

计算机辅助几何设计

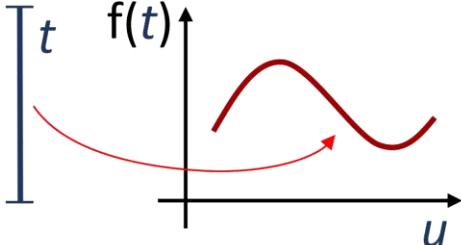
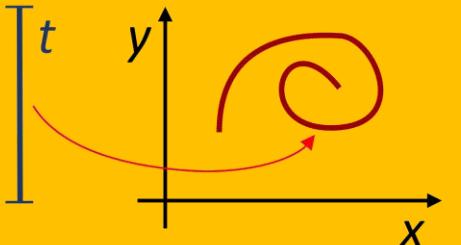
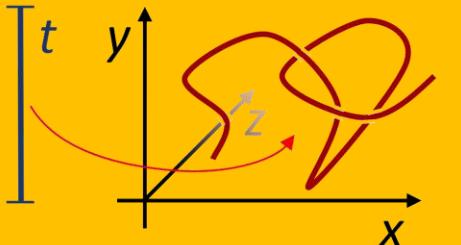
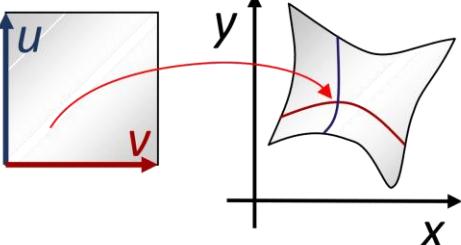
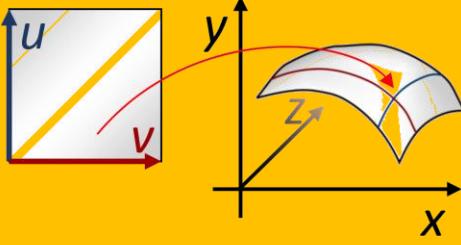
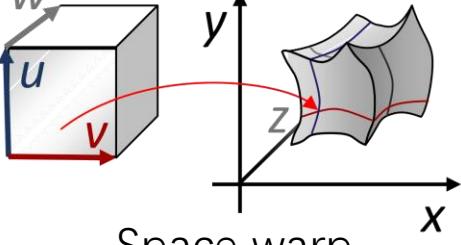
2021秋学期

Spline Surfaces

Tensor Product Surfaces · Total Degree Surfaces

陈仁杰

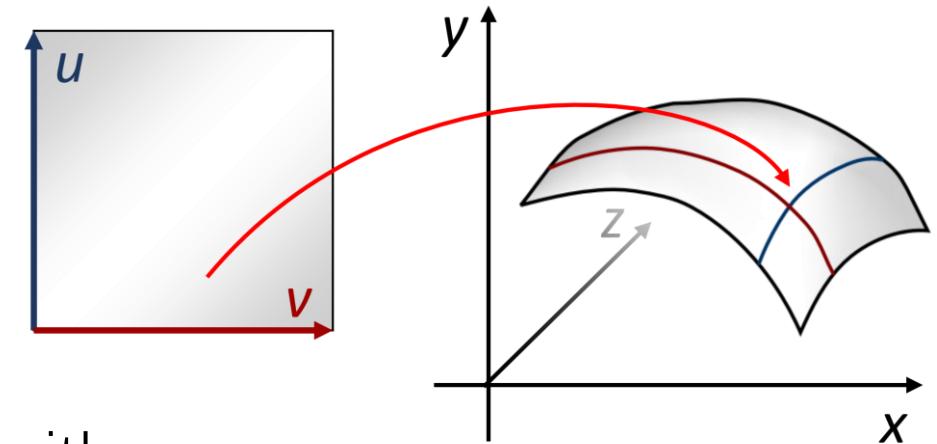
中国科学技术大学

	Output: 1D	Output: 2D	Output: 3D
Input: 1D	 <p>Function graph</p>	 <p>Plane curve</p>	 <p>Space curve</p>
Input: 2D		 <p>Plane warp</p>	 <p>Surface</p>
Input: 3D			 <p>Space warp</p>

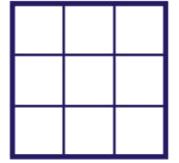
Spline Surfaces

Parametric spline surfaces:

- Two parameter coordinates (u, v)
- Piecewise bivariate polynomials
(rational surfaces \rightarrow homogeneous coords)
- Assemble multiple pieces to form a surface with continuity
- Each piece is called *spline patch*

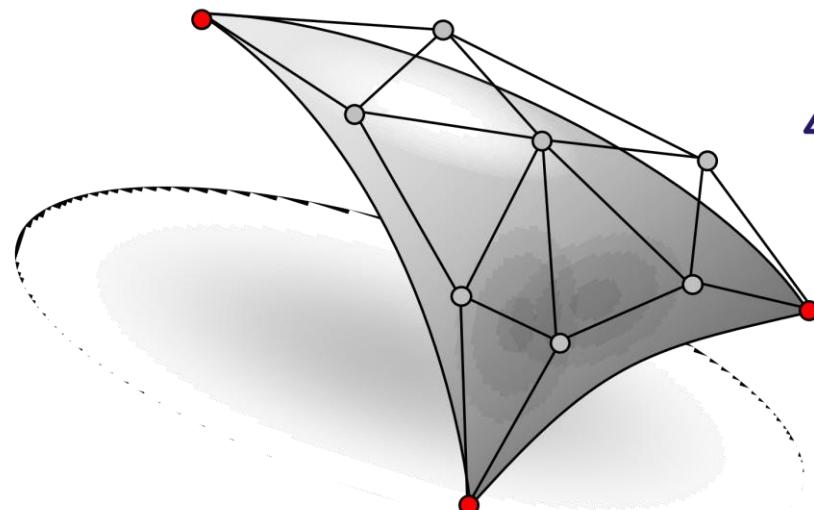
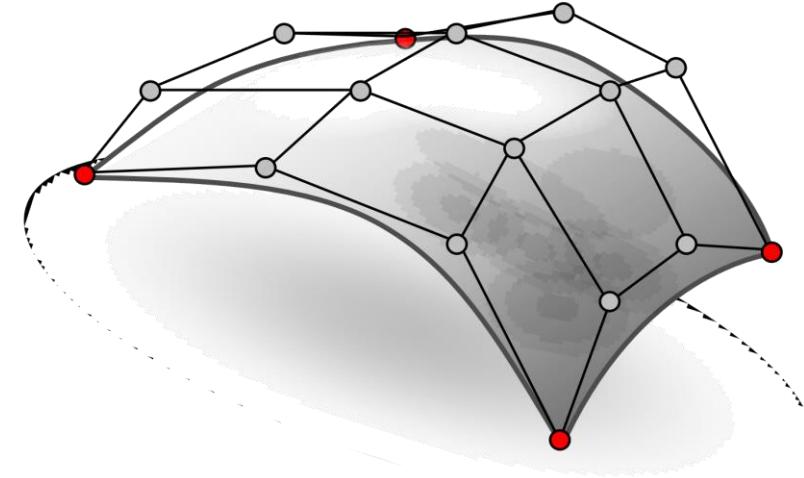


Spline Surfaces



Two different approaches

- Tensor product surfaces
 - Simple construction
 - Everything carries over from curve case
 - Quad patches
 - Degree anisotropy
- Total degree surfaces
 - Not as straightforward
(blossoming will help)
 - Isotropic degree
 - Triangle patches
 - “natural” generalization of curves



Tensor Product Surfaces

Tensor Product Surfaces

Simple Idea

- Given a basis for a one dimensional function space on the interval $t \in [t_0, t_1] \rightarrow \mathbb{R}^d$:

$$\mathbf{B}^{(curv)} := \{b_1(t), \dots, b_n(t)\}$$

- Build a new basis with two parameters by taking all possible products:

$$\mathbf{B}^{(surf)} := \{b_1(u)b_1(v), b_1(u)b_2(v), \dots, b_n(u)b_n(v)\}$$

Tensor Product Surfaces

Tensor product basis

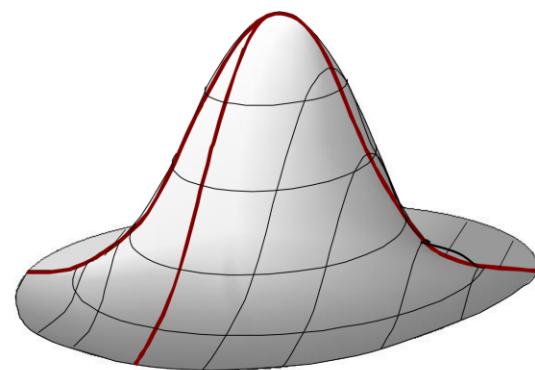
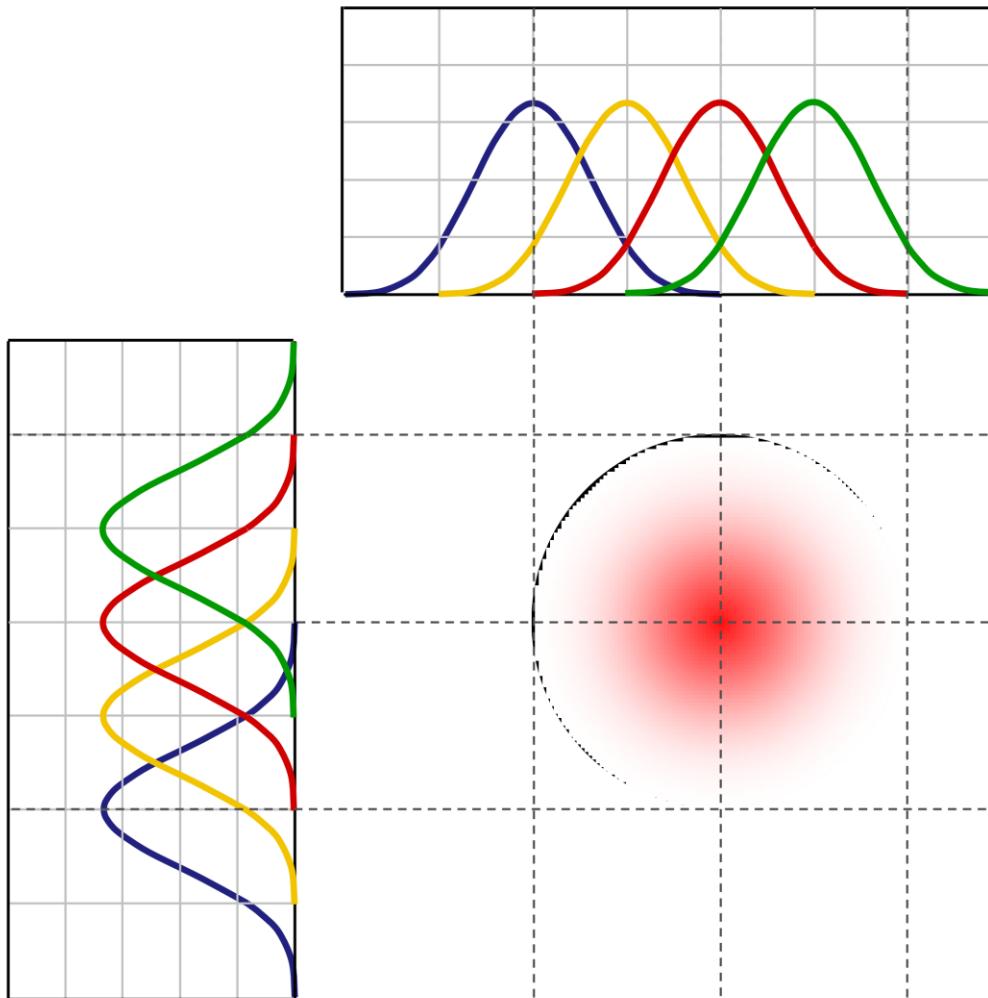
	$b_1(u)$	$b_2(u)$	$b_3(u)$	$b_4(u)$
$b_1(v)$	$b_1(v)b_1(u)$	$b_1(v)b_2(u)$	$b_1(v)b_3(u)$	$b_1(v)b_4(u)$
$b_2(v)$	$b_2(v)b_1(u)$	$b_2(v)b_2(u)$	$b_2(v)b_3(u)$	$b_2(v)b_4(u)$
$b_3(v)$	$b_3(v)b_1(u)$	$b_3(v)b_2(u)$	$b_3(v)b_3(u)$	$b_3(v)b_4(u)$
$b_4(v)$	$b_4(v)b_1(u)$	$b_4(v)b_2(u)$	$b_4(v)b_3(u)$	$b_4(v)b_4(u)$

Monomial Example

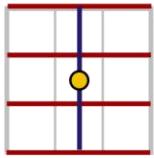
Tensor product basis of cubic monomials

	1	u	u^2	u^3
1	1	u	u^2	u^3
v	v	vu	vu^2	vu^3
v^2	v^2	v^2u	v^2u^2	v^2u^3
v^3	v^3	v^3u	v^3u^2	v^3u^3

Example



Tensor Product Surfaces



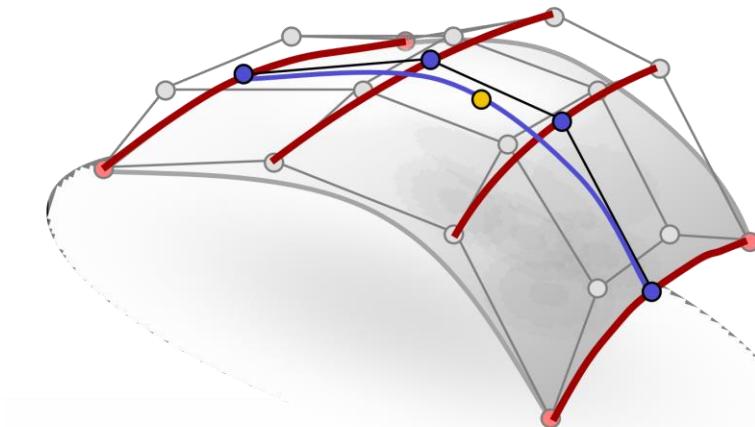
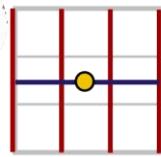
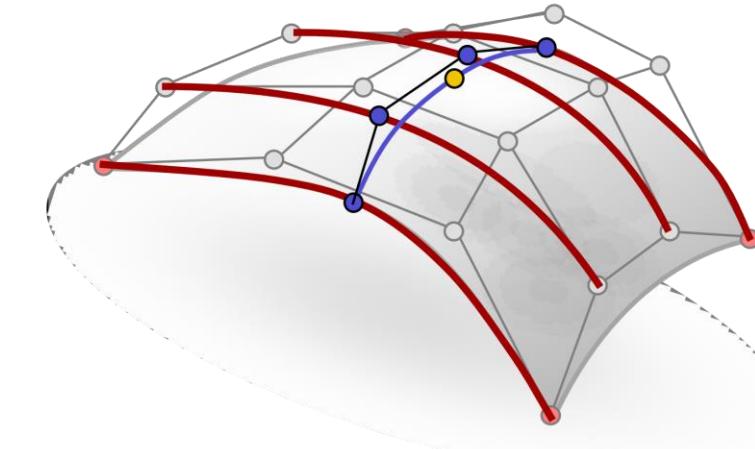
Tensor Product Surfaces

$$\mathbf{f}(u, v) = \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j}$$

$$= \sum_{i=1}^n b_i(u) \sum_{j=1}^n b_j(v) \mathbf{p}_{i,j}$$

$$= \sum_{j=1}^n b_j(v) \sum_{i=1}^n b_i(u) \mathbf{p}_{i,j}$$

- “Curves of Curves”
- Order does not matter



Properties

Properties of tensor product surfaces:

- Linear invariance: Obvious
- Affine invariance?
 - Needs partition of unity property
 - Assume basis $B^{(curv)} := \{b_1(t), \dots, b_n(t)\}$ forms a partition of unity, i.e.: $\sum_{i=1}^n b_i(v) = 1$
 - Then we get:

$$\sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) = \sum_{i=1}^n b_i(u) \sum_{j=1}^n b_j(v) = \sum_{j=1}^n b_j(v) \cdot 1 = 1$$

- Affine invariance for tensor product surfaces is induced by the corresponding property of the employed curve basis

Properties

Properties of tensor product surfaces:

- Convex Hull?
 - Assume basis $\mathbf{B}^{(curv)} := \{b_1(t), \dots, b_n(t)\}$ forms a partition of unity and it is nonnegative (≥ 0) on $t \in [t_0, t_1]$
 - Obviously, we then have:
$$\sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \geq 0$$

$\geq 0 \quad \geq 0$
- So we have the convex hull property on $[t_0, t_1]^2$
- The convex hull property for tensor product surface is induced by the property of the employed curve basis

Partial Derivatives

Computing partial derivatives:

- First derivatives:

$$\frac{\partial}{\partial u} \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j} = \sum_{j=1}^n b_j(v) \sum_{i=1}^n \left(\frac{d}{du} b_i \right)(u) \mathbf{p}_{i,j}$$

$$\frac{\partial}{\partial v} \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j} = \sum_{i=1}^n b_i(u) \sum_{j=1}^n \left(\frac{d}{dv} b_j \right)(v) \mathbf{p}_{i,j}$$

- Just spline-curve combinations of curve derivatives

Partial Derivatives

Computing partial derivatives:

- Second derivatives:

$$\frac{\partial}{\partial u^2} \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j} = \sum_{j=1}^n b_j(v) \sum_{i=1}^n \left(\frac{d^2}{du^2} b_i \right)(u) \mathbf{p}_{i,j}$$

$$\frac{\partial^2}{\partial u \partial v} \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j} = \frac{\partial}{\partial v} \sum_{j=1}^n b_j(v) \sum_{i=1}^n \left(\frac{d}{du} b_i \right)(u) \mathbf{p}_{i,j}$$

$$= \sum_{j=1}^n \left(\frac{d}{dv} b_j \right)(v) \sum_{i=1}^n \left(\frac{d}{du} b_i \right)(u) \mathbf{p}_{i,j}$$

Partial Derivatives

Computing partial derivatives:

- General derivatives:

$$\begin{aligned} \frac{\partial^{r+s}}{\partial u^r \partial v^s} \sum_{i=1}^n \sum_{j=1}^n b_i(u) b_j(v) \mathbf{p}_{i,j} &= \sum_{j=1}^n \left(\frac{d^s}{dv^s} b_j \right)(v) \sum_{i=1}^n \left(\frac{d^r}{du^r} b_i \right)(u) \mathbf{p}_{i,j} \\ &= \sum_{i=1}^n \left(\frac{d^r}{du^r} b_i \right)(u) \sum_{j=1}^n \left(\frac{d^s}{dv^s} b_j \right)(v) \mathbf{p}_{i,j} \end{aligned}$$

Normal Vectors

We can compute normal vectors from partial derivatives:

$$\mathbf{n}(u, v) = \frac{\left(\sum_{j=1}^n b_j(v) \sum_{i=1}^n \frac{d}{du} b_i(u) \mathbf{p}_{i,j} \right) \times \left(\sum_{j=1}^n \frac{d}{dv} b_j(v) \sum_{i=1}^n b_i(u) \mathbf{p}_{i,j} \right)}{\left\| \left(\sum_{j=1}^n b_j(v) \sum_{i=1}^n \frac{d}{du} b_i(u) \mathbf{p}_{i,j} \right) \times \left(\sum_{j=1}^n \frac{d}{dv} b_j(v) \sum_{i=1}^n b_i(u) \mathbf{p}_{i,j} \right) \right\|}$$

- Problem: degenerate cases
 - Collinear tangents
 - Irregular parametrization
- Need extra code to handle special cases

Tensor Product Surfaces

Tensor Product Bézier Surfaces

Tensor Product Bézier Spline Surfaces

Tensor Product Bézier Surfaces

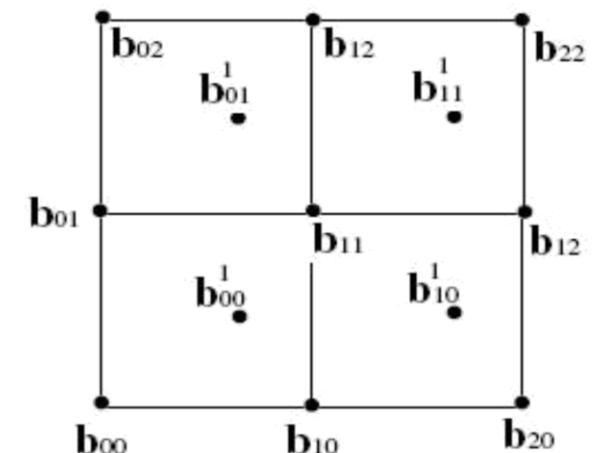
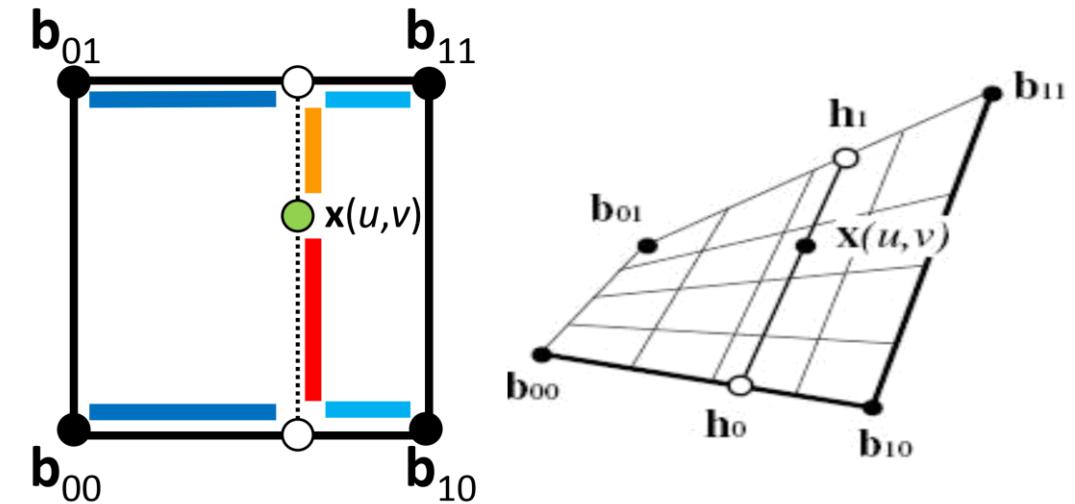
Bézier curves:
repeated linear interpolation

now a different setup:

4 points $\mathbf{b}_{00}, \mathbf{b}_{10}, \mathbf{b}_{11}, \mathbf{b}_{01}$
Parameter area $[0,1] \times [0,1]$

→ bilinear interpolation:
repeated linear interpolation

repeated bilinear interpolation:
Gives us tensor product Bézier surfaces
(example) shows quadratic Bézier Surfaces)



Some formulas for this setup

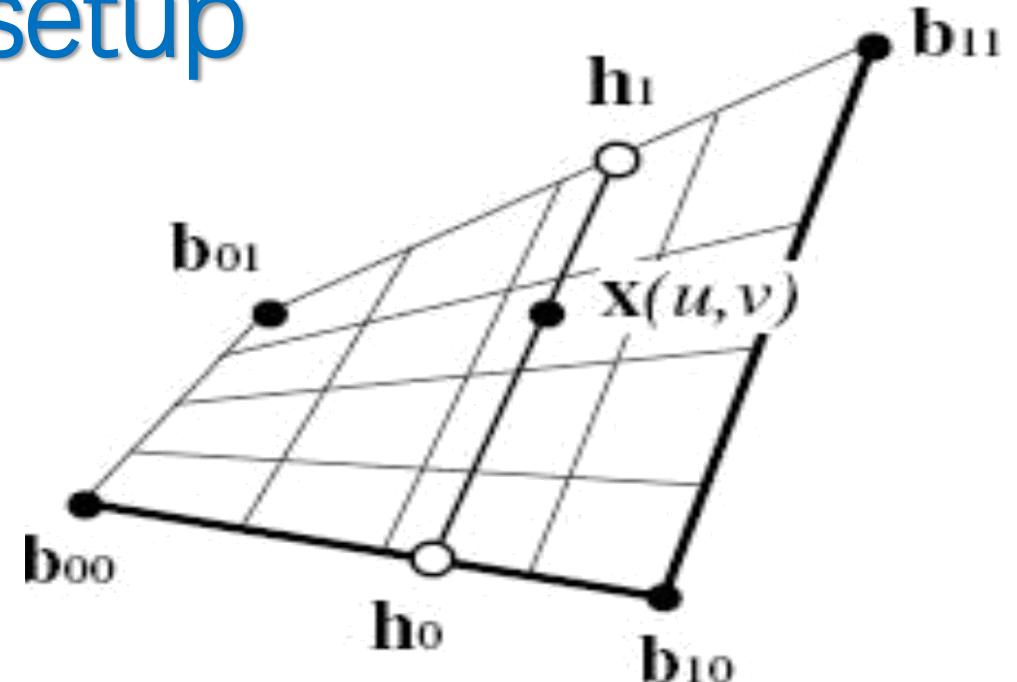
$$\mathbf{h}_0 = (1 - u)\mathbf{b}_{00} + u\mathbf{b}_{10}$$

$$\mathbf{h}_1 = (1 - u)\mathbf{b}_{01} + u\mathbf{b}_{11}$$

$$x(u, v) = (1 - v)\mathbf{h}_0 + v\mathbf{h}_1$$

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

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



Some formulas for this setup

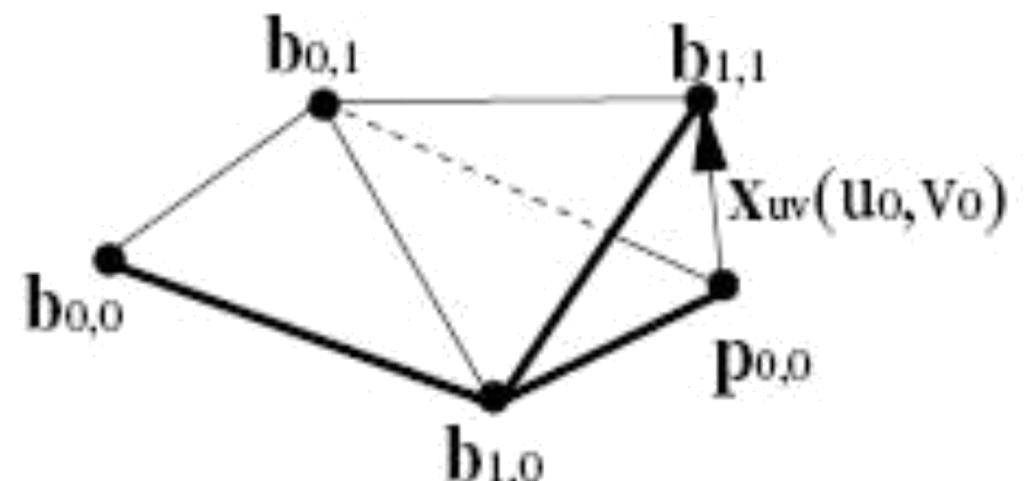
Derivatives of bilinear surfaces

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

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

$$\mathbf{x}_{uu}(u, v) = \mathbf{x}_{vv}(u, v) = 0$$

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



Some formulas for this setup

Biquadratic surfaces

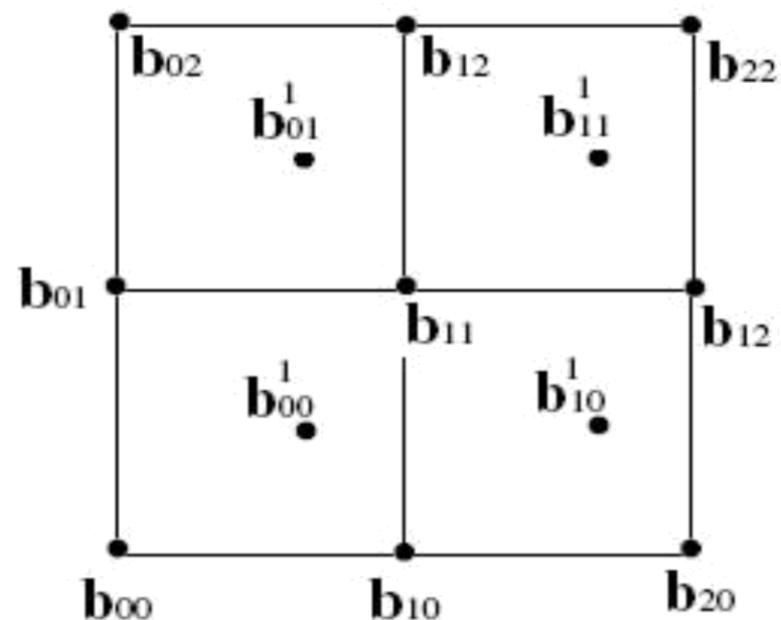
$$\mathbf{b}_{00}^1 = (1-u)(1-v)\mathbf{b}_{00} + u(1-v)\mathbf{b}_{10} + (1-u)v\mathbf{b}_{01} + uv\mathbf{b}_{11}$$

$$\mathbf{b}_{10}^1 = (1-u)(1-v)\mathbf{b}_{10} + u(1-v)\mathbf{b}_{20} + (1-u)v\mathbf{b}_{11} + uv\mathbf{b}_{21}$$

$$\mathbf{b}_{01}^1 = (1-u)(1-v)\mathbf{b}_{01} + u(1-v)\mathbf{b}_{11} + (1-u)v\mathbf{b}_{02} + uv\mathbf{b}_{12}$$

$$\mathbf{b}_{11}^1 = (1-u)(1-v)\mathbf{b}_{11} + u(1-v)\mathbf{b}_{21} + (1-u)v\mathbf{b}_{12} + uv\mathbf{b}_{22}$$

$$\begin{aligned}x(u, v) &= (1-u)(1-v)\mathbf{b}_{00}^1 + u(1-v)\mathbf{b}_{10}^1 + (1-u)v\mathbf{b}_{01}^1 + uv\mathbf{b}_{11}^1 \\&= \sum_{i=0}^2 \sum_{j=0}^2 B_i^2(u) B_j^2(v) \mathbf{b}_{i,j}\end{aligned}$$



Bézier Patches

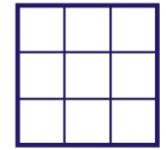
Bézier Patches:

- Use tensor product Bernstein basis

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

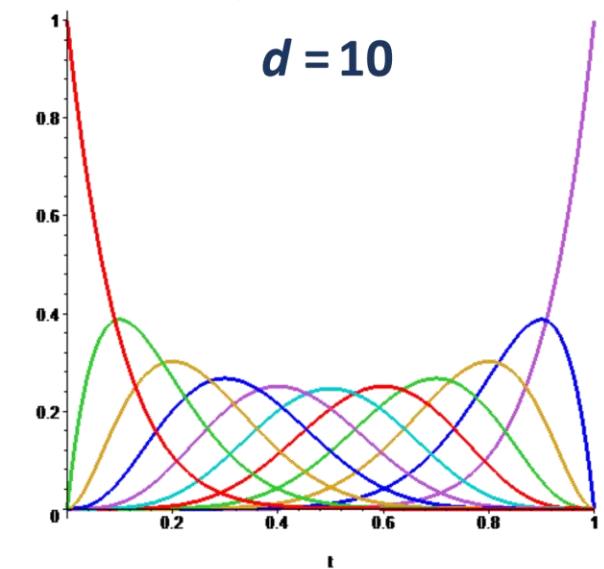
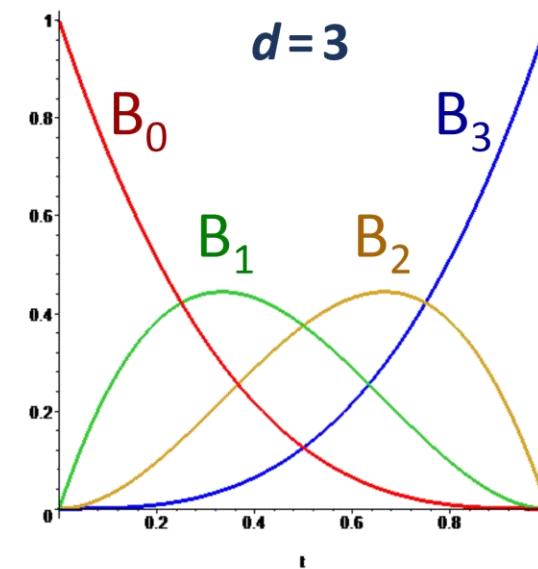
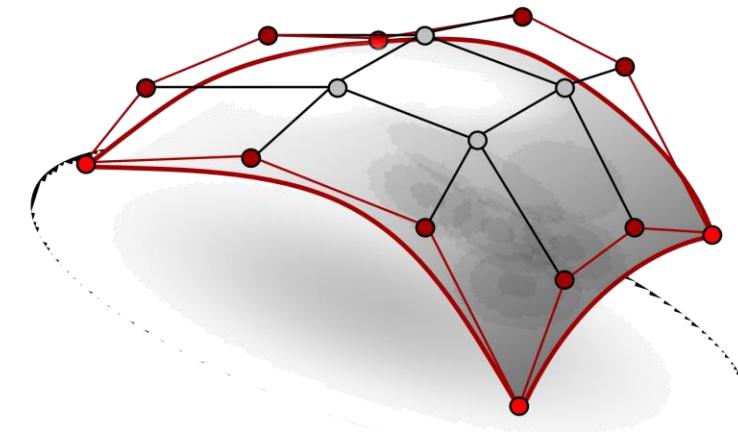
- We get automatically:
 - Affine invariance
 - Convex hull property

Bézier Patches



Bézier Patches:

- Remember endpoint interpolation:
 - Boundary curves are Bézier curves of the boundary control points



Bézier Patches

Bézier Patches:

- Tangent vectors:
 - First derivatives at boundary points are proportional to differences of control points:

$$\frac{\partial}{\partial u} \mathbf{f}(u, v) \Big|_{u=0} = \sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(v) B'_j^{(d)}(0) \mathbf{p}_{i,j}$$

$$= d \sum_{j=0}^d B_j^{(d)}(v) (\mathbf{p}_{1,j} - \mathbf{p}_{0,j})$$

$$\frac{\partial}{\partial u} \mathbf{f}(u, v) \Big|_{u=1} = d \sum_{j=0}^d B_j^{(d)}(v) (\mathbf{p}_{d,j} - \mathbf{p}_{d-1,j})$$

Continuity Conditions

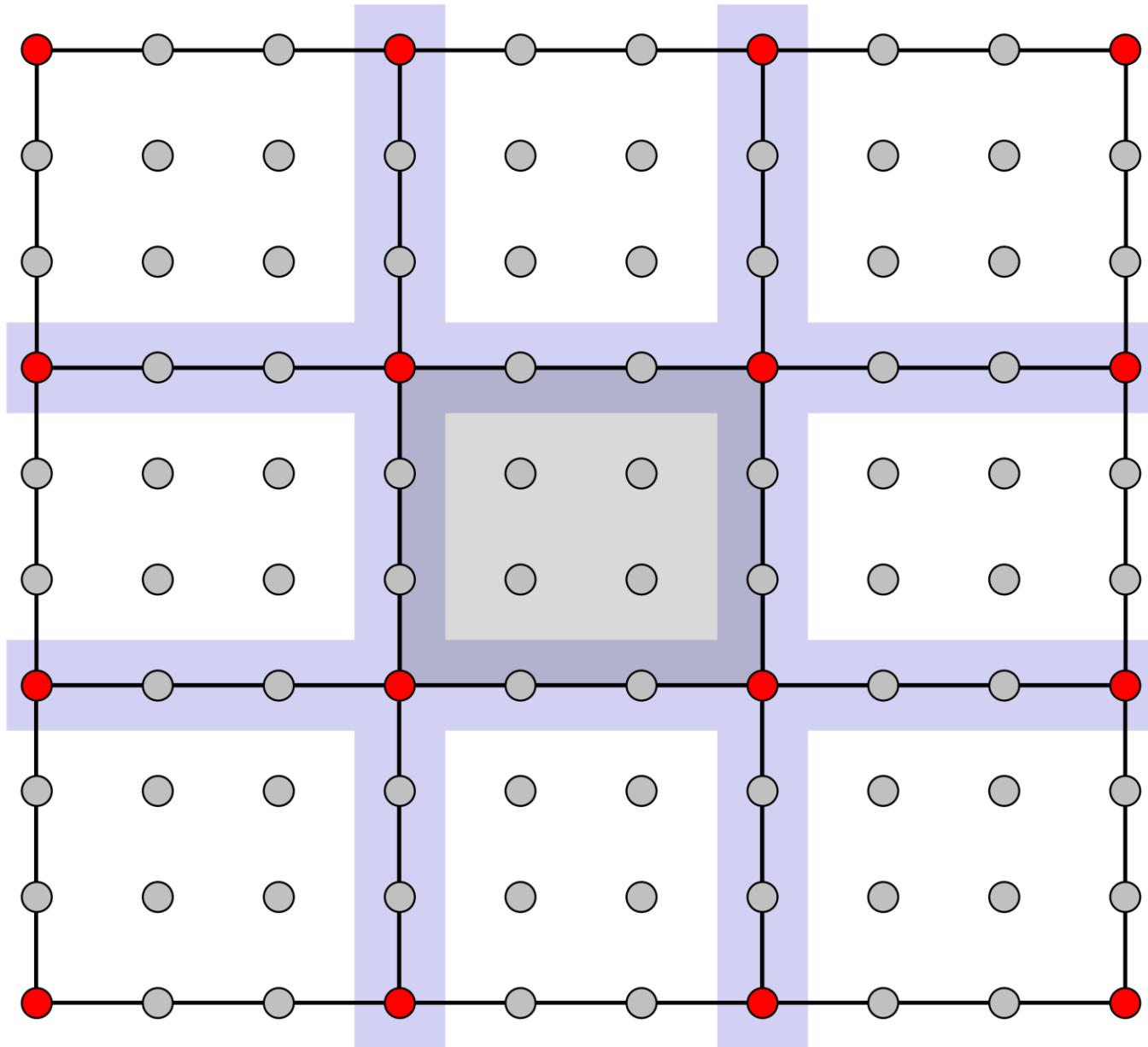
For C^0 continuity:

- Boundary control points must match

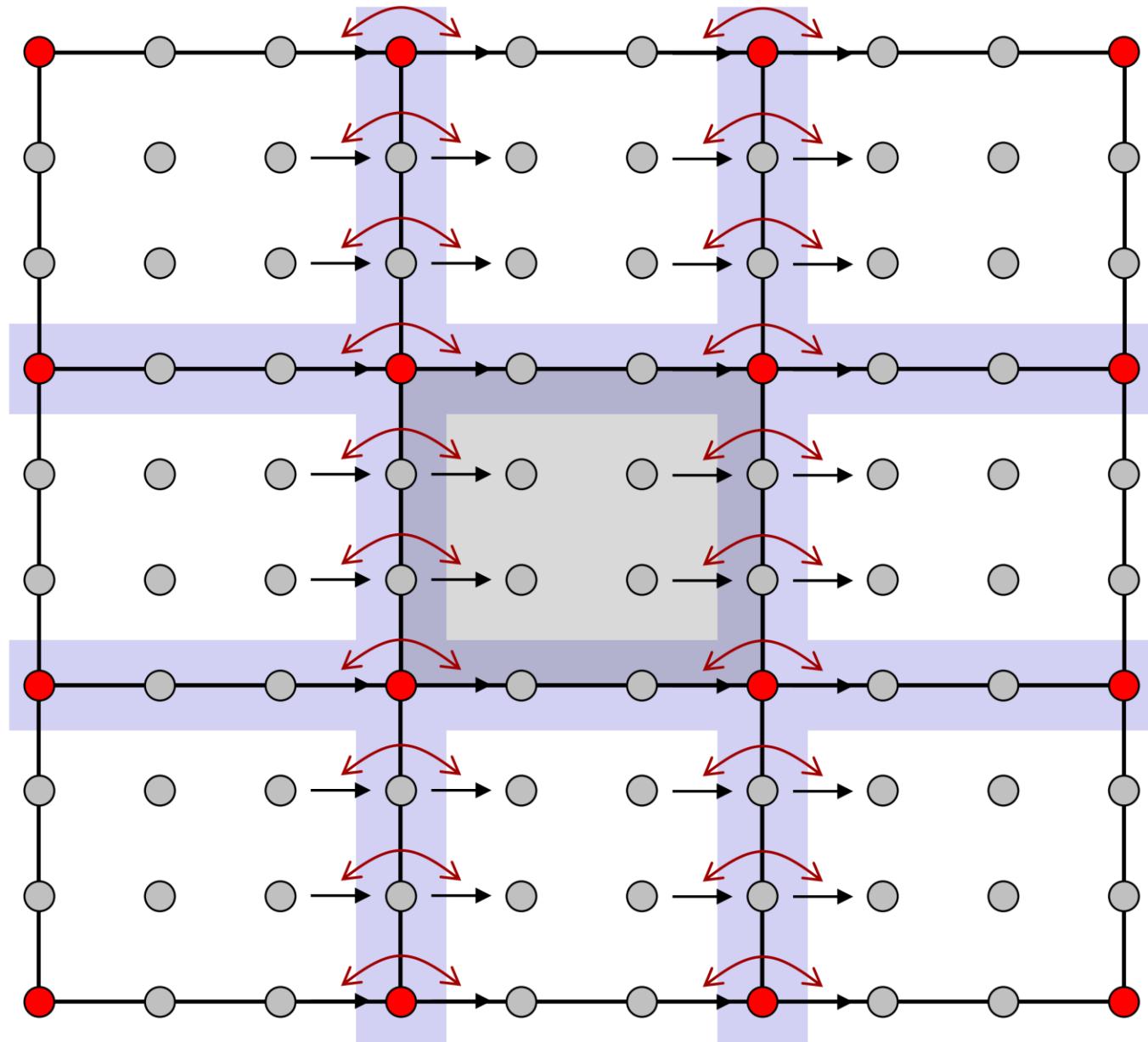
For C^1 continuity:

- Difference vectors must match at the boundary

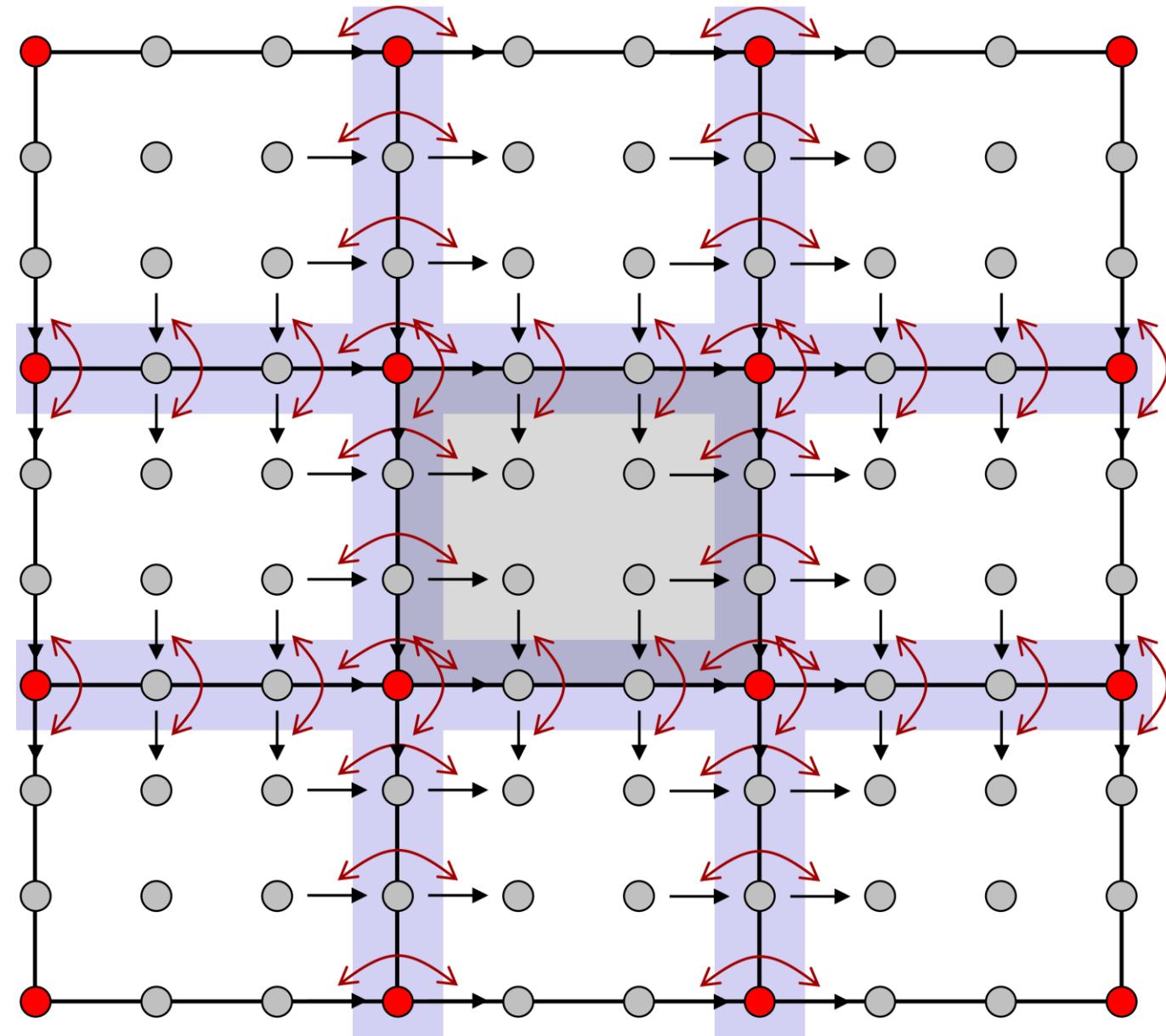
C^0 Continuity



C^1 Continuity



C^1 Continuity



Polars & Blossoms

Blossoms for tensor product surfaces:

- Polar form of a polynomial tensor product surfaces of degree d :

$$F: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^n \quad F(u, v)$$

$$f: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^n \quad f(u_1, \dots, u_d; v_1, \dots, v_d)$$

- Required properties:

- **Diagonality:** $f(u, \dots, u; v, \dots, v) = F(u, v)$

- **Symmetry:** $f(u_1, \dots, u_d; v_1, \dots, v_d) = f(u_{\pi(1)}, \dots, u_{\pi(d)}; v_{\mu(1)}, \dots, v_{\mu(d)})$
for all permutations of indices π, μ

- **Multi-affine:** $\sum \alpha_k = 1$

$$\Rightarrow f\left(u_1, \dots, \sum \alpha_k u_i^{(k)}, \dots, u_d; v_1, \dots, v_d\right)$$

$$= \alpha_1 f\left(u_1, \dots, u_i^{(1)}, \dots, u_d; v_1, \dots, v_d\right) + \dots + \alpha_n f\left(u_1, \dots, u_i^{(n)}, \dots, u_d; v_1, \dots, v_d\right)$$

and $f\left(u_1, \dots, u_d; v_1, \dots, \sum \alpha_k v_i^{(k)}, \dots, v_d\right)$

$$= \alpha_1 f\left(u_1, \dots, u_d; v_1, \dots, v_i^{(1)}, \dots, v_d\right) + \dots + \alpha_n f\left(u_1, \dots, u_d; v_1, \dots, v_i^{(n)}, \dots, v_d\right)$$

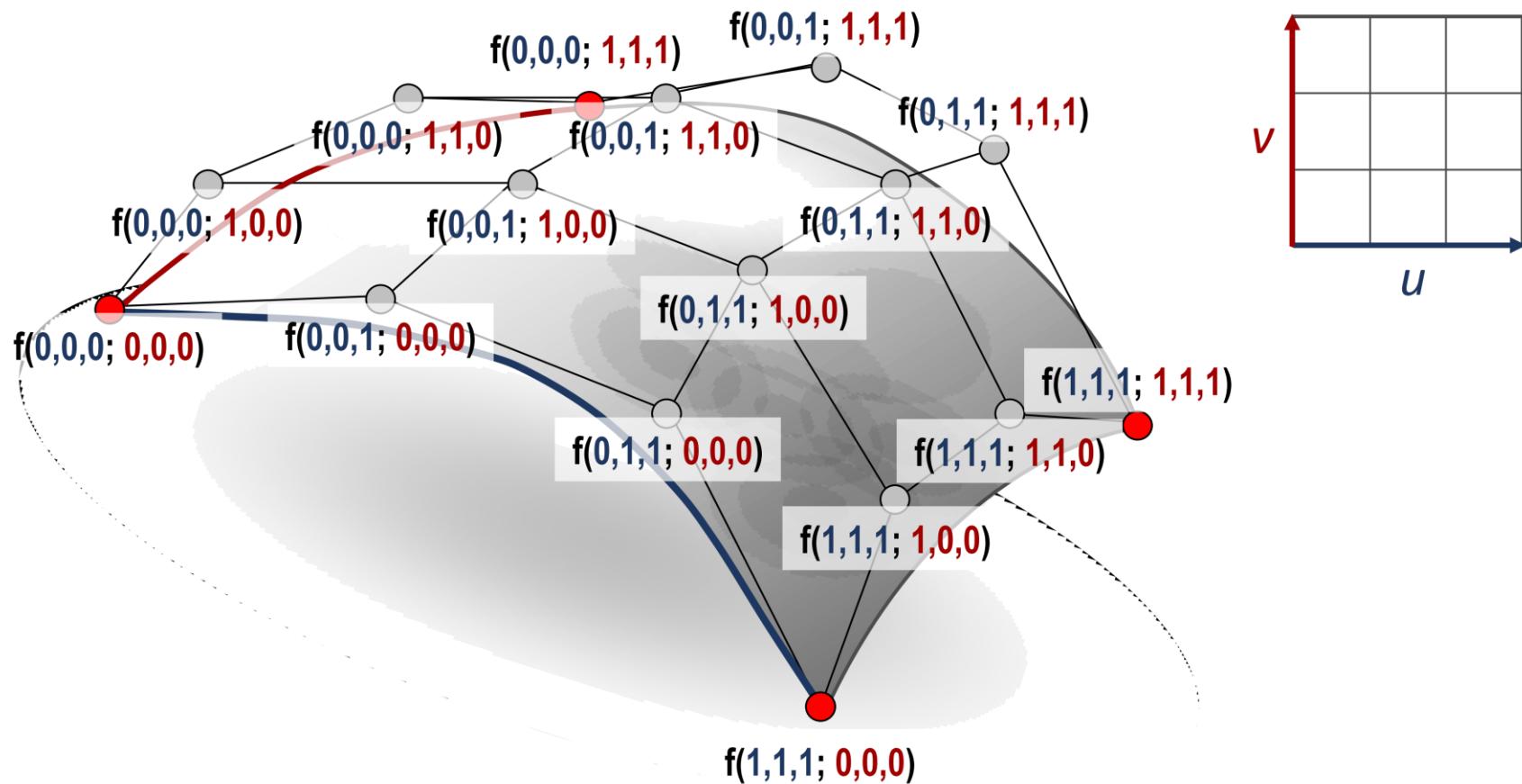
Short Summary

Polar forms for tensor product surfaces:

- Polar separately in u and v
- Notation: $f(\underline{u_1, \dots, u_d}; \underline{v_1, \dots, v_d})$
 u -parameters v -parameters
- Can be used to derive properties/algorithms similar to the curve case

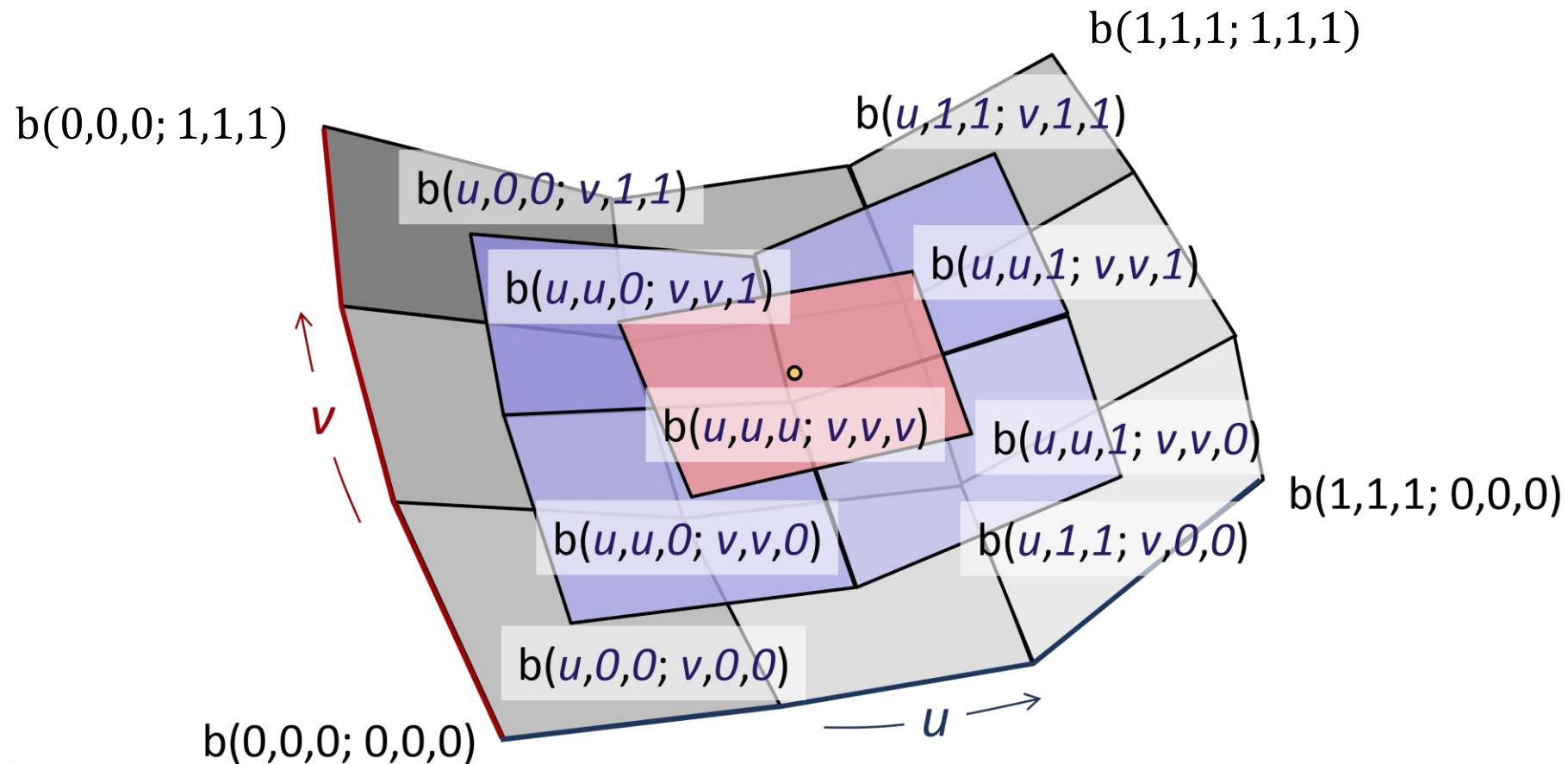
Bézier Control Points

Bézier control points in blossom notation:



de Casteljau Algorithm

de Casteljau algorithm for tensor product surfaces



Tensor Product Surfaces

Tensor Product B-Spline Surfaces

B-Spline Patches

B-Spline Patches

- More general than Bézier patches
(we get Bézier patches as a special case)
- First, we fix a degree d
- Then, we need knot sequences in u and v direction:
 $(u_1, \dots, u_n), (v_1, \dots, v_m)$
- And a corresponding array of control points

$$\begin{array}{ccccccc} d_{0,0} & \cdots & d_{n-d+1,0} & & & & \\ & \cdots & & \cdots & & & \\ d_{0,m-d+1} & \cdots & d_{n-d+1,m-d+1} & & & & \end{array}$$

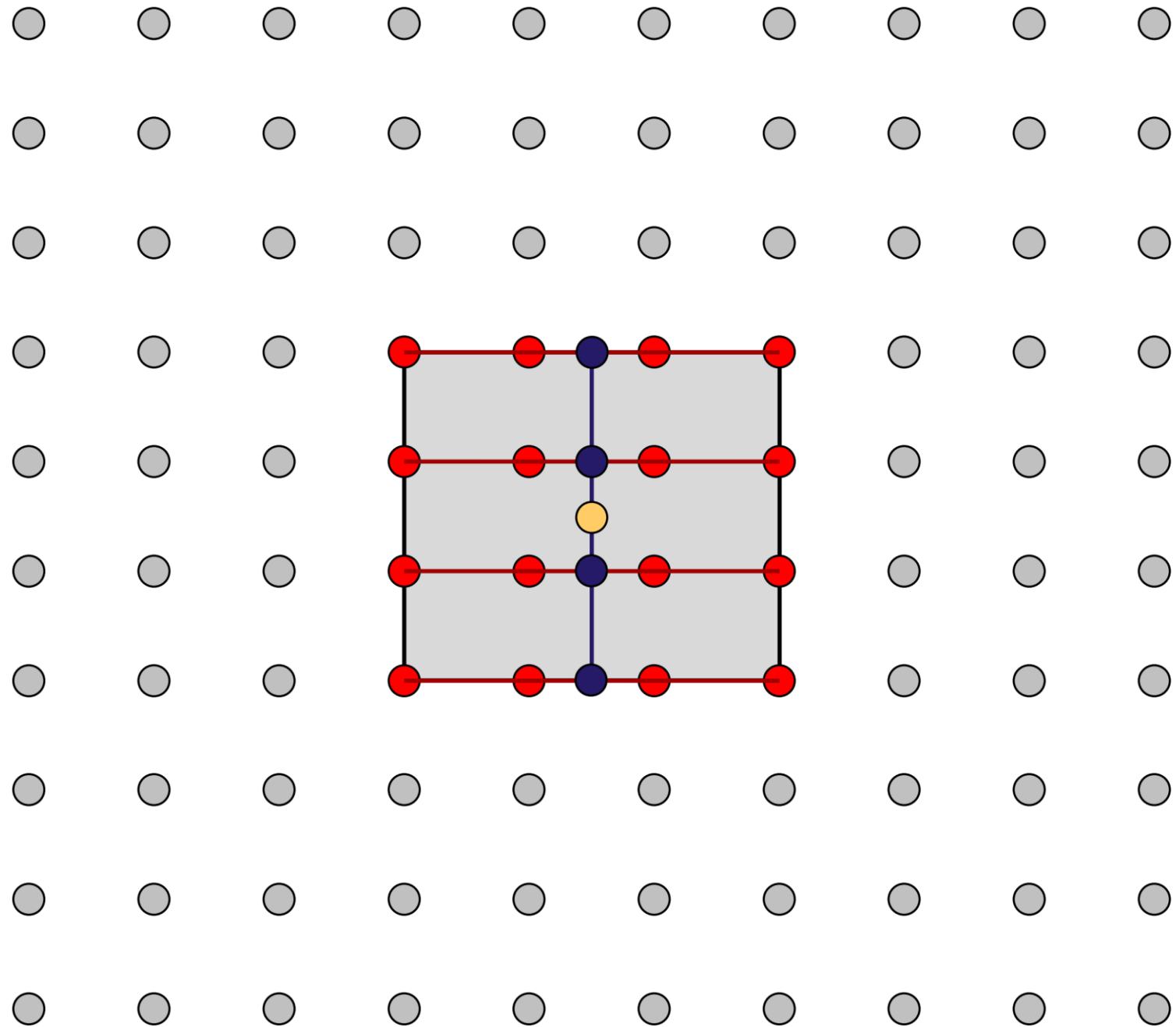
B-Spline Patches

Then, obtain a parametric B-spline patch as:

$$\mathbf{f}(u, v) = \sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \mathbf{p}_{i,j}$$

- We can evaluate the patches using the de Boor Algorithm:
 - “Curves of curves” idea
 - Determine the knots/control points influencing (u, v) ,
These will be no more than $(d + 1) \times (d + 1)$ points
 - Compute $(d + 1)$ v -direction control points along u -direction,
Performing $(d + 1)$ curve evaluations
 - Then evaluate the curve in v -direction
 - (or the other way around, interchanging u, v -directions)

Illustration



B-Spline Patches

Alternative:

- 2D de Boor algorithm
- Works similar to the 2D de Casteljau algorithm but with different weights
(we can use tensor–product blossoming to derive the weights)

Tensor Product Surfaces

Rational Patches

Rational Patches

Rational Patches

- We can use rational Bézier/B-splines to create the patches (“rational Bézier patches” / “NURBS-patches”)
- Idea:
 - Form a parametric surface in 4D, homogenous space
 - Then project to $\omega = 1$ to obtain the surface in Euclidian 3D space
- In short: Just use homogeneous coordinates everywhere

Rational Patch

Rational Bézier Patch:

$$\mathbf{f}^{(hom)}(u, v) = \sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(u) B_j^{(d)}(v) \begin{pmatrix} \omega_{i,j} \mathbf{p}_{i,j} \\ \omega_{i,j} \end{pmatrix}$$

$$\mathbf{f}^{(Eucl)}(u, v) = \frac{\sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(u) B_j^{(d)}(v) \omega_{i,j} \mathbf{p}_{i,j}}{\sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(u) B_j^{(d)}(v) \omega_{i,j}}$$

Rational Patch

Rational B-Spline Patch:

$$\mathbf{f}^{(hom)}(u, v) = \sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \begin{pmatrix} \omega_{i,j} \mathbf{p}_{i,j} \\ \omega_{i,j} \end{pmatrix}$$

$$\mathbf{f}^{(Eucl)}(u, v) = \frac{\sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \omega_{i,j} \mathbf{p}_{i,j}}{\sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \omega_{i,j}}$$

Remark: Rational Patches

Observation:

- Euclidian surface is not a tensor product surface
 - Denominator depends on both u and v
- Homogeneous space: 4D surface is a tensor product surface.

$$\mathbf{f}^{(Eucl)}(u, v) = \frac{\sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(u) B_j^{(d)}(v) \omega_{i,j} \mathbf{p}_{i,j}}{\sum_{i=0}^d \sum_{j=0}^d B_i^{(d)}(u) B_j^{(d)}(v) \omega_{i,j}}$$

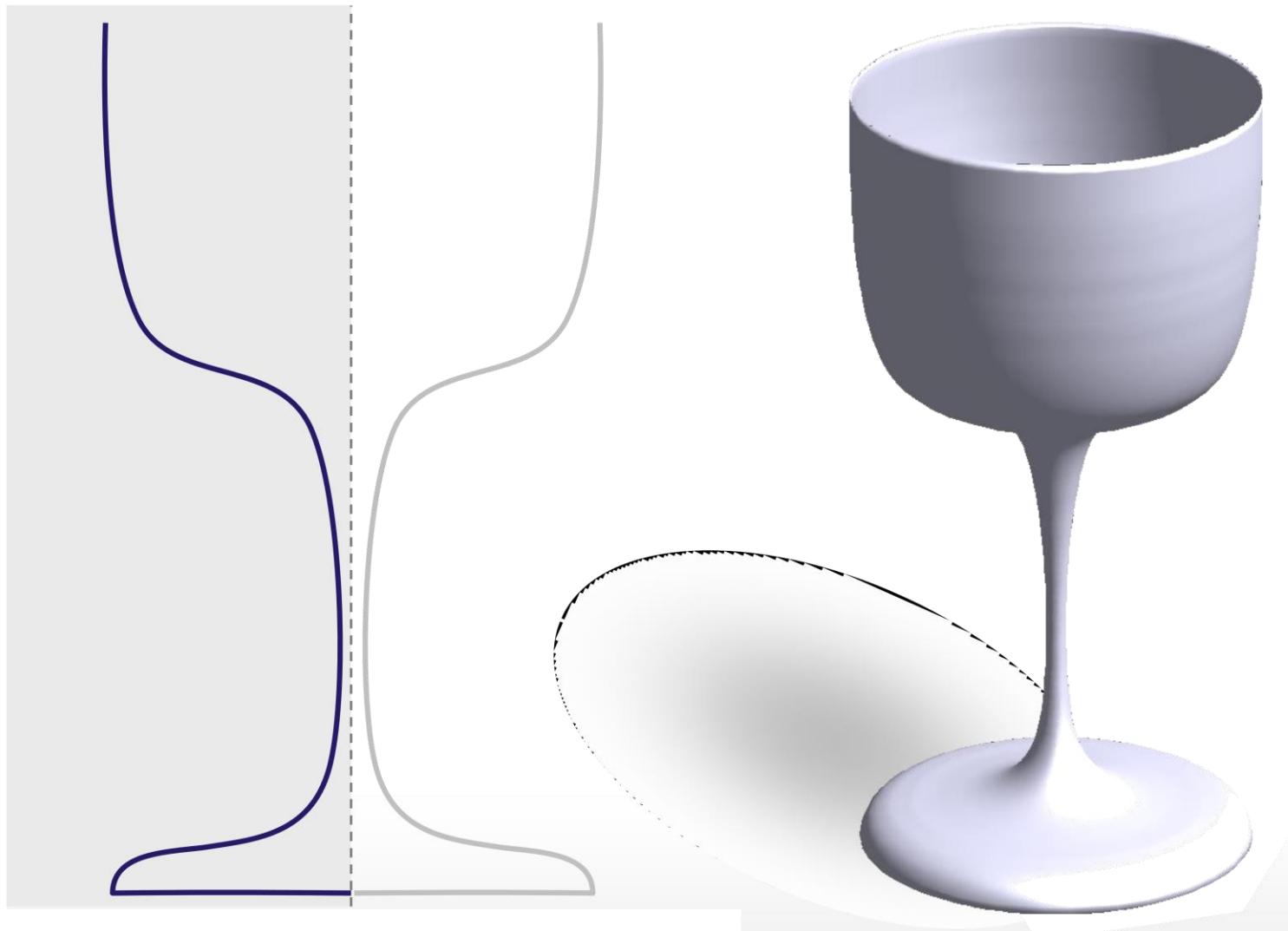
$$\mathbf{f}^{(Eucl)}(u, v) = \frac{\sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \omega_{i,j} \mathbf{p}_{i,j}}{\sum_{i=0}^d \sum_{j=0}^d N_i^{(d)}(u) N_j^{(d)}(v) \omega_{i,j}}$$

Surfaces of Revolution

Advantages of rational patches:

- Rational patches can represent surfaces of revolution exactly.
- Examples:
 - Cylinders
 - Cones
 - Spheres
 - Ellipsoids
 - Tori
- Question: given a cross section curve, how do we get the control points for the 3D surface?

Surfaces of Revolution



Surfaces of Revolution

Given:

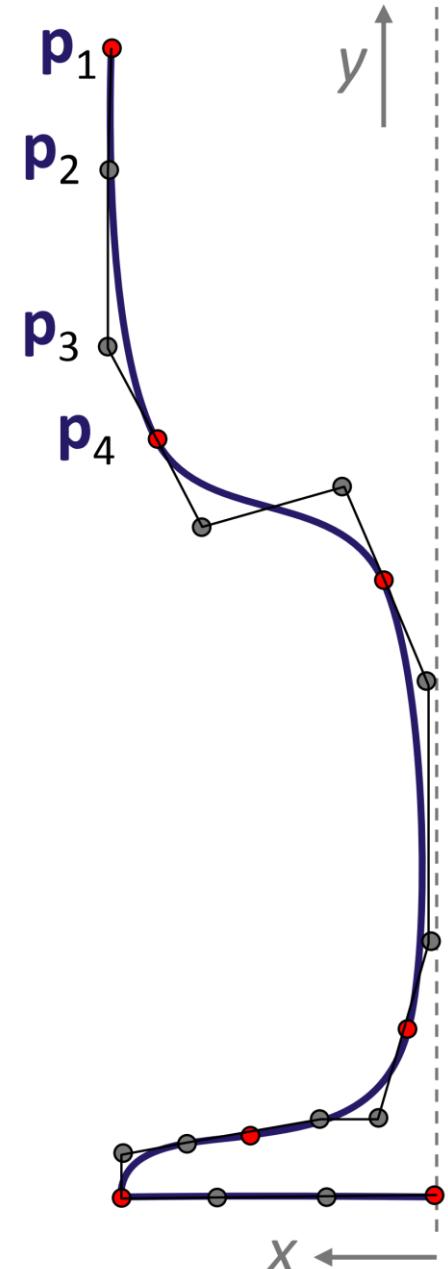
- Control points p_1, \dots, p_n of curve (“generatrix”)

We want to compute:

- Control points $p_{i,j}$ of a rational surface

Such that:

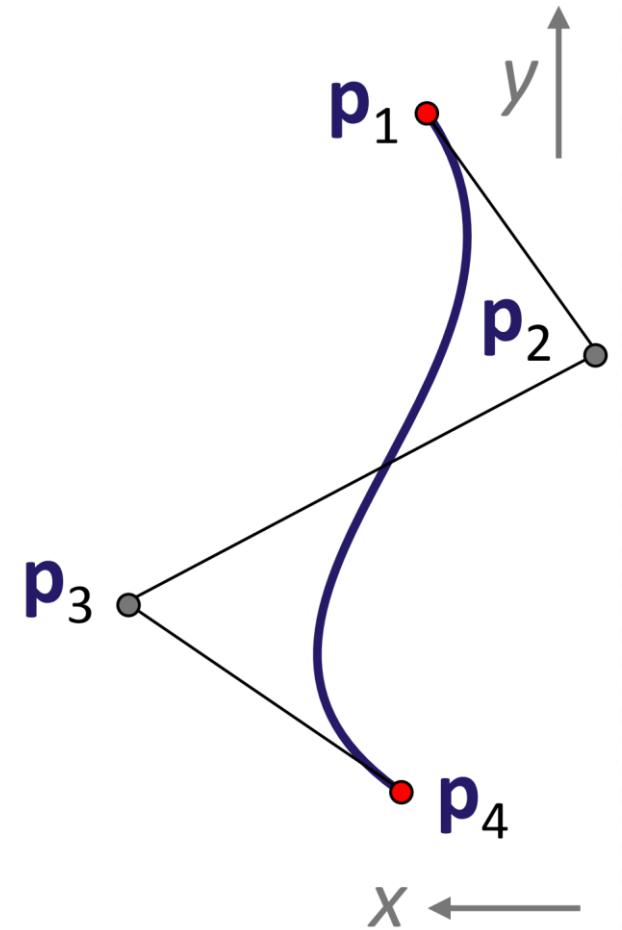
- The surface describes the surface of revolution that we obtain by rotating the curve around the y axis (w.l.o.g.)



Surfaces of Revolution

Simplification:

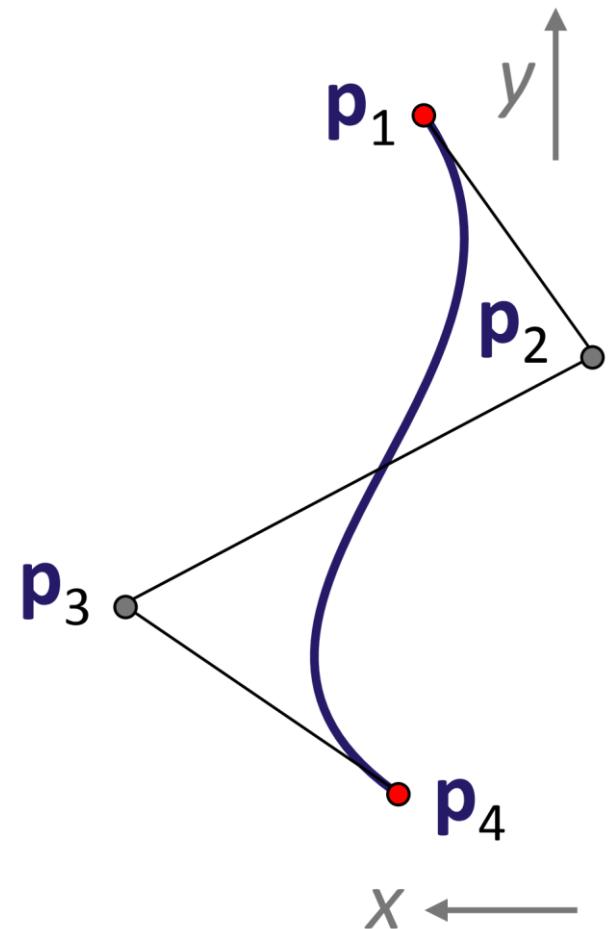
- We look only at a single rational Bézier segment
- Applying the scheme to multiple segments together is straightforward
- The same idea also works for B-splines



Surfaces of Revolution

Construction:

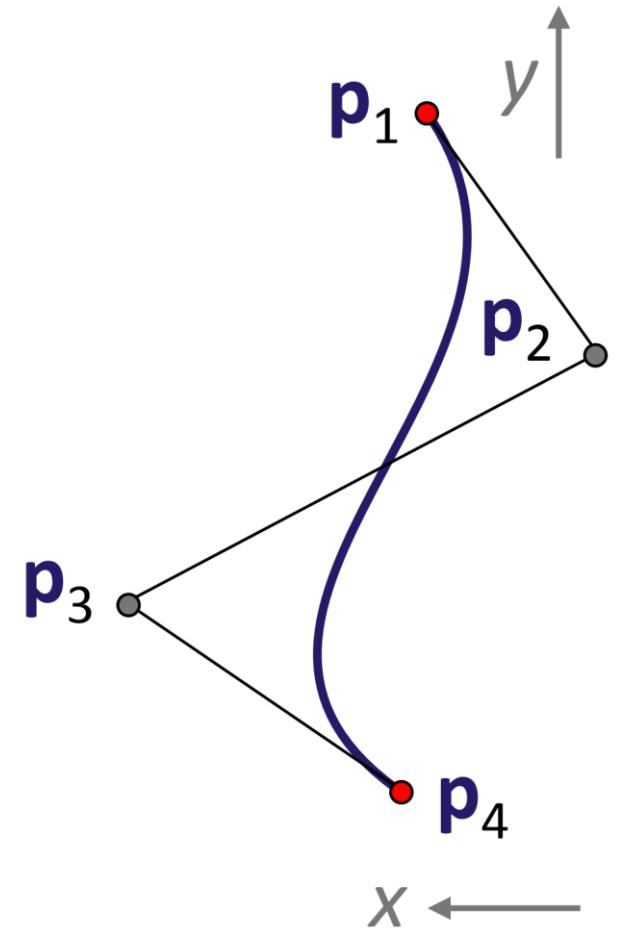
- We are given control points $\mathbf{p}_1, \dots, \mathbf{p}_{d+1}$ (d is the degree in u direction)
- We introduce a new parameter v
- In v direction, we use quadratic Bézier curves (2nd degree basis in v -direction)



Surfaces of Revolution

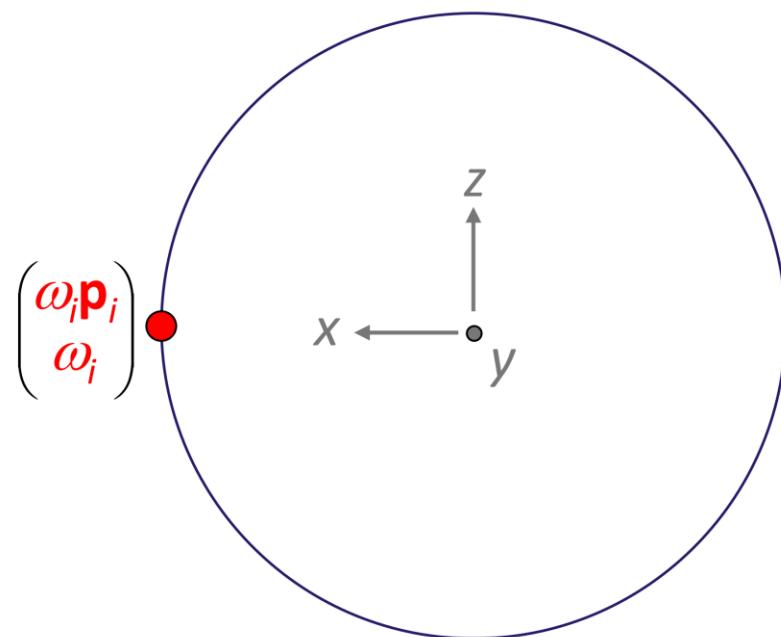
Key Idea:

- For u -direction curves: control points (and thus the curves) must move on circles around the y -axis
- Circles must have the same parametrization (this is easy)
- This means, the control points rotate around the y -axis
- Affine invariance will make the whole curve rotate, we get the desired surface of revolution



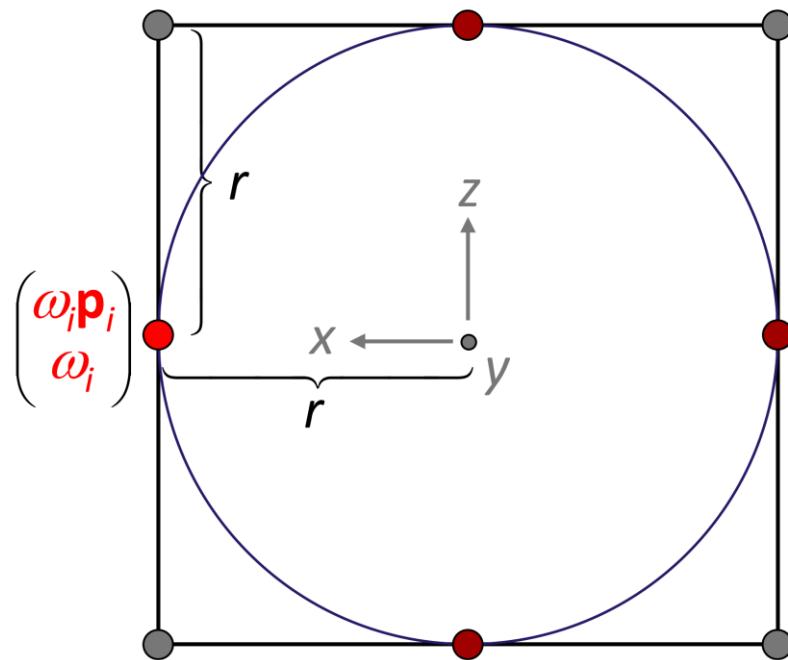
Surface of Revolution

Making one point rotate around the y-axis:



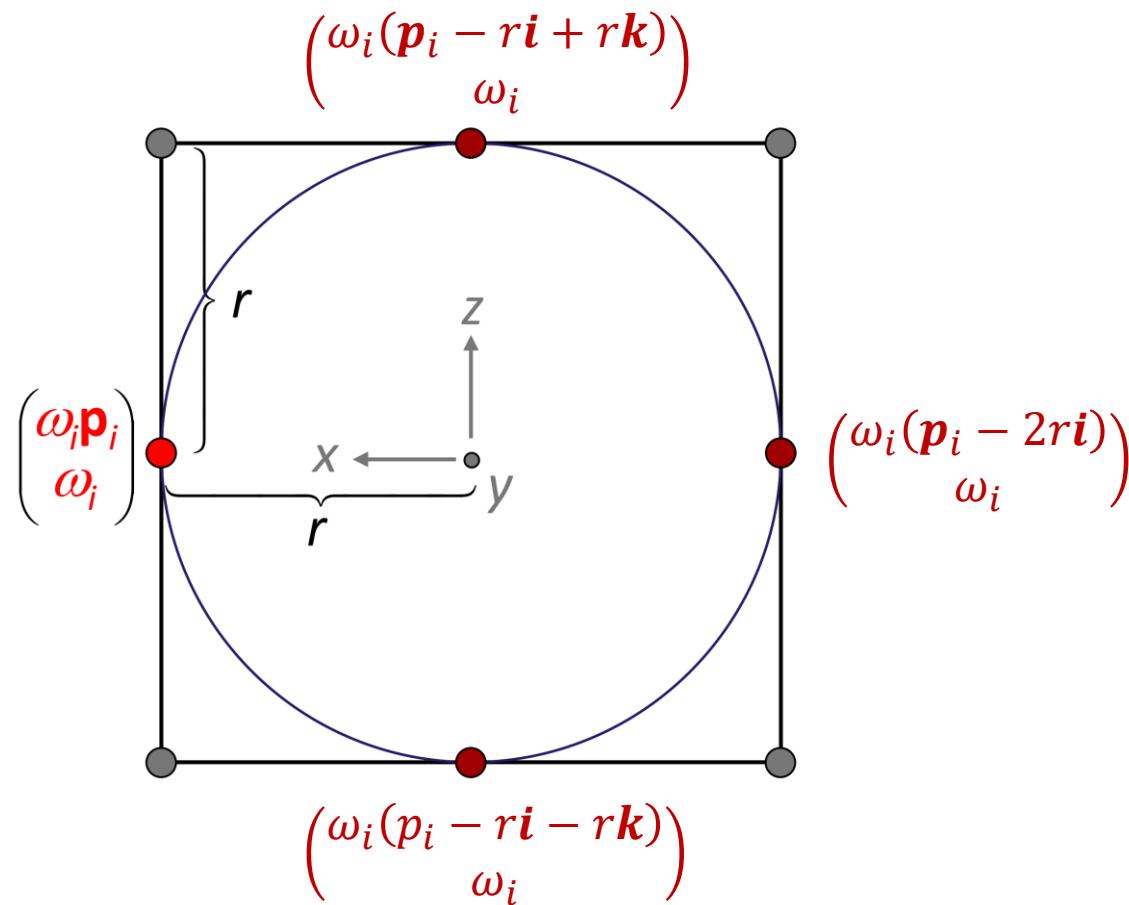
Surface of Revolution

Making one point rotate around the y-axis:



Surface of Revolution

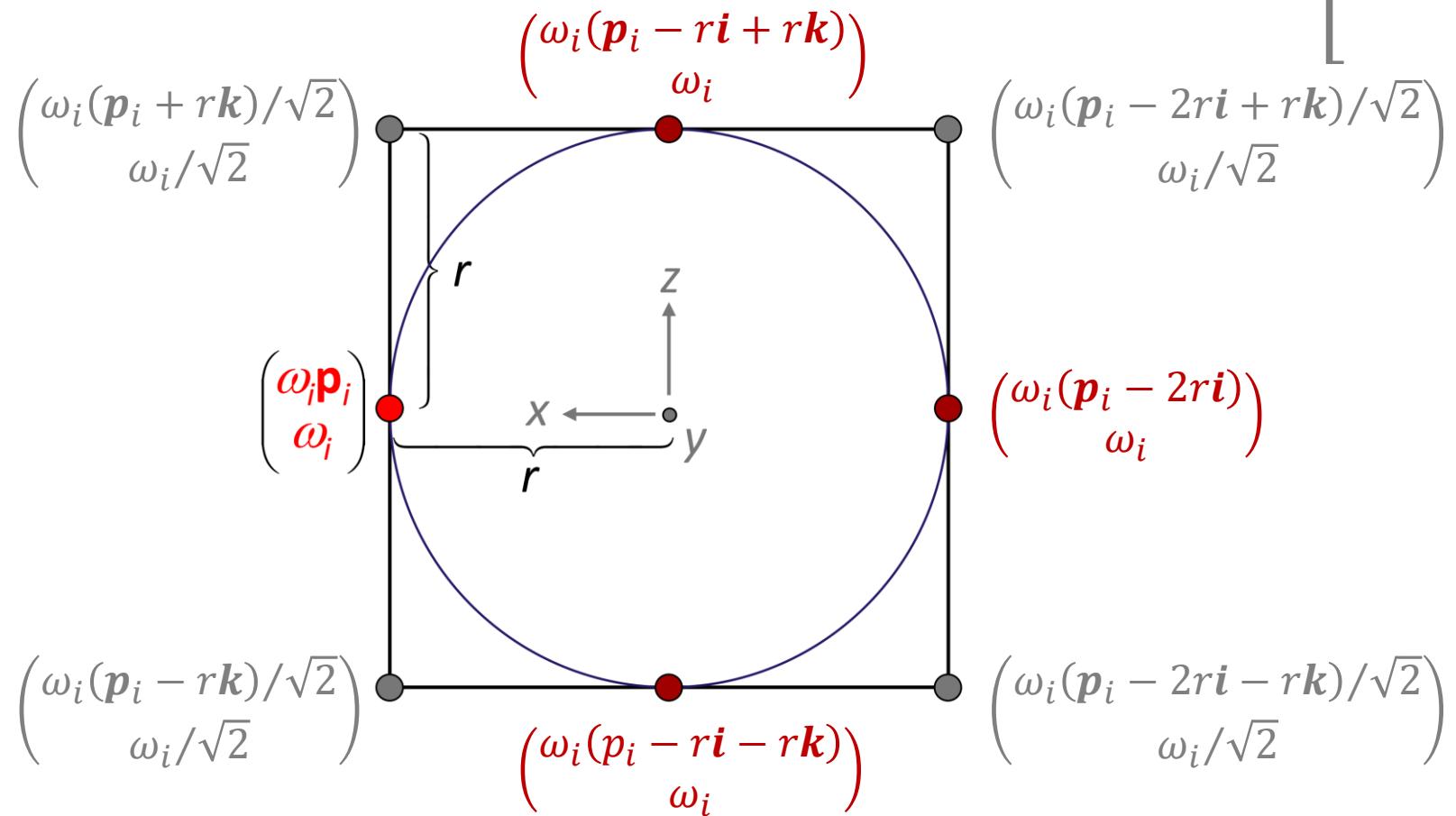
Making one point rotate around the y-axis:



$$i := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, k := \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Surface of Revolution

Making one point rotate around the y-axis:



$$i := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, k := \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Remark

What we get:

- We obtain 4 segments, i.e. 4 patches for each Bézier segment
- A similar construction with 3 segments exists as well

Does the scheme yield a circle for weights $\neq 1$ in the generatrix curve?

- Common factors in weights cancel out
- Therefore, we still obtain a circle at these points
- Parametrization does not change either

Benefit

With this construction, it is straightforward to create:

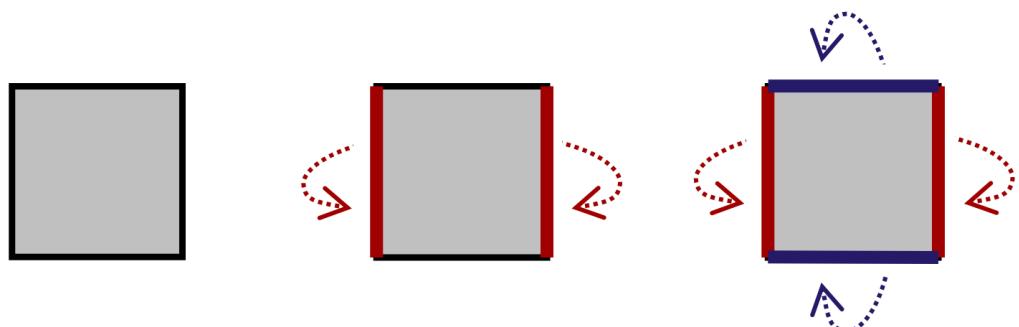
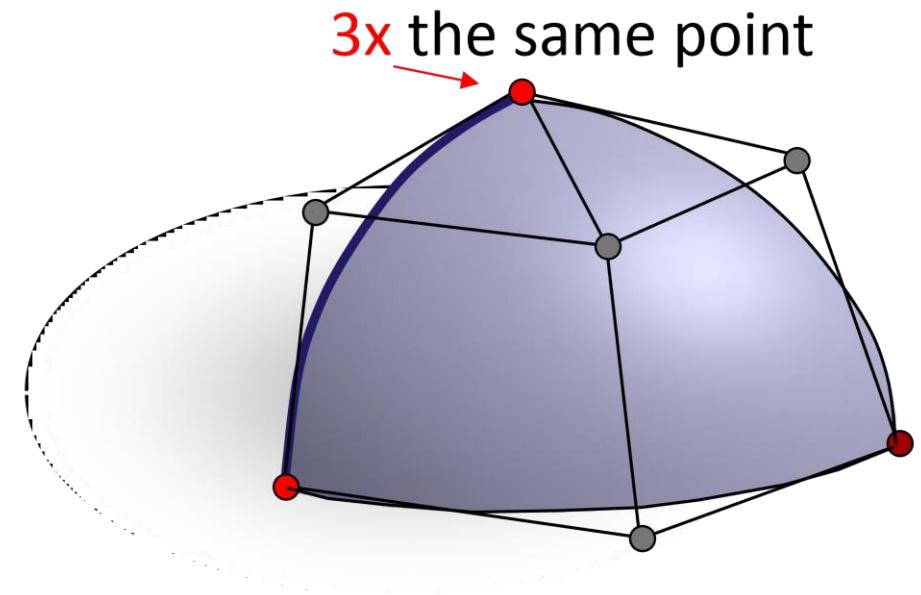
- Spheres
- Tori
- Cylinders
- Cones

And affine transformations of these (e.g. ellipsoids)

Parametrization Restrictions

Remaining problem:

- The sphere and the cone are not regularly parametrized (double control points)
- Might cause trouble (normal computation, tessellation)
- In general: no sphere, or n -tori ($n > 1$) can be parametrized without degeneracies
- What works: open surfaces, cylinders, tori



Curves on Surfaces, trimmed NURBS

Quad patch problem:

- All of our shapes are parameterized over rectangular regions
- General boundary curves are hard to create
- Topology fixed to a disc (or cylinder, torus)
- No holes in the middle
- Assembling complicated shapes is painful
 - Lots of pieces
 - Continuity conditions for assembling pieces become complicated
 - Cannot use C^2 B-splines continuity along boundaries when using multiple pieces

Curves on Surfaces, trimmed NURBS

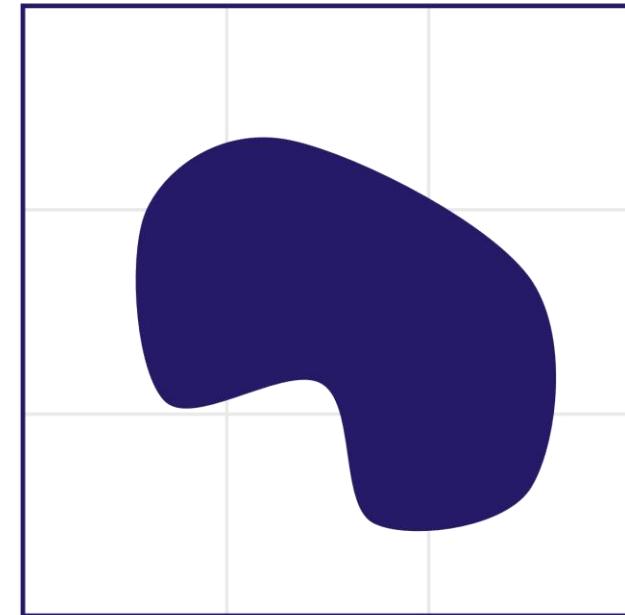
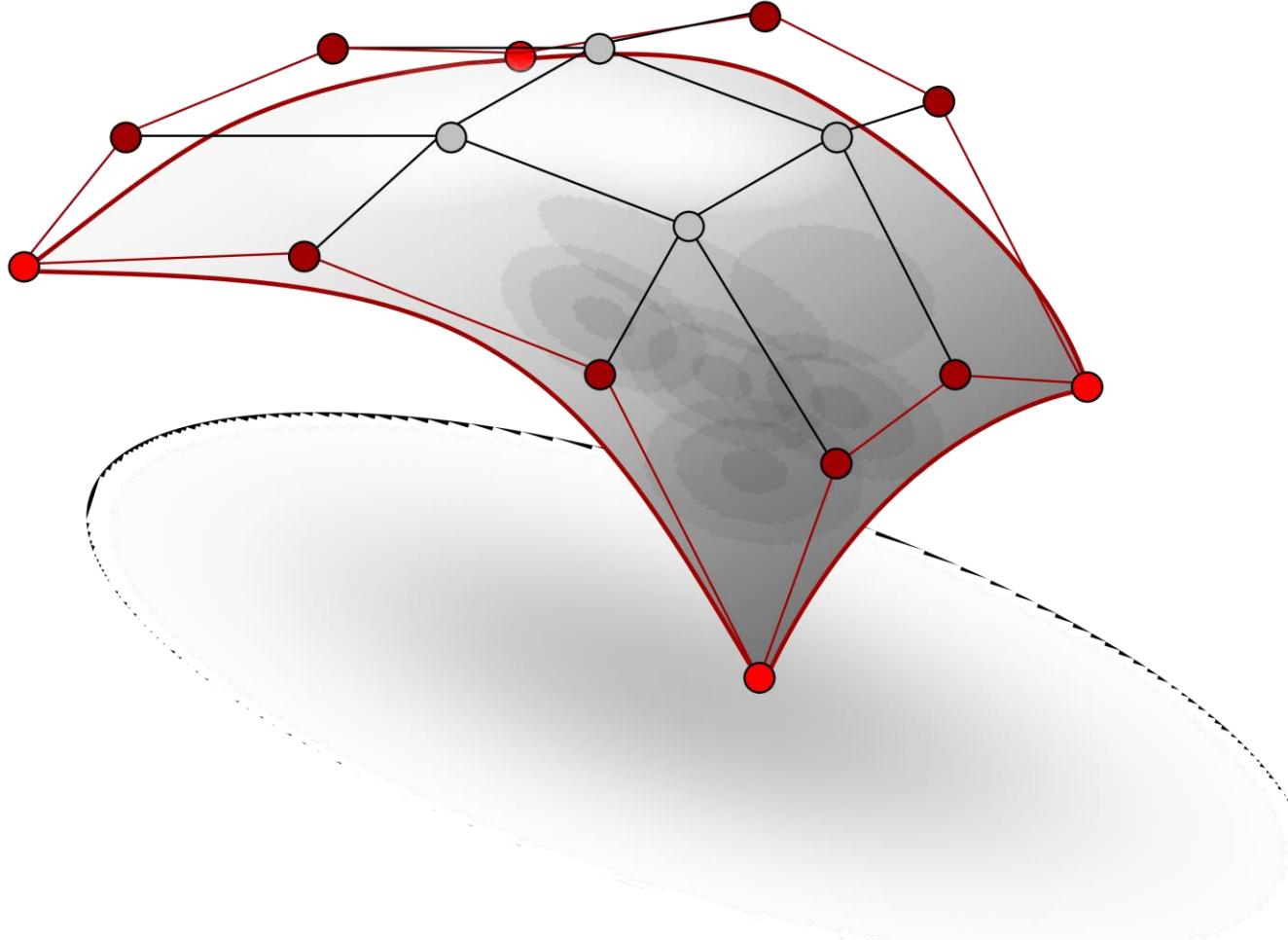
Consequence:

- We need more control over the parameter domain
- One solution is *trimming* using *curves on surfaces (CONS)*
- Standard tool in CAD: *trimmed NURBS*

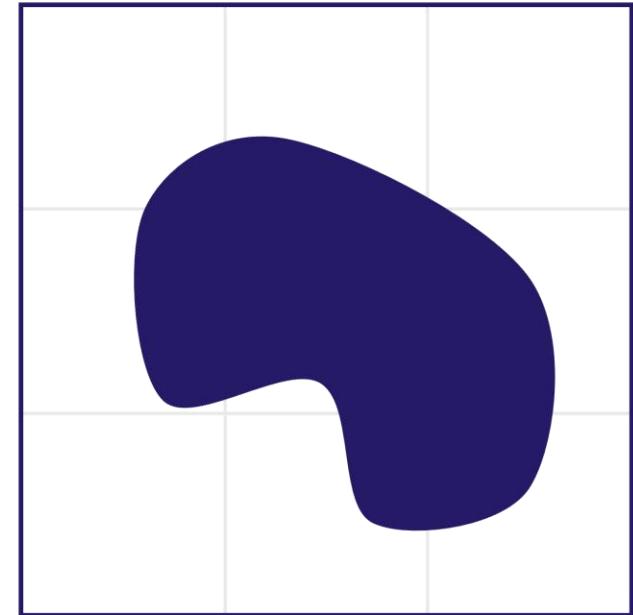
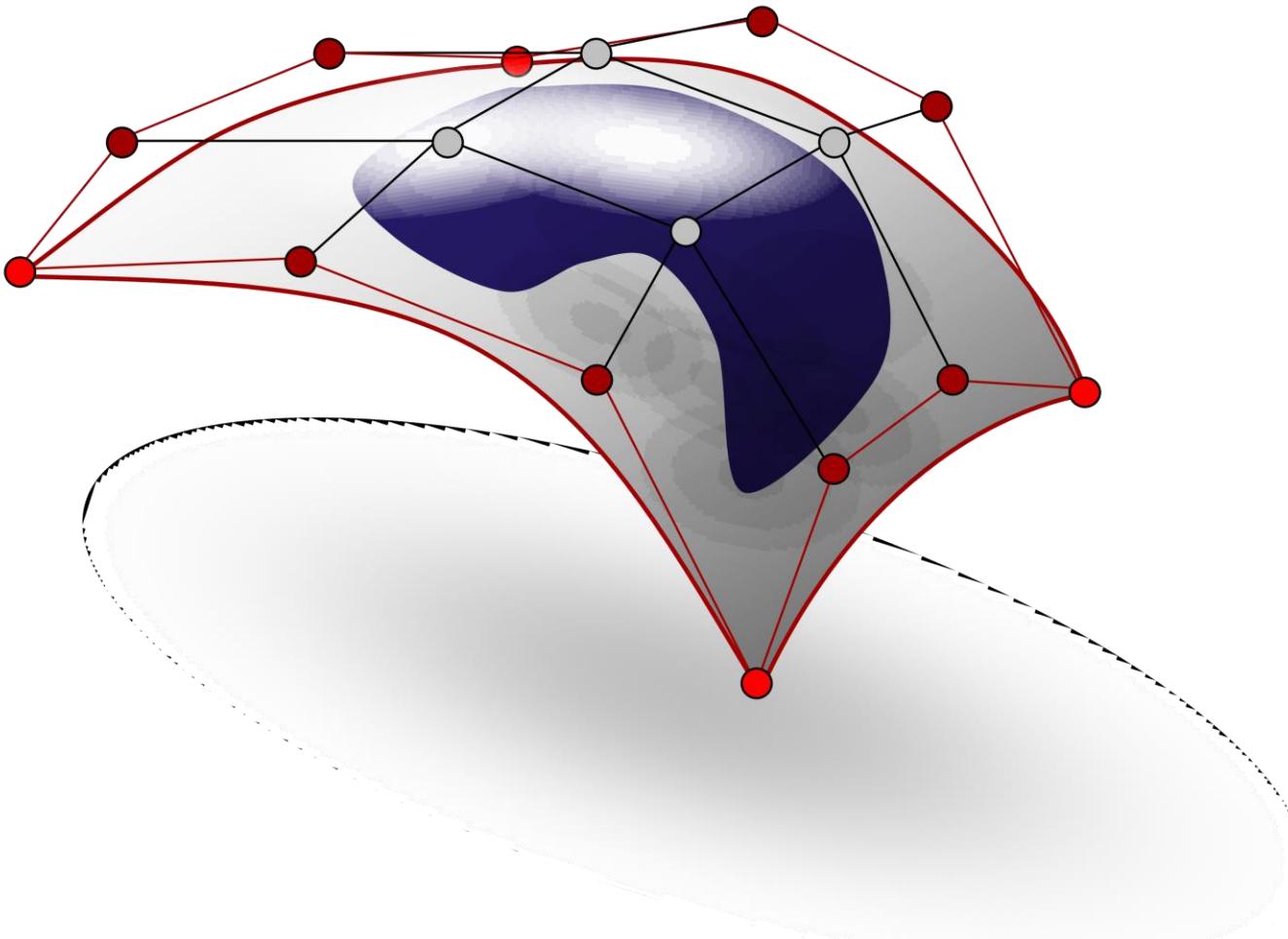
Basic idea:

- Specify a curve in the parameter domain that encapsulates one (or more) pieces of area
- Tessellate the parameter domain accordingly to cut out the trimmed piece (rendering)

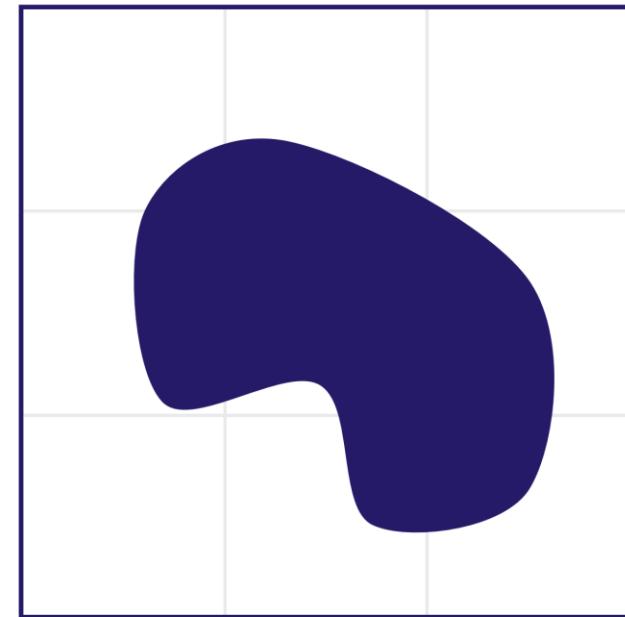
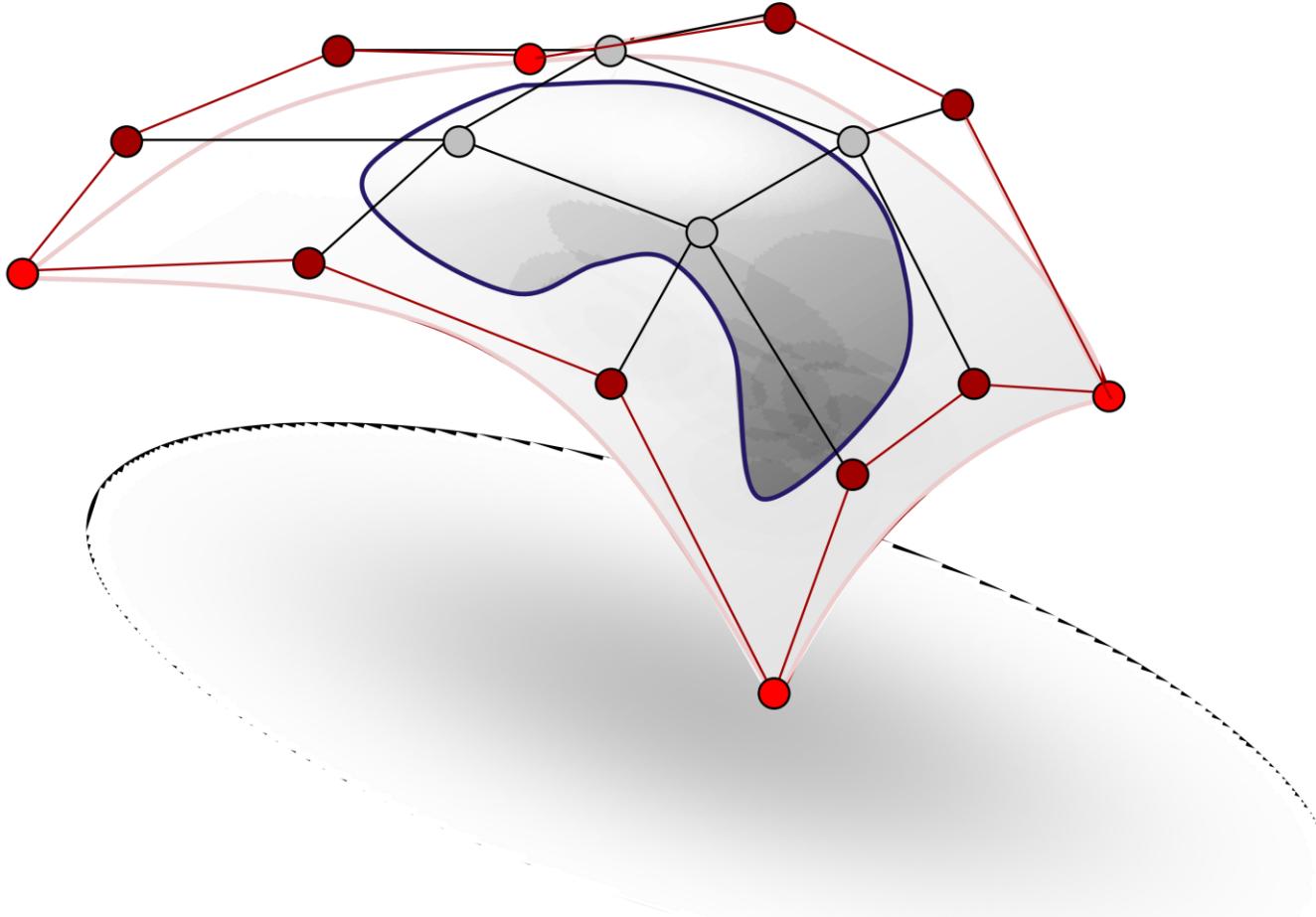
Curves-on-Surfaces (CONS)



Curves-on-Surfaces (CONS)



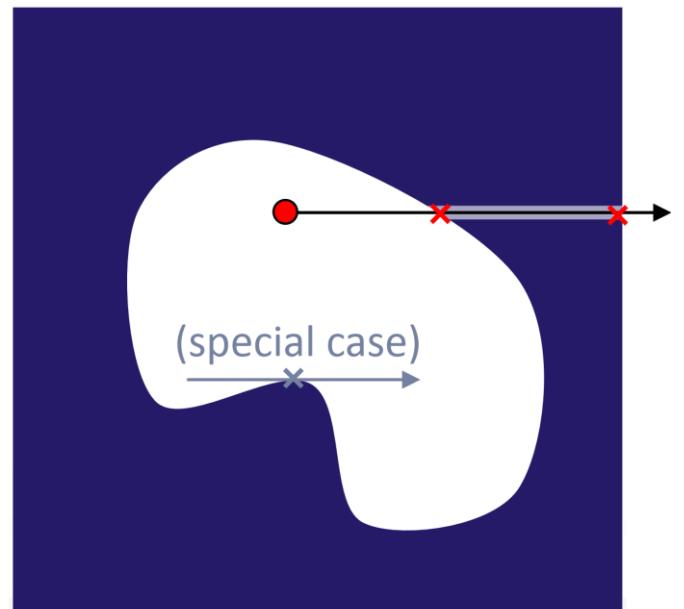
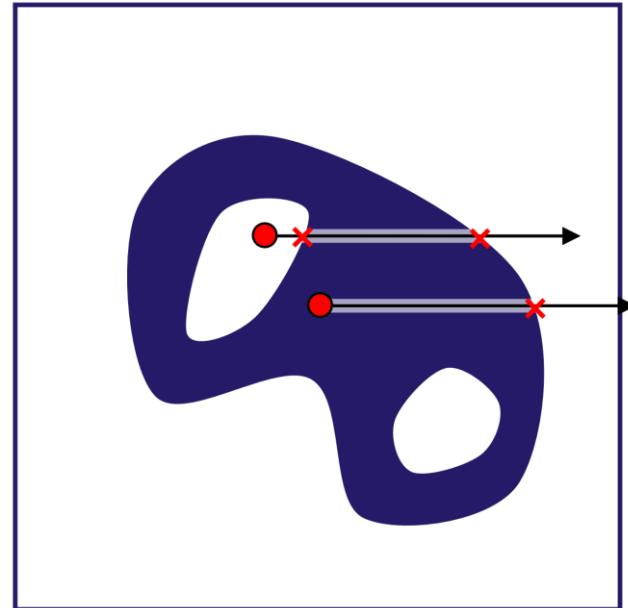
Curves-on-Surfaces (CONS)



General Shapes

General shapes with holes:

- Draw multiple curves
- Inside / outside test:
 - If any ray in the parameter domain intersects the boundary curves an odd number of times, the point is inside
 - Outside otherwise
 - Implementation needs to take care of special cases (critical points with respect to normal of the ray)
 - Nasty, but doable

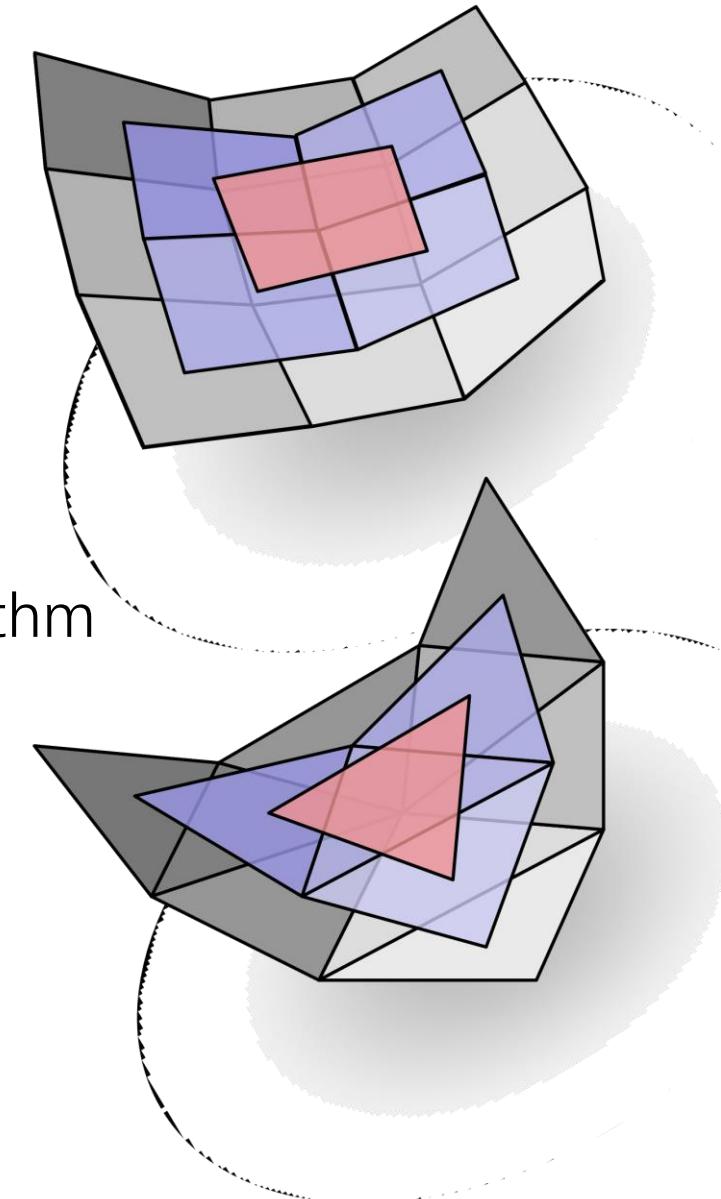


Total Degree Surfaces

Bézier Triangles

Alternative surface definition: Bézier triangles

- Constructed according to given total degree
 - Completely symmetric: degree anisotropy
- Can be derived using a triangular de Casteljau algorithm
 - Blossoming formalism is very helpful for defining Bézier Triangles
 - Barycentric interpolation of blossom values



Blossoms for Total Degree Surfaces

Blossom with points as arguments:

- Polar form degree d with points as input and output:

$$\begin{aligned} \mathbf{F}: \mathbb{R}^n &\rightarrow \mathbb{R}^m \\ \mathbf{f}: \mathbb{R}^{d \times n} &\rightarrow \mathbb{R}^m \end{aligned}$$

points as arguments

- Required Properties:

- Diagonality: $\mathbf{f}(\mathbf{t}, \mathbf{t}, \dots, \mathbf{t}) = \mathbf{F}(\mathbf{t})$
- Symmetry: $\mathbf{f}(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_d) = \mathbf{f}(\mathbf{t}_{\pi(1)}, \mathbf{t}_{\pi(2)}, \dots, \mathbf{t}_{\pi(d)})$
for all permutations of indices π

- Multi-affine: $\sum \alpha_k = 1$

$$\begin{aligned} \Rightarrow \mathbf{f}(\mathbf{t}_1, \dots, \sum \alpha_k \mathbf{t}_i^{(k)}, \dots, \mathbf{t}_d) \\ = \alpha_1 \mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_i^{(1)}, \dots, \mathbf{t}_d) + \dots + \alpha_n \mathbf{f}(\mathbf{t}_1, \dots, \mathbf{t}_i^{(n)}, \dots, \mathbf{t}_d) \end{aligned}$$

Example

Example: bivariate monomial basis

- In powers of (u, v) :

$$B = \{1, u, v, u^2, uv, v^2\}$$

- Blossom form: multilinear in (u_1, u_2, v_1, v_2)

$$\begin{aligned} B = \{1, \\ \frac{1}{2}(u_1 + u_2), \frac{1}{2}(v_1 + v_2), \\ u_1 u_2, \frac{1}{4}(u_1 v_1 + u_1 v_2 + u_2 v_1 + u_2 v_2), v_1 v_2\} \end{aligned}$$

Barycentric Coordinates

Barycentric Coordinates:

- Planar case:

Barycentric combinations of 3 points

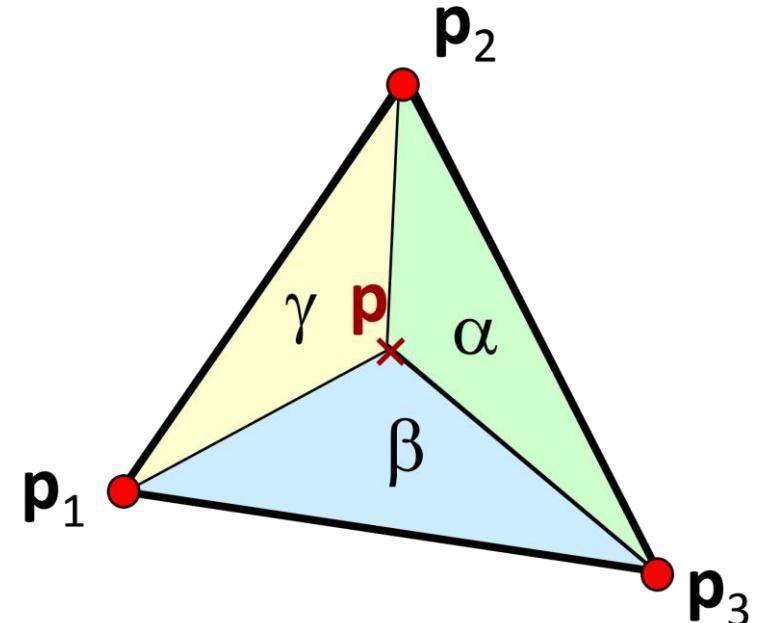
$$\mathbf{p} = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3, \text{ with } \alpha + \beta + \gamma = 1$$

$$\gamma = 1 - \alpha - \beta$$

- Area formulation

$$\gamma = 1 - \alpha - \beta$$

$$\alpha = \frac{\text{area}(\Delta(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \beta = \frac{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \gamma = \frac{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}))}{\text{area}(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}$$



Barycentric Coordinates

Barycentric Coordinates:

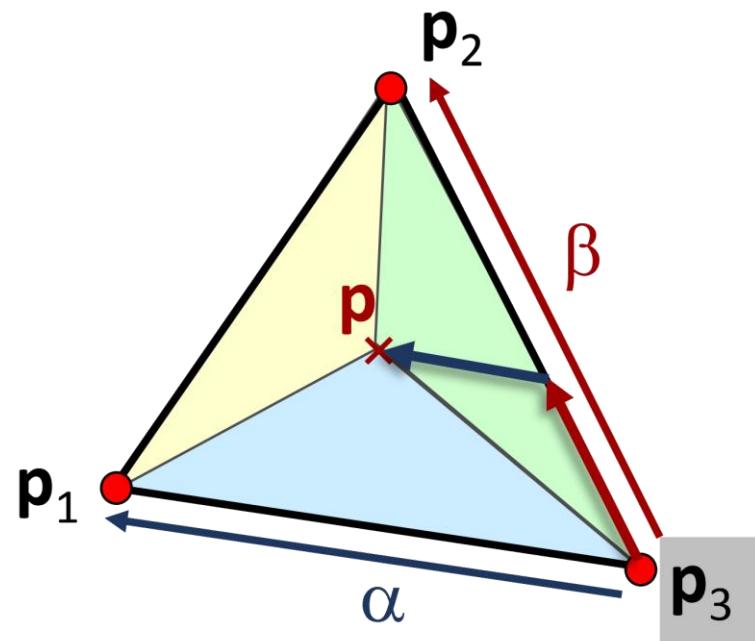
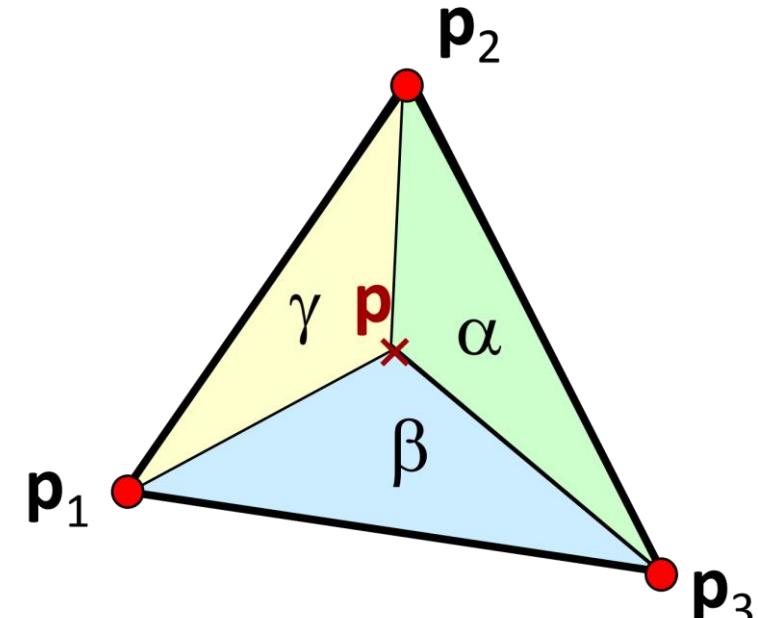
- Linear formulation:

$$\mathbf{p} = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3$$

$$= \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + (1 - \alpha - \beta) \mathbf{p}_3$$

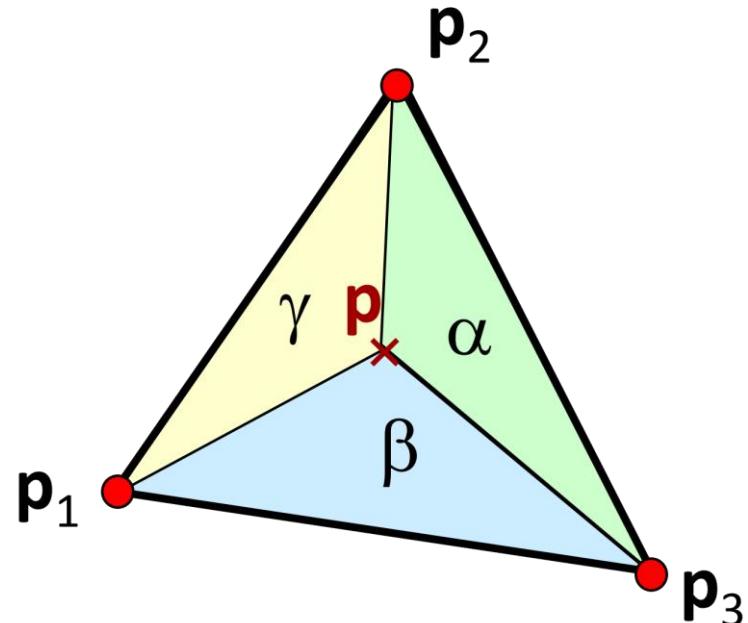
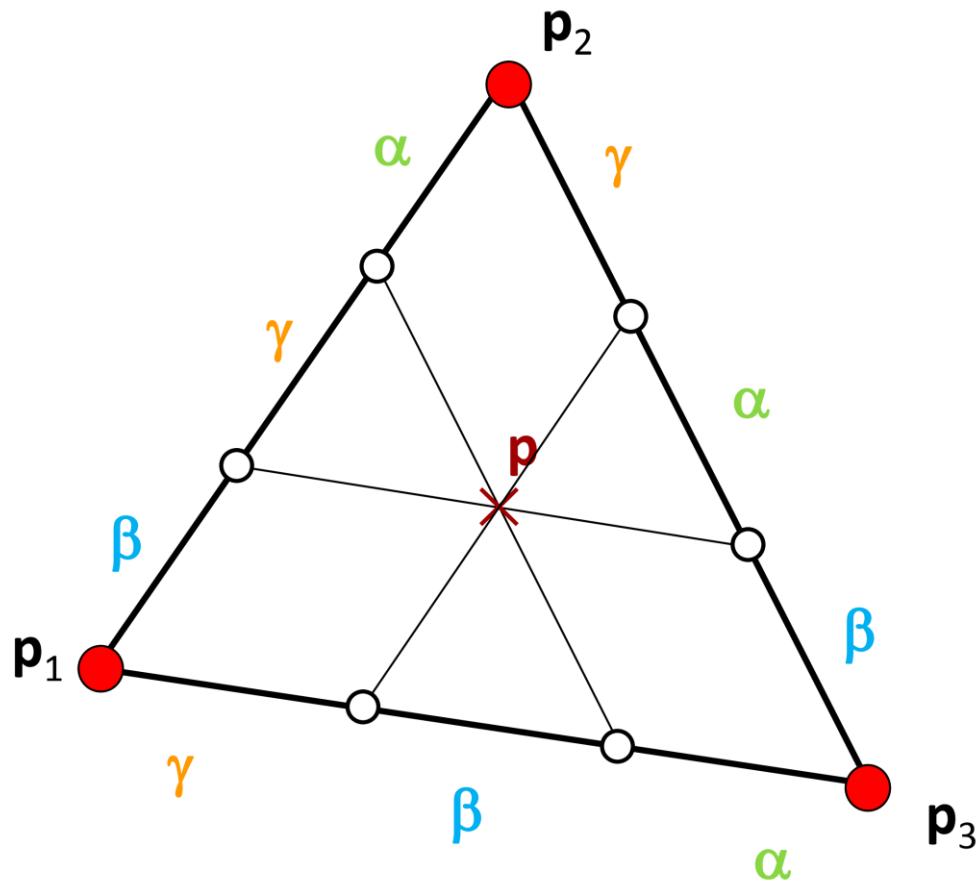
$$= \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \mathbf{p}_3 - \alpha \mathbf{p}_3 - \beta \mathbf{p}_3$$

$$= \mathbf{p}_3 + \underline{\alpha(\mathbf{p}_1 - \mathbf{p}_3)} + \underline{\beta(\mathbf{p}_2 - \mathbf{p}_3)}$$



Barycentric Coordinates

$$\mathbf{p} = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3, \text{ with } \alpha + \beta + \gamma = 1$$



Bézier Triangles: Overview

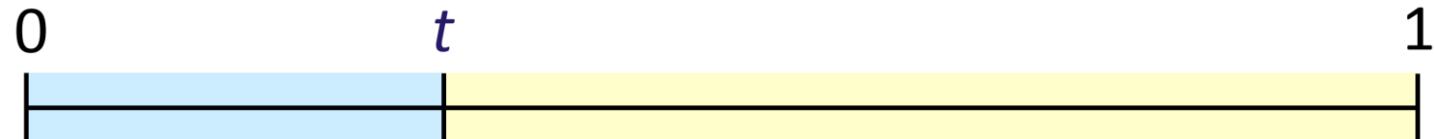
Bézier Triangles: Main Ideas

- Use 3D points as inputs to the blossoms
- These are Barycentric coordinates of a parameter triangle $\{a, b, c\}$
- Use 3D points as outputs
- Form control points by multiplying parameter points, just as in the curve case: $p(\underbrace{a, \dots, a}_{i}, \underbrace{b, \dots, b}_{j}, \underbrace{c, \dots, c}_{k})$
- De Casteljau Algorithm: compute polynomial values $p(x, \dots, x)$ by barycentric interpolation

Plugging in the Barycentric Coord's

Analog: 2D curves in barycentric coordinates

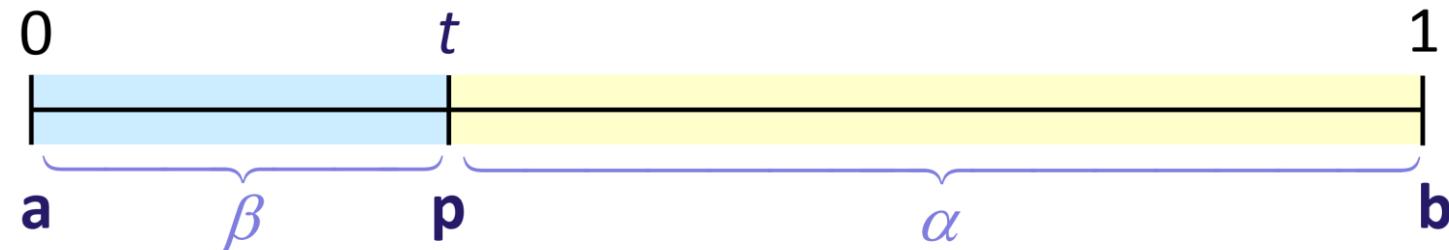
- Barycentric coordinates for 2D curves:



Plugging in the Barycentric Coord's

Analog: 2D curves in barycentric coordinates

- Barycentric coordinates for 2D curves:



- $p = \alpha a + \beta b, \quad \alpha + \beta = 1$

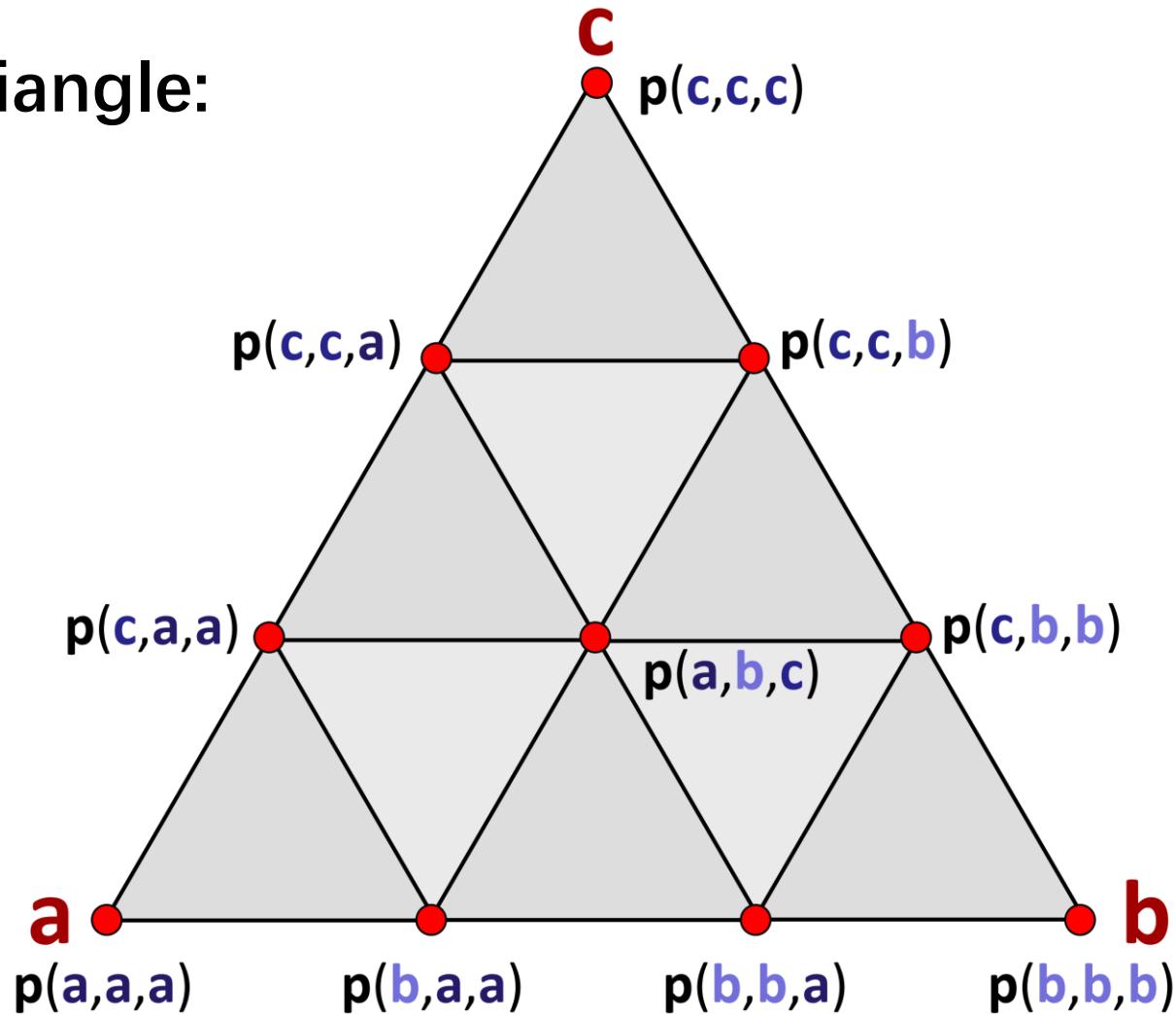
- Bézier splines:

$$\mathbf{F}(t) = \sum_{i=0}^d \binom{d}{i} (1-t)^i t^{d-i} \mathbf{f}(\underbrace{\mathbf{a}, \dots, \mathbf{a}}_i, \underbrace{\mathbf{b}, \dots, \mathbf{b}}_{d-i}) \quad (\text{standard form})$$

$$\mathbf{F}(\mathbf{p}) = \sum_{\substack{i+j=d \\ i \geq 0, j \geq 0}} \frac{d!}{i!j!} \alpha^i \beta^j \mathbf{f}(\underbrace{\mathbf{a}, \dots, \mathbf{a}}_i, \underbrace{\mathbf{b}, \dots, \mathbf{b}}_j) \quad (\text{barycentric form})$$

Example

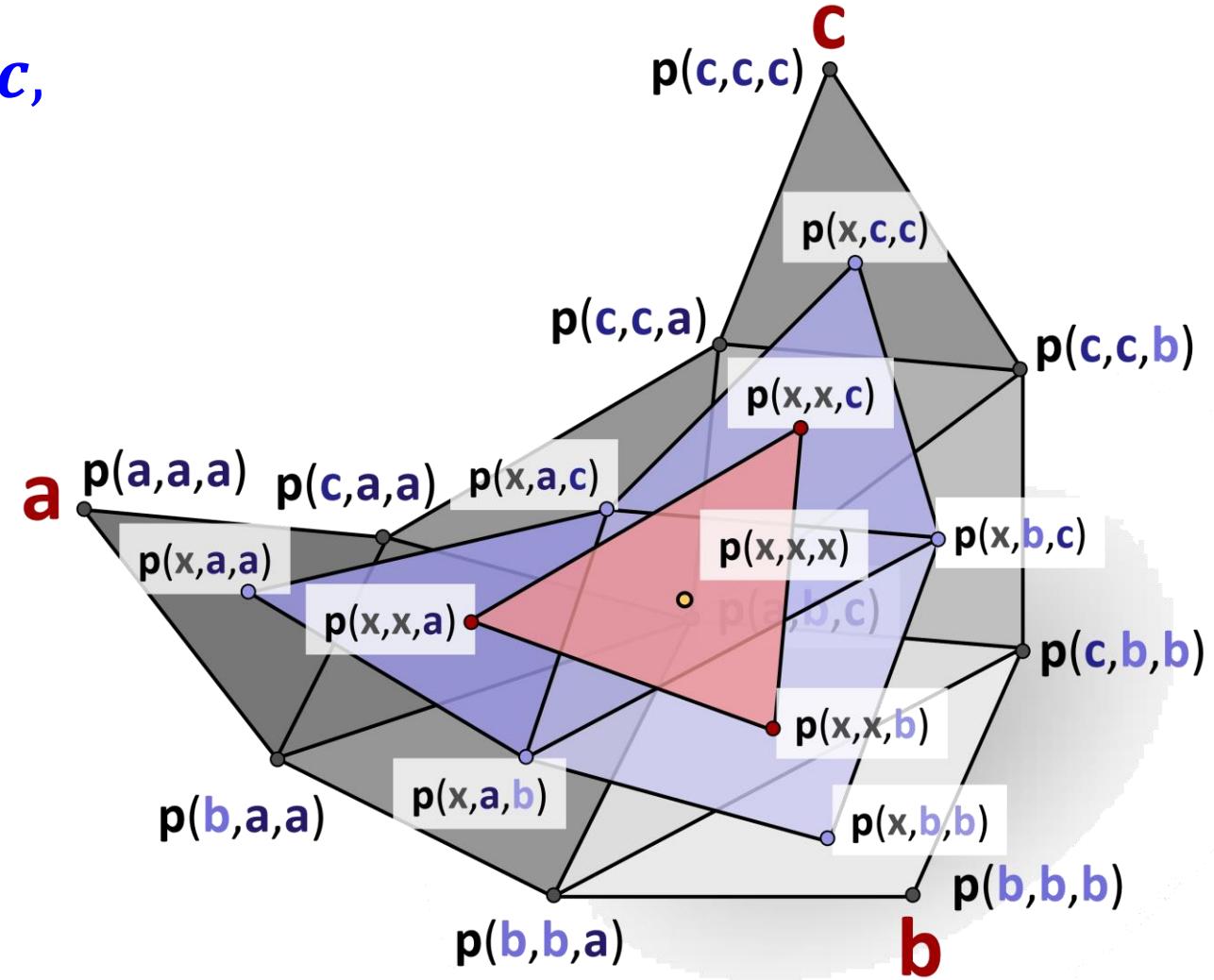
Cubic Bézier Triangle:



De Casteljau Algorithm

$$\mathbf{x} = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c},$$

$$\alpha + \beta + \gamma = 1$$



Bernstein Form

Writing this recursion out, we obtain:

$$F(\mathbf{x}) = \sum_{\substack{i+j+k=d \\ i,j,k \geq 0}} \frac{d!}{i! j! k!} \alpha^i \beta^j \gamma^k f(\underbrace{a, \dots, a}_i, \underbrace{b, \dots, b}_j, \underbrace{c, \dots, c}_k)$$

$$\mathbf{x} = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c},$$

$$\alpha + \beta + \gamma = 1$$

- This is the *Bernstein form* of a Bézier triangle surface
- (Proof by induction)

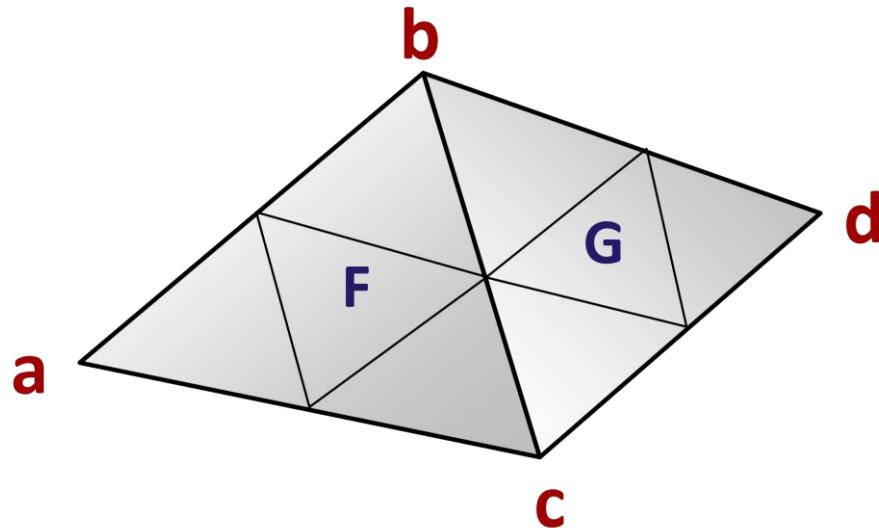
Continuity

We need to assemble Bézier triangles continuously:

- What are the conditions for C^0 , C^1 continuity?
- As an example, we will look at the quadratic case...
- (Try the cubic case as an exercise)

Continuity

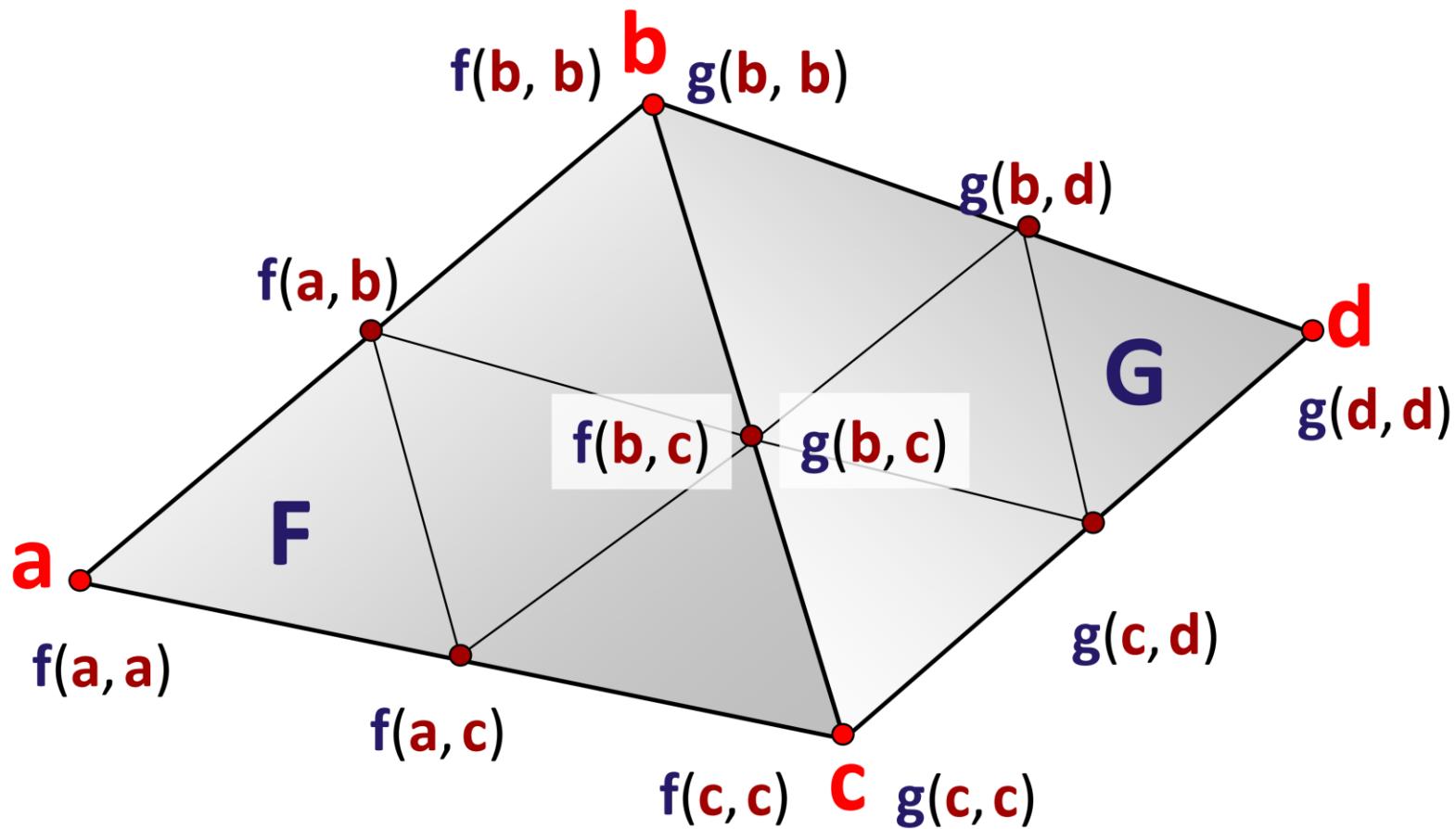
Situation:



- Two Bézier triangles meet along a common edge.
 - Parametrization: $T_1 = \{a, b, c\}$, $T_2 = \{c, b, d\}$
 - Polynomial surfaces $F(T_1)$, $G(T_2)$
 - Control points:
 - $F(T_1)$: $f(a, a), f(a, b), f(b, b), f(a, c), f(c, c), f(b, c)$
 - $G(T_2)$: $g(d, d), g(d, b), g(b, b), g(d, c), g(c, c), g(b, c)$

Continuity

Situation:



Continuity

C^0 continuity:

- The points on the boundary have to agree:

$$f(\mathbf{b}, \mathbf{b}) = g(\mathbf{b}, \mathbf{b})$$

$$f(\mathbf{b}, \mathbf{c}) = g(\mathbf{b}, \mathbf{c})$$

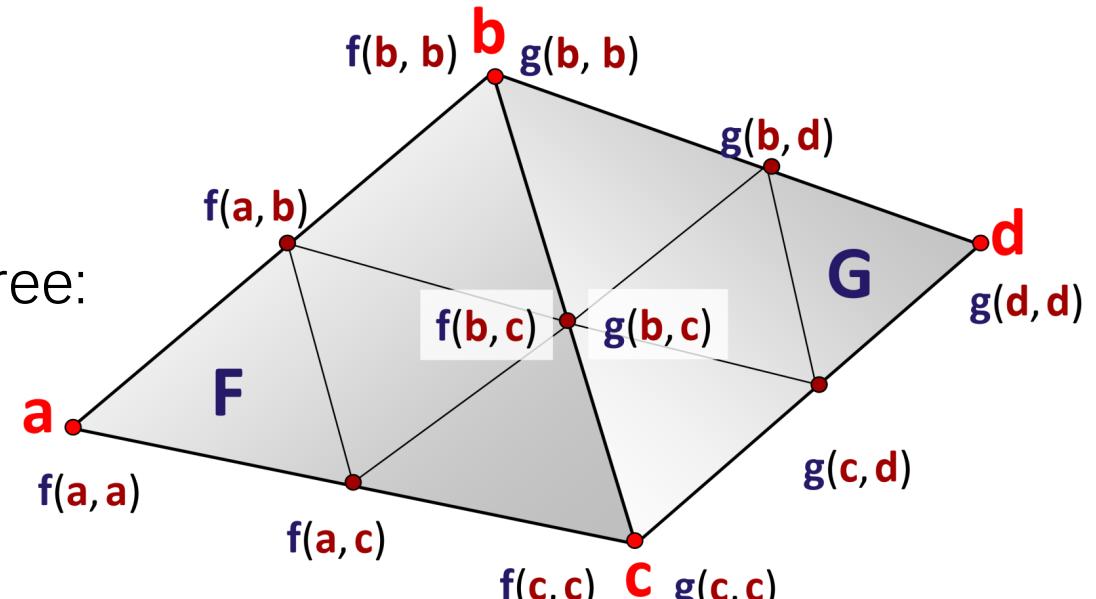
$$f(\mathbf{c}, \mathbf{c}) = g(\mathbf{c}, \mathbf{c})$$

- Proof: Let $\mathbf{x} := \beta\mathbf{b} + \gamma\mathbf{c}$, $\beta + \gamma = 1$

$$\begin{aligned} f(\mathbf{x}, \mathbf{x}) &= \beta f(\mathbf{b}, \mathbf{x}) + \gamma f(\mathbf{c}, \mathbf{x}) \\ &= \beta^2 f(\mathbf{b}, \mathbf{b}) + 2\beta\gamma f(\mathbf{b}, \mathbf{c}) + \gamma^2 f(\mathbf{c}, \mathbf{c}) \end{aligned}$$

$$\begin{array}{ccc} \parallel & \parallel & \parallel \\ g(\mathbf{b}, \mathbf{b}) & g(\mathbf{b}, \mathbf{c}) & g(\mathbf{c}, \mathbf{c}) \end{array}$$

$$\begin{aligned} &= \beta^2 g(\mathbf{b}, \mathbf{b}) + 2\beta\gamma g(\mathbf{b}, \mathbf{c}) + \gamma^2 g(\mathbf{c}, \mathbf{c}) \\ &= \beta g(\mathbf{b}, \mathbf{x}) + \gamma g(\mathbf{c}, \mathbf{x}) = g(\mathbf{x}, \mathbf{x}) \end{aligned}$$



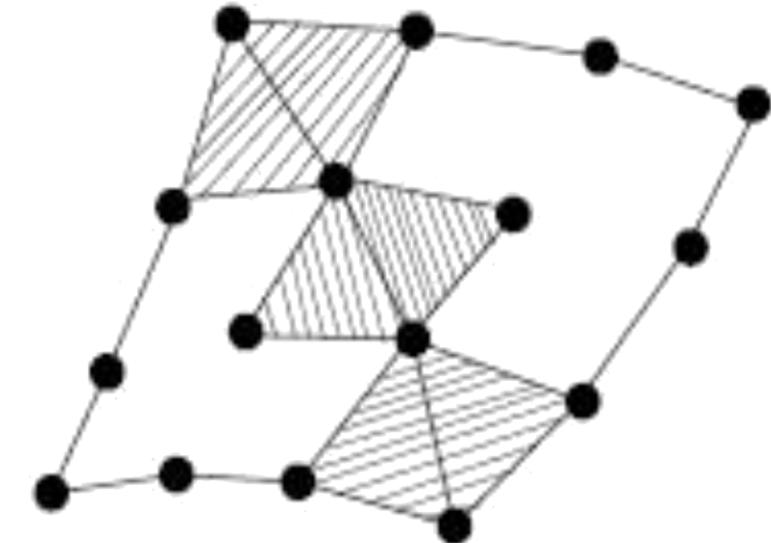
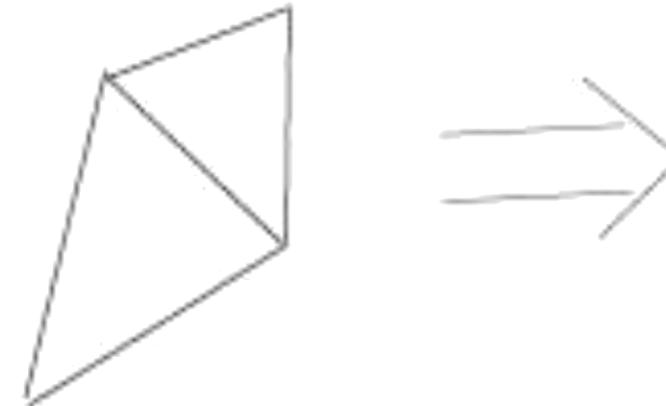
Continuity

C^1 continuity:

- We need C^0 continuity.

In addition:

- Points at hatched quadrilaterals are coplanar
- Hatched quadrilaterals are an affine image of the same parameter quadrilateral



Continuity

C^1 continuity:

- We need C^0 continuity.

In addition:

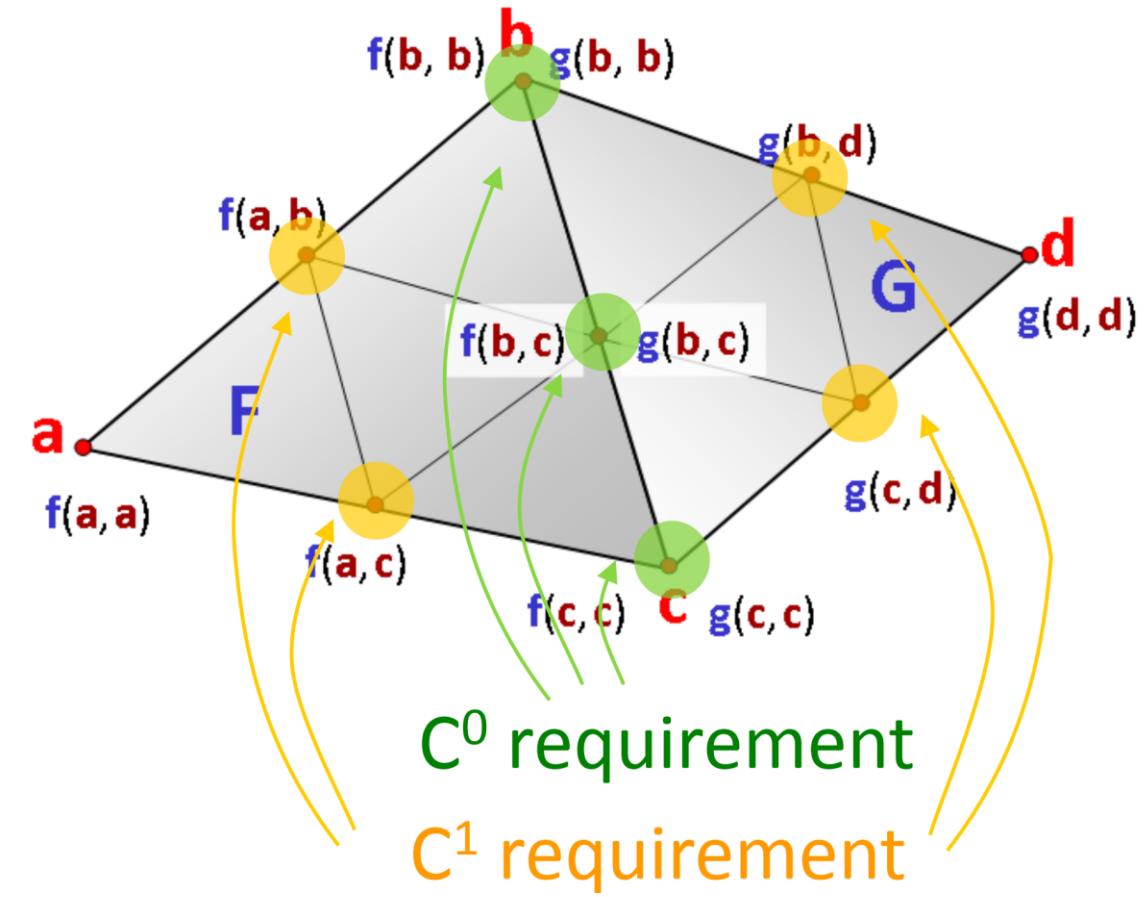
The blossoms have to agree partially:

$$f(\mathbf{a}, \mathbf{b}) = g(\mathbf{a}, \mathbf{b})$$

$$f(\mathbf{b}, \mathbf{d}) = g(\mathbf{b}, \mathbf{d})$$

$$f(\mathbf{a}, \mathbf{c}) = g(\mathbf{a}, \mathbf{c})$$

$$f(\mathbf{c}, \mathbf{d}) = g(\mathbf{c}, \mathbf{d})$$



Continuity

C^1 continuity: Proof

- Derivatives:

$$\frac{\partial}{\partial \hat{d}} F(x)|_{x=p} = f(p, \hat{d})$$

(similar to the curve case)

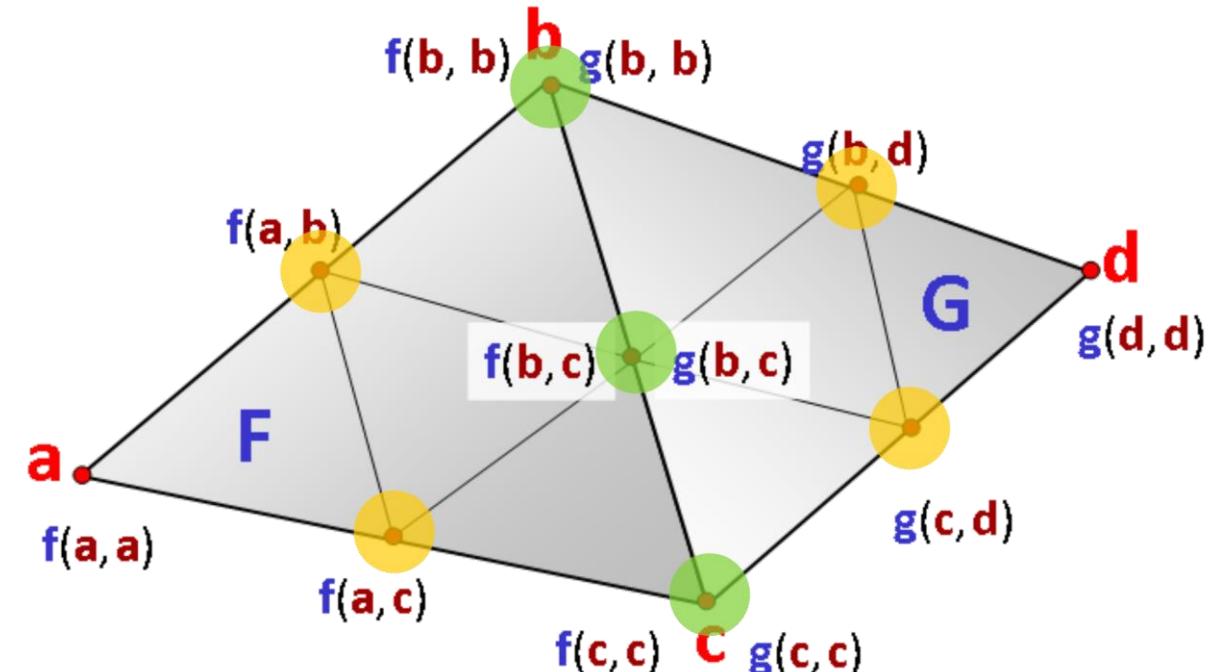
- C^1 -Continuity:

$$\forall x \in \mathbb{R}^3: f(p, x) = g(p, x)$$

- We have to show

$$\forall x \in \mathbb{R}^3: \begin{cases} f(b, x) = g(b, x) \\ f(c, x) = g(c, x) \end{cases}$$

- $\Rightarrow C^1$ continuity follows for all boundary points (by interp.)



Continuity

C^1 continuity: Proof

- So we have to show

$$\forall x \in \mathbb{R}^3: \begin{cases} f(b, x) = g(b, x) \\ f(c, x) = g(c, x) \end{cases}$$

- Proof:

Write $x = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c}$ (coordinate system)

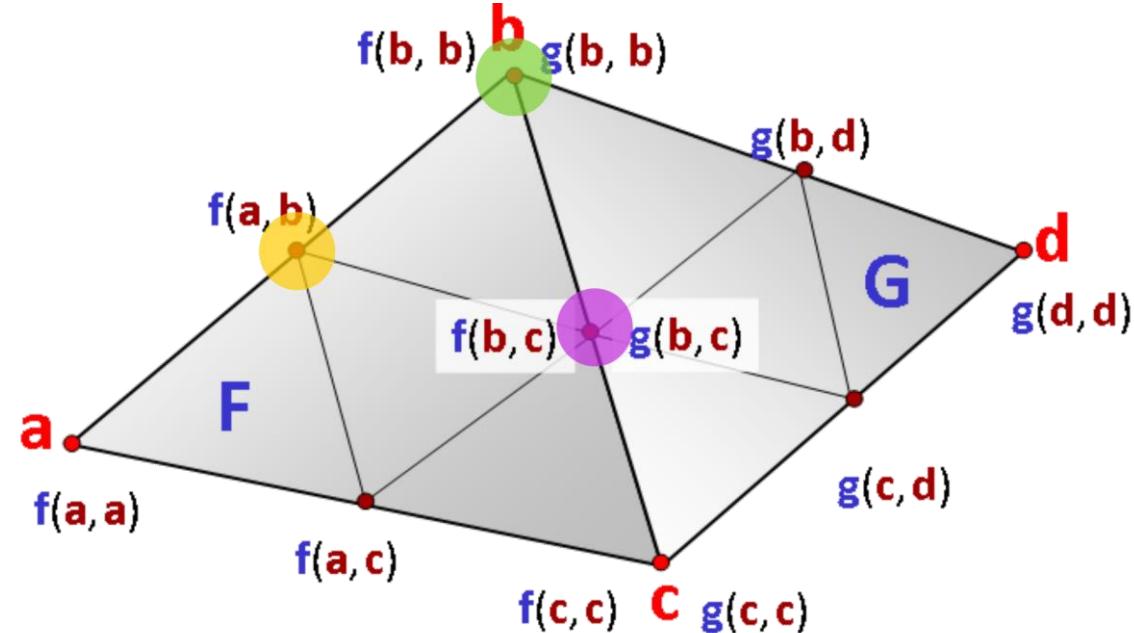
$$f(\mathbf{b}, x) = \alpha f(\mathbf{a}, \mathbf{b}) + \beta f(\mathbf{b}, \mathbf{b}) + \gamma f(\mathbf{b}, \mathbf{c})$$

$$g(\mathbf{b}, x) = \alpha g(\mathbf{a}, \mathbf{b}) + \beta g(\mathbf{b}, \mathbf{b}) + \gamma g(\mathbf{b}, \mathbf{c})$$

$$f(\mathbf{b}, x) = g(\mathbf{b}, x) \Leftrightarrow \alpha f(\mathbf{a}, \mathbf{b}) + \beta f(\mathbf{b}, \mathbf{b}) + \gamma f(\mathbf{b}, \mathbf{c})$$

$$= \alpha g(\mathbf{a}, \mathbf{b}) + \beta g(\mathbf{b}, \mathbf{b}) + \gamma g(\mathbf{b}, \mathbf{c})$$

$$\Leftrightarrow f(\mathbf{a}, \mathbf{b}) = g(\mathbf{a}, \mathbf{b})$$



(same for the other conditions)

Continuity

So what does this mean?

- The blossoms have to agree partially:

$$f(\mathbf{a}, \mathbf{b}) = g(\mathbf{a}, \mathbf{b})$$

$$f(\mathbf{b}, \mathbf{d}) = g(\mathbf{b}, \mathbf{d})$$

$$f(\mathbf{a}, \mathbf{c}) = g(\mathbf{a}, \mathbf{c})$$

$$f(\mathbf{c}, \mathbf{d}) = g(\mathbf{c}, \mathbf{d})$$

- The points must be coplanar

(with edge points):

$$f(\mathbf{a}, \mathbf{b}), g(\mathbf{b}, \mathbf{d}), g(\mathbf{b}, \mathbf{b}), g(\mathbf{b}, \mathbf{c})$$

- The points in \mathbf{F} must be affine images of the points in \mathbf{G}

