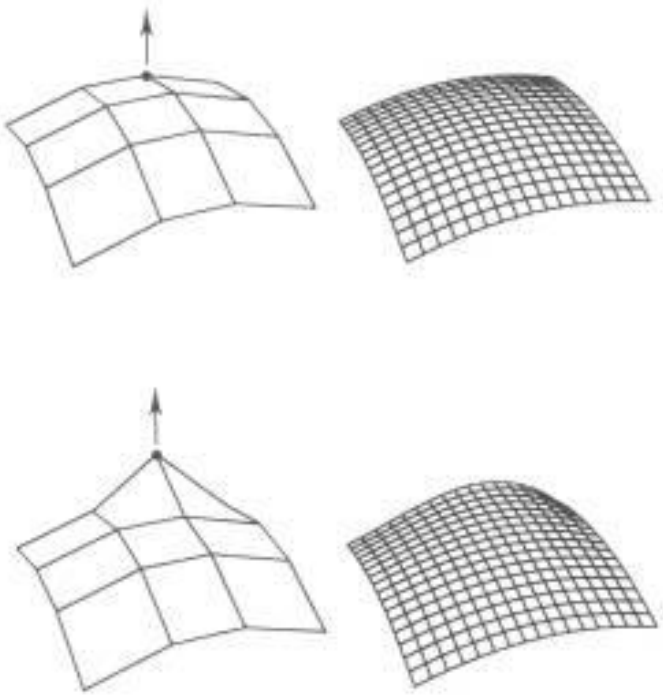
Properties

- Linear Precision: when all the control points lie on a plane, then the patch lies on the same plane.

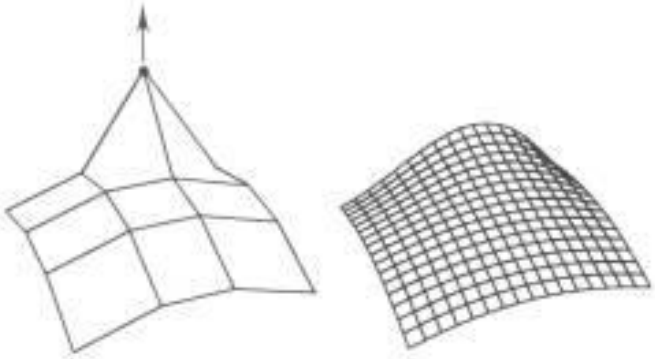
- Shape approximation: the control mesh approximates the shape of the patch







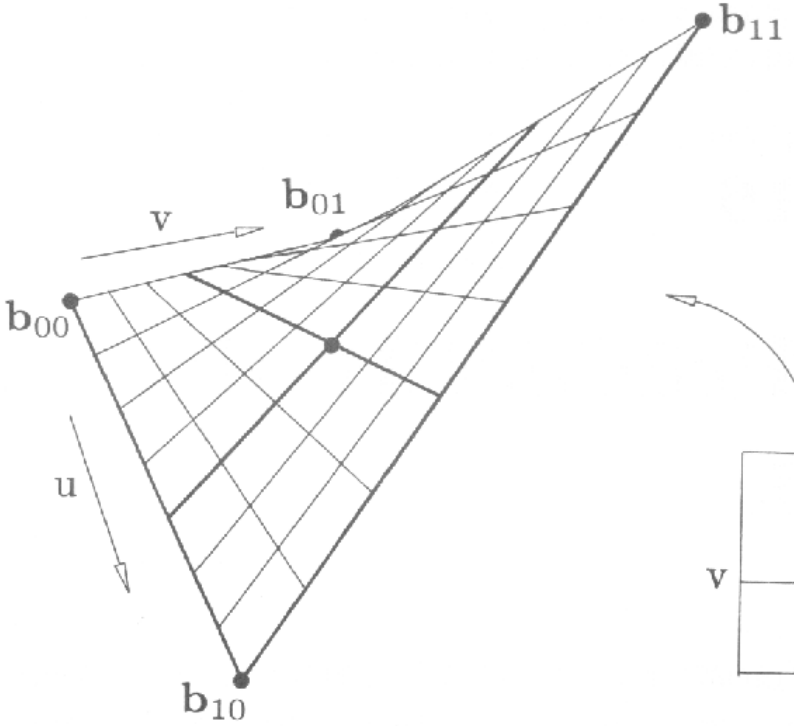
- Surface shape control

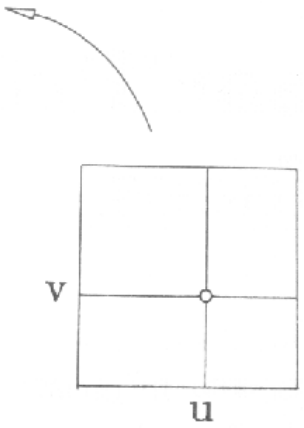


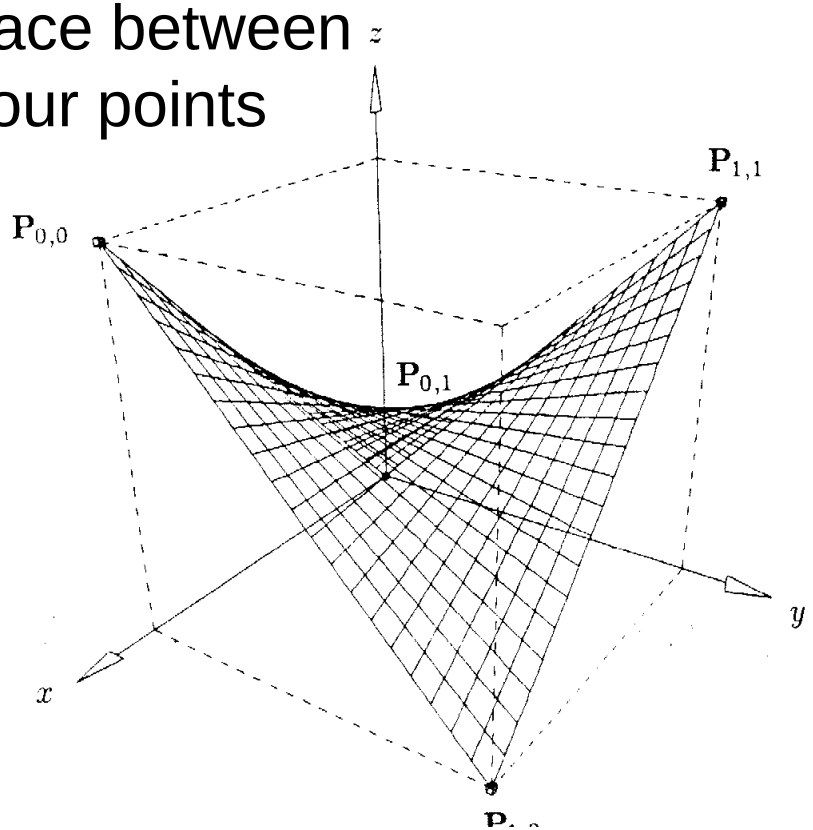
Bilinear Surface: Bézier patch of degree (1,1)

Hyperbolic paraboloid

The "simplest" surface between four points





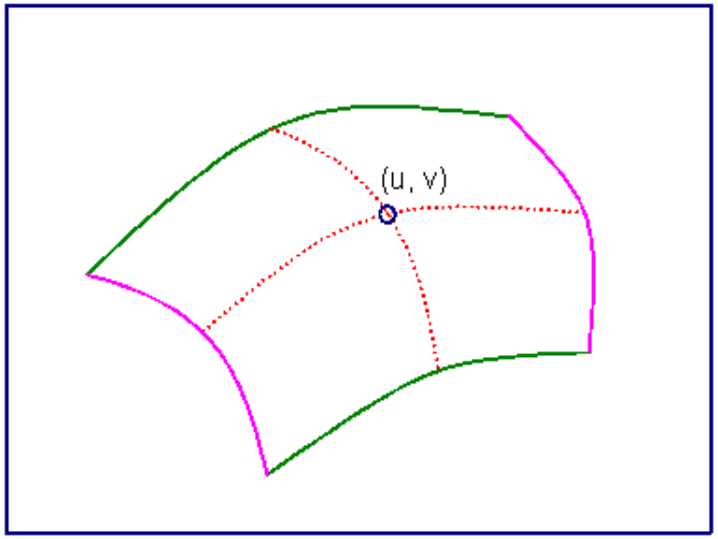


$$s(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1} b_{ij} B_j(u) B_i(v) = [(1-u)b_{00} + ub_{10}](1-v) + [(1-u)b_{01} + ub_{11}]v$$

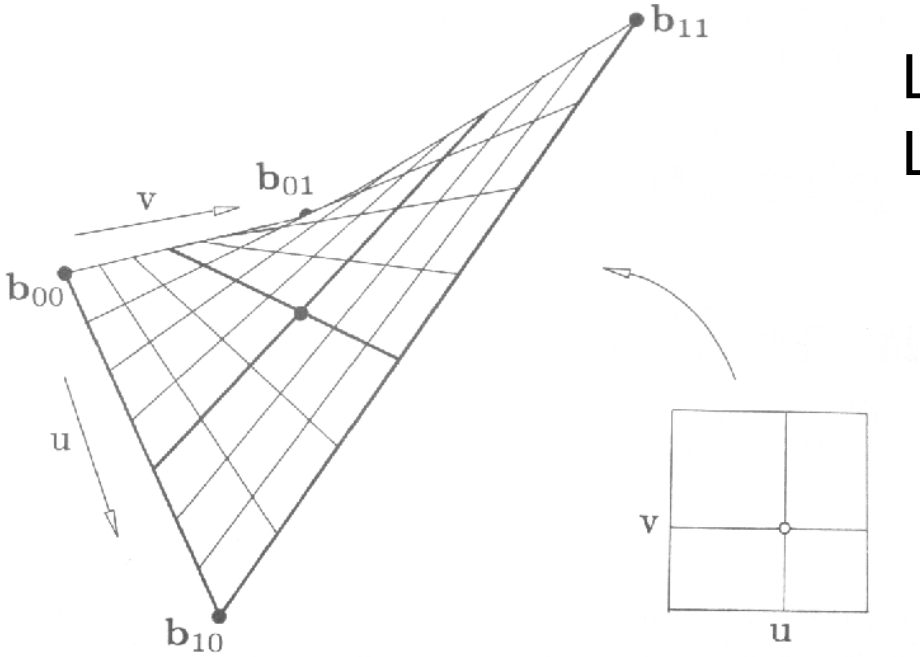
Bézier Patch Evaluation mode I by direct de Casteljau

Compute the point on the Bézier patch at a given (u,v) by repeated application of bilinear interpolation.

Suppose we are given a rectangular array of control points b_{ij}
The de Casteljau algorithm evaluates s(u,v)



Bi-Lerp: bilinear interpolation



$Lerp(b_{00},b_{01},v)=(1-v)\,b_{00}+vb_{01}$

$Lerp(b_{10},b_{11},v)=(1-v)\,b_{10}+vb_{11}$

Bi-Lerp(b_{00} **,** b_{01}**,** b_{10}**,** b_{11}**,u,v)=**

Lerp(Lerp(b_{00}**,** b_{01}**,v),** **Lerp(**b_{10}**,** b_{11}**,v),u)=**

$$[(1-v)b_{00}+vb_{01}](1-u)+[(1-v)b_{10}+vb_{11}]u$$

The de Casteljau Evaluation Algorithm

$$\text{set} \quad b_{ij}^{00} = b_{ij}$$

$$\text{for} \quad r = 1,..,n$$

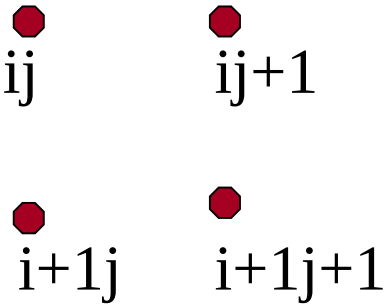
$$\text{for} \quad i,j = 0,..,n-r$$

$$b_{ij}^{rr} = BiLerp(b_{ij}^{r-1r-1}, b_{i+1j}^{r-1r-1}, b_{ij+1}^{r-1r-1}, b_{i+1j+1}^{r-1r-1}, u, v)$$

$$\text{end}$$

$$\text{end}$$

$$s(u,v) = b_{00}^{nn}$$



$$BiLerp(..) = [(1-u)b_{ij}^{r-1r-1} + ub_{i+1j}^{r-1r-1}](1-v) +$$
$$[(1-u)b_{ij+1}^{r-1r-1} + ub_{i+1j+1}^{r-1r-1}]v$$

Example: computing a point on a Bézier patch using de Casteljau algorithm

Bézier patch of degree 2 at (u,v)=(0.5,0.5)

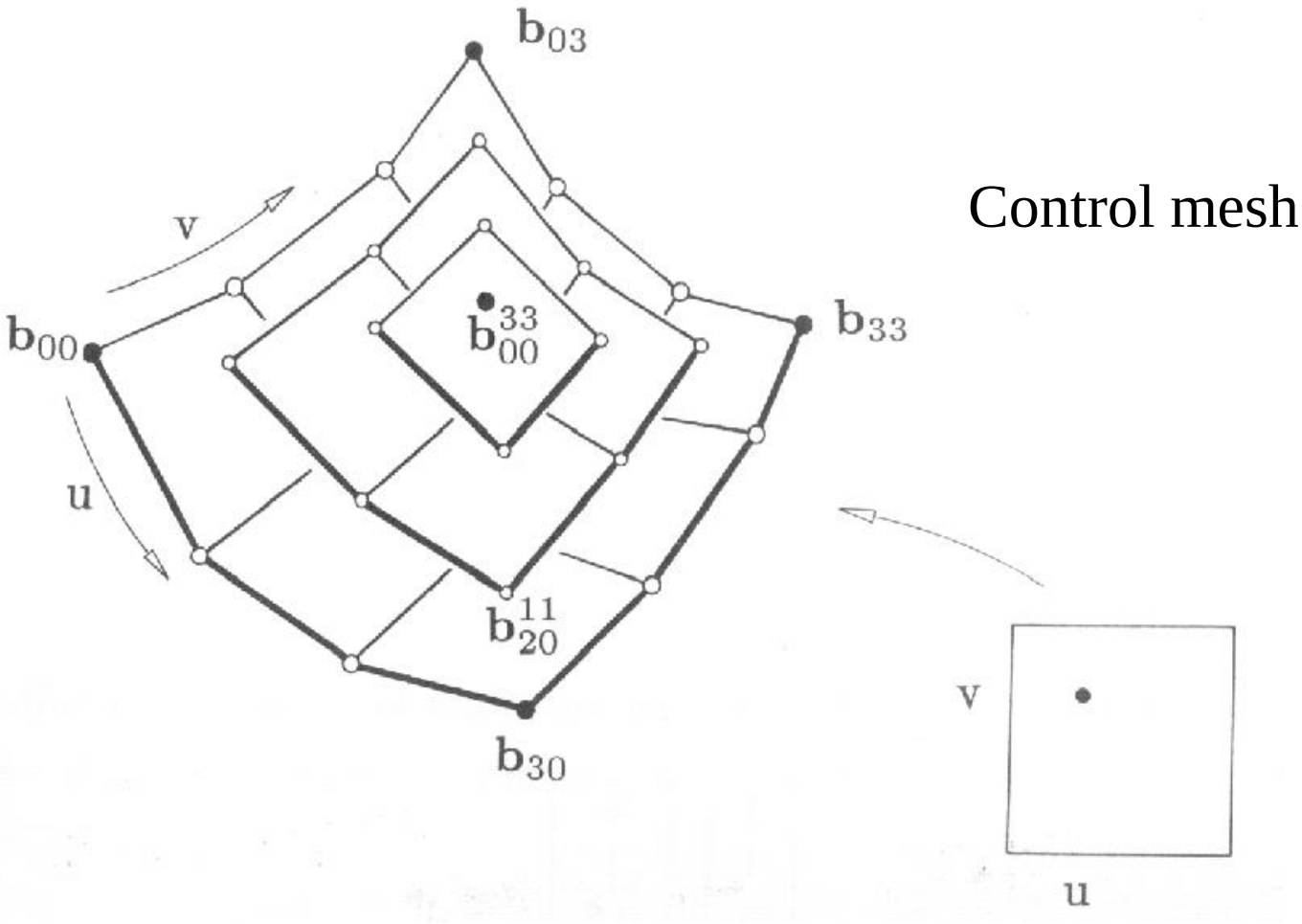
$$\begin{bmatrix} \begin{bmatrix}0\\0\\0\end{bmatrix} & \begin{bmatrix}2\\0\\0\end{bmatrix} & \begin{bmatrix}4\\0\\0\end{bmatrix} \\ \begin{bmatrix}0\\2\\0\end{bmatrix} & \begin{bmatrix}2\\2\\0\end{bmatrix} & \begin{bmatrix}4\\2\\2\end{bmatrix} \\ \begin{bmatrix}0\\4\\0\end{bmatrix} & \begin{bmatrix}2\\4\\4\end{bmatrix} & \begin{bmatrix}4\\4\\4\end{bmatrix} \end{bmatrix} \quad \textbf{Step 1} \Rightarrow \quad \begin{bmatrix} \begin{bmatrix}1\\1\\0\end{bmatrix} & \begin{bmatrix}3\\1\\0.5\end{bmatrix} \\ \begin{bmatrix}1\\3\\1\end{bmatrix} & \begin{bmatrix}3\\3\\2.5\end{bmatrix} \end{bmatrix} \quad \textbf{Step 2} \Rightarrow \quad \begin{bmatrix}2\\2\\1\end{bmatrix}$$

The point on the surface is:

Control mesh

Example

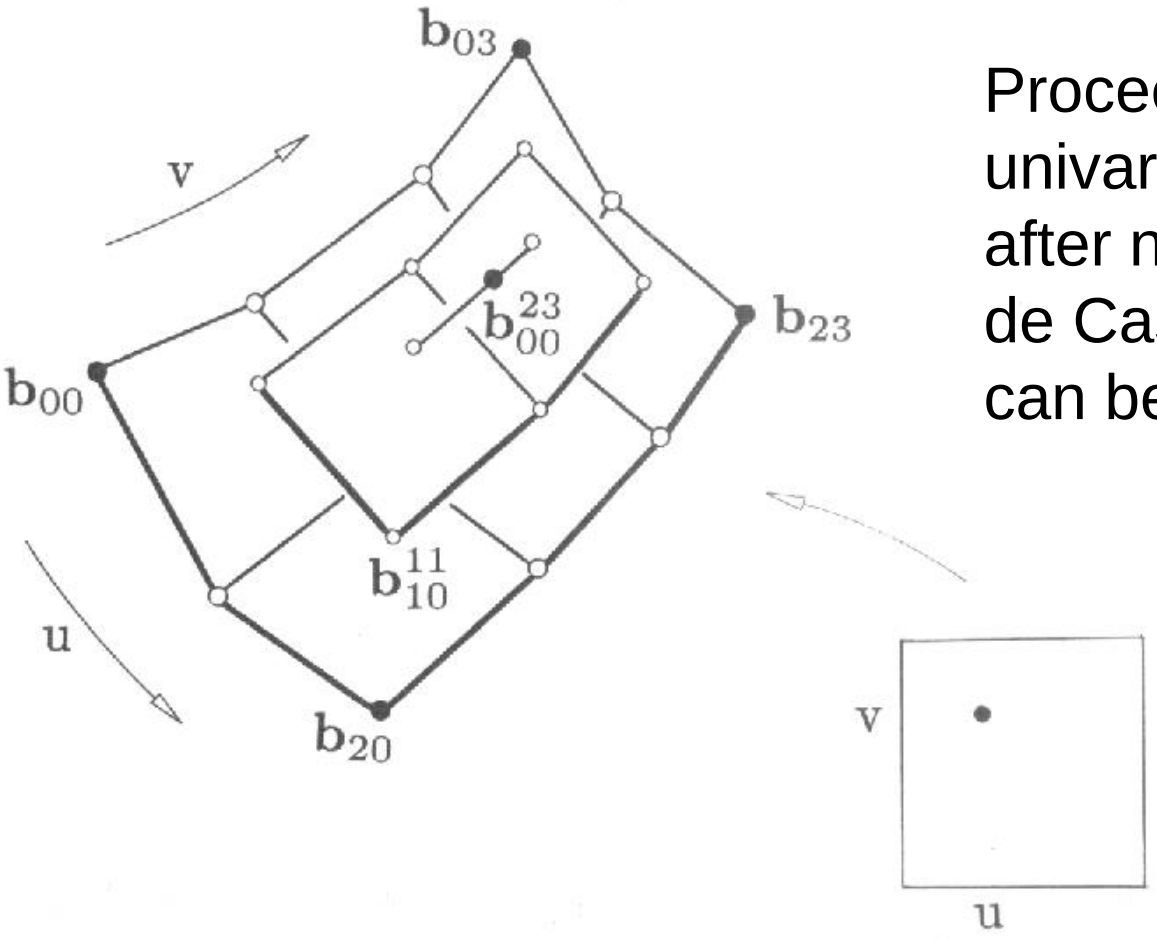
Bézier patch of degree 3 at (u,v)



Control mesh

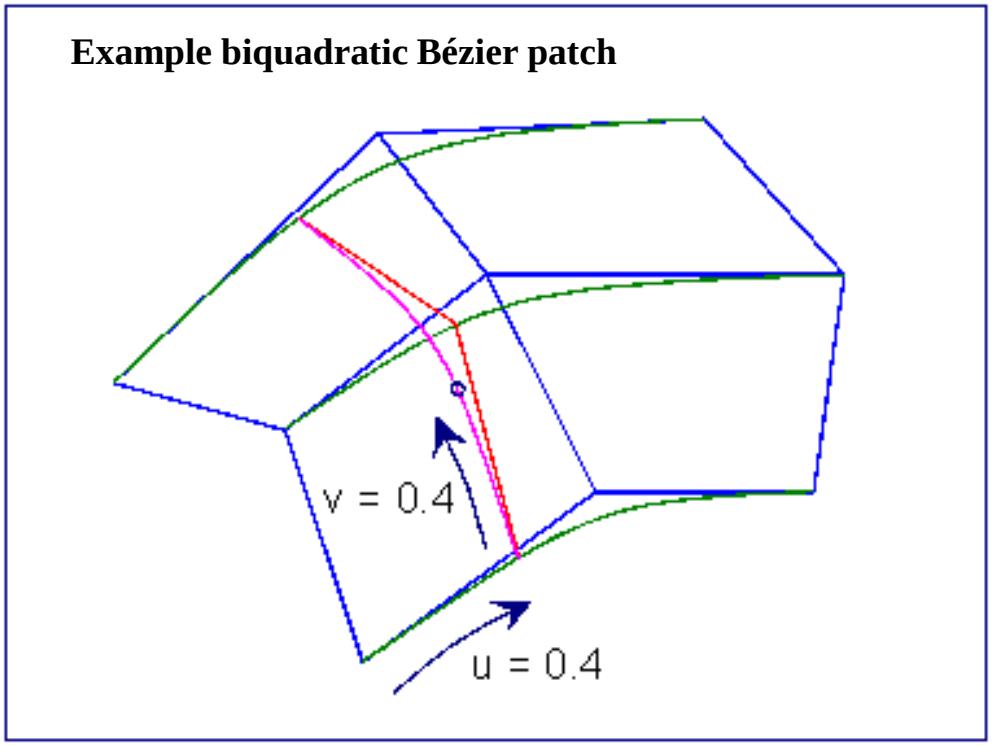
Example

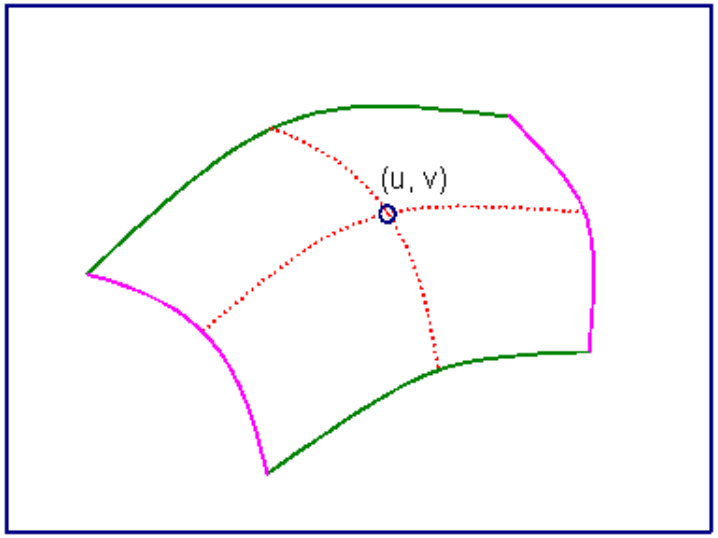
Bézier patch of degree (2,3) at (u,v)



Proceeds in a univariate manner after no more direct de Casteljau steps can be performed

Bézier Patch: Evaluation mode II





Compute the point on the patch at a given (u,v) by evaluating univariate de Casteljau algorithm for curves
first in u, then in v

Example: computing a point on a Bézier patch using univariate de Casteljau

Bézier patch of degree 2 at (u,v)=(0.5,0.5)

$$\begin{bmatrix} \begin{bmatrix}0\\0\\0\end{bmatrix} & \begin{bmatrix}2\\0\\0\end{bmatrix} & \begin{bmatrix}4\\0\\0\end{bmatrix} \\ \begin{bmatrix}0\\2\\0\end{bmatrix} & \begin{bmatrix}2\\2\\0\end{bmatrix} & \begin{bmatrix}4\\2\\2\end{bmatrix} \\ \begin{bmatrix}0\\4\\0\end{bmatrix} & \begin{bmatrix}2\\4\\4\end{bmatrix} & \begin{bmatrix}4\\4\\4\end{bmatrix} \end{bmatrix} \xrightarrow{\text{Step 1}} \begin{bmatrix} \begin{bmatrix}2\\0\\0\end{bmatrix} \\ \begin{bmatrix}2\\2\\0.5\end{bmatrix} \\ \begin{bmatrix}2\\4\\3\end{bmatrix} \end{bmatrix} \xrightarrow{\text{Step 2}} \begin{bmatrix}2\\2\\1\end{bmatrix}$$

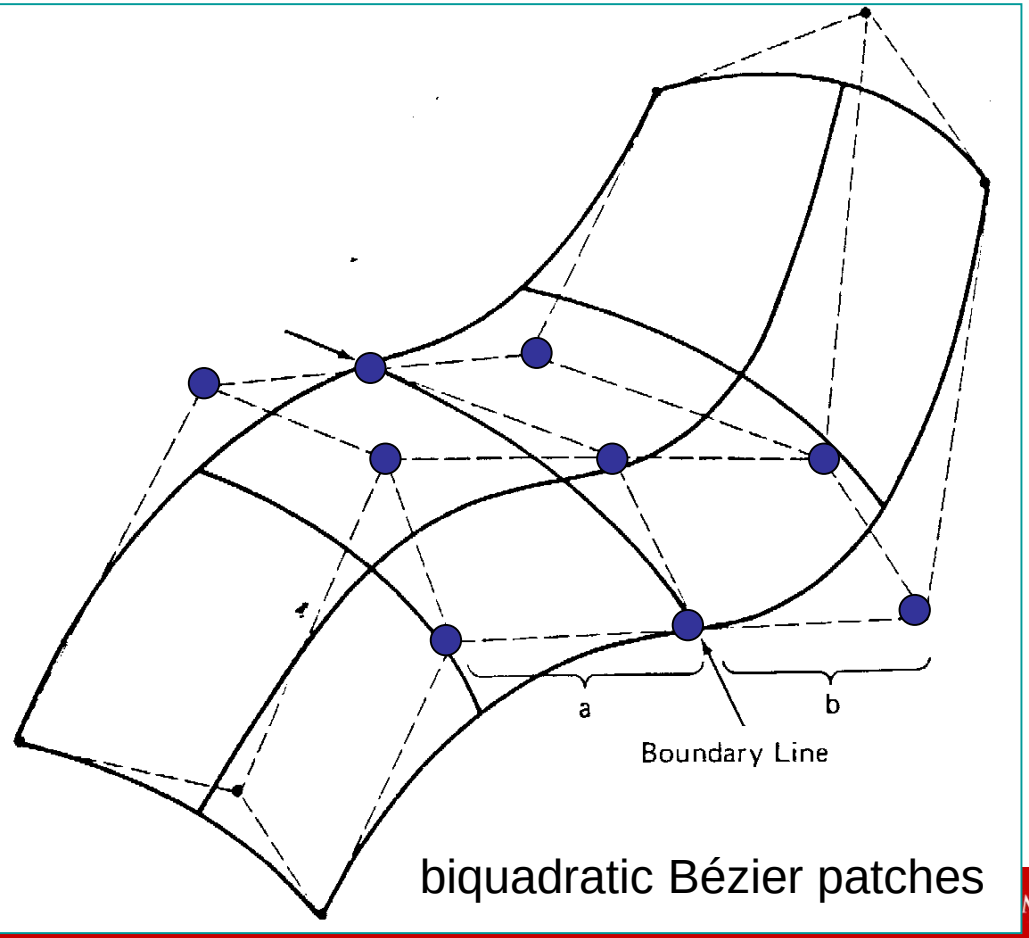
Control mesh

Step 1: evaluate each row of Bézier points for u=0.5

Step 2: evaluate the obtained curve s(0.5,v) for v=0.5

Joining Bézier patches

Two adjacent patches are C^r across their common boundary if and only if all rows of their control mesh vertices can be interpreted as control polygons of C^r Bézier curves



biquadratic Bézier patches

C^1 continuity:

(a) **blue CP must be collinear**

(b) **All be in the same ratio (a/b)**

Without (b) there is no continuous tangent plane

Tensor product Spline Surfaces

Given two knot vectors U and V on a parametric domain [0,1]x[0,1],

$$U = \left\{ \underset{n}{a,..,a}, u_1,..,u_K, \underset{n}{b,..,b} \right\} \quad V = \left\{ \underset{m}{a,..,a}, v_1,..,v_L, \underset{m}{b,..,b} \right\}$$

the spline surface of orders n and m defined by the bidirectional net of control points P_{ij} is:

$$s(u,v) = \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} P_{ij} N_{i,n}(u) N_{j,m}(v)$$

where the CP define the Control Mesh

$N_{i,n}(u), N_{j,m}(v)$ univariate B-Spline basis functions of degree n-1 and m-1

Spline surfaces

$$s(u,v) = \begin{pmatrix} s_x(u,v) \\ s_y(u,v) \\ s_z(u,v) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} x_{ij} N_{i,n}(u) N_{j,m}(v) \\ \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} y_{ij} N_{i,n}(u) N_{j,m}(v) \\ \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} z_{ij} N_{i,n}(u) N_{j,m}(v) \end{pmatrix}$$

$P_{ij=}$ (x_{ij},y_{ij},z_{ij}) control points Control mesh : KxL

Properties

- The properties of the tensor product basis functions

$$N_{i,n}(u)N_{j,m}(v)$$

follow from the corresponding properties of the univariate basis functions (nonnegativity, partition of unity, ..)

Local support:

$$N_{i,n}(u)N_{j,m}(v) = 0$$

$$\textit{if } (u,v) \textit{ is outside the rectangle } [u_i, u_{i+n}) \times [v_j, v_{j+m})$$

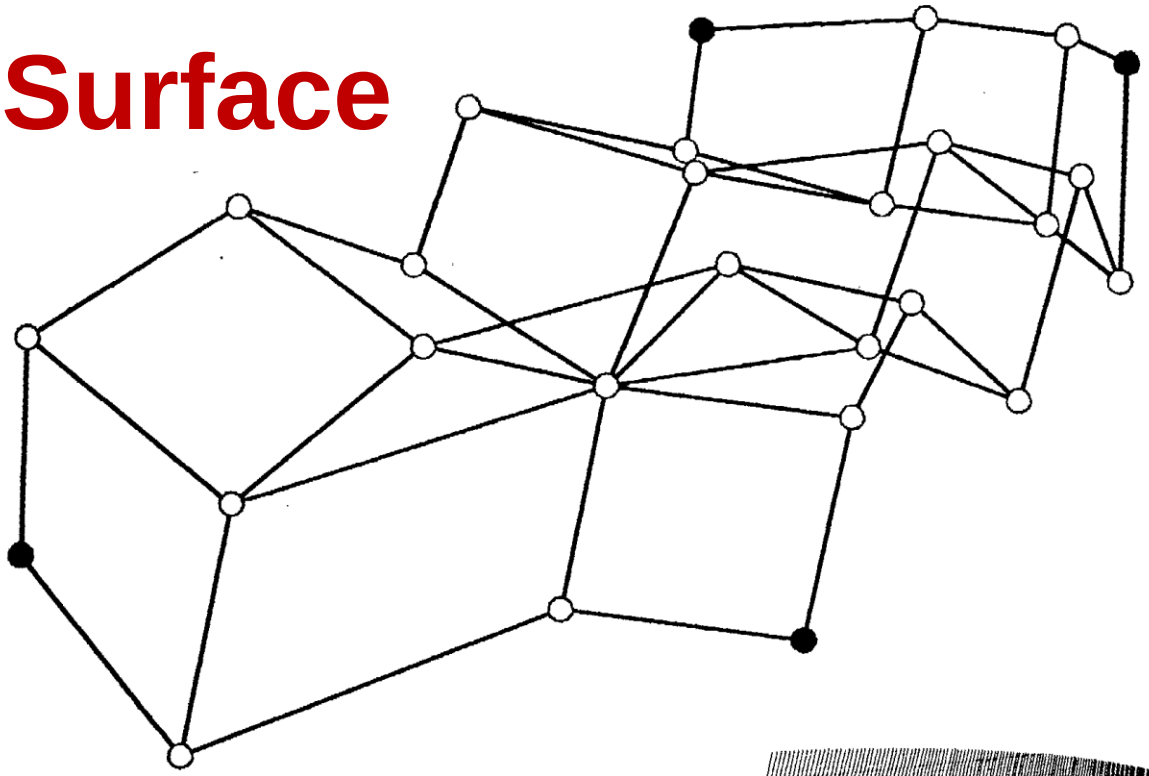
- The spline surface have the following properties:
  - Affine invariance
  - Local convex hull property
  - Local modification: if P_{ij} is moved it affects the surface only in the rectangle $[u_i, u_{i+n}) \times [v_j, v_{j+m})$
  - No variation diminishing property for spline surface

Spline Surface Evaluation at (u,v)

- Find the knot span in which u in $[u_l,u_{l+1})$;
- Compute the n nonzero basis functions in $[u_l,u_{l+1})$;
- Find the knot span in which v in $[v_h,v_{h+1})$;
- Compute the m nonzero basis functions in $[v_h,v_{h+1})$;
- Multiply the values of the nonzero basis functions with the corresponding control points:

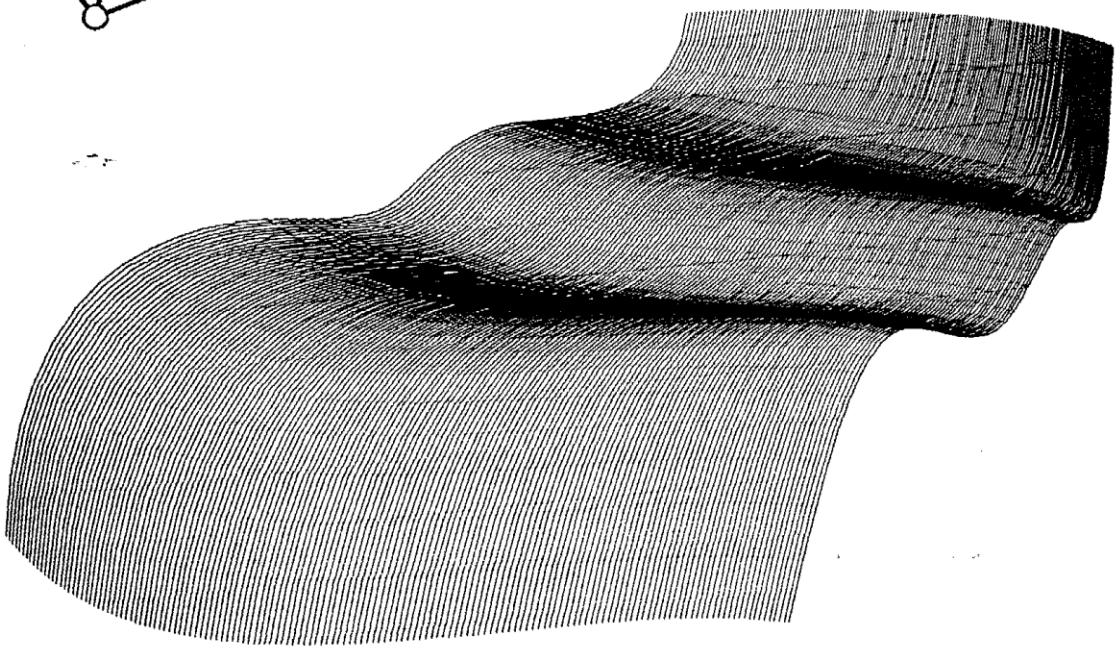
$$s(u,v) = \sum_{i=l-n+1}^{l} \sum_{j=h-m+1}^{h} P_{ij} N_{i,n}(u) N_{j,m}(v)$$

Spline Surface



UxV=[0,1]x[0,1]
Parametric domain

Example of tensor product bicubic spline (6x3 patch)



Non Uniform Rational B-Splines (NURBSs)

Let s^w be a spline surface in the homogeneous space 4D:

$$s^w(u,v) = \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} P_{ij}{}^w N_{i,n}(u) N_{j,m}(v)$$

Project into 3D

$$s(u,v) = \begin{pmatrix} s_x(u,v) \\ s_y(u,v) \\ s_z(u,v) \end{pmatrix} = \frac{\sum_{i=1}^{n+K} \sum_{j=1}^{m+L} w_{ij} P_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+K} \sum_{j=1}^{m+L} w_{ij} N_{i,n}(u) N_{j,m}(v)}$$

$\mathbf{P_{ij}}$ control points -- Control mesh : **KxL**

UxV=[0,1]x[0,1] Parametric domain, knot vectors

$\mathbf{N_{i,n}(u), N_{j,m}(v)}$ B-Spline basis functions

$\mathbf{w_{ij}}$ **weights ≥0**

Non Uniform Rational B-Splines (NURBSs)

A NURBS surface of degree n-1 in the u-direction and m-1 in the v direction is a bivariate vector-valued piecewise rational function of the form:

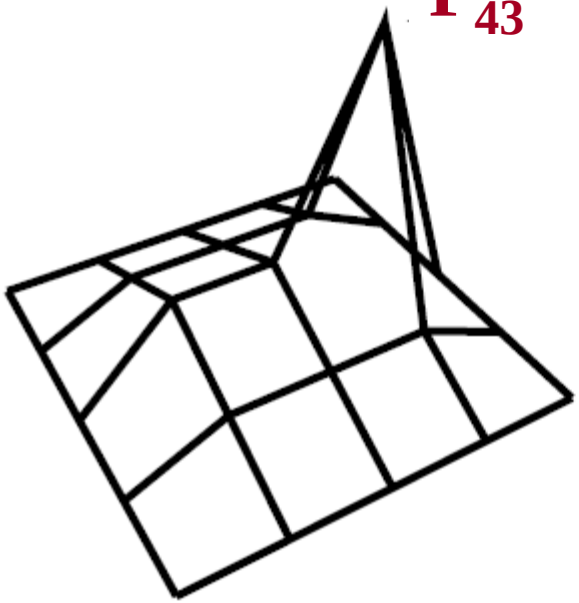
$$s(u,v) = \frac{\sum_{i=1}^{n+K}\sum_{j=1}^{m+L} w_{ij} P_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+K}\sum_{j=1}^{m+L} w_{ij} N_{i,n}(u) N_{j,m}(v)}$$

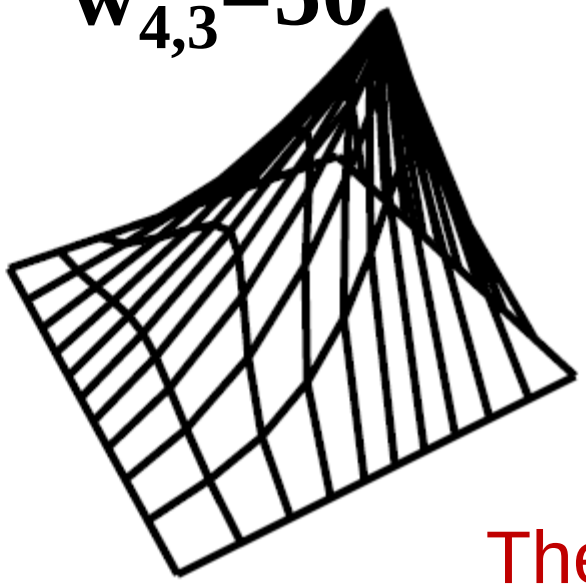
Introducing the piecewise rational basis functions:

$$R_{ij}(u,v) = \frac{w_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+K}\sum_{j=1}^{m+L} w_{ij} N_{i,n}(u) N_{j,m}(v)}$$

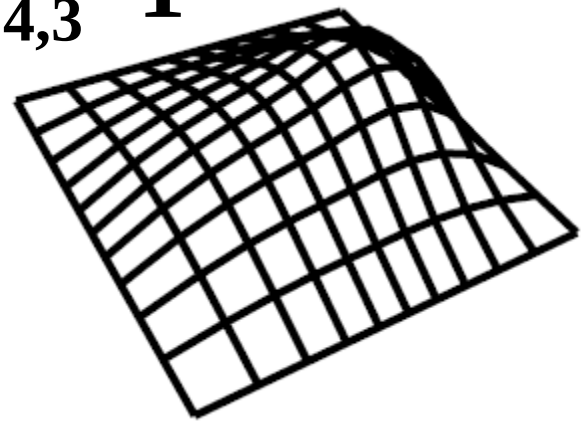
The surface can be written as

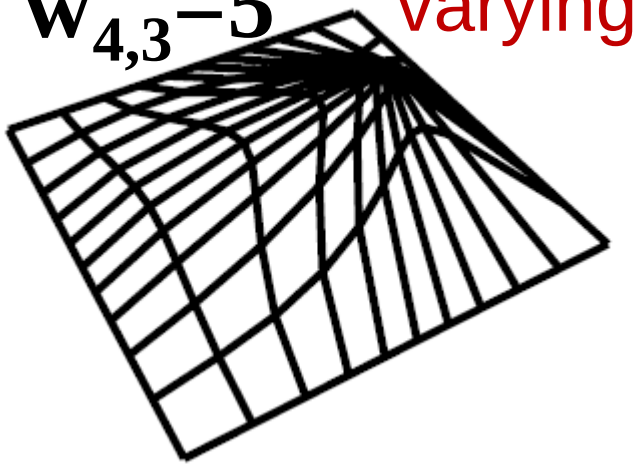
$$s(u,v) = \sum_{i=1}^{n+K}\sum_{j=1}^{m+L} P_{ij} R_{i,j}(u,v)$$





The effect of varying the weights





If $w_i=1$ for all i, then the NURBS is a non-rational spline surface

- Rational Bézier Patch: A NURBS surface without internal knots and open knot vector

- Closed Surface (no periodic): First column of the CP grid (or first row) coincides with the last column (or row)

- Corner: If all the CP in a subgrid (n-1)x(m-1) coincide, then the surface interpolates that CP, modelling a corner.

  By using multiply coincident CP, visual discontinuities can be created where there are no corresponding discontinuity in the basis functions.

- Planarity: If all the CP in a subgrid nxm lie on a plane, then the surface lies on the same plane.

NURBS Surface Evaluation at (u,v)

Apply spline surface procedure where the control points are multiplied by the weights

$$P_{ij}=(w_{ij}x_{ij},w_{ij}y_{ij},w_{ij}z_{ij},w_{ij})$$

Compute the four contributions

(S_x,S_y,S_z,S_w) where

$$S_x = \sum_{i=l-n+1}^{l} \sum_{j=h-m+1}^{h} x_{ij} w_{ij} N_{i,n}(u) N_{j,m}(v)$$

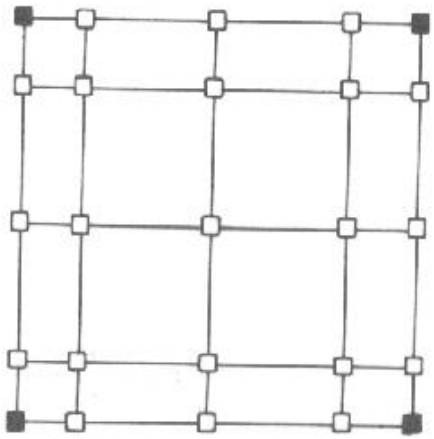
The point on the NURBS surface is:

$$x=S_x/S_w,\ \ y=S_y/S_w,\ z=S_z/S_w$$

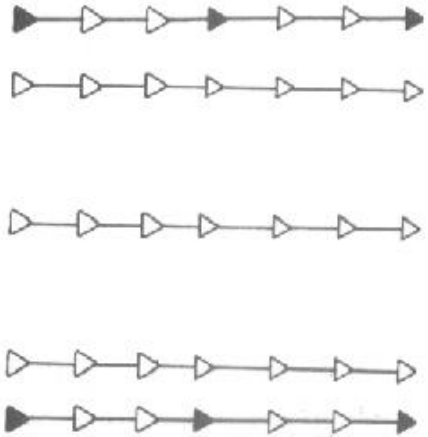
Spline Surface in piecewise Bézier patches form

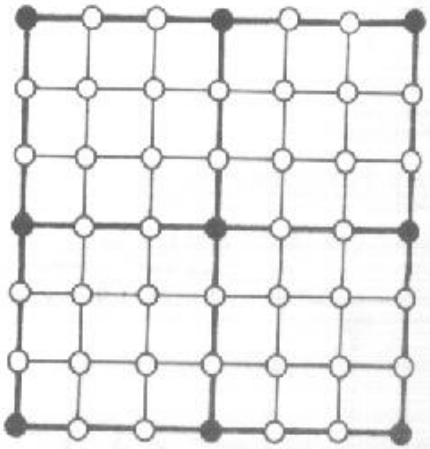
$$s(u,v) = \sum_{i=1}^{n+K} N_{i,n}(u)\left[\sum_{j=1}^{m+L} P_{ij} N_{j,m}(v)\right]$$

Curve in u direction



Bicubic spline





Piecewise bicubic Bézier

-convert each spline curve in u direction into piecewise Bézier curves by knot insertion until *multiplicity* = *degree*

-Analogously in v direction

How to store a spline/NURBS surface

FILENAME: namefile.snurbs

DEGREE_U_V

2 2

N.C.P._U_V

5 9

N.KNOTS_U_V

8 12

COORD.C.P.(X,Y,Z,W)

0.000000e+00 0.000000e+00 1.000000e+00 0.00000e+00

….

KNOTS_U

0.000000e+00

….

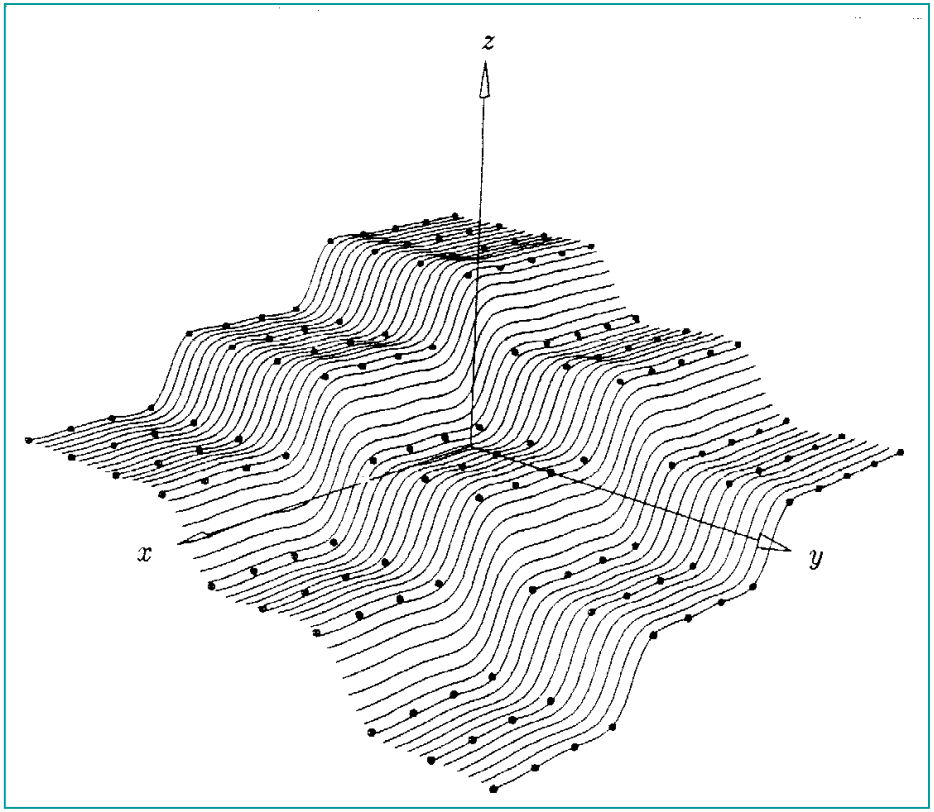
KNOTS_V

0.000000e+00

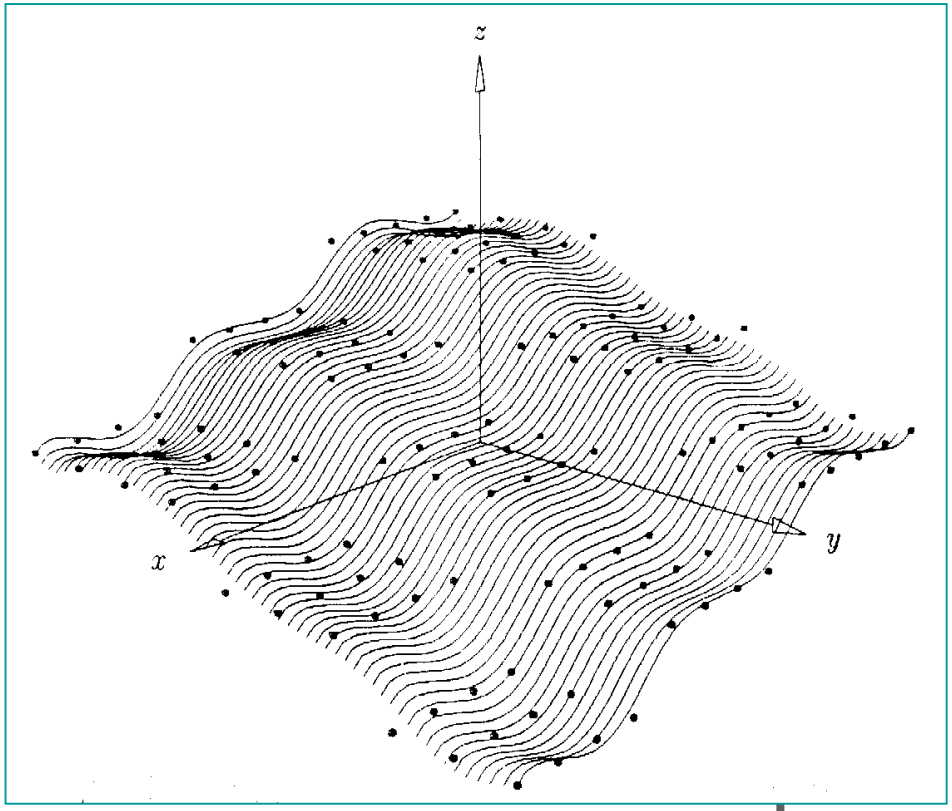
….

Surface fitting

interpolant

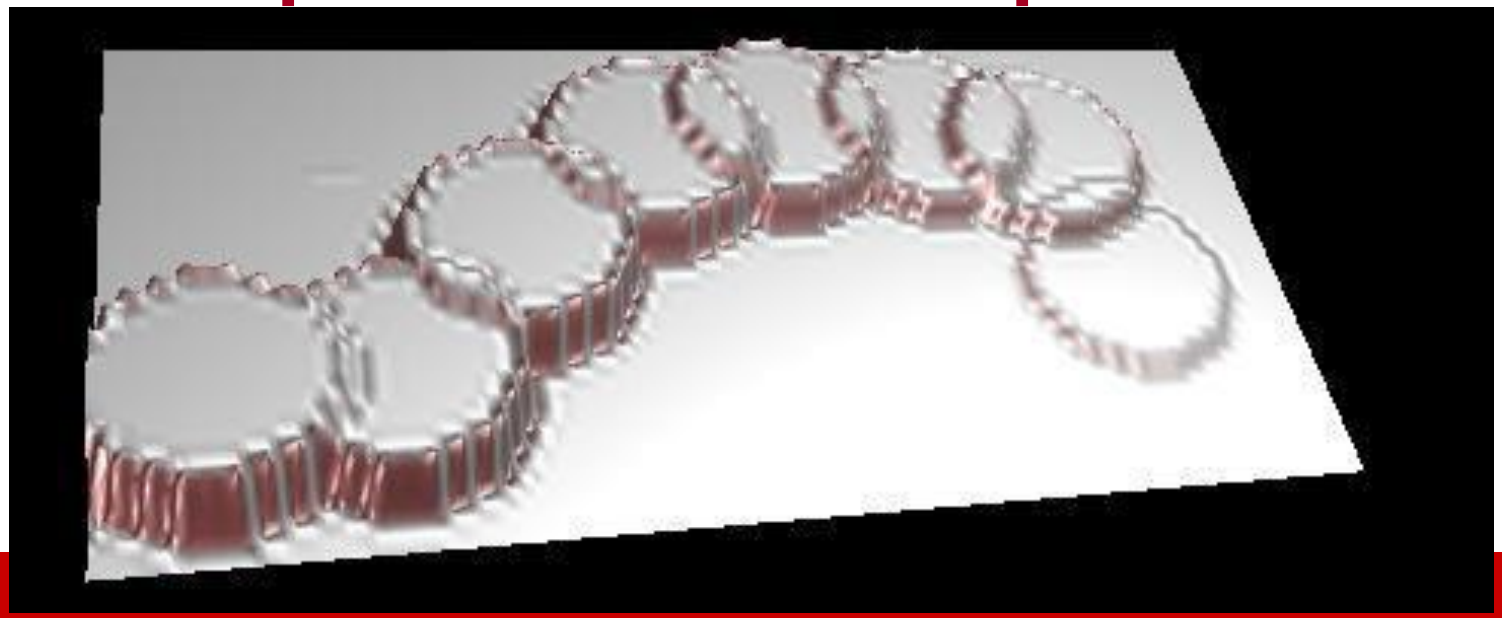


approximant

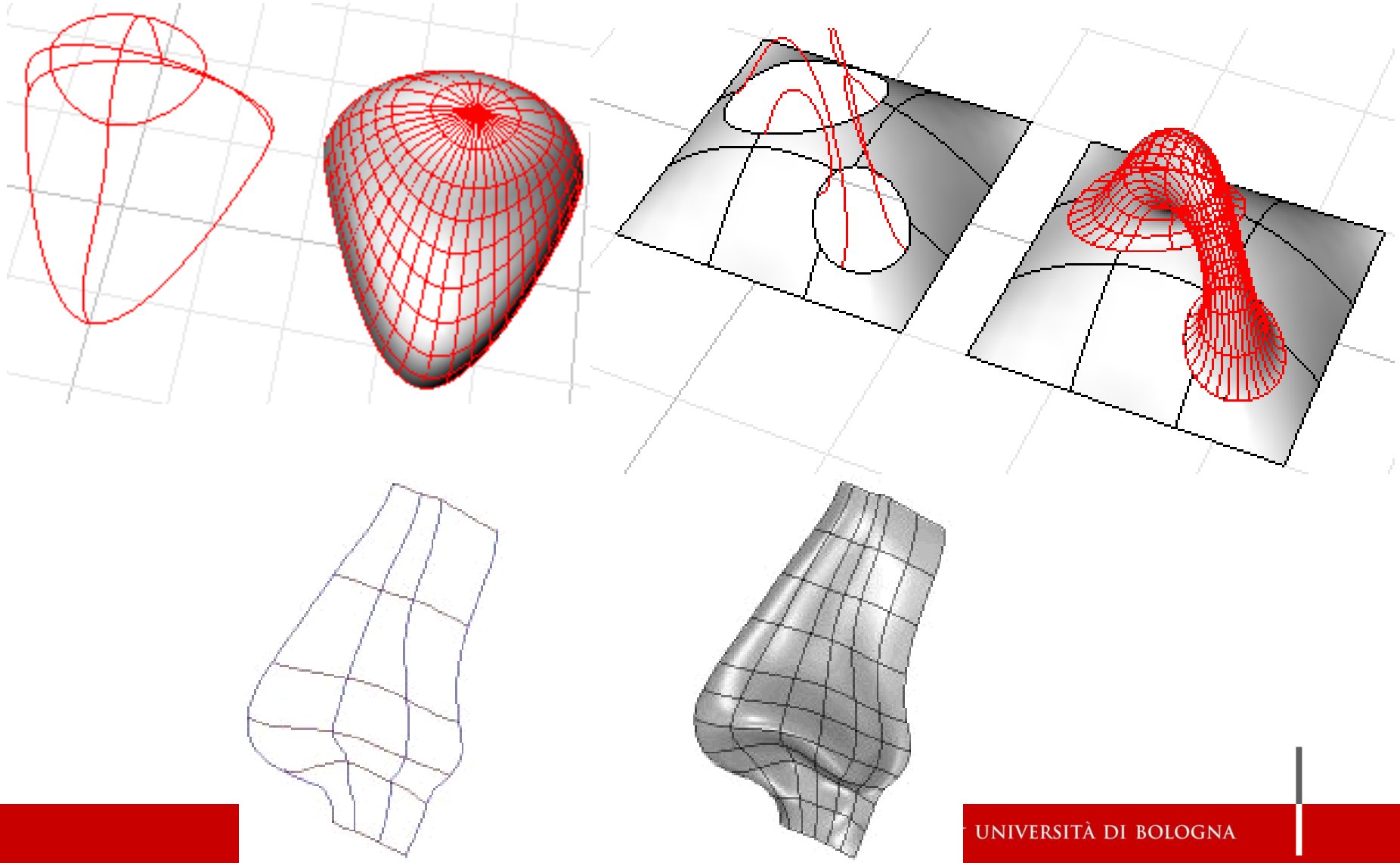


From a gray scale image

.. to a spline surface interpolant



Spline Surface by interpolating a network of curve



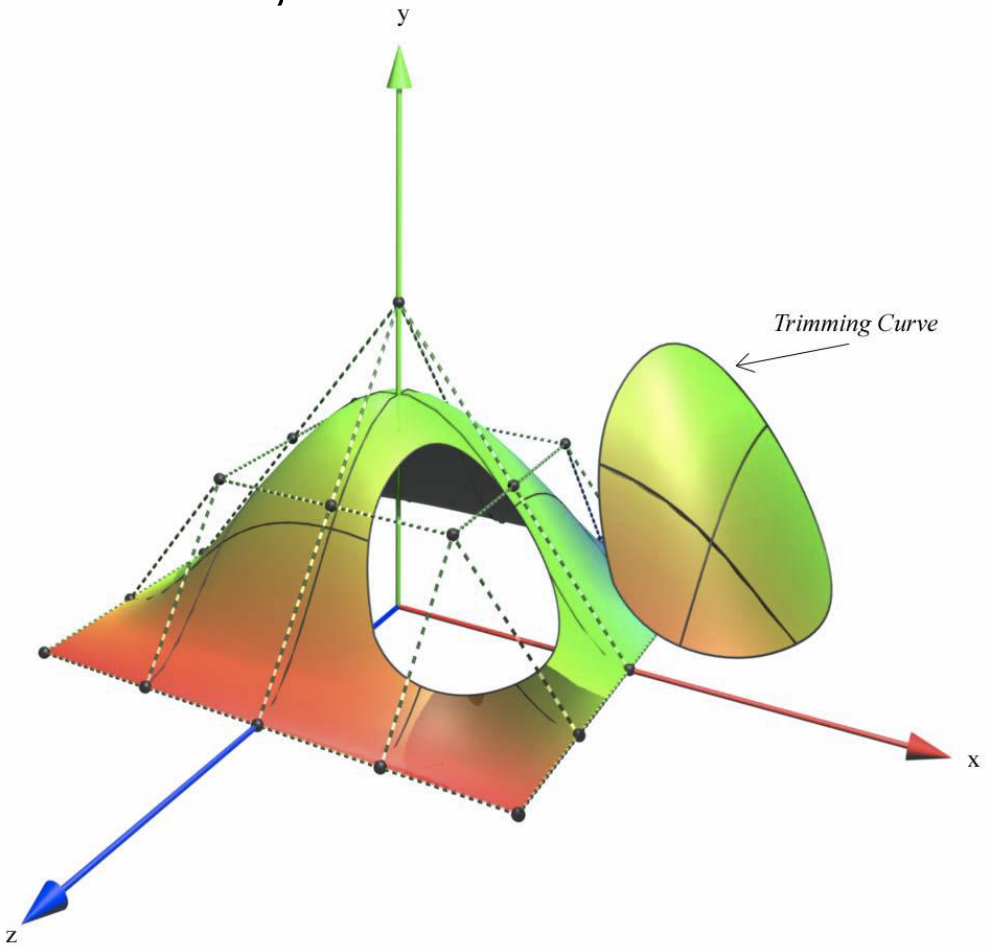
NURBS limits

- NURBS surfaces have rectangular topology

- Arbitrary topologies can be obtained by collapsing CP, which can cause bad parameterizations

Trimmed NURBS surfaces

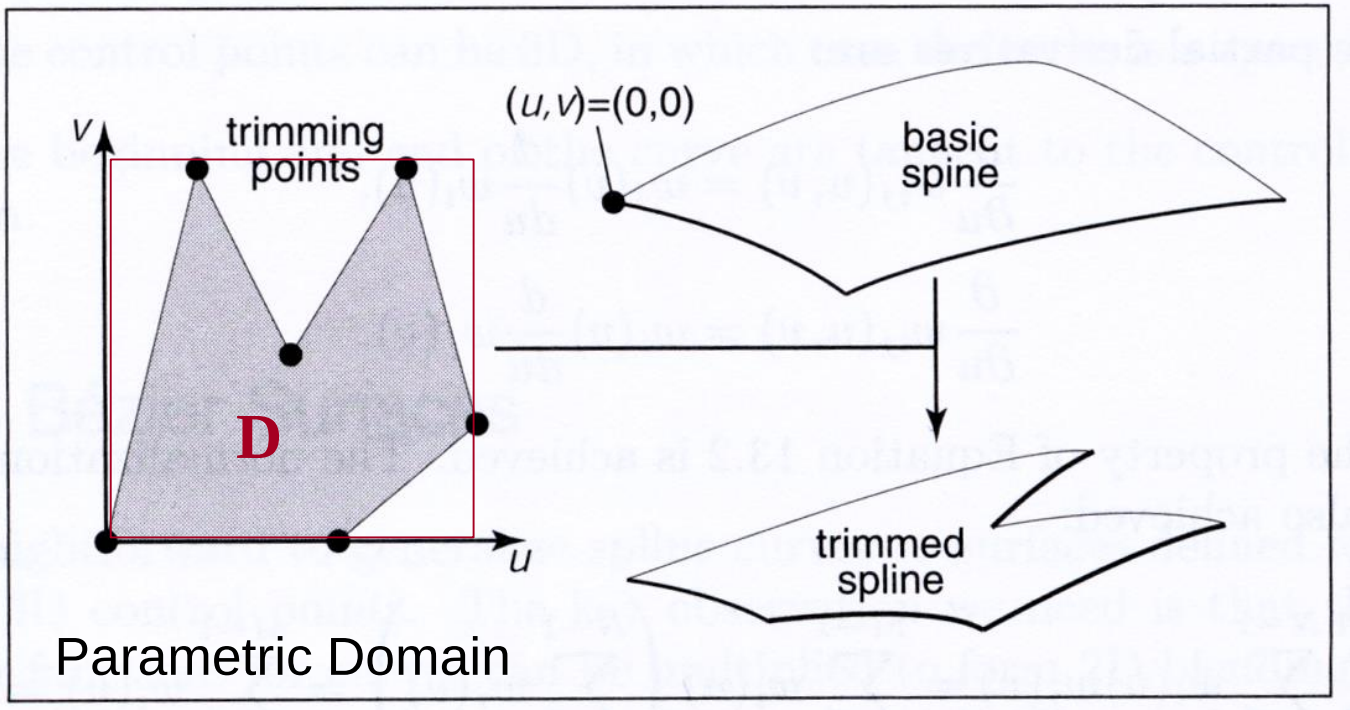
A trimmed NURBS surface is one in which specified patches have been trimmed out, or removed.



Trimmed NURBS surfaces

The surface S(u,v) is limited to a subdomain D in UxV of the parametric space, called Trimming Region (TR)

- TR specifies in the parametric domain regions of interest
- Visualize the original surface S(u,v) only on TR

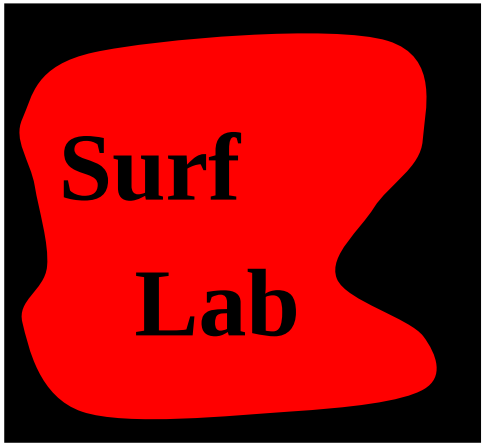


Trimming Region

- A trimming region is defined by a set of closed trimming loops in the parameter space of a surface.
- A trimming loop consists of a closed NURBS curve and/or piecewise linear curve.
- Self intersecting curves are not allowed.

$$c_k(t) = (x_k(t), y_k(t)) =$$

$$c_k(t) = \frac{\sum_{i=1}^{pk+K} w_i P_i N_{i,pk}(t)}{\sum_{i=1}^{pk+K} w_i N_{i,pk}(t)} \qquad k = 1, .., M$$



Parametric Domain

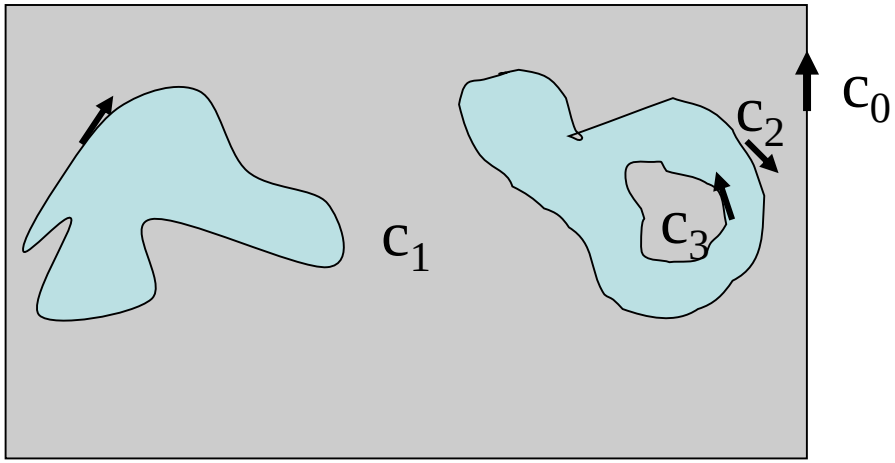
Trimming region

Loops (curves c(t)) may be nested, but a nested loop must be oriented oppositely from the loop that contains it. The outermost loop must be oriented counter-clockwise.

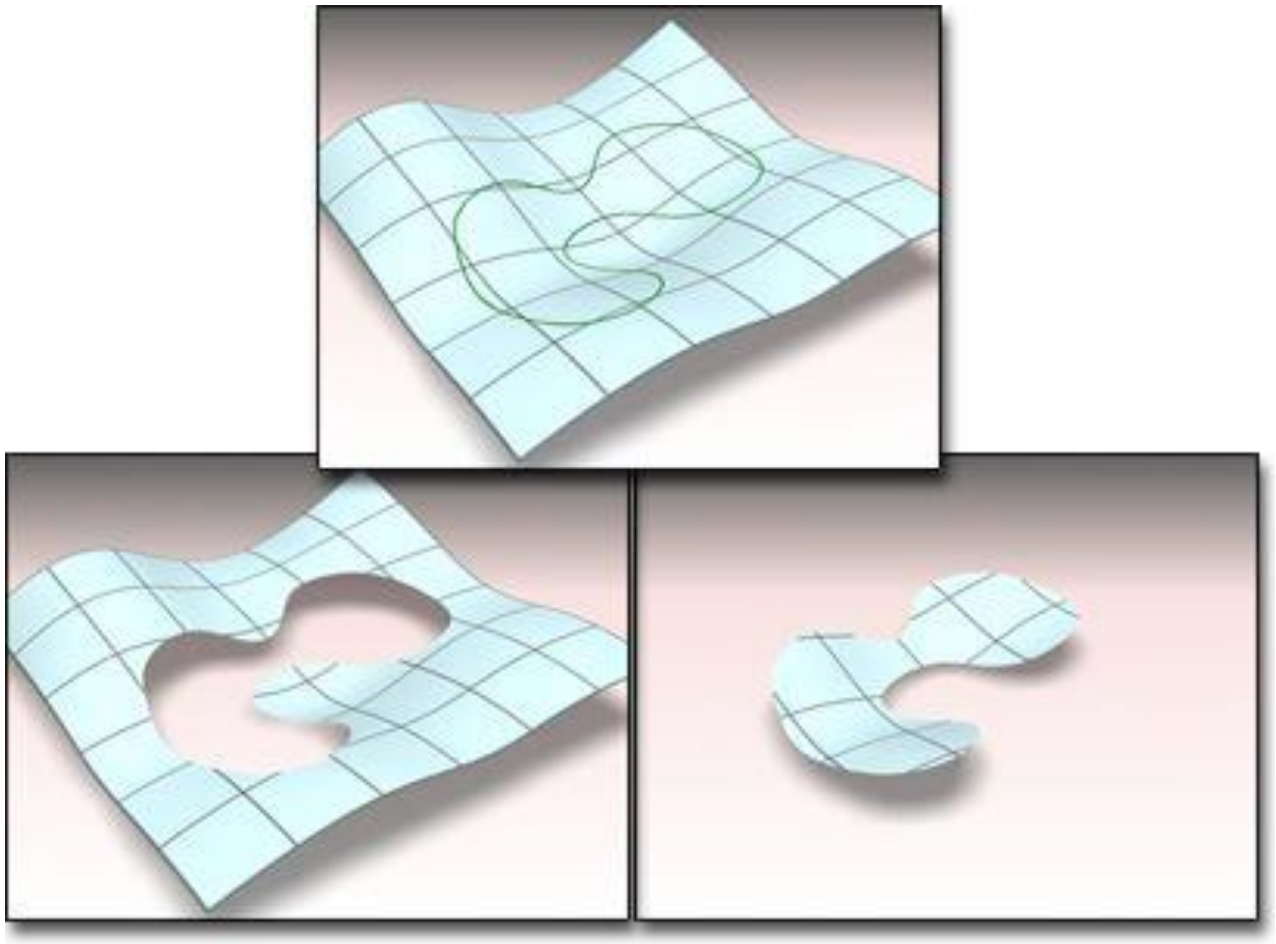
Outer loops: oriented counter-clockwise

Inner loops: oriented clockwise

c_0 outer
c_1 inner
c_2 inner
c_3 outer



The trimming region D is defined by the left-region of the outer loop removing the right-regions of the inner loops.



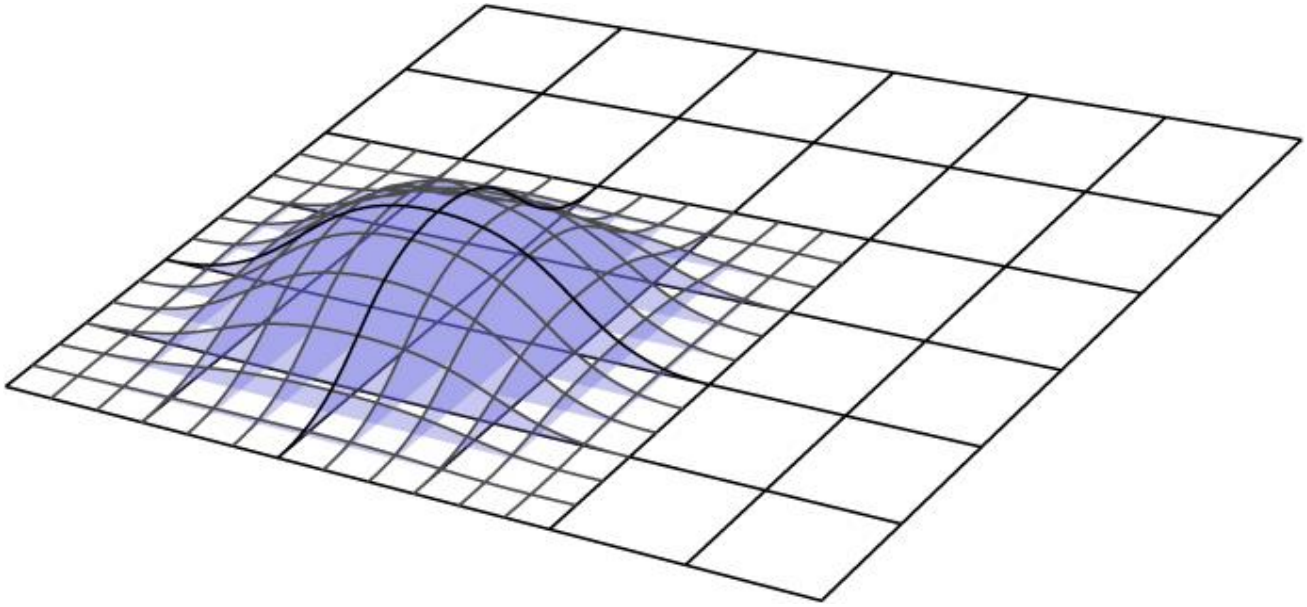
Trimmed NURBS surfaces

Applications:

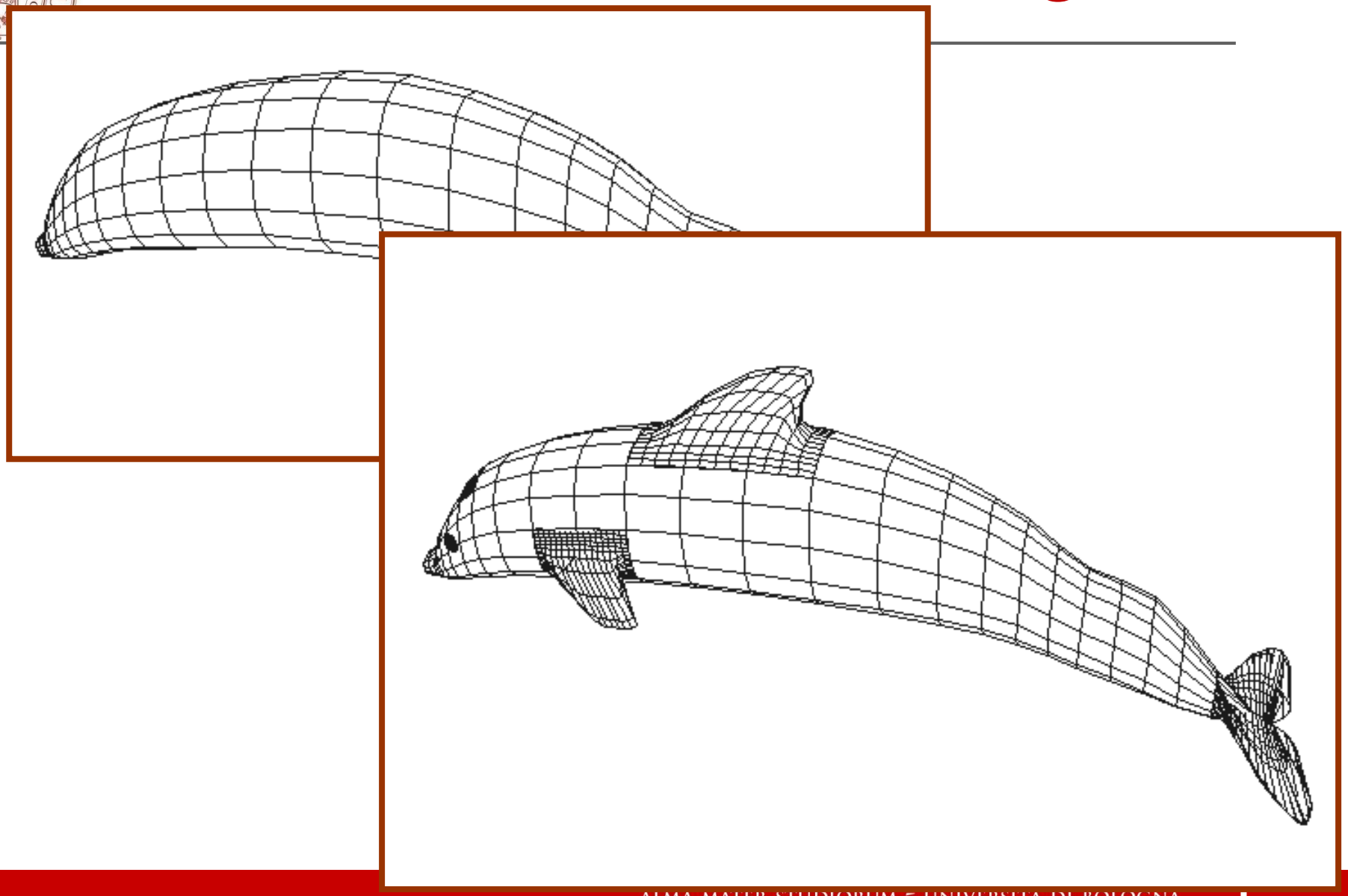
- **Hierarchical modelling**
- Composing solids by Boolean Operations

Trimmed NURBS surfaces: hierarchical modelling

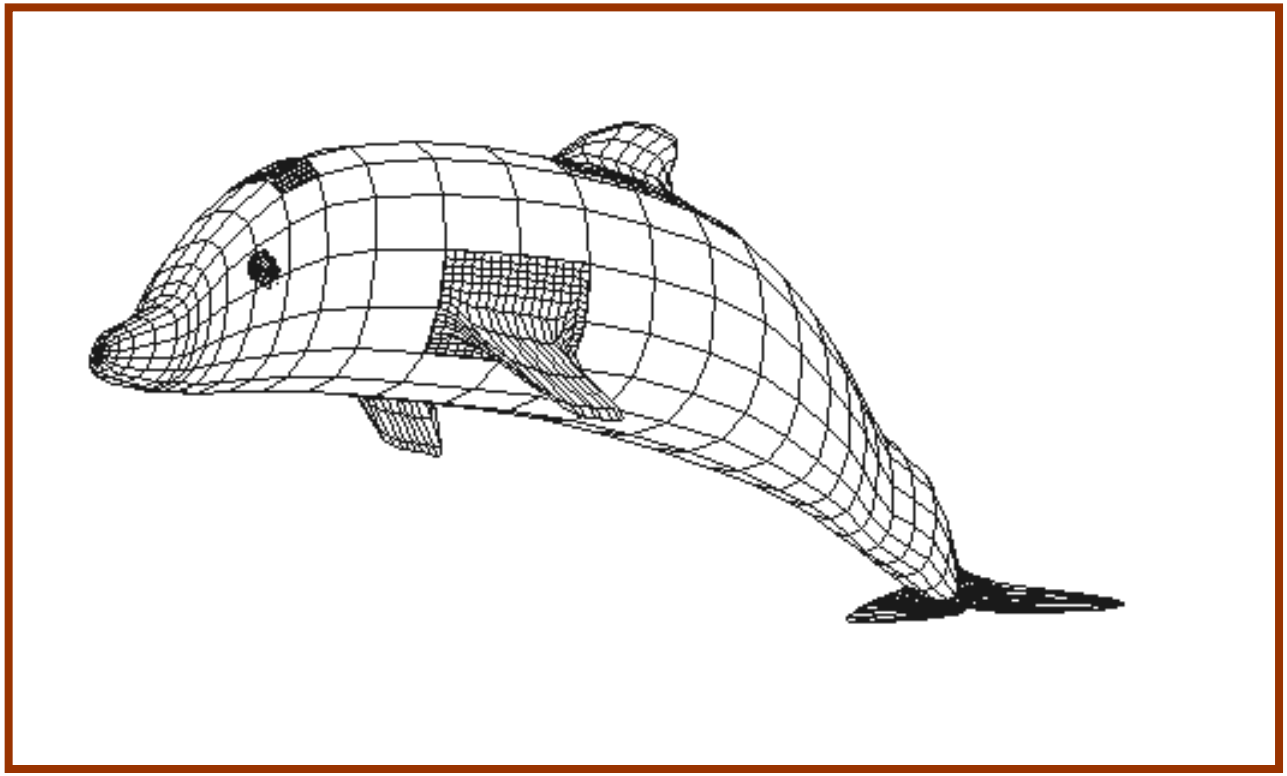
- Using Trimmed NURBS
  - Restrict the domain into regions of interest
  - The original surface is unchanged
  - Construct local details on the trimming regions
- Locally modify the surface adding geometric details without any parametric changes.



Hierarchical modelling



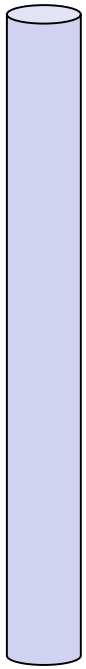
Hierarchical modelling

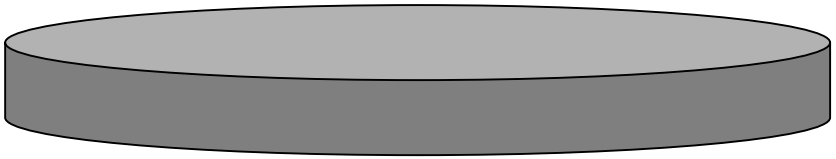


Hierarchical modeling

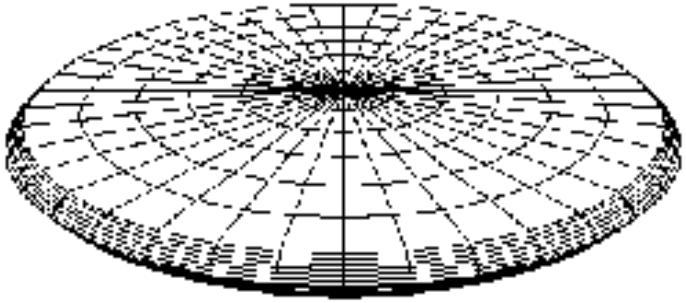
Courtesy of CGGroup , xcmodel , Univ. Bologna

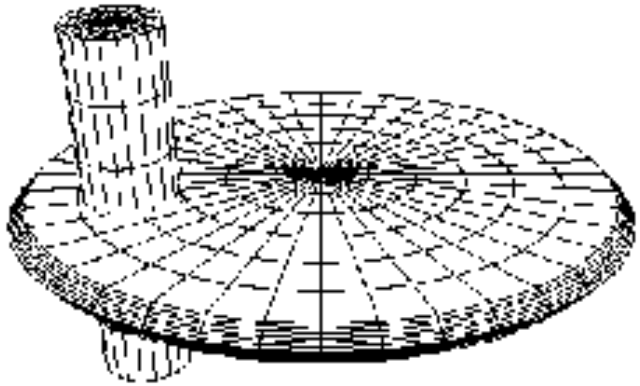
Trimmed NURBS surfaces: solid modeling by boolean operations

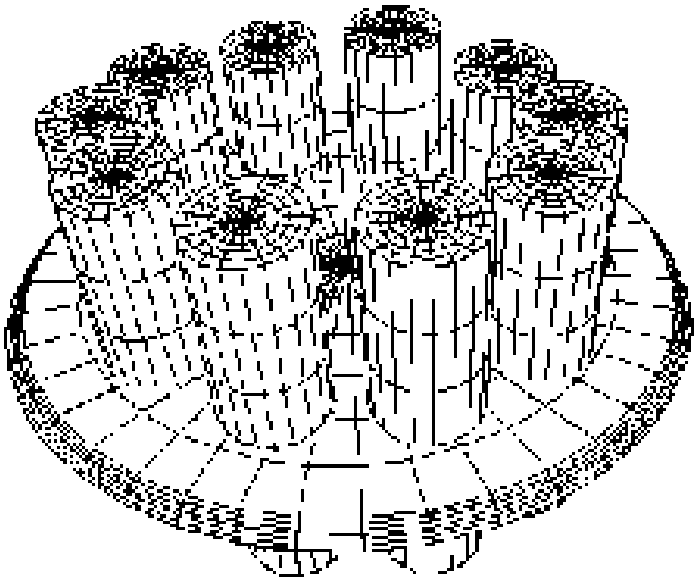


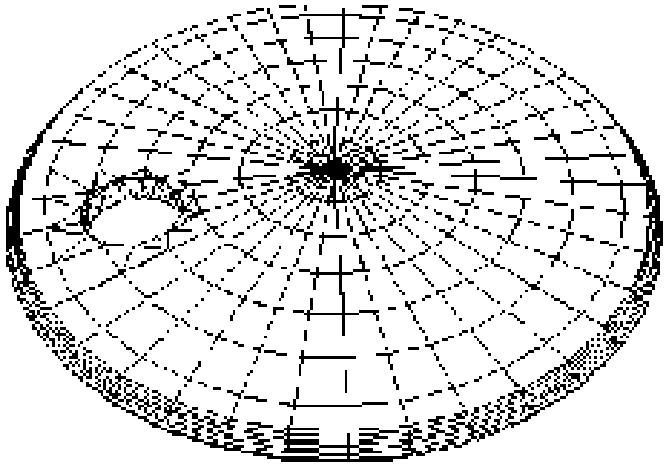


Trimmed NURBS: solid modeling



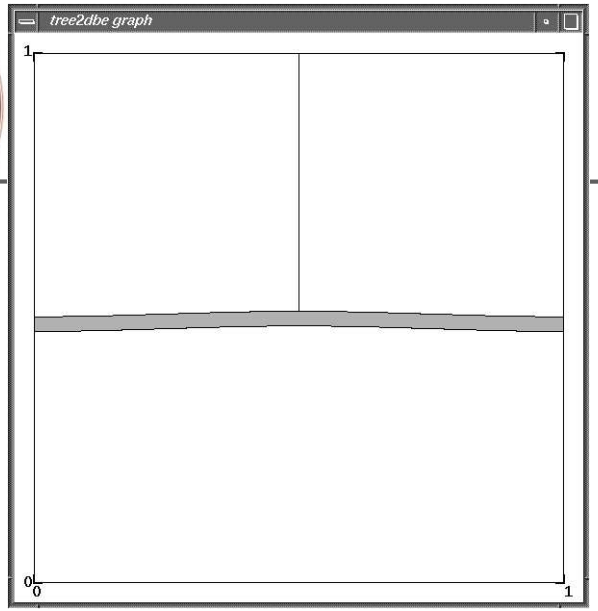




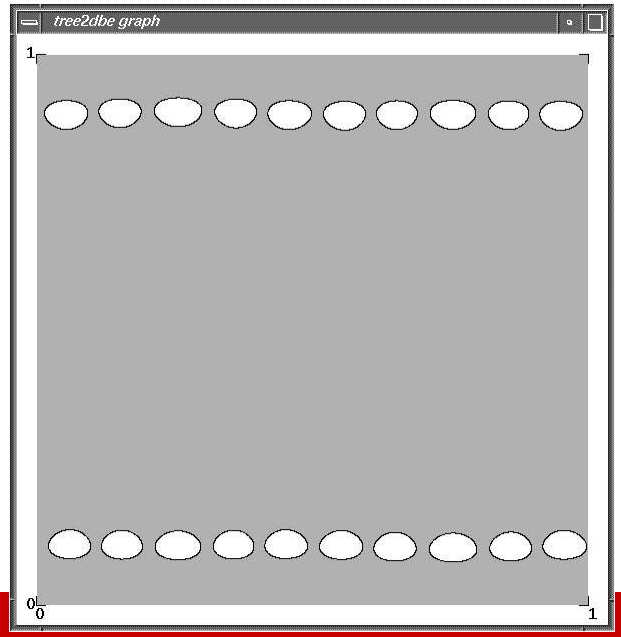


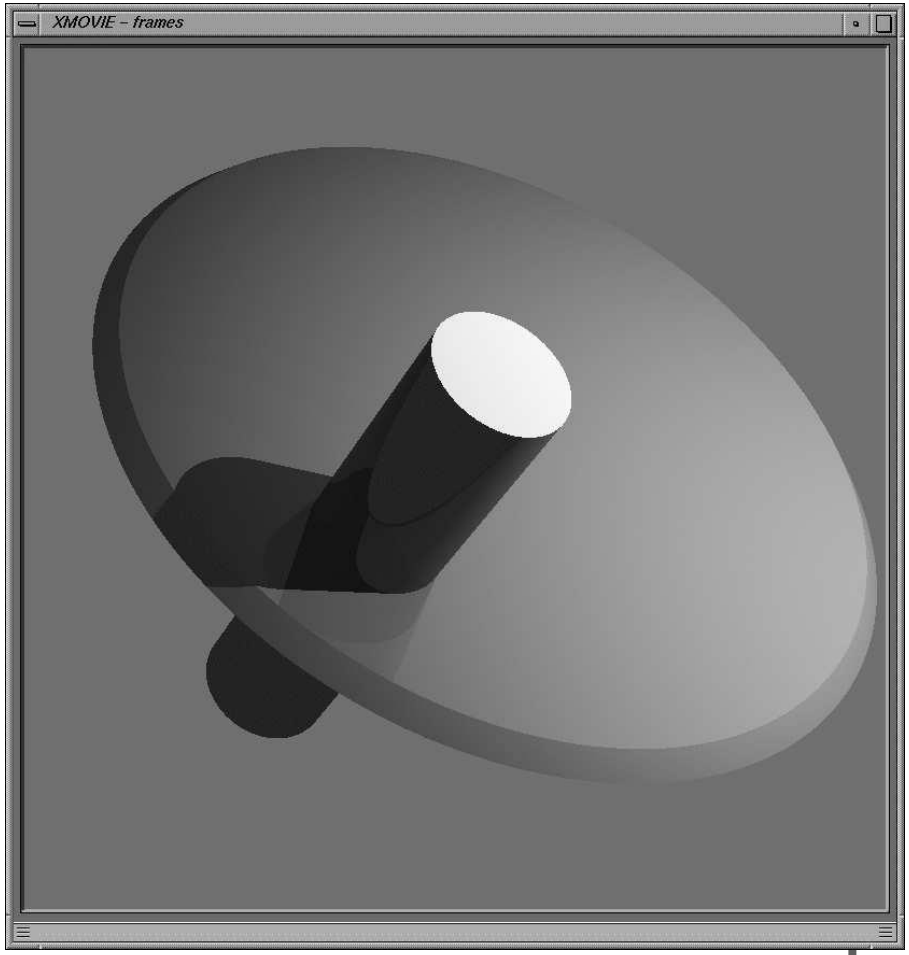
Composing solids: Difference between a disk and a cylinder

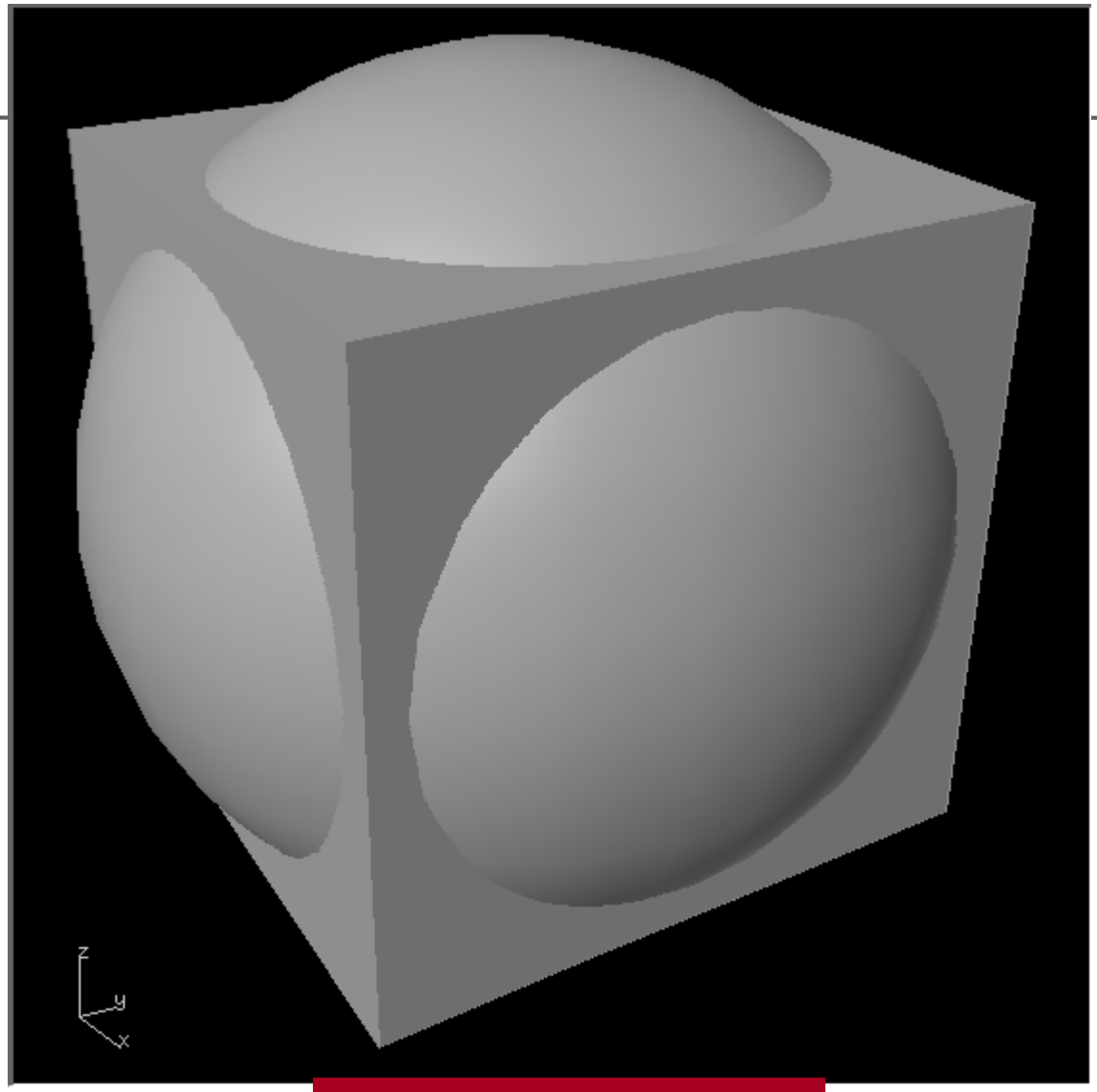
Cylinder part



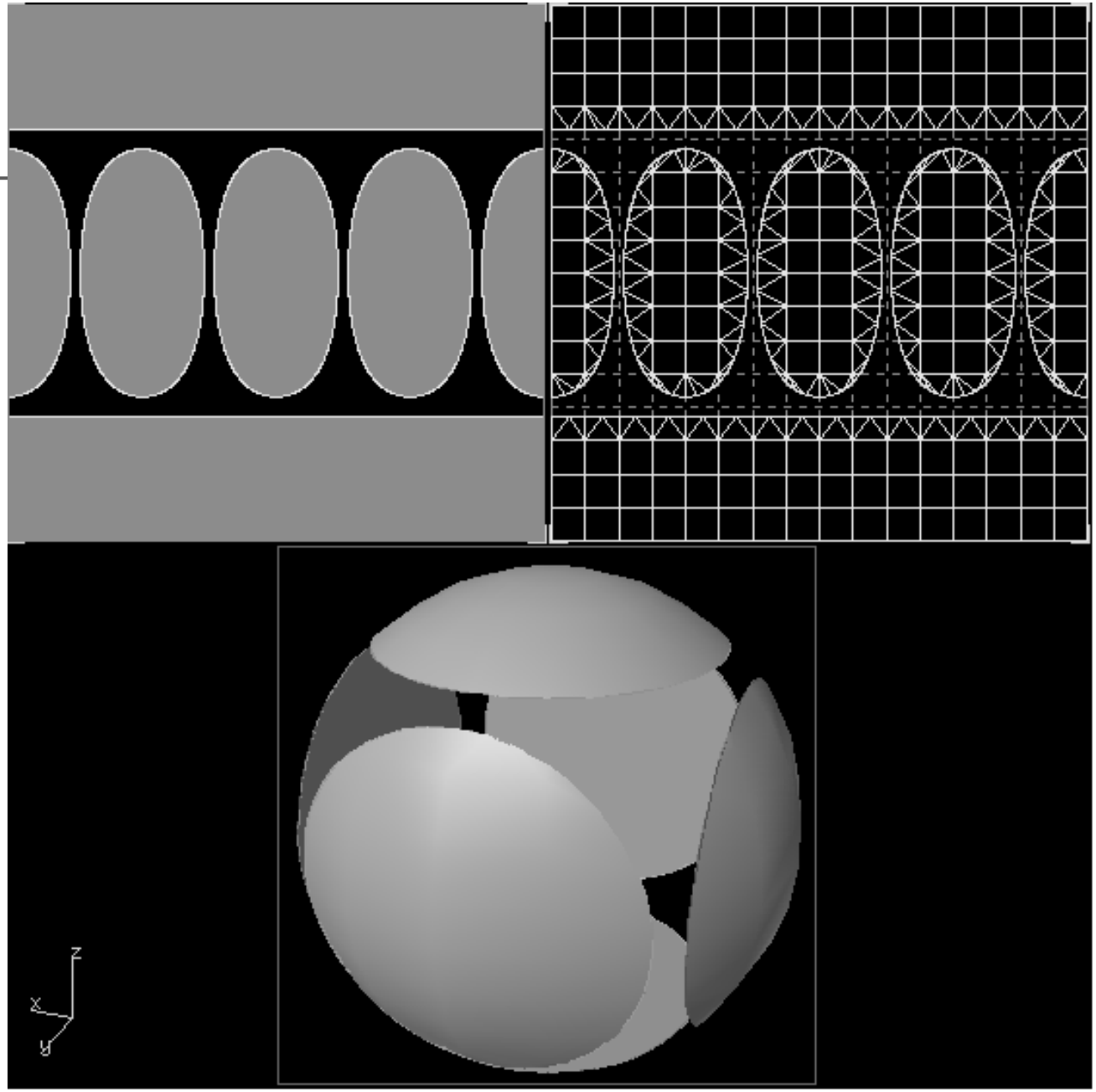
Disk part



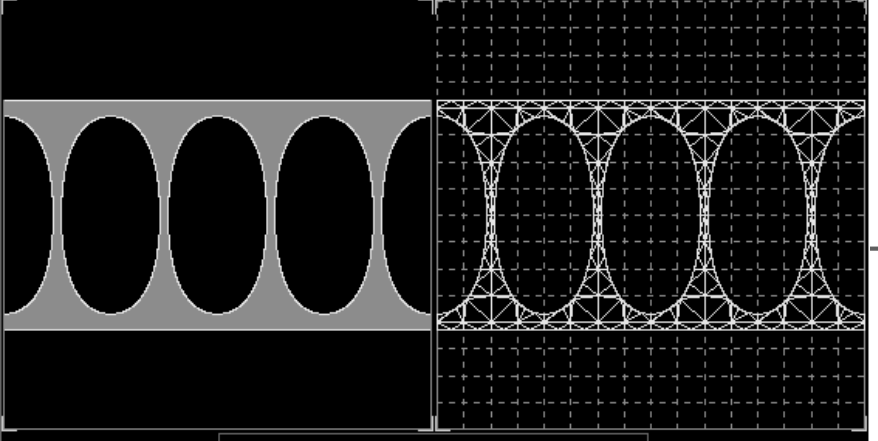


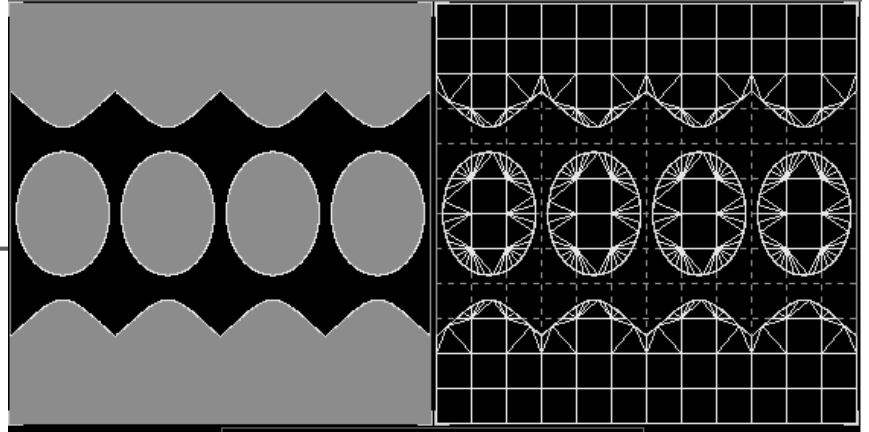


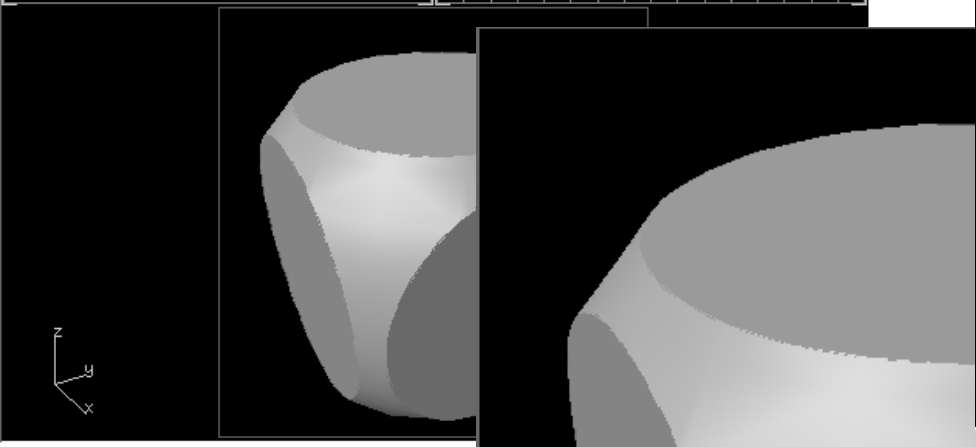
Sphere union Cube

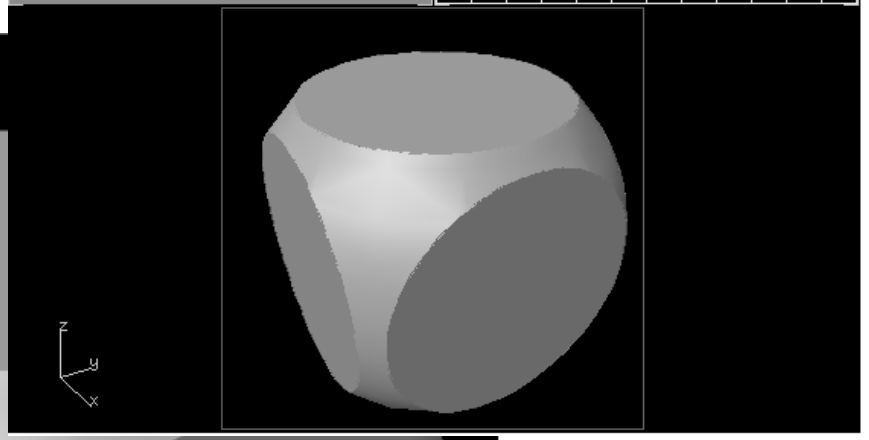


Sphere - Cube









sfera

cubo

Sphere intersection cube

Cross sectional design: from curves to surfaces

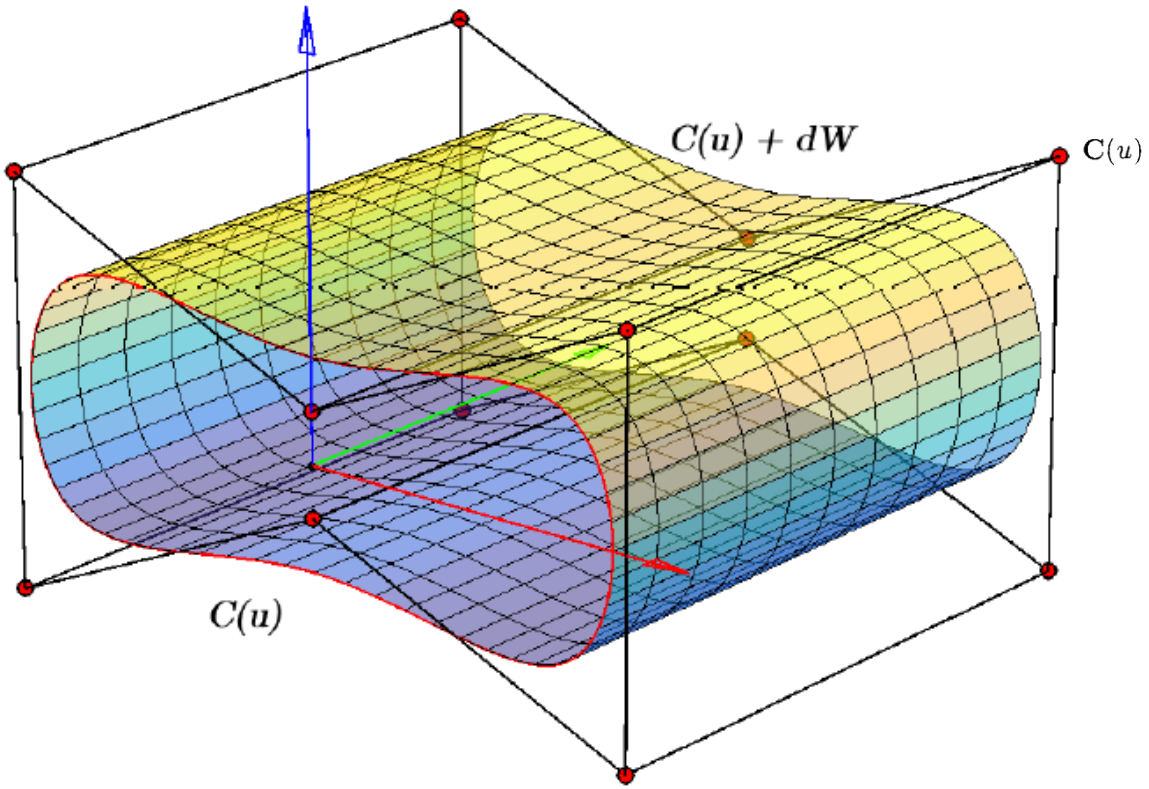
Model the shape of a surface by modifing its 3D CP in a 2D window is a difficult task,

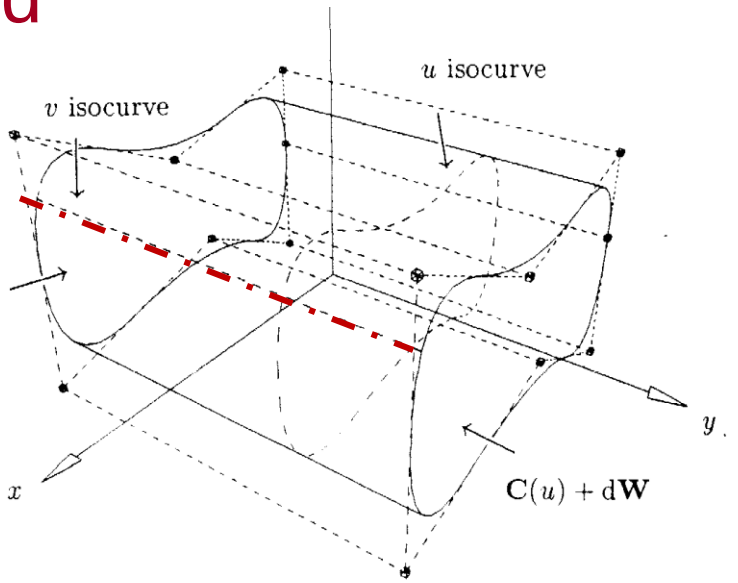
We need tools to construct surfaces from curves automatically.

- **Extruded Surface**
- **Ruled Surface**
- Surfaces of Revolution
- **Skinned Surface**
- **Swept Surface**
- **....**

Extruded Surface

Obtained by moving a profile curve c(u) in a given direction W for a a given distance d





Extruded Surface

$$c(u) = \sum_{i=1}^{n+K} P_i N_{i,n}(u) \quad \text{knot vector } U$$

- Extrusion direction: W (unitary vector in v direction)
- Extrusion offset: d

- For fixed u, s(u,v) is a straight segment from c(u) to c(u)+dW
- For fixed v , s(u,v) is the curve

$$s(u,\underline{v}) = c(u) + \underline{v}dW = \sum_{i=1}^{n+K} (P_i + \underline{v}dW) N_{i,n}(u)$$

Extruded Surface

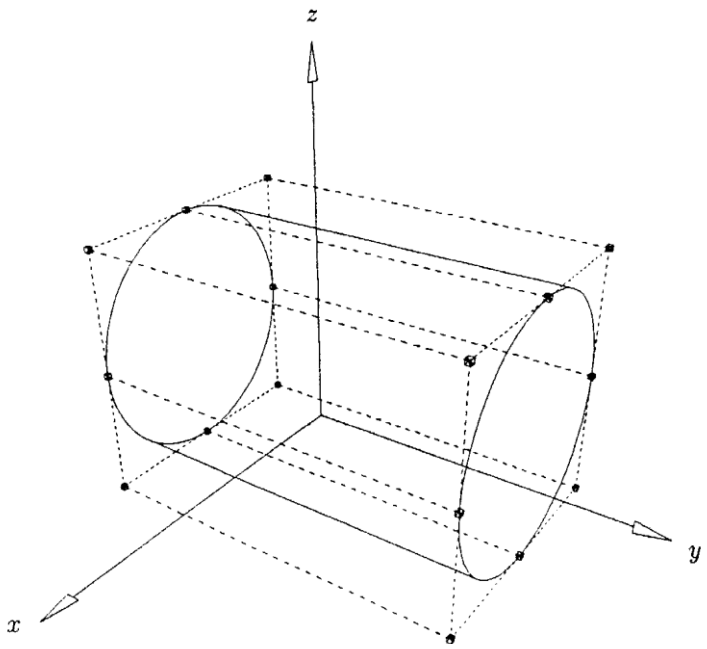
Extruded surface :

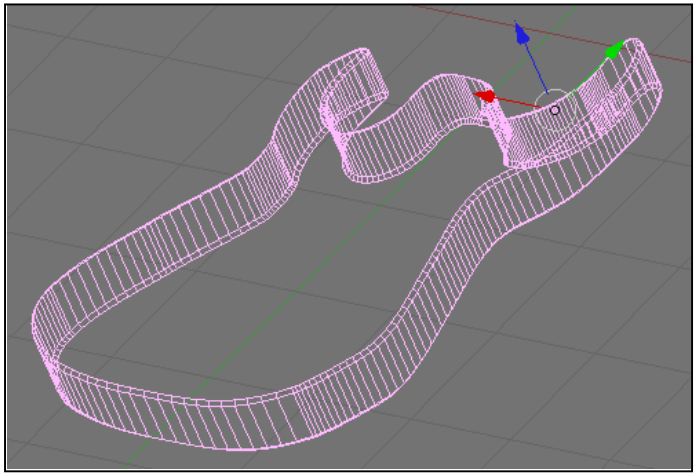
$$s(u,v) = \sum_{i=1}^{n+K} \sum_{j=1}^{2} P_{ij} N_{i,n}(u) N_{j,2}(v)$$

- Control points $\mathbf{P_{i1}=P_i;}$ $\mathbf{P_{i2}=P_i+dW}$;
- Knot vectors: UxV, **V=[0,0,1,1]**

- If c(u) is rational (NURBS) with weights w_i, then s(u,v) is rational with weights $\mathbf{w_{i1}=w_{i2}=w_i}$

Extruded Surfaces





$$s(u,v) = \sum_{i=1}^{9}\sum_{j=1}^{2} P_{ij} N_{i,3}(u) N_{j,2}(v)$$

The cylinder is obtained by traslating the NURBS circle (9 points) a distance d along a vector normal to the plane of the circle.

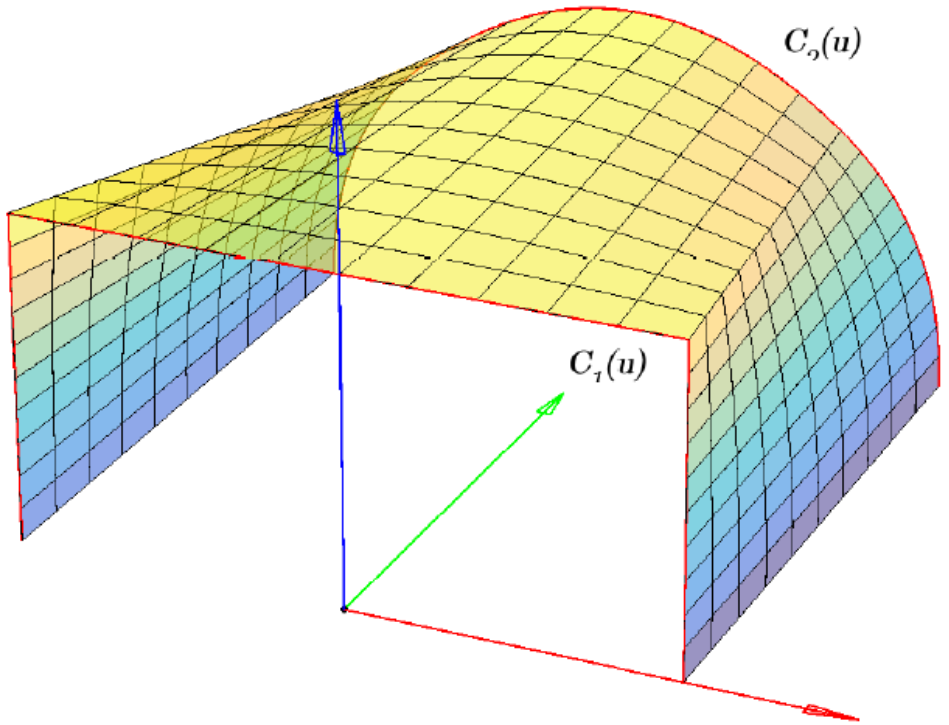
Ruled Surface

Obtained by linear interpolation in v direction between curves **$c_1(u)$** and **$c_2(u)$** defined on U parametric domain.
For fixed u, s(u,v) is a straight segment joining c_1 and c_2

$$c_1(u) = \sum_{i=1}^{n+K} P_i N_{i,n}(u)$$

$$c_2(u) = \sum_{j=1}^{n+K} T_j N_{j,n}(u)$$

Same degree n-1
Same knot vector U



Ruled Surface

The spline ruled surface on the parametric domain UxV, V=[0,0,1,1] is defined as :

$$s(u,v) = \sum_{i=1}^{n+K} \sum_{j=1}^{2} w_{ij} P_{ij} N_{i,n}(u) N_{j,2}(v)$$

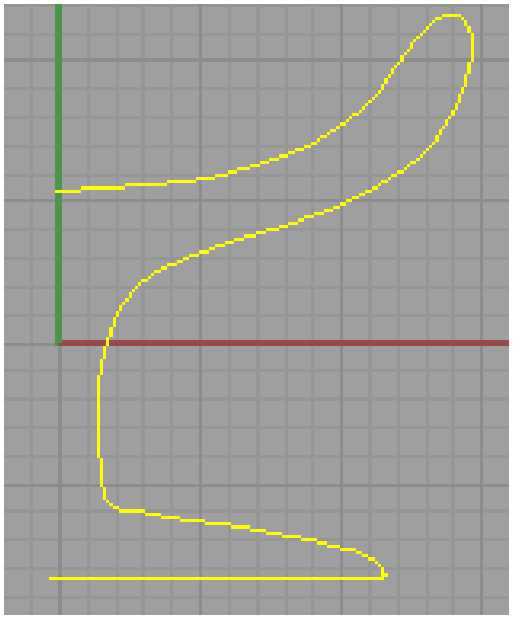
$$p_{i1} = P_i; \quad p_{i2} = T_i;$$

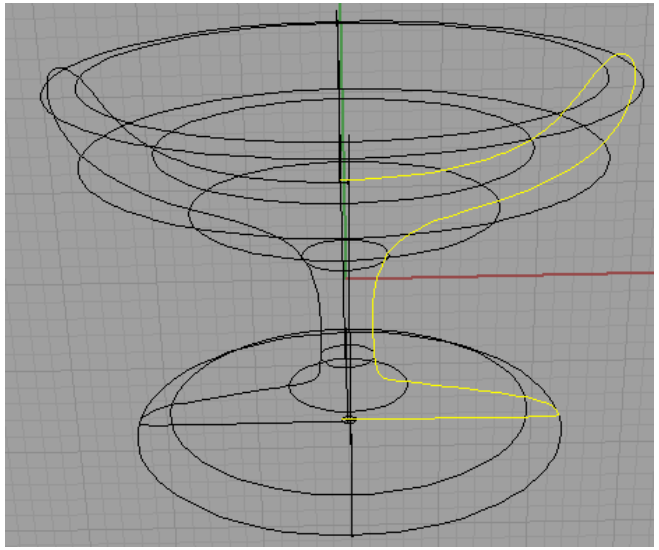
If the two curves are NURBS, then the ruled surface is rational as well, with weights

$$w_{i1} = w_i^{[1]}; \quad w_{i2} = w_i^{[2]};$$

Surfaces of revolution

Given a profile curve **c**(*t*) in the plane, the surface is defined by spinnig it through an arbitrary angle around an axis





Surfaces of revolution

Profile curve in the x-z plane (revolve it about the z axis):

$$c(v) = \sum_{j=1}^{m+K} P_j N_{j,m}(v) \quad Knot\ vector\ V$$

$$P_j = (x_j, 0, z_j)^T$$

Control points

For fixed u=u0, s(u0,v) is the isoparametric curve c(v) rotated by a given angle around the z axis

For fixed v=v0, s(u,v0) is a circle in x-y plane, with its center on the z axis

Surfaces of revolution

NURBS surface of revolution s(u,v):

$$s(u,v) = \sum_{i=1}^{9} \sum_{j=1}^{m+K} P_{ij} R_{i,3}(u) R_{j,m}(v)$$

circle in u

Knot vector for a 9-points circle

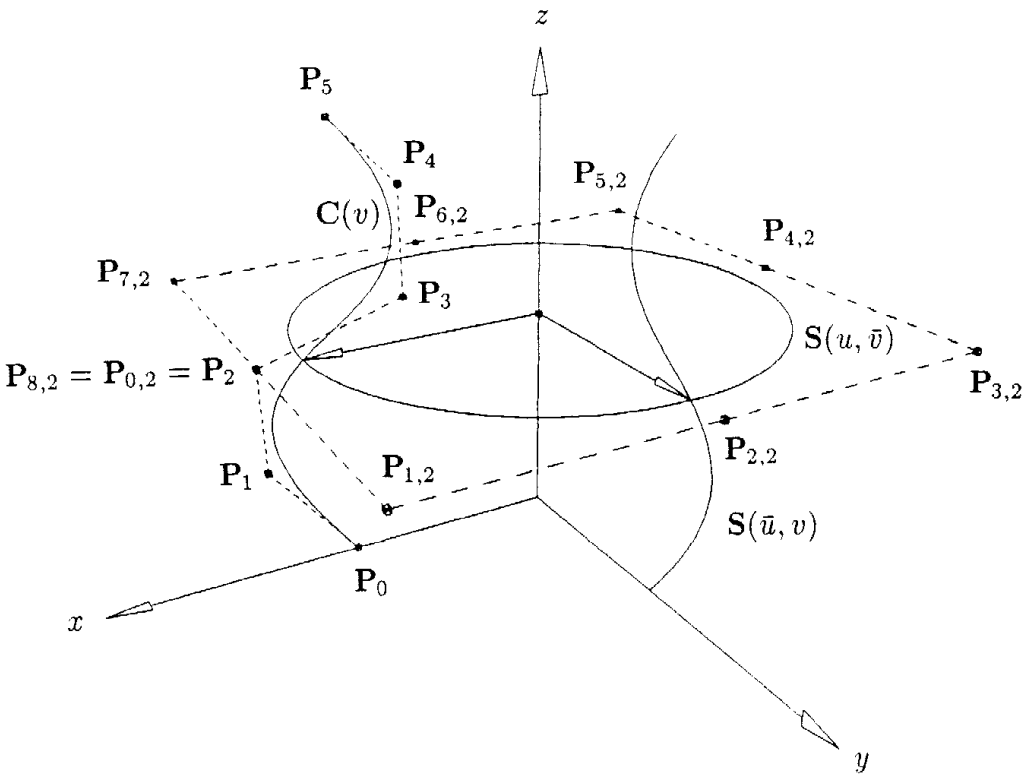
$$U = \{0,0,0,1/4,1/4,1/2,1/2,3/4,3/4,1,1,1\}$$

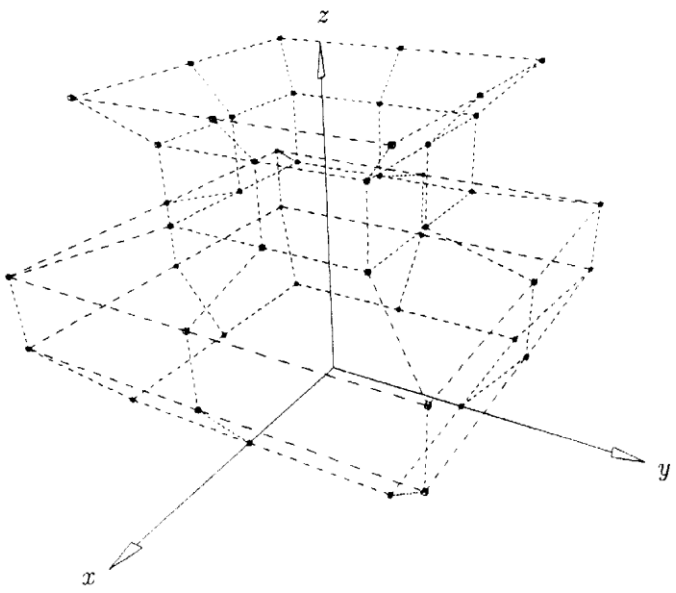
$$w_i = \left\{1,\sqrt{2}/2,1,\sqrt{2}/2,1,\sqrt{2}/2,1,\sqrt{2}/2,1\right\}$$

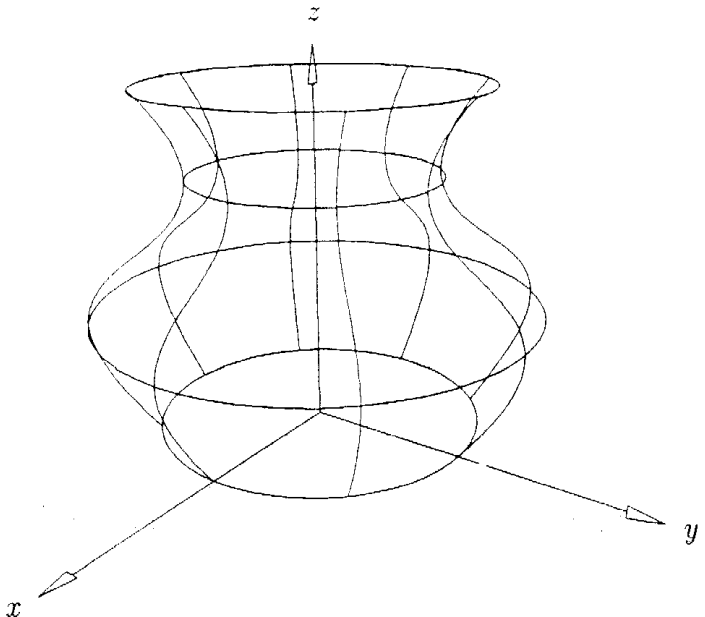
$$P_{ij} = \begin{cases} P_j \quad i = 1 \\ \text{rotation } P_j \text{ of } 45° \quad i = 2,3,..,9 \\ \text{(for fixed j CP lie on the } z{=}zj \text{ plane)} \end{cases}$$

Weights for the NURBS circle curve

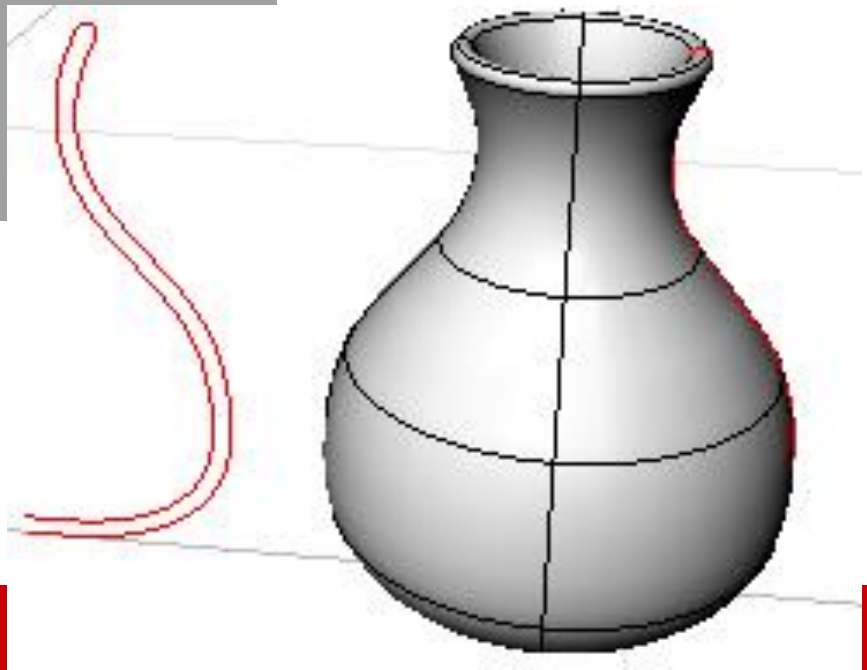
$$w_{ij} = w_i w_j$$


z
$\mathbf{P}_5$
$\mathbf{P}_4$
$\mathbf{C}(v)$
$\mathbf{P}_{6,2}$
$\mathbf{P}_{5,2}$
$\mathbf{P}_{4,2}$
$\mathbf{P}_{7,2}$
$\mathbf{P}_3$
$\mathbf{S}(u, \bar{v})$
$\mathbf{P}_{8,2} = \mathbf{P}_{0,2} = \mathbf{P}_2$
$\mathbf{P}_{3,2}$
$\mathbf{P}_{2,2}$
$\mathbf{P}_1$
$\mathbf{P}_{1,2}$
$\mathbf{S}(\bar{u}, v)$
$\mathbf{P}_0$
x
y


z
x
y


z
x
y

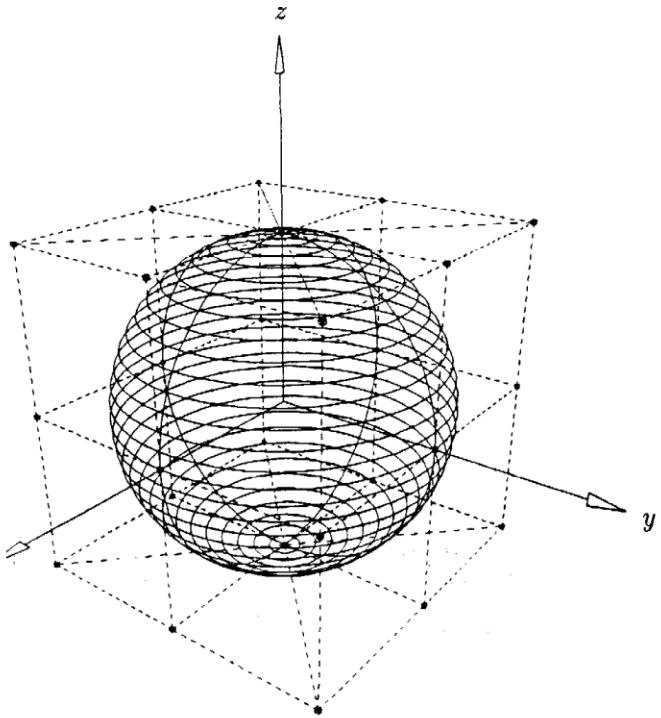
Example: profile curve and Surfaces of revolution



Sphere as a revolution NURBS surface

Revolving about the z-axis a half-circle (unitary ray, centered at the origin) c(u) of order 3

$$c(u) = \frac{\sum_{i=1}^{5} w_i P_i N_{i,3}(u)}{\sum_{j=1}^{5} w_j N_{j,3}(u)}$$

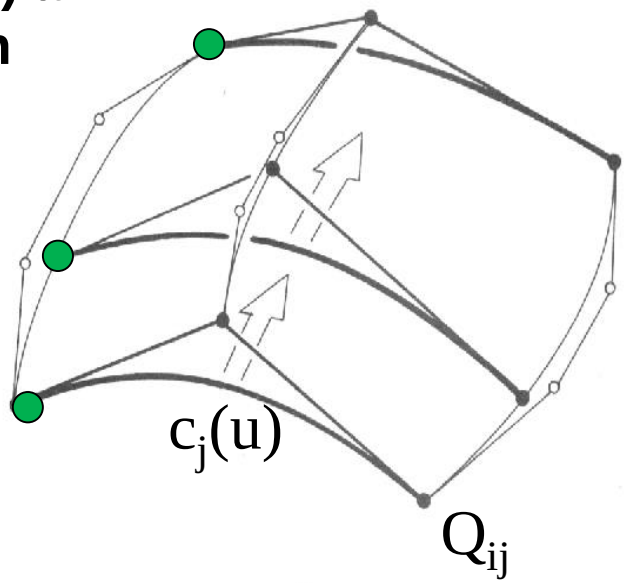


$$U = \left\{0,0,0,\frac{1}{2},\frac{1}{2},1,1,1\right\}, \quad W = \left\{1,\frac{\sqrt{2}}{2},1,\frac{\sqrt{2}}{2},1\right\}$$

CP at the North and South poles are repeated 9 times,

Skinned Surface

Skinning is the process of interpolating (blending) a given set of NURBS curves (section curves with common degree and number of CP) to form a surface.

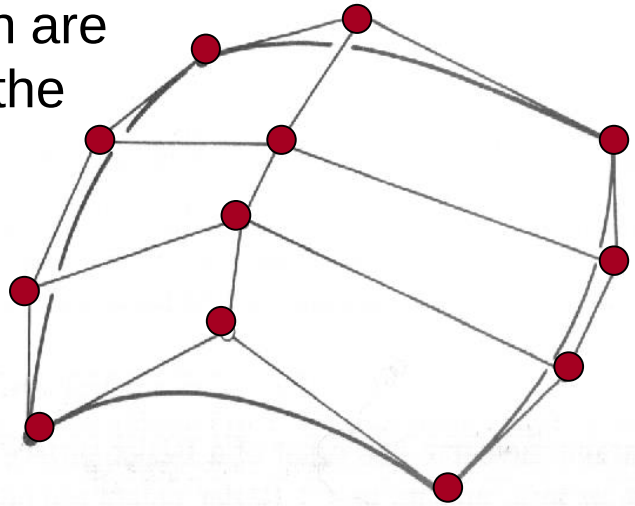


Section curves of degree n-1 and knot vector U:

$$c_j^w(u) = \sum_{i=1}^{n+K} Q_{ij}^w N_{i,n}(u), \quad j = 1,..,L+m$$

For each index i, the CP Q_{ij}^w (points ●) in v direction are interpolated in the homogeneous space, obtaining the curves

$$c_i^w(v) = \sum_{j=1}^{m+L} P_{ij}^w N_{i,m}(v), \quad i = 1,..,K+n$$

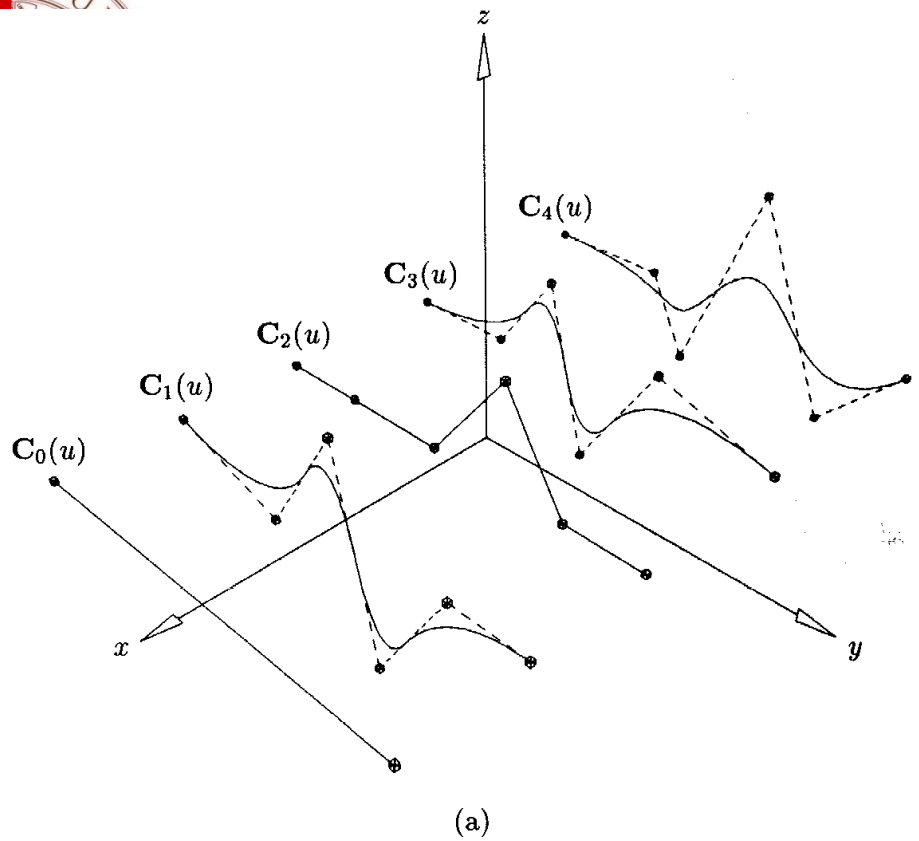


Skinned Surface

The skinned surface is defined by the computed CP $\mathbf{P}_{ij}^{w}$

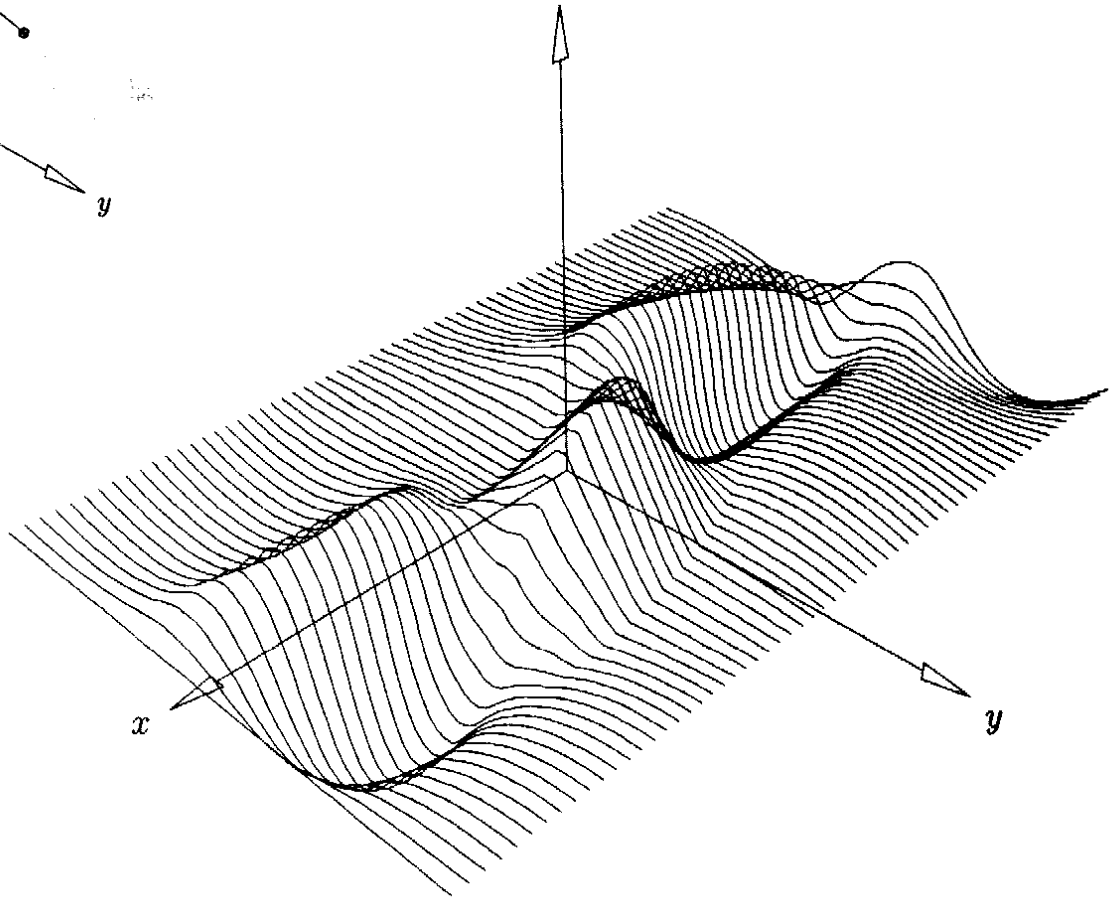
$$s^{w}(u,v) = \sum_{i=1}^{n+K} \sum_{j=1}^{m+L} P^{w}_{ij} N_{i,n}(u) N_{j,m}(v)$$

Example



(a)

Section curves



Skinned surface

Skinned Surface

ALGORITHM

INPUT: **m** — **degree** in direction v (m-1<=K).

$\{v_k\}$, k=1,..,m+L — **interpolation nodes**

V — **knot vector** in direction v

section curves : — **degree n, CP Q, U knot vector**

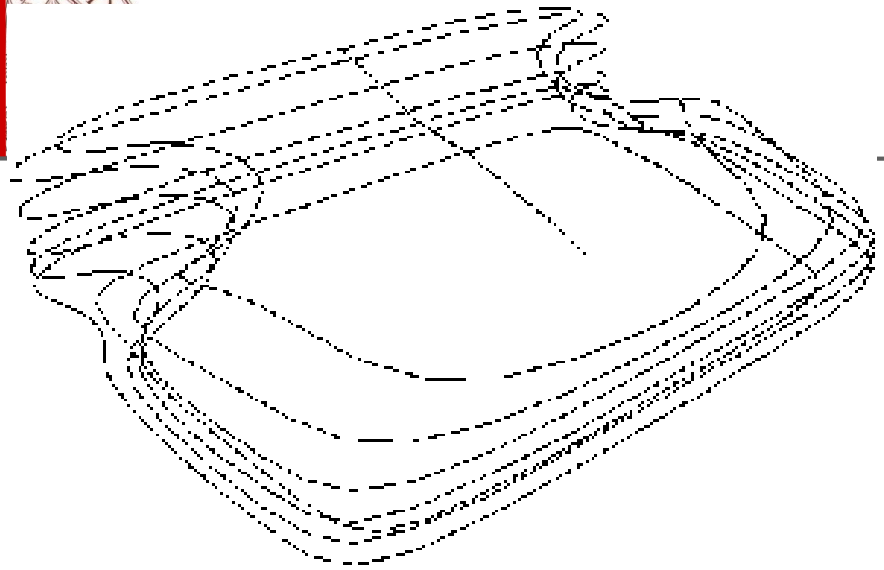
OUTPUT: **P_{ij}** control point of the skinned surface

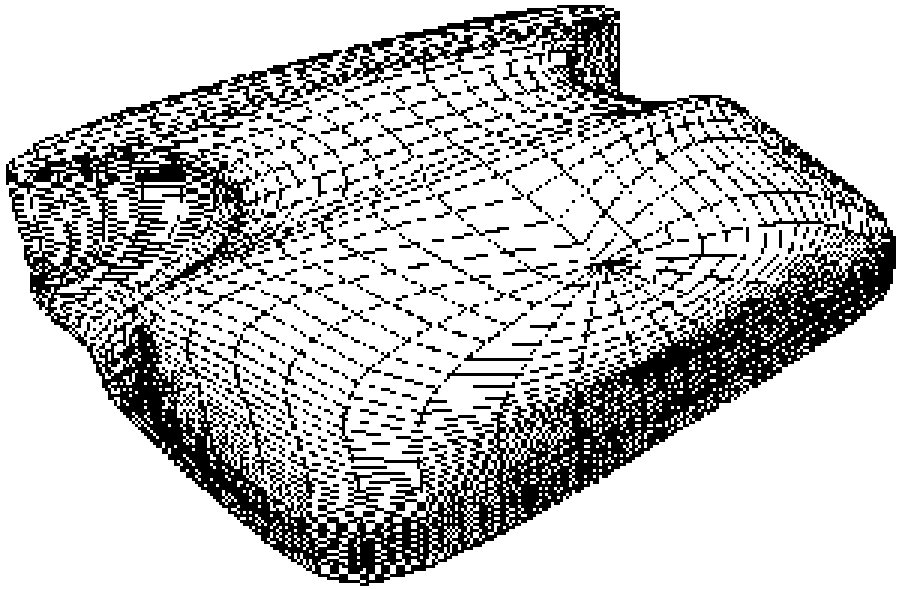
Compute n+K **interpolant curves** through the control points Q_{ij}^w of the section curves (solve n+K linear systems)

```
for i=1 to n+K
   P(i,:)=curveinterp_skinning(n+K,Q,m-1,V,v_k);
end
```

(P_{ij}^w is the j-th control point of the curve which interpolates the points $Q_{i1}^w,\ldots,Q_{im+L}^w$.)

Skinning example

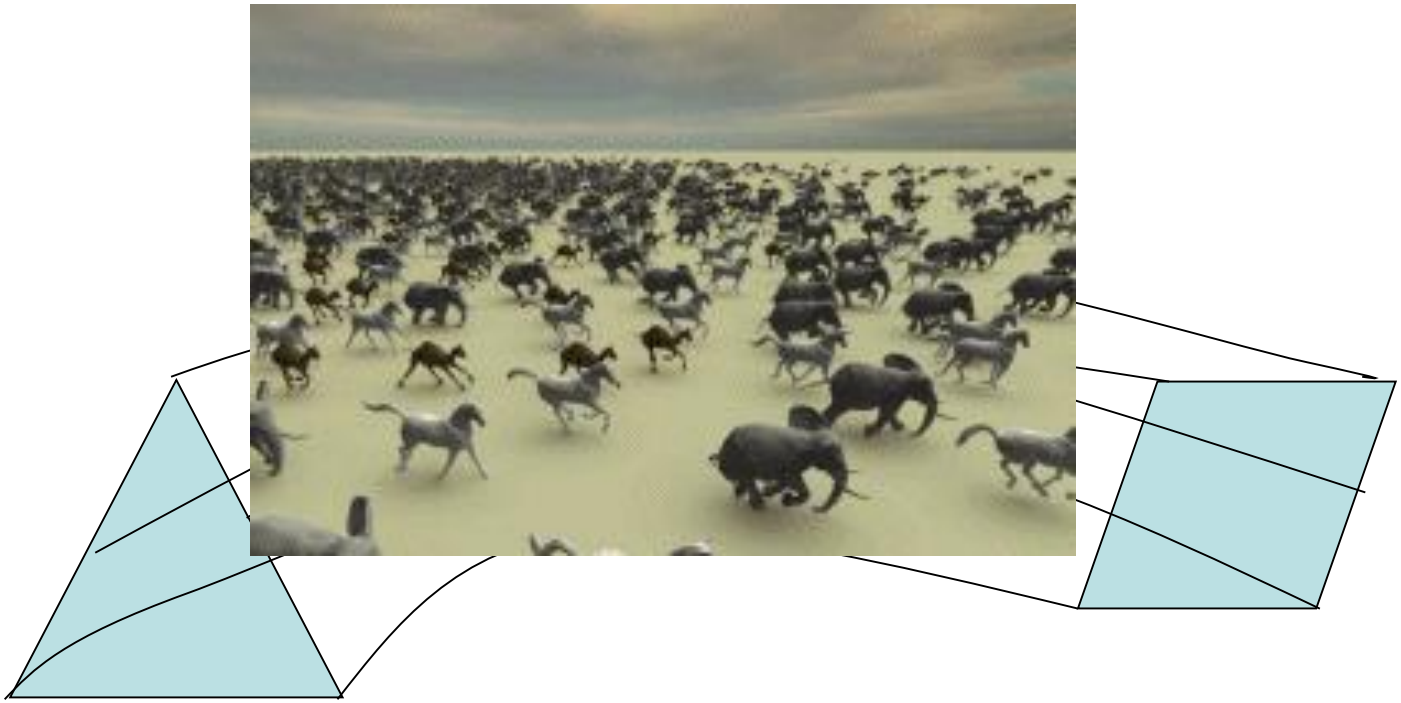




Skinning for animation (morphing)

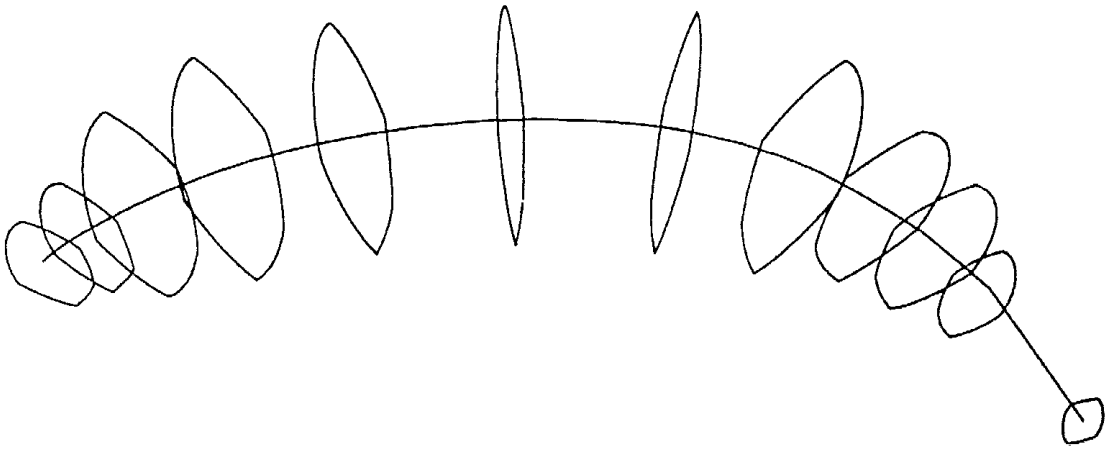
- Do
  Au

5), 24(3),

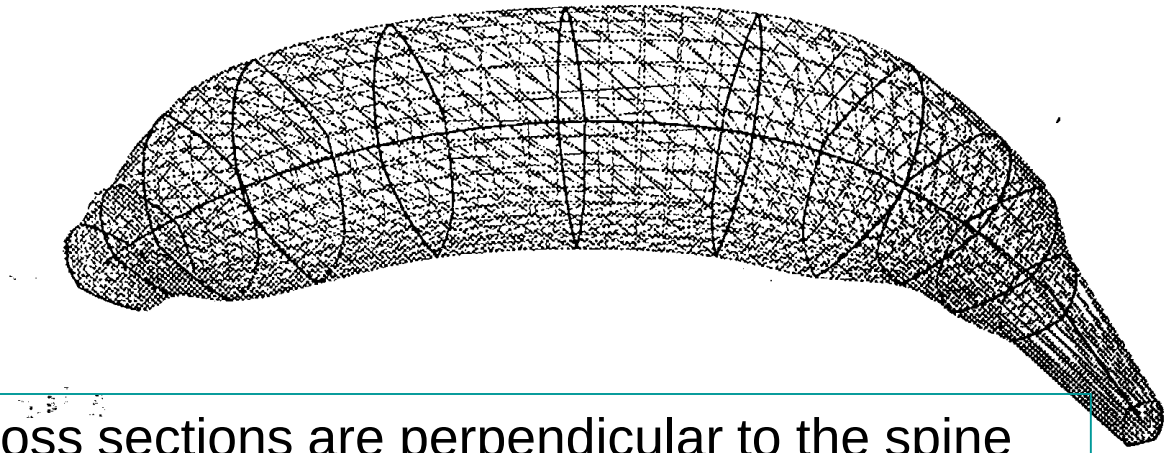


Swept Surfaces

Surface defined by a *cross sectional curve* moving along a *spine.*
Simple version: a single 3D curve for spine and a single 2D curve for the cross section



Sweeping example:
several cross sections rather than just one.



The planes containing the cross sections are perpendicular to the spine

Swung surfaces

It's a generalization of a surface of revolution where the trajectory curve is not necessary circular.

Profile Curve P(u) defined in the xz plane:

$$P(u) = \sum_{i=1}^{n+K} P_i R_{i,n}(u), \quad P_i = (P_{x_i}, 0, P_{z_i})^T$$

Trajectory Curve T(v) defined in the xy plane:

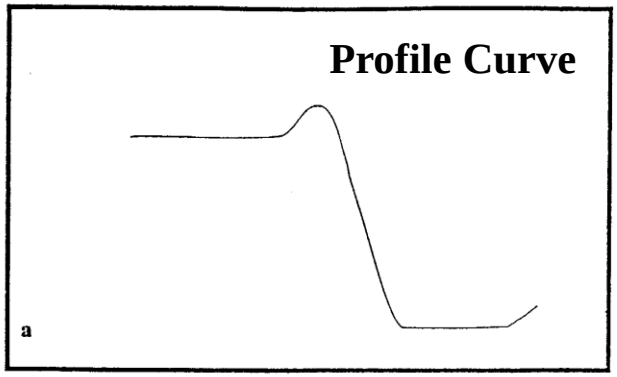
$$T(v) = \sum_{j=1}^{m+L} T_j R_{j,m}(v), \quad T_j = (T_{x_j}, T_{y_j}, 0)^T$$

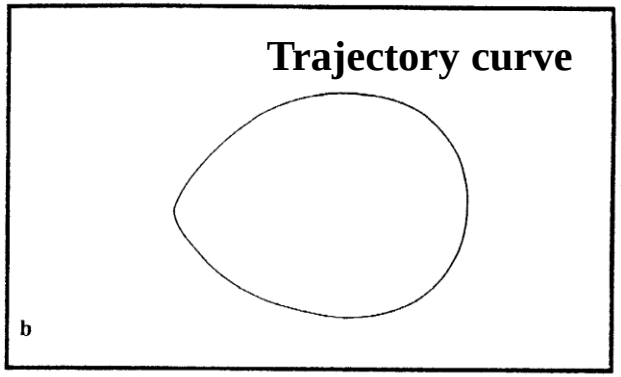
Swung surface

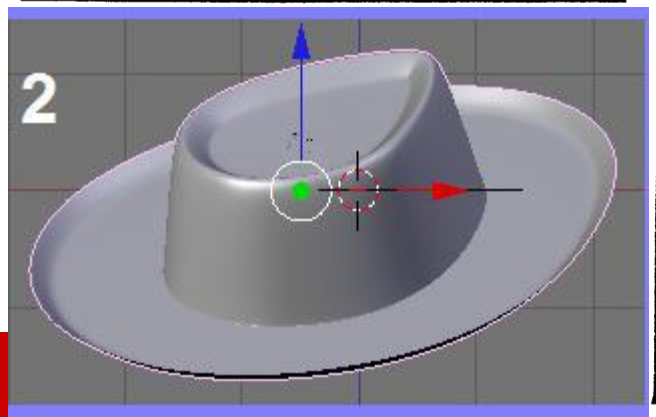
Swinging P(u) about the z axis and simultaneously scaling it according to T(v), s is an arbitrary scaling factor.

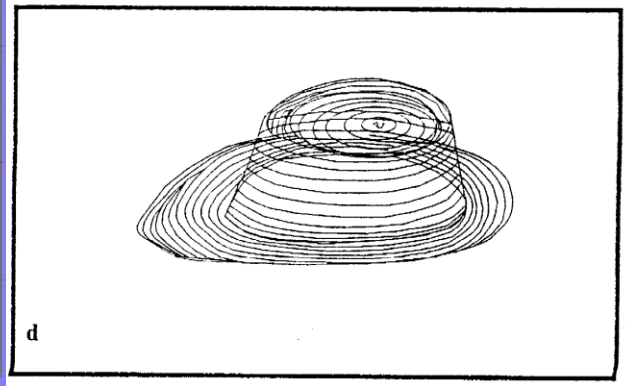
S(u,v) has a NURBS representation given by:

$$s(u,v) = \left(sP_x(u)T_x(v), sP_x(u)T_y(v), P_z(u)\right)^T$$









Swung surface

Fixing u yields curves having the shape of T(v) but scaled in the x and y directions.

Fixing v=$\underline{v}$ the isoparametric curve $C_{\underline{v}}(u)$ are obtained by rotating P(u) into the plane containing the vector ($T_x(\underline{v})$, $T_y(\underline{v})$,0), and scaling the x and y coords. of the rotated curve with the factor s|T(v)|.

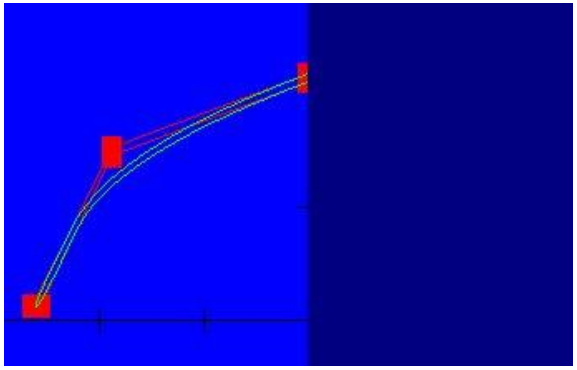
Control points of the swung surface:

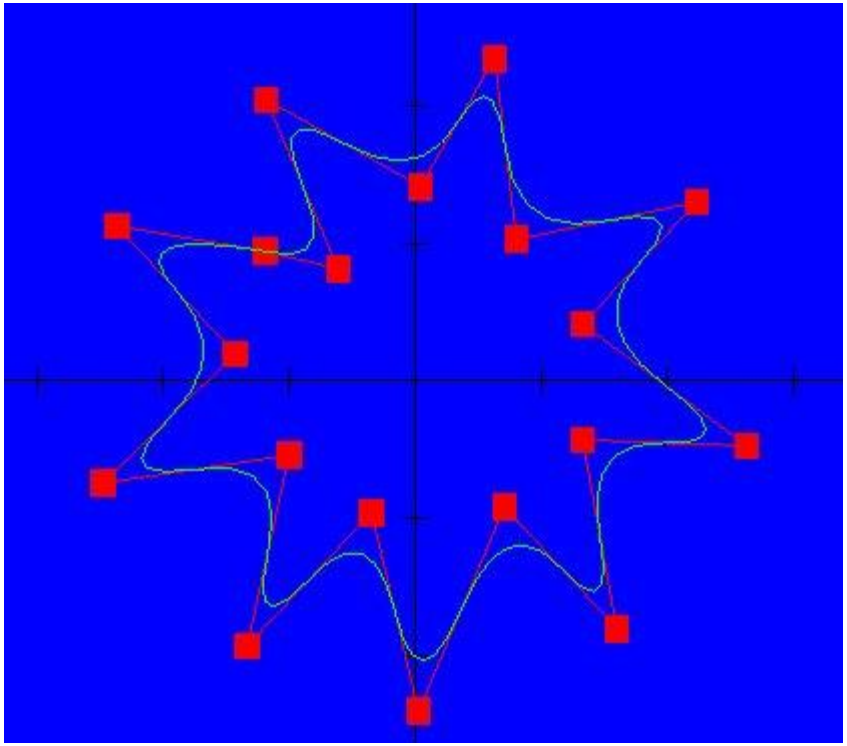
$$Q_{ij} = \left(sP_{x_i} T_{x_j}, sP_{x_i} T_{y_j}, P_{z_i} \right)^T$$

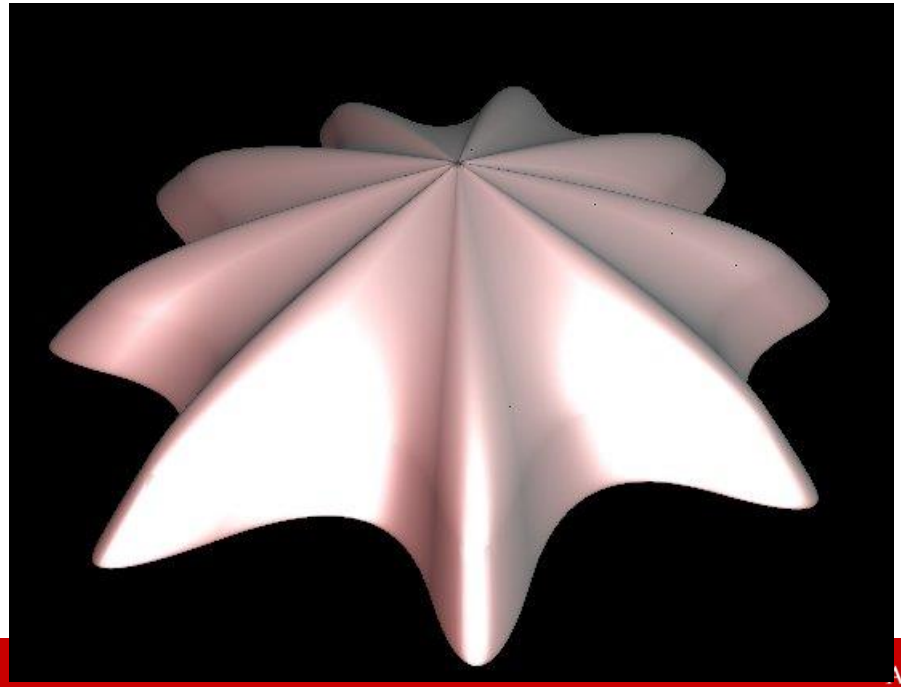
and weights: $w_{ij} = w_i \cdot w_j$

U and V knot vectors for s(u,v) are those defining P(u) and T(v)

Swinging Example





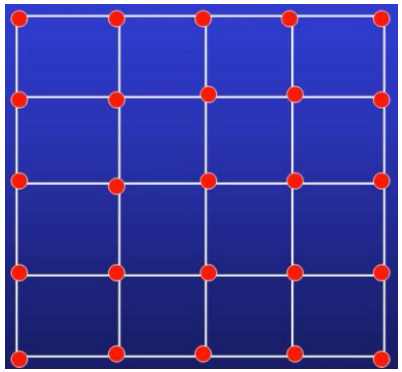


Domain Space Tessellation

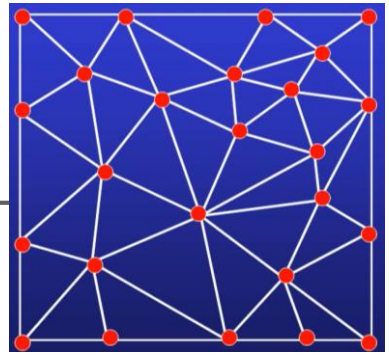
- ***Tessellation*** is the process of taking a complex surface and approximating it with a set of simpler surfaces (like triangles)

- The most straightforward way to tessellate a parametric surface is ***uniform tessellation***

With this method, we simply choose some resolution in *u* and *v* and uniformly divide up the surface like a grid
- This method is very efficient to compute, as the cost of evaluating the surface reduces to approximately the same cost as evaluating a curve
- However, as the generated mesh is uniform, it may have more triangles than it needs in flatter areas and fewer than it needs in highly curved areas

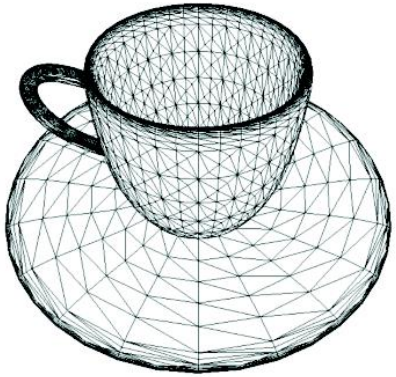


Adaptive Tessellation



- The goal of a tessellation is to provide the fewest triangles necessary to accurately represent the original surface

- For a curved surface, this means that we want more triangles in areas where the curvature is high, and fewer triangles in areas where the curvature is low

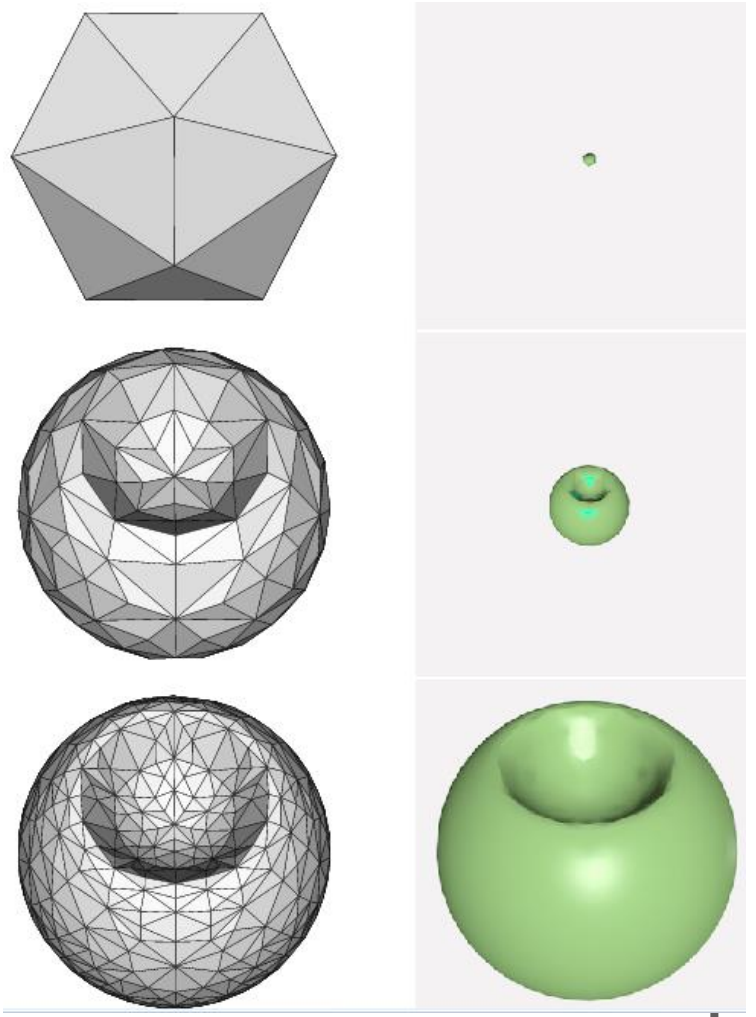


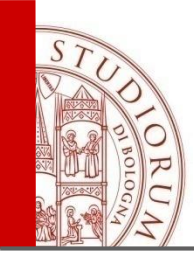
Adaptive Tessellation:
View-Dependent Rendering of Parametric Surfaces

- We may also want more triangles in areas that are closer to the camera, and fewer farther away

  Level Of Details

- *Adaptive tessellation* schemes are designed to address these requirements





Draw a Bézier patch: adaptive subdivision method

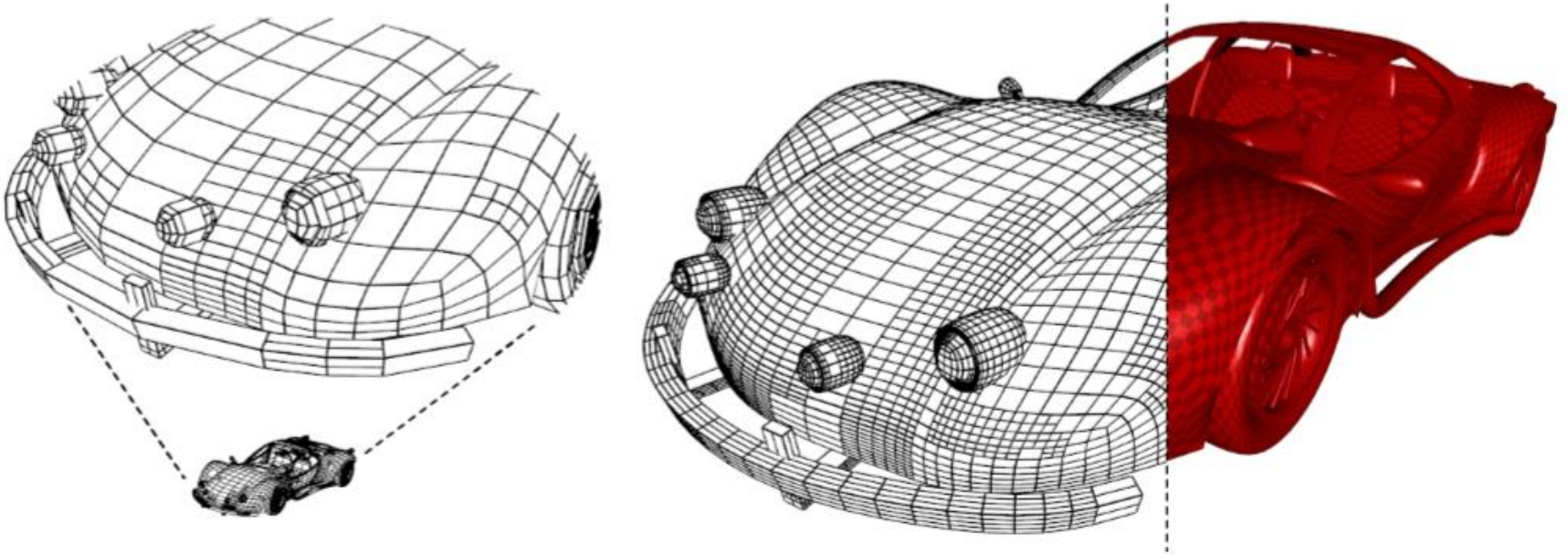
- basic approach: recursively test flatness/view dependent
  - if the patch s(u,v) is not flat enough,
  - subdivide into four using curve subdivision twice in v=1/2 and u=1/2, and
  - recursively process each subpatch
- as with curves, convex hull property is useful for termination testing (is inherited from the curves)

Flat test: (convex hull flatness test)

Construct a plane interpolating 3 noncollinear CP

Compute the distances d_i from the remaining CP from this plane. $D=max\ |d_i|$

If (D < Tolerance Tol) then the patch is considered flat and is approximated as a flat quadrilateral.



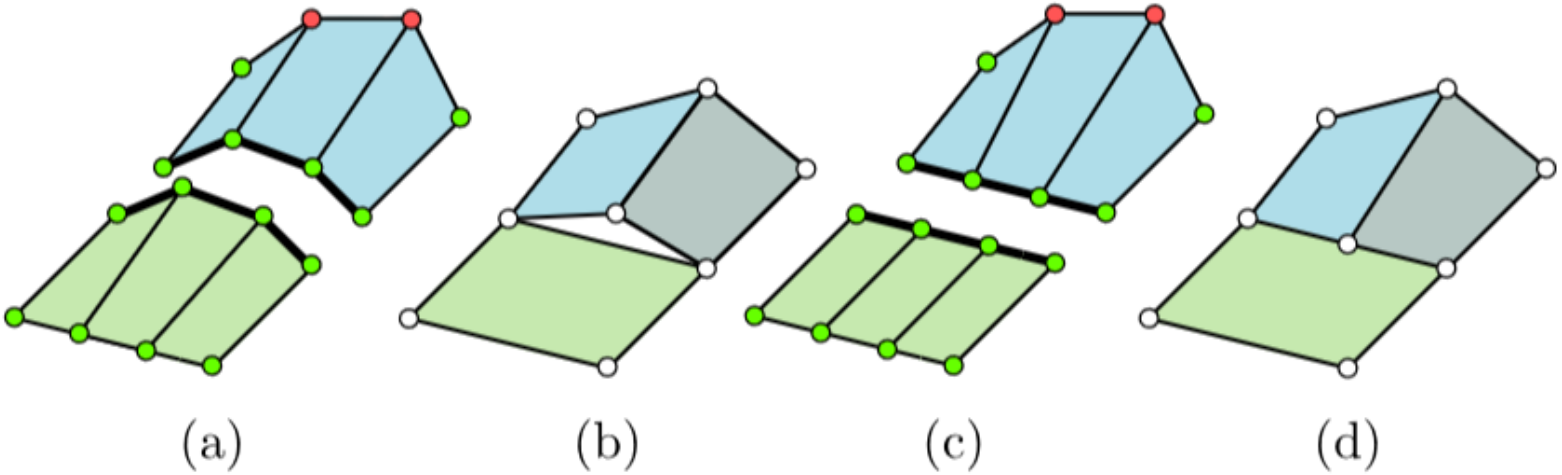
- Adaptively subdivide rational Bézier patches until a view-dependent error metric is satisfied. For a1600x1200 image of the car model (right) render 192k quads at 143 fps on a NVIDIA GTX 280

Real-Time View-Dependent Rendering of Parametric Surfaces, C. Eisenacher, Q. Meyer, C.Loop Microsoft Research

Crack Problem

With adaptive subdivision, must take care with cracks

- A surface is subdivided and its neighbor is not: small gaps or small overlaps can appear in the surface.



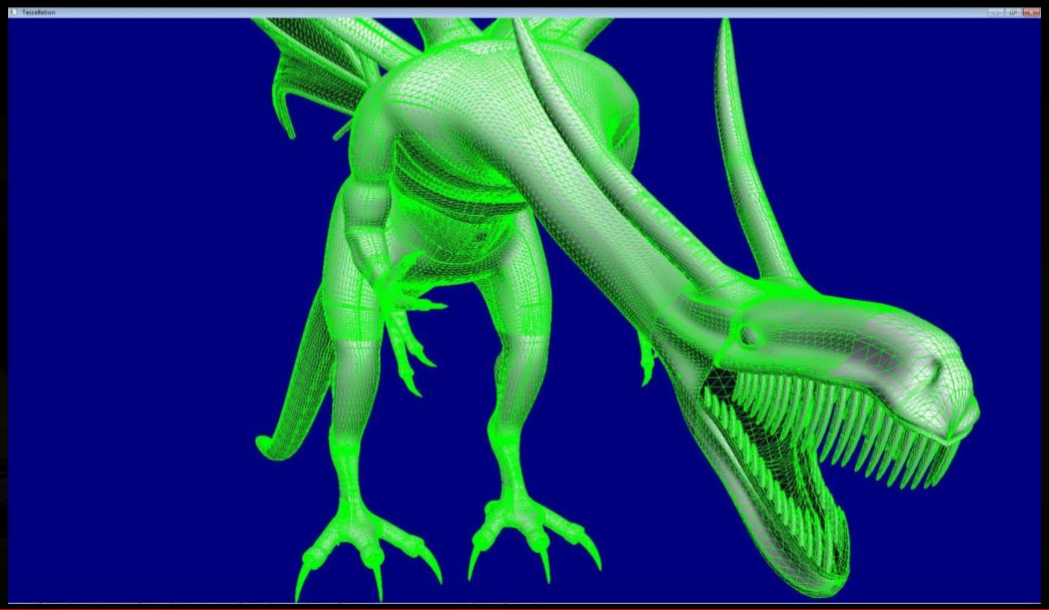
Crack prevention: Patches sharing an edge (a), are subdivided to different levels, generating a crack (b). If an edge is close to being linear (control points green) we set it to linear (c) and subdivision cracks are avoided (d).

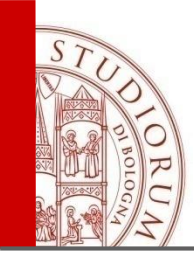
Tesselating NURBS surfaces with CUDA Tesselator shader

OpenGL 4.x

- Vertex
- Tessellation Control
- Tessellator
- Tessellation Eval.
- Geometry
- Fragment

Results (438 surf, 3.6M tri in 36.4 ms)





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Serena Morigi

Dipartimento di Matematica

serena.morigi@unibo.it

http://www.dm.unibo.it/~morigi