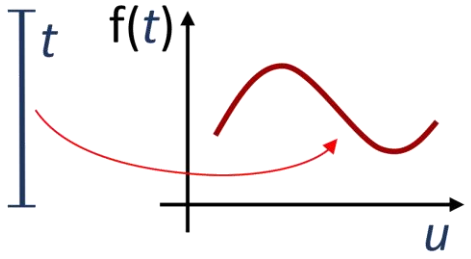
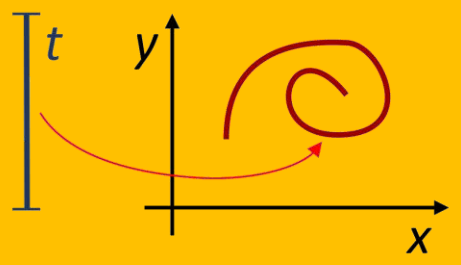
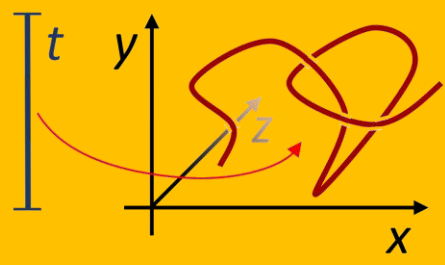
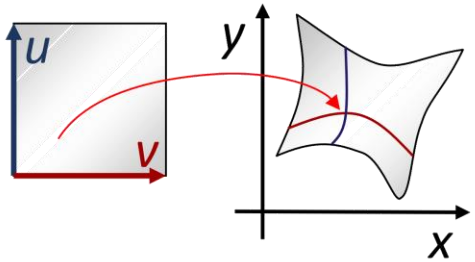
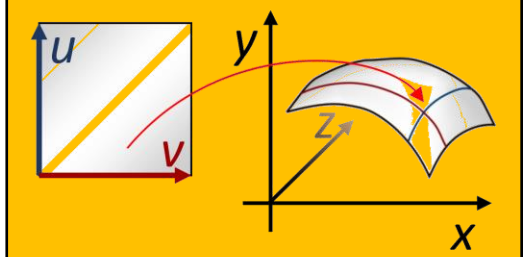
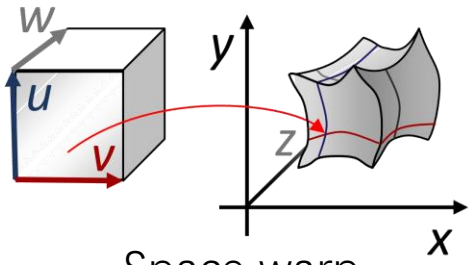
计算机辅助几何设计
2019秋学期

Spline Surfaces

Tensor Product Surfaces · Total Degree Surfaces

陈仁杰

中国科学技术大学

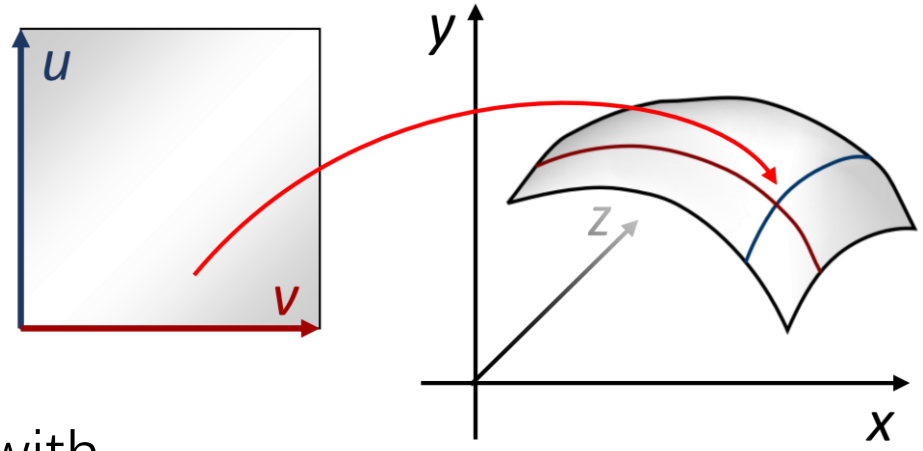
	Output: 1D	Output: 2D	Output: 3D
Input: 1D	 Function graph	 Plane curve	 Space curve
Input: 2D		 Plane warp	 Surface
Input: 3D			 Space warp

Spline Surfaces

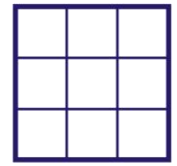
Parametric spline surfaces:

- Two parameter coordinates (u, v)
- Piecewise bivariate polynomials

  (rational surfaces → homogeneous coords)
- Assemble multiple pieces to form a surface with continuity
- Each piece is called *spline patch*

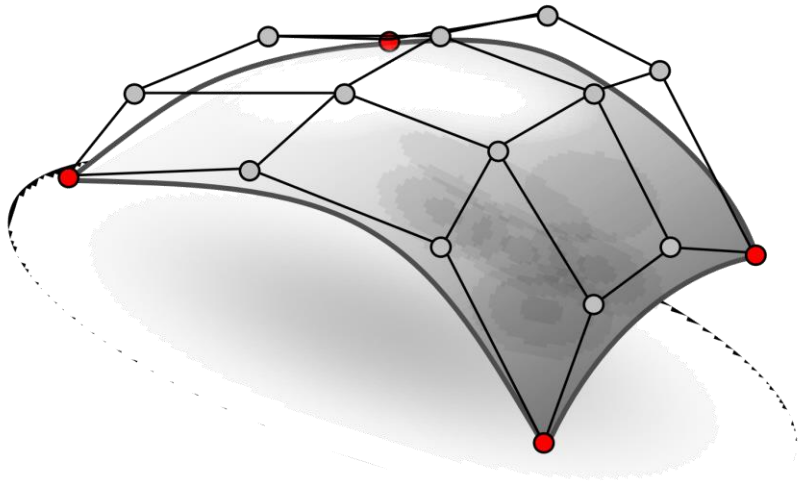


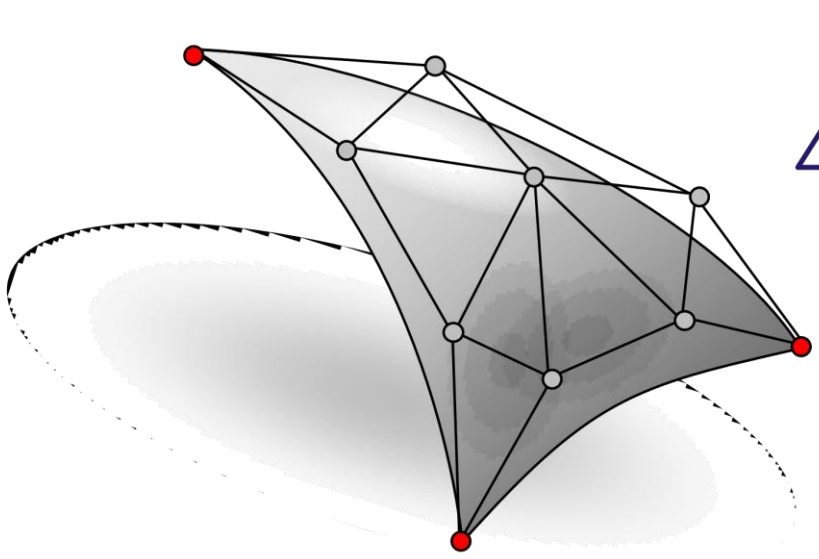
Spline Surfaces



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 $t \in [t_0, t_1] \to \mathbb{R}^d$:

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

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

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

Tensor Product Surfaces

Tensor product basis

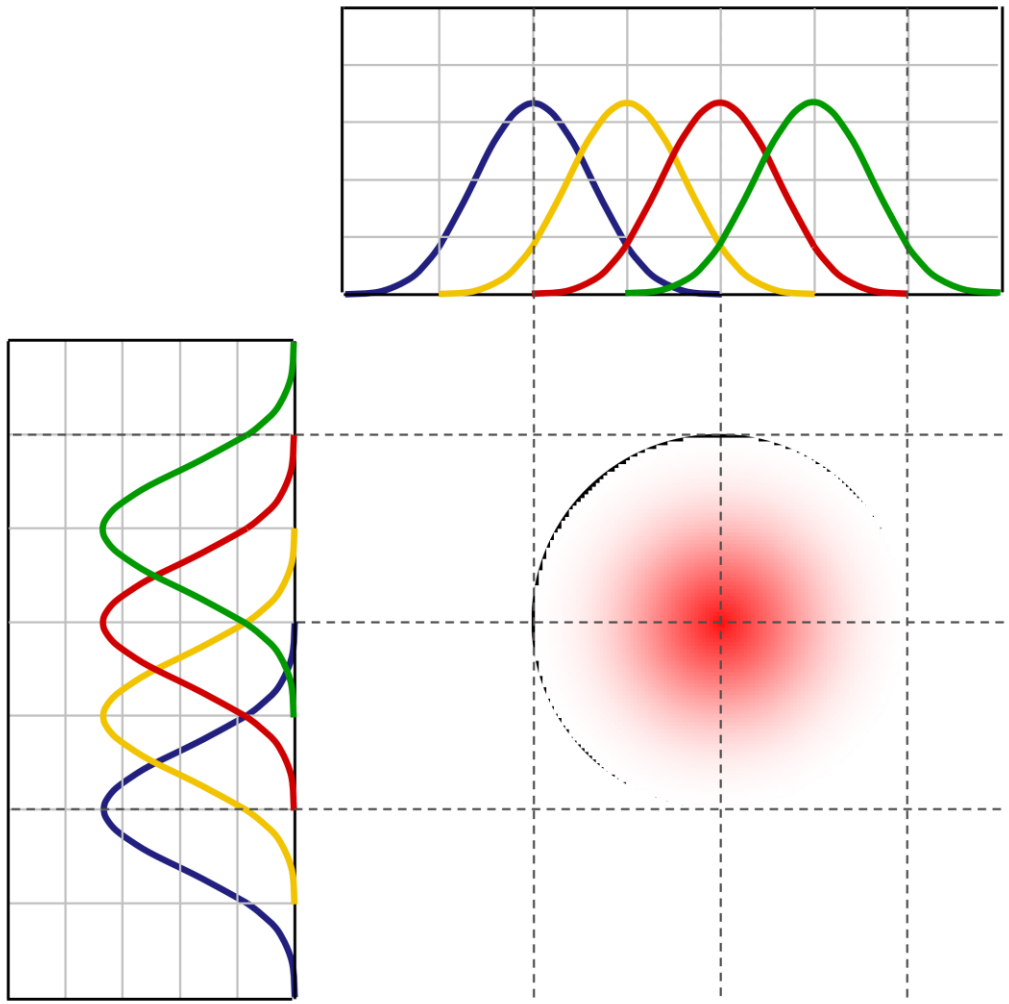
	$b_1(u)$	$b_2(u)$	$b_3(u)$	$b_4(u)$
$b_1(v)$	$b_1(v)b_1(u)$	$b_1(v)b_2(u)$	$b_1(v)b_3(u)$	$b_1(v)b_4(u)$
$b_2(v)$	$b_2(v)b_1(u)$	$b_2(v)b_2(u)$	$b_2(v)b_3(u)$	$b_2(v)b_4(u)$
$b_3(v)$	$b_3(v)b_1(u)$	$b_3(v)b_2(u)$	$b_3(v)b_3(u)$	$b_3(v)b_4(u)$
$b_4(v)$	$b_4(v)b_1(u)$	$b_4(v)b_2(u)$	$b_4(v)b_3(u)$	$b_4(v)b_4(u)$

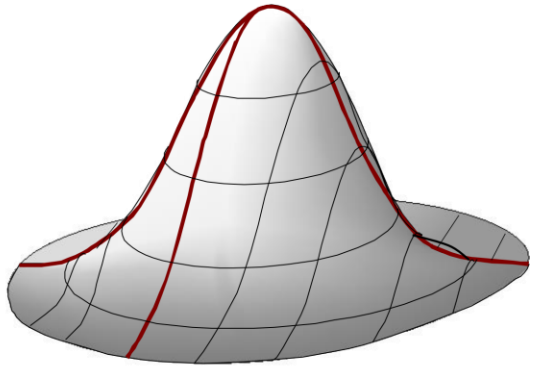
Monomial Example

Tensor product basis of cubic monomials

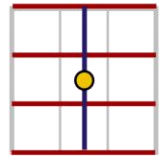
	1	u	u^2	u^3
1	1	u	u^2	u^3
v	v	vu	vu^2	vu^3
v^2	v^2	v^2u	v^2u^2	v^2u^3
v^3	v^3	v^3u	v^3u^2	v^3u^3

Example





Tensor Product Surfaces

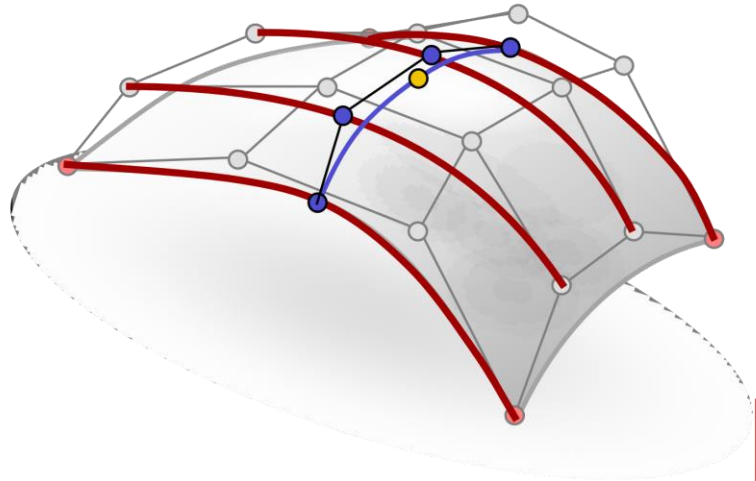


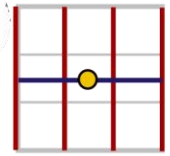
Tensor Product Surfaces

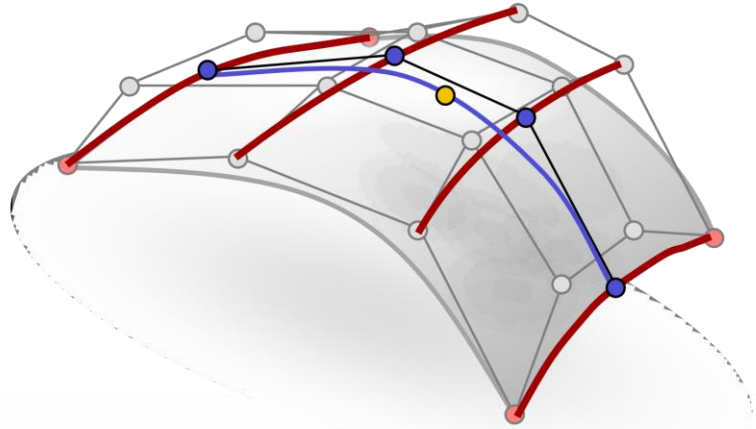
$$\boldsymbol{f}(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_i(u)b_j(v)\boldsymbol{p}_{i,j}$$

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

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







- “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 $\boldsymbol{B}^{(curv)} \coloneqq \{b_1(t), \dots, b_n(t)\}$ forms a partition of unity and it is nonnegative (≥ 0) on $t \in [t_0, t_1]$
  - Obviously, we then have:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \underbrace{b_i(u)}_{\geq 0} \underbrace{b_j(v)}_{\geq 0} \geq 0$$

  - 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:

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

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

- Just spline-curve combinations of curve derivatives

Partial Derivatives

Computing partial derivatives:

- Second derivatives:

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

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

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

Partial Derivatives

Computing partial derivatives:

- General derivatives:

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

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

Normal Vectors

We can compute normal vectors from partial derivatives:

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

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

Tensor Product Surfaces

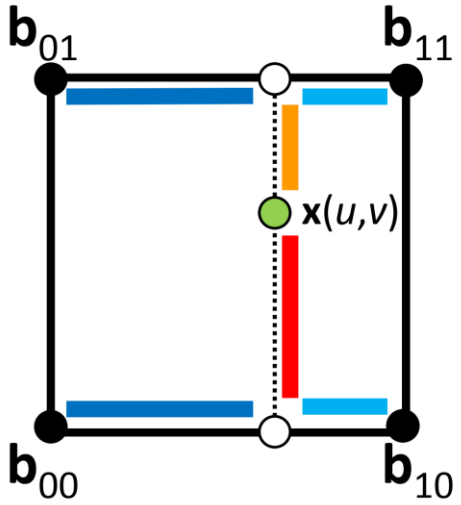
Tensor Product Bezier Surfaces

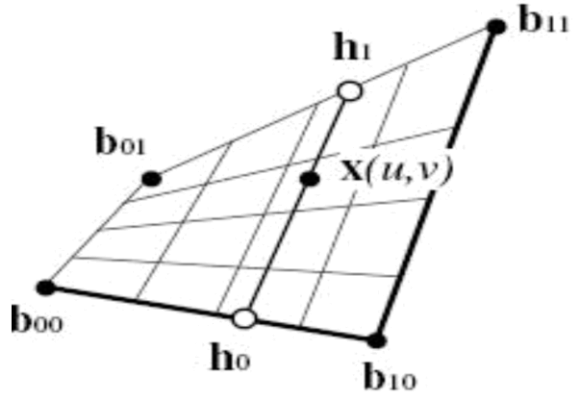
Tensor Product Bezier Spline Surfaces

Tensor Product Bezier Surfaces

Bezier curves:
repeated linear interpolation

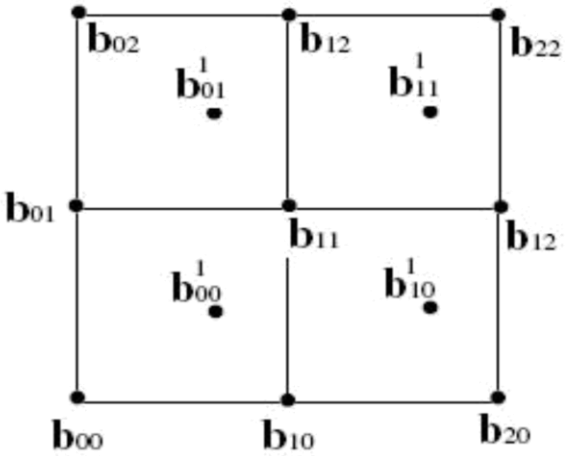
now a different setup:
4 points $\boldsymbol{b}_{00}, \boldsymbol{b}_{10}, \boldsymbol{b}_{11}, \boldsymbol{b}_{01}$
Parameter area $[0,1] \times [0,1]$





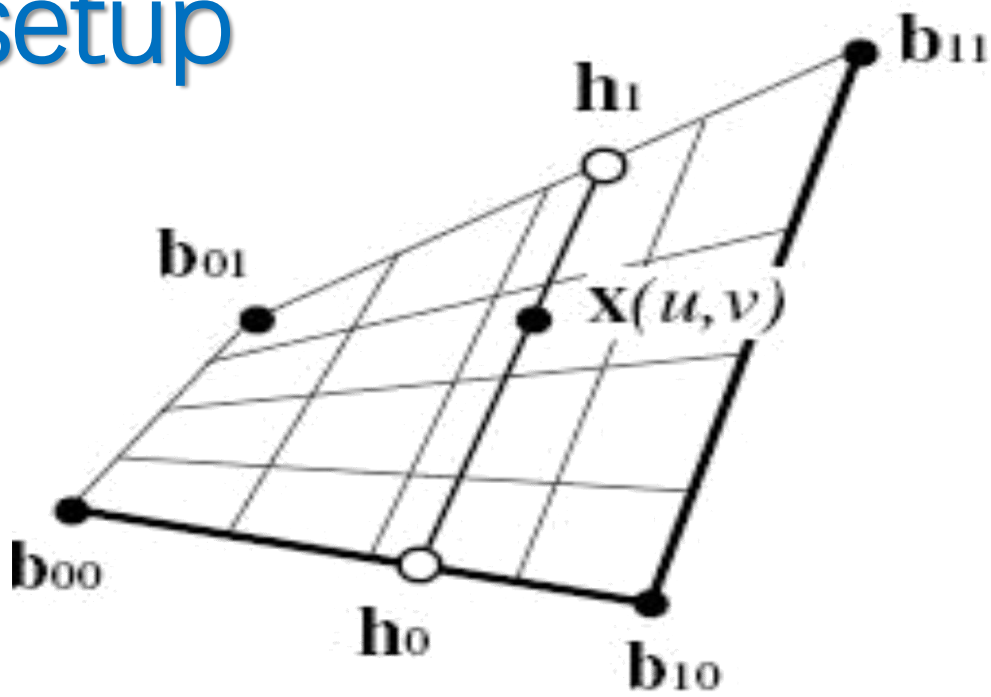
bilinear interpolation:
repeated linear interpolation

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



Some formulas for this setup

$$\boldsymbol{h}_0 = (1-u)\boldsymbol{b}_{00} + u\boldsymbol{b}_{10}$$
$$\boldsymbol{h}_1 = (1-u)\boldsymbol{b}_{01} + u\boldsymbol{b}_{11}$$



$$\boldsymbol{x}(u,v) = (1-v)\boldsymbol{h}_0 + v\boldsymbol{h}_1$$
$$= (1-v)[(1-u)\boldsymbol{b}_{00} + u\boldsymbol{b}_{10}] + v[(1-u)\boldsymbol{b}_{01} + u\boldsymbol{b}_{11}]$$

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

Some formulas for this setup

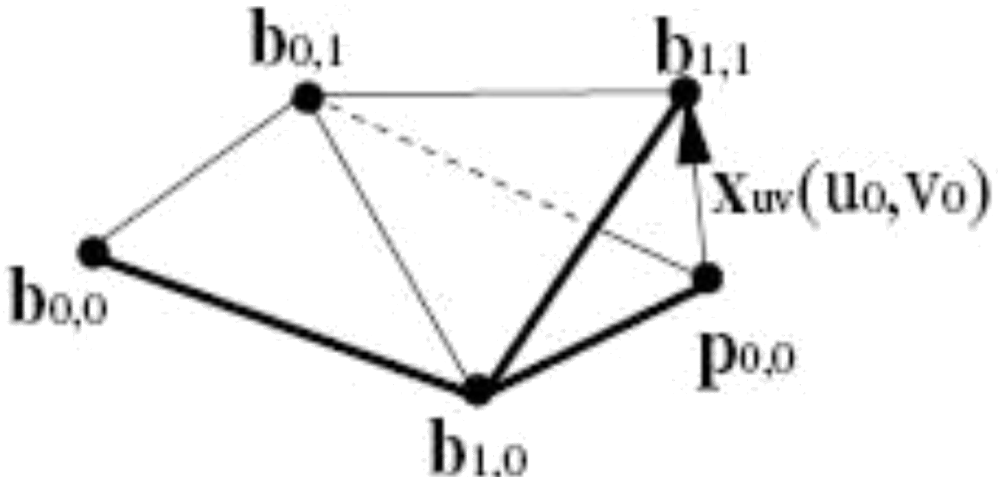
Derivatives of bilinear surfaces

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

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

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

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



Some formulas for this setup

Biquadratic surfaces

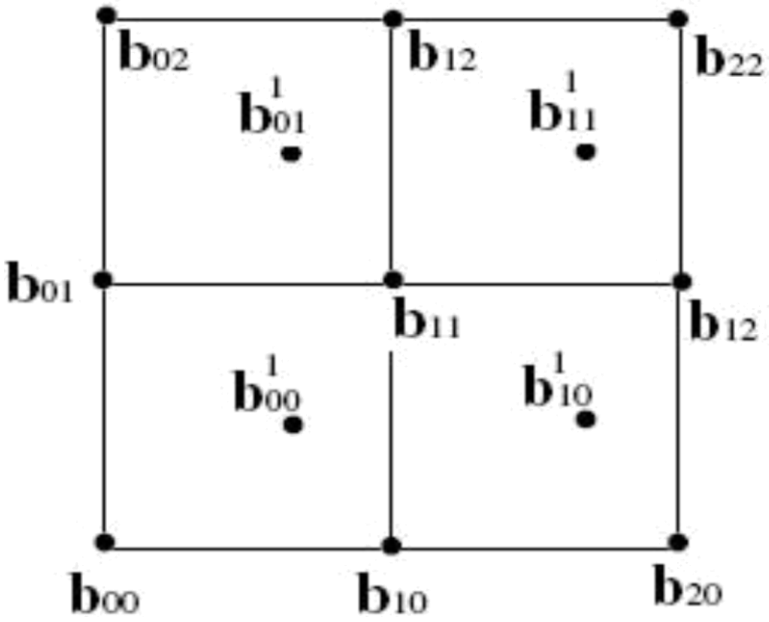
$$\boldsymbol{b}_{00}^{1} = (1-u)(1-v)\boldsymbol{b}_{00} + u(1-v)\boldsymbol{b}_{10} + (1-u)v\boldsymbol{b}_{01} + uv\boldsymbol{b}_{11}$$

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

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

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

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



Bezier Patches

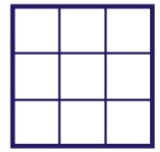
Bezier Patches:

- Use tensor product Bernstein basis

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

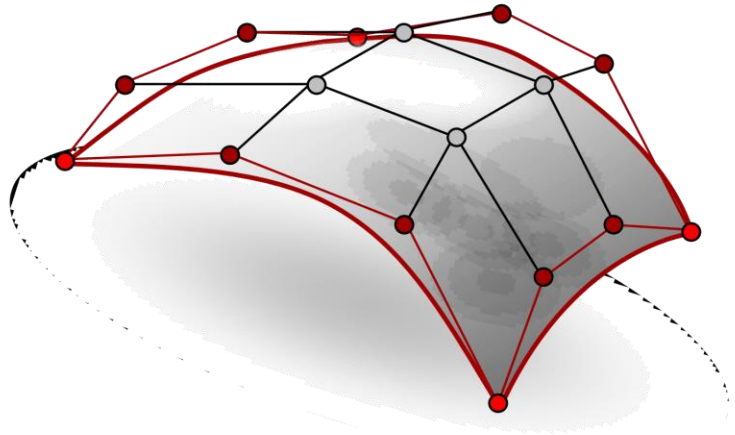
- We get automatically:
  - Affine invariance
  - Convex hull property

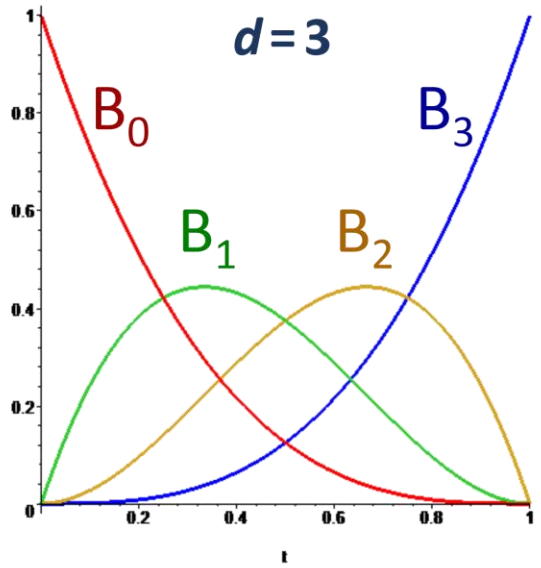
Bezier Patches

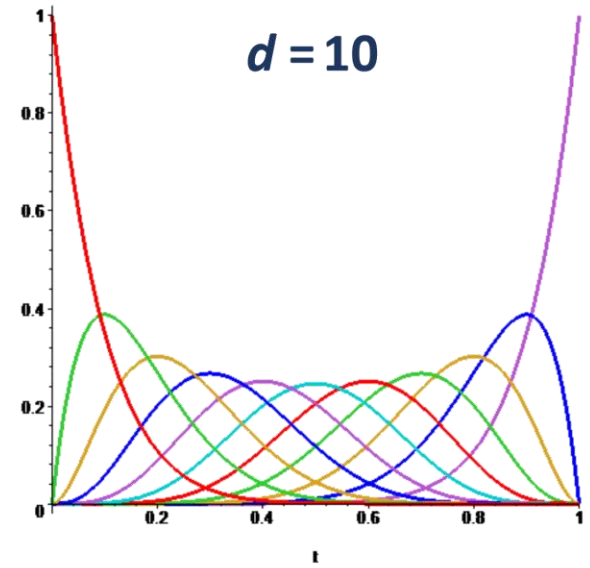


Bezier Patches:

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







Bezier Patches

Bezier Patches:

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

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

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

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

Continuity Conditions

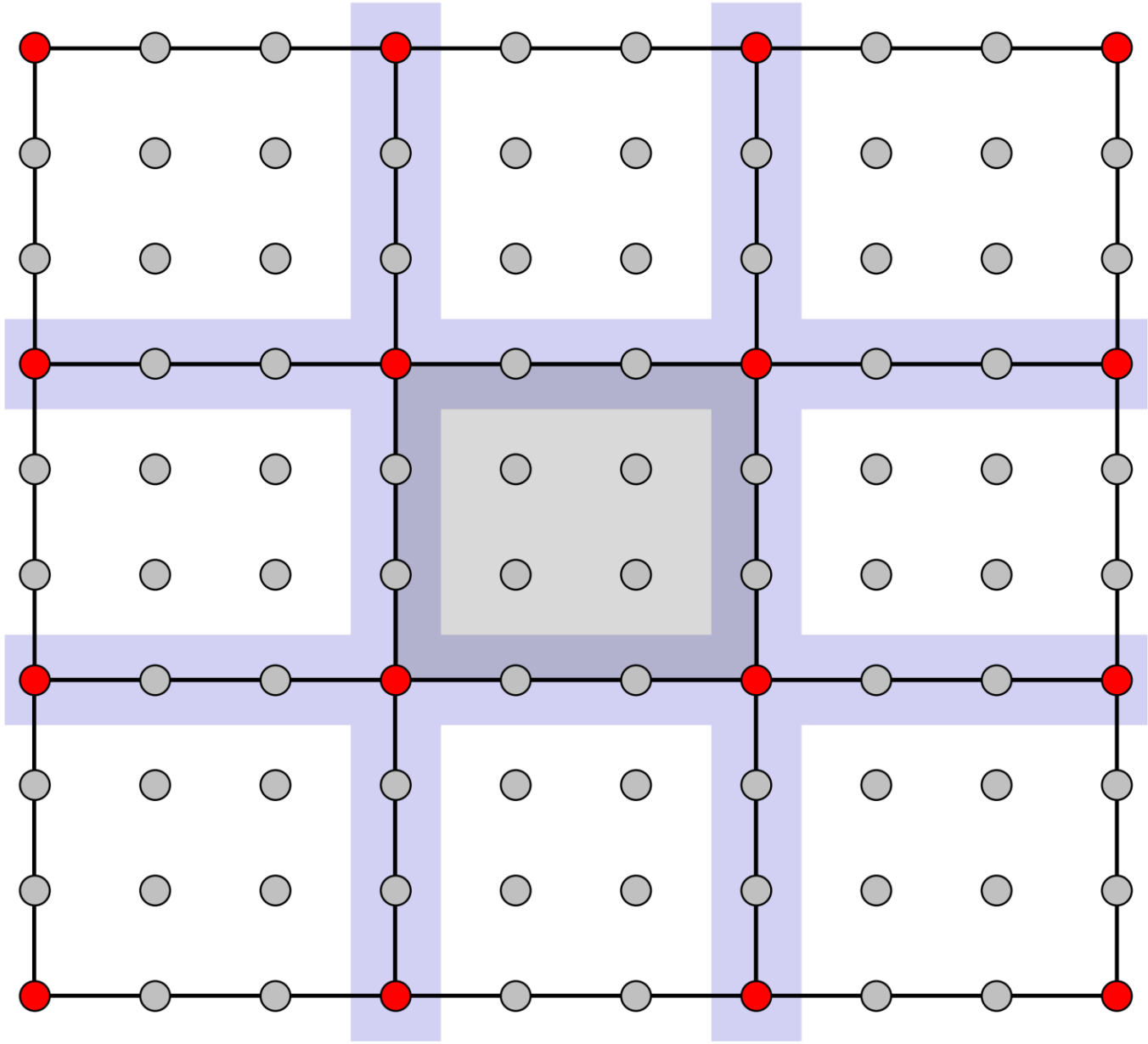
For C^0 continuity:

- Boundary control points must match

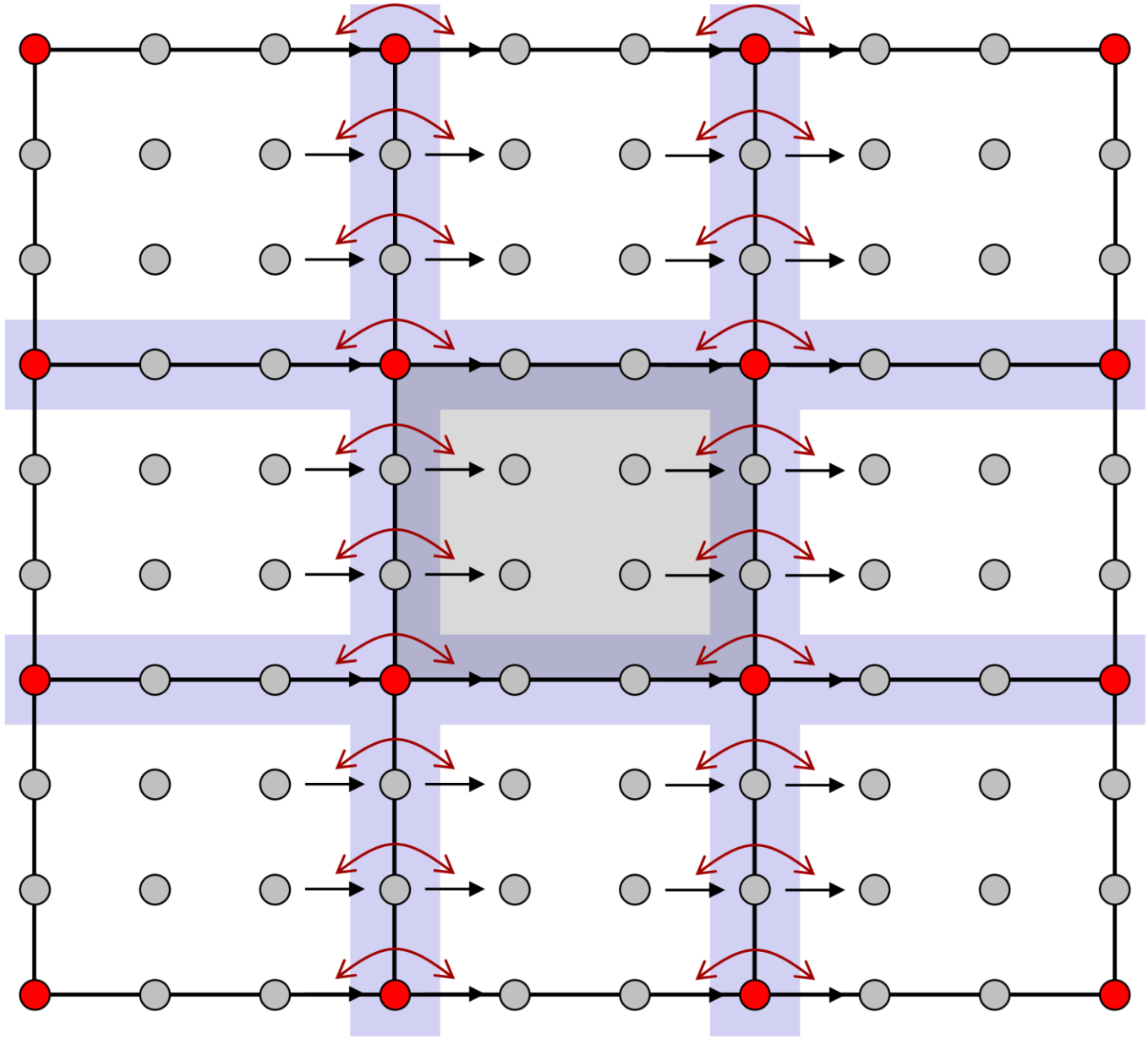
For C^1 continuity:

- Difference vectors must match at the boundary

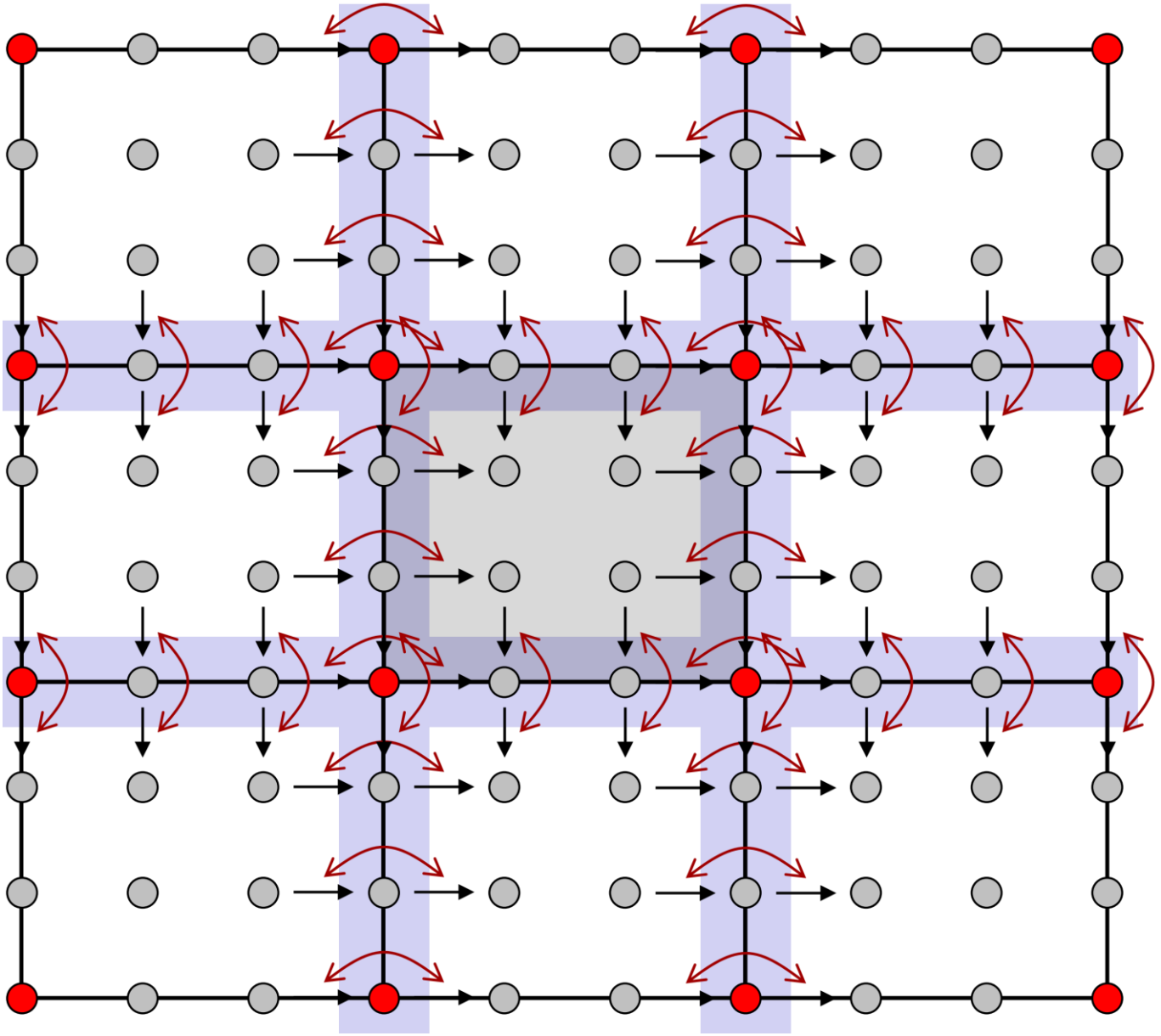
C^0 Continuity



C^0 Continuity



C^0 Continuity



Polars & Blossoms

Blossoms for tensor product surfaces:

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

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

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

- Required properties:
  - **Diagonality**: $\boldsymbol{f}(u, \dots, u; v, \dots, v) = F(u,v)$
  - **Symmetry**: $\boldsymbol{f}(u_1, \dots, u_d; v_1, \dots, v_d) = f\left(u_{\pi(1)}, \dots, u_{\pi(d)}; v_{\mu(1)}, \dots, v_{\mu(d)}\right)$
    for all permutations of indices π, μ
  - **Multi-affine:** $\sum \alpha_k = 1$

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

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

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

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

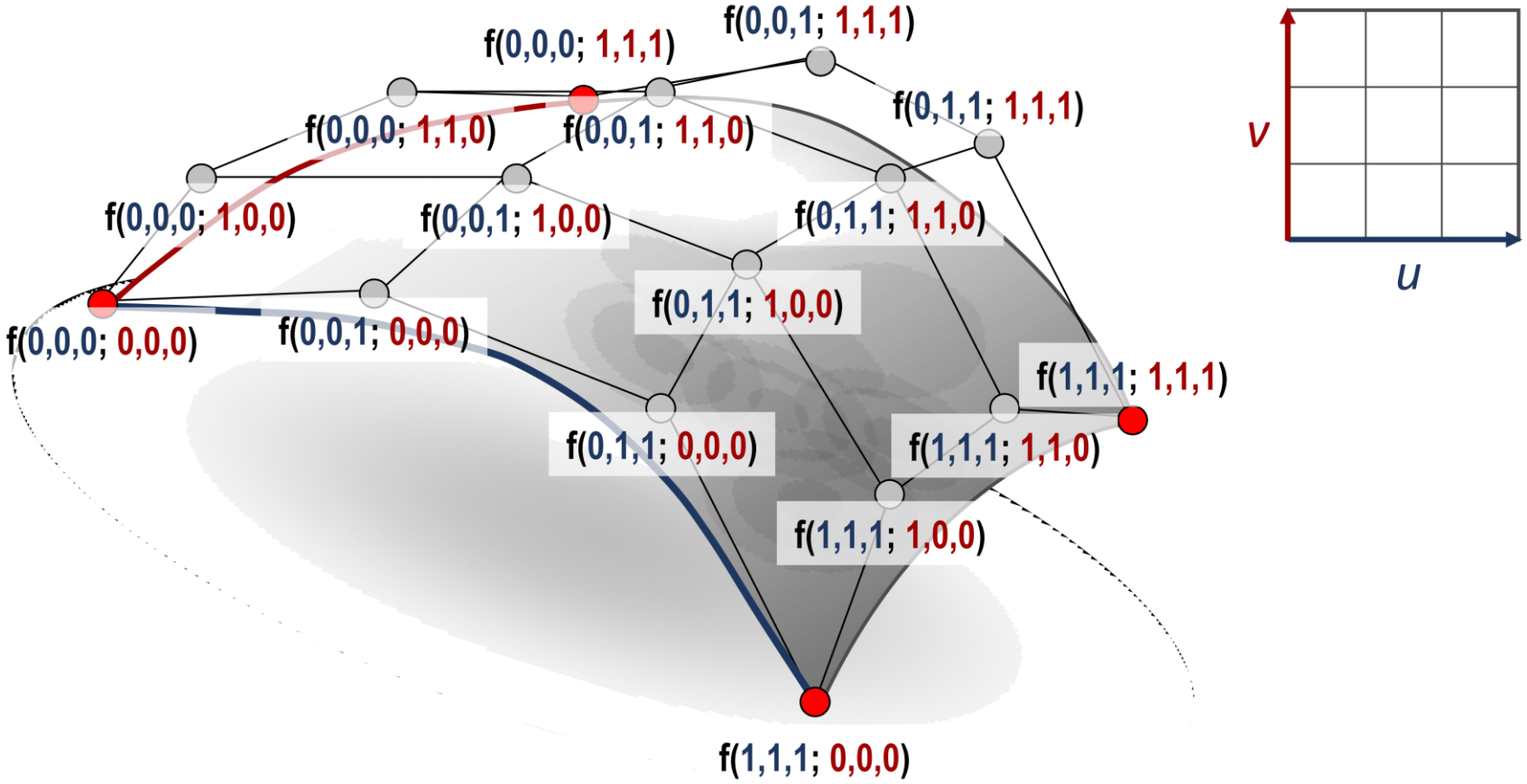
Short Summary

Polar forms for tensor product surfaces:

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

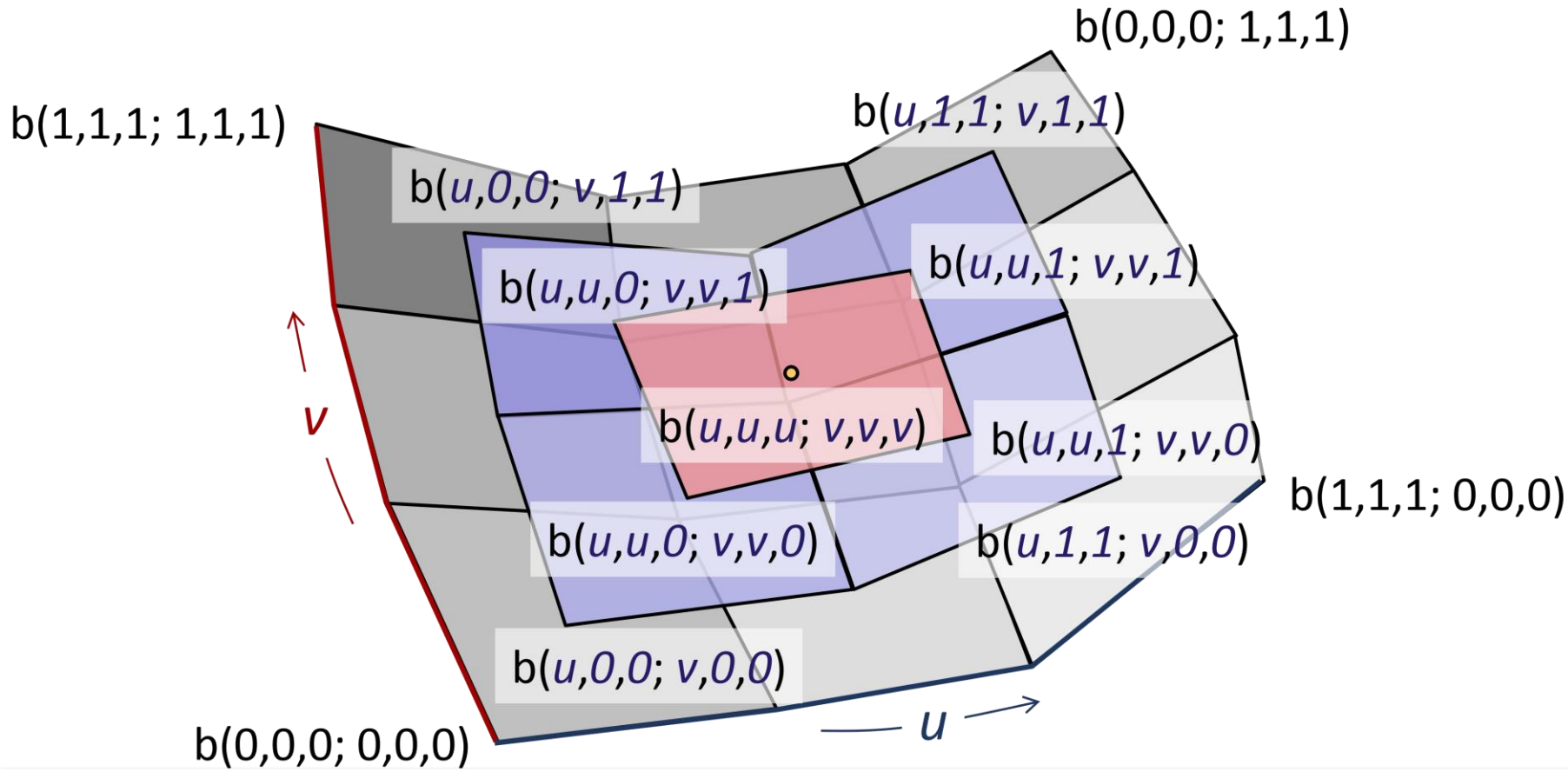
Bezier Control Points

Bezier 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 Bezier patches
  (we get Bezier 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

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

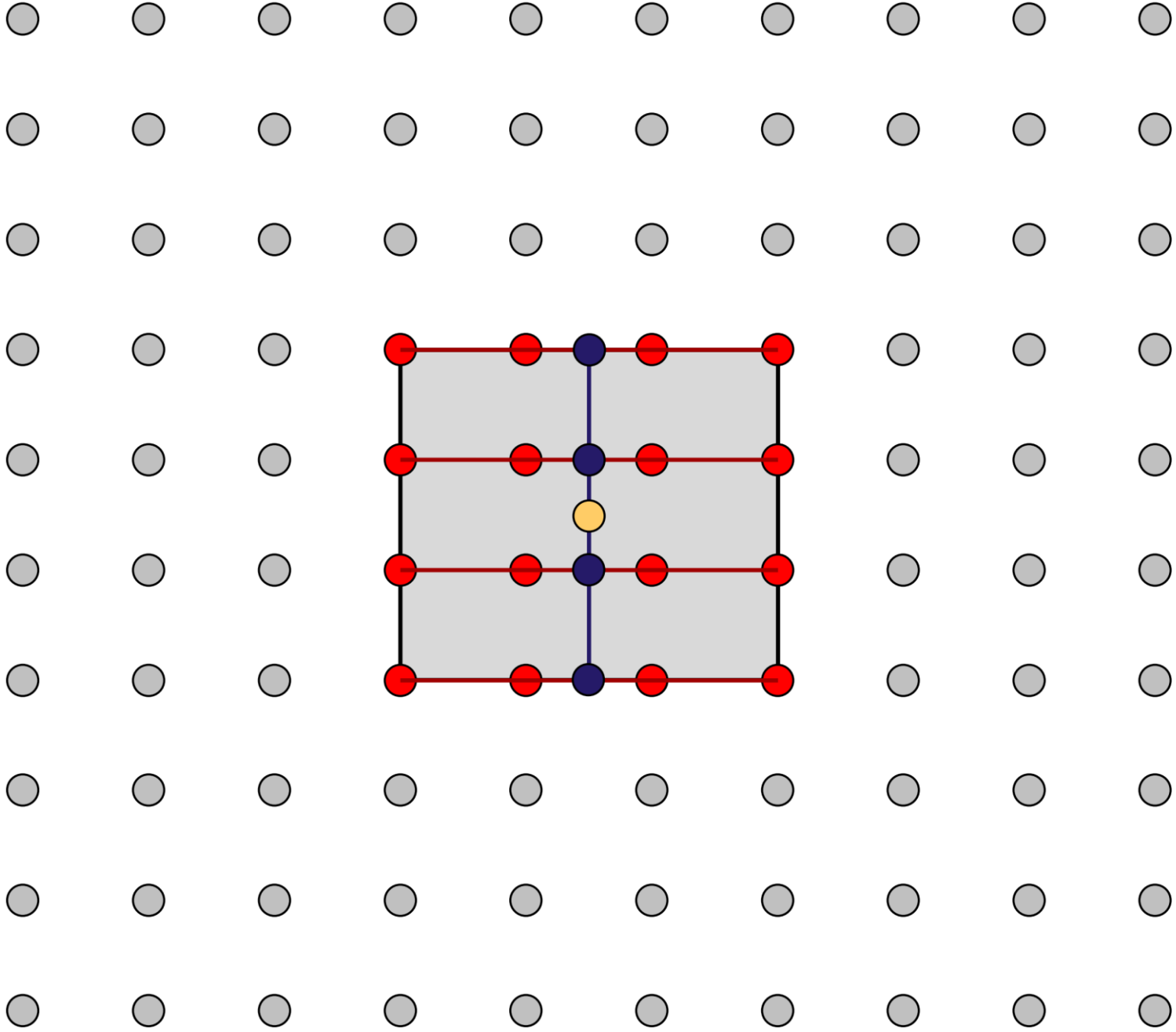
B-Spline Patches

Then, obtain a parametric B-spline patch as:

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

- 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 Bezier/B-splines to create the patches ("rational Bezier 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 Bezier Patch:

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

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

Rational Patch

Rational B-Spline Patch:

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

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

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.

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

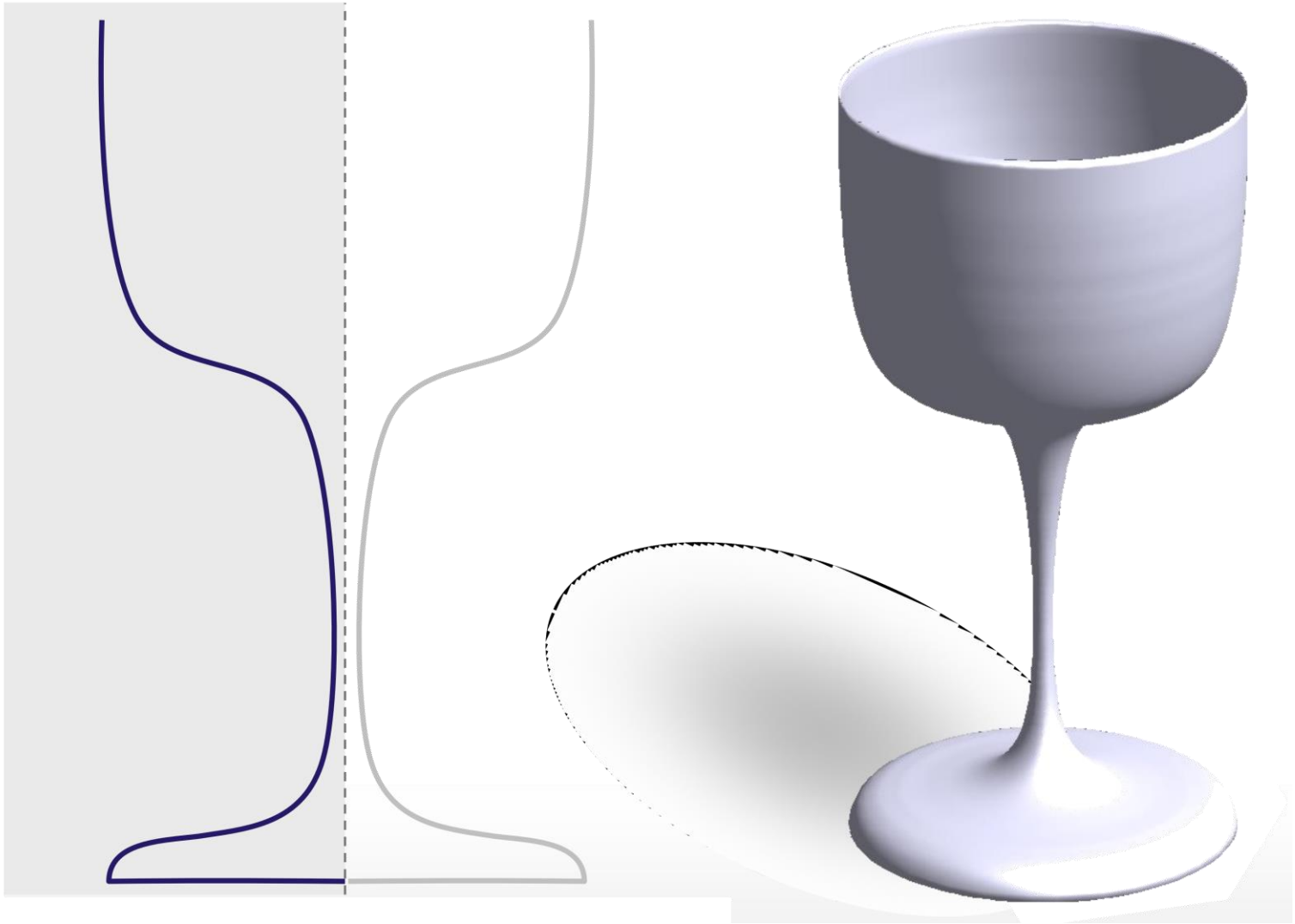
$$\boldsymbol{f}^{(Eucl)}(u,v) = \frac{\sum_{i=0}^{d}\sum_{j=0}^{d} N_i^{(d)}(u)N_j^{(d)}(v)\omega_{i,j}\boldsymbol{p}_{i,j}}{\sum_{i=0}^{d}\sum_{j=0}^{d} N_i^{(d)}(u)N_j^{(d)}(v)\omega_{i,j}}$$

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:

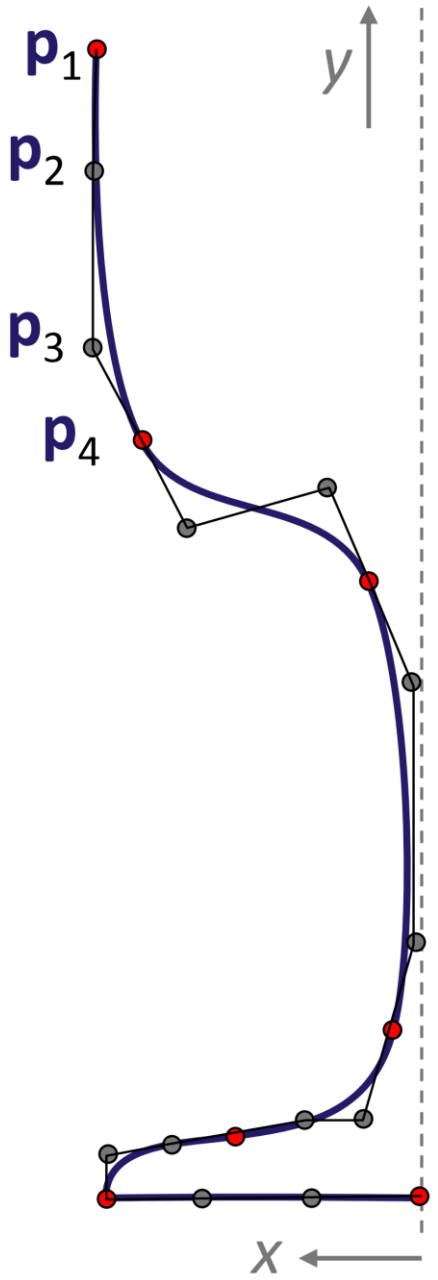
- Control points $p_1, \ldots, p_n$ of curve ("generatrix")

We want to compute:

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

Such that:

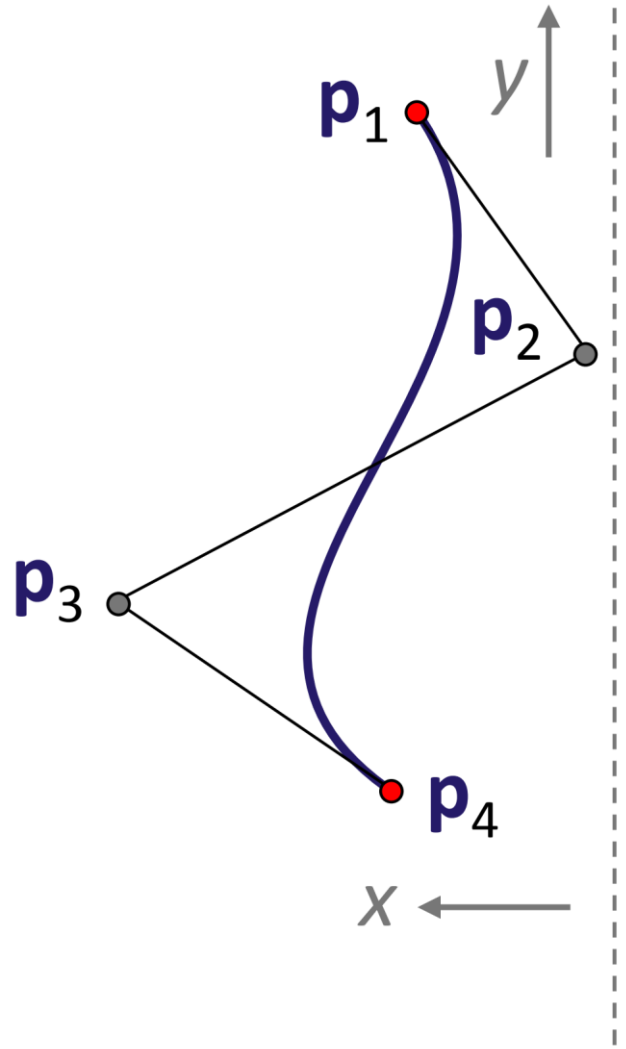
- The surface describes the surface of revolution that we obtain by rotating the curve around the y axis (w.l.o.g.)



Surfaces of Revolution

Simplification:

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



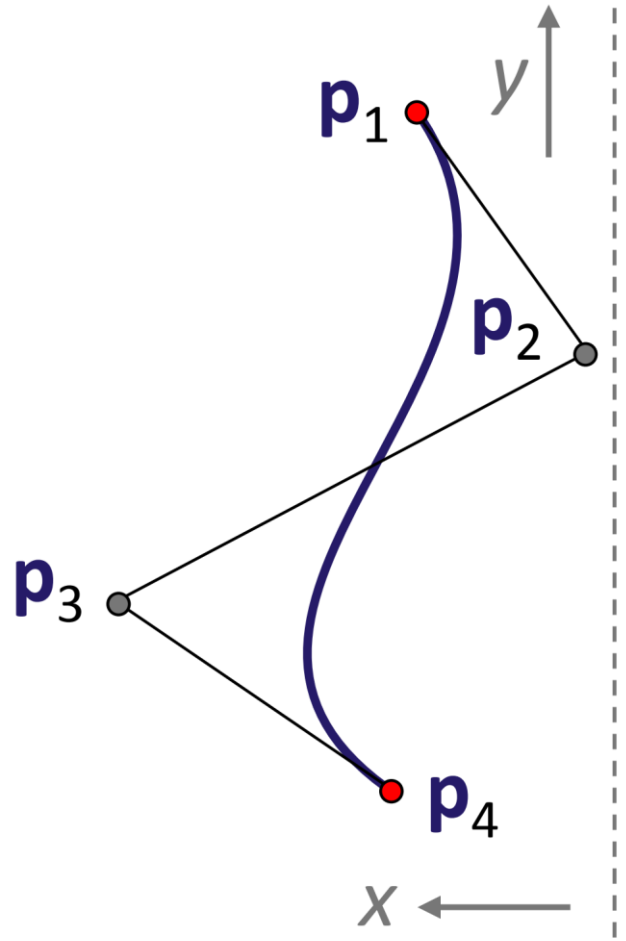
Surfaces of Revolution

Construction:

- We are given control points

  $\boldsymbol{p}_1, \cdots, \boldsymbol{p}_{d+1}$

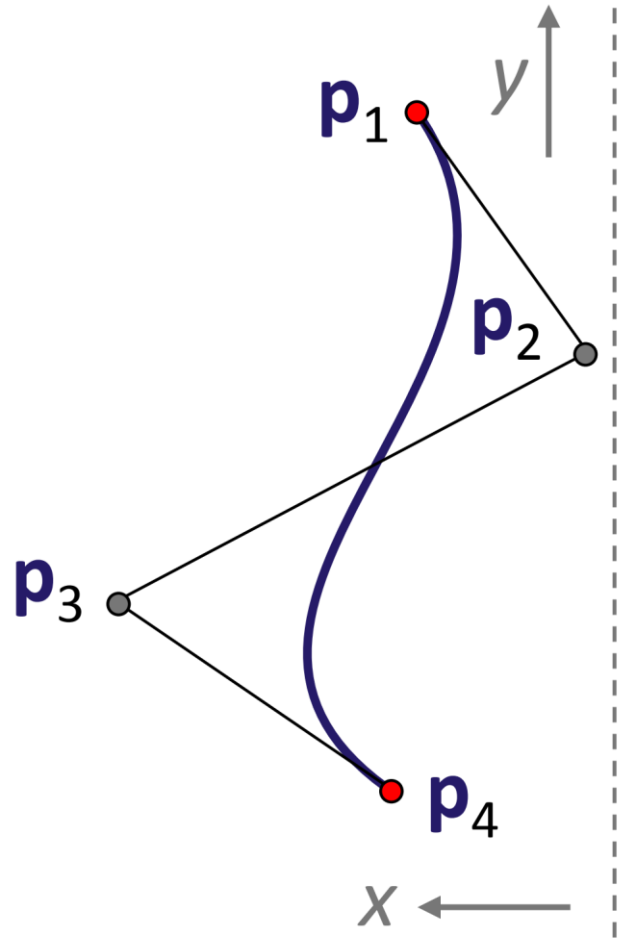
  (d is the degree in u direction)
- We introduce a new parameter v
- In v direction, we use quadratic Bezier curves (2nd degree basis in v-direction)



Surfaces of Revolution

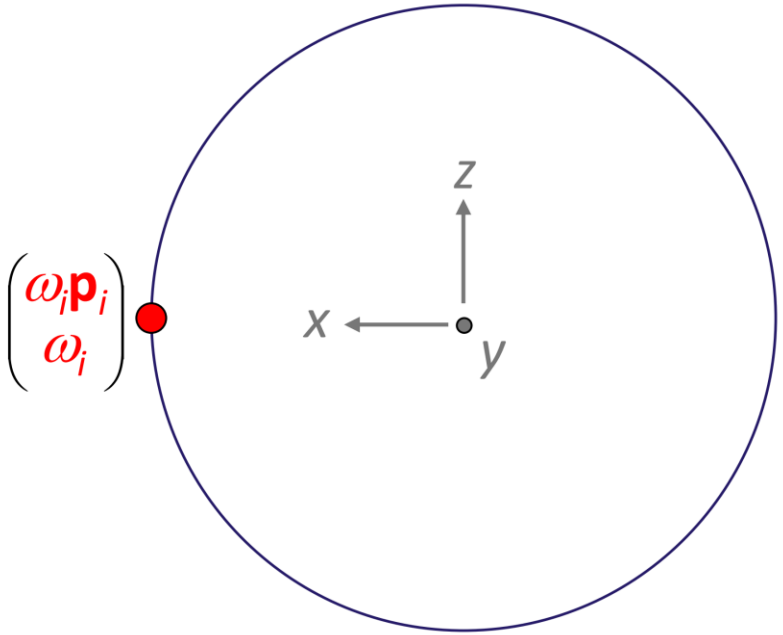
Key Idea:

- 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



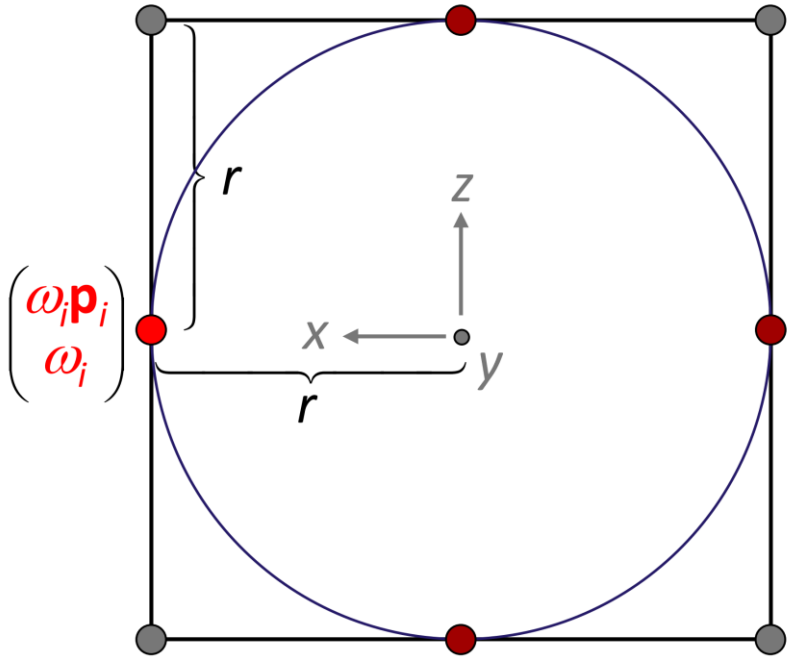
Surface of Revolution

Making one point rotate around the y-axis:



Surface of Revolution

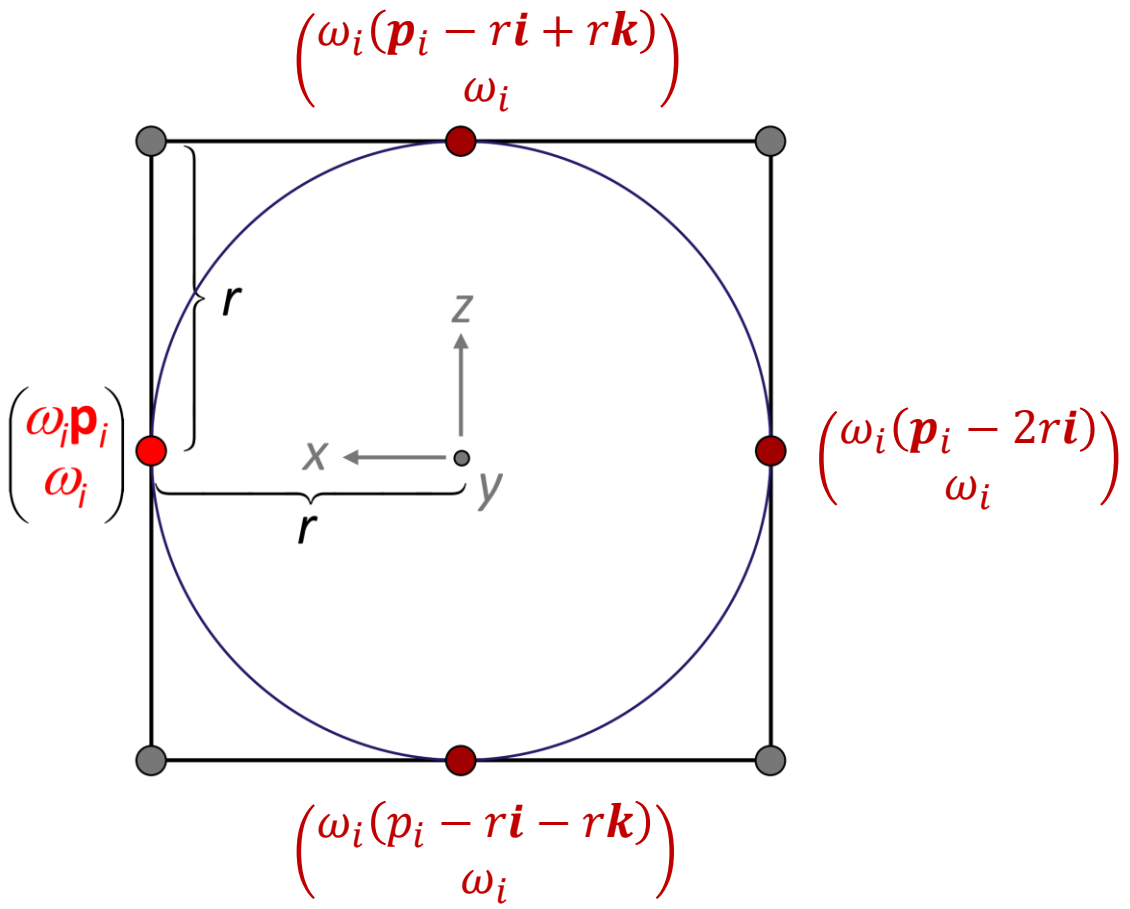
Making one point rotate around the y-axis:



Surface of Revolution

Making one point rotate around the y-axis:

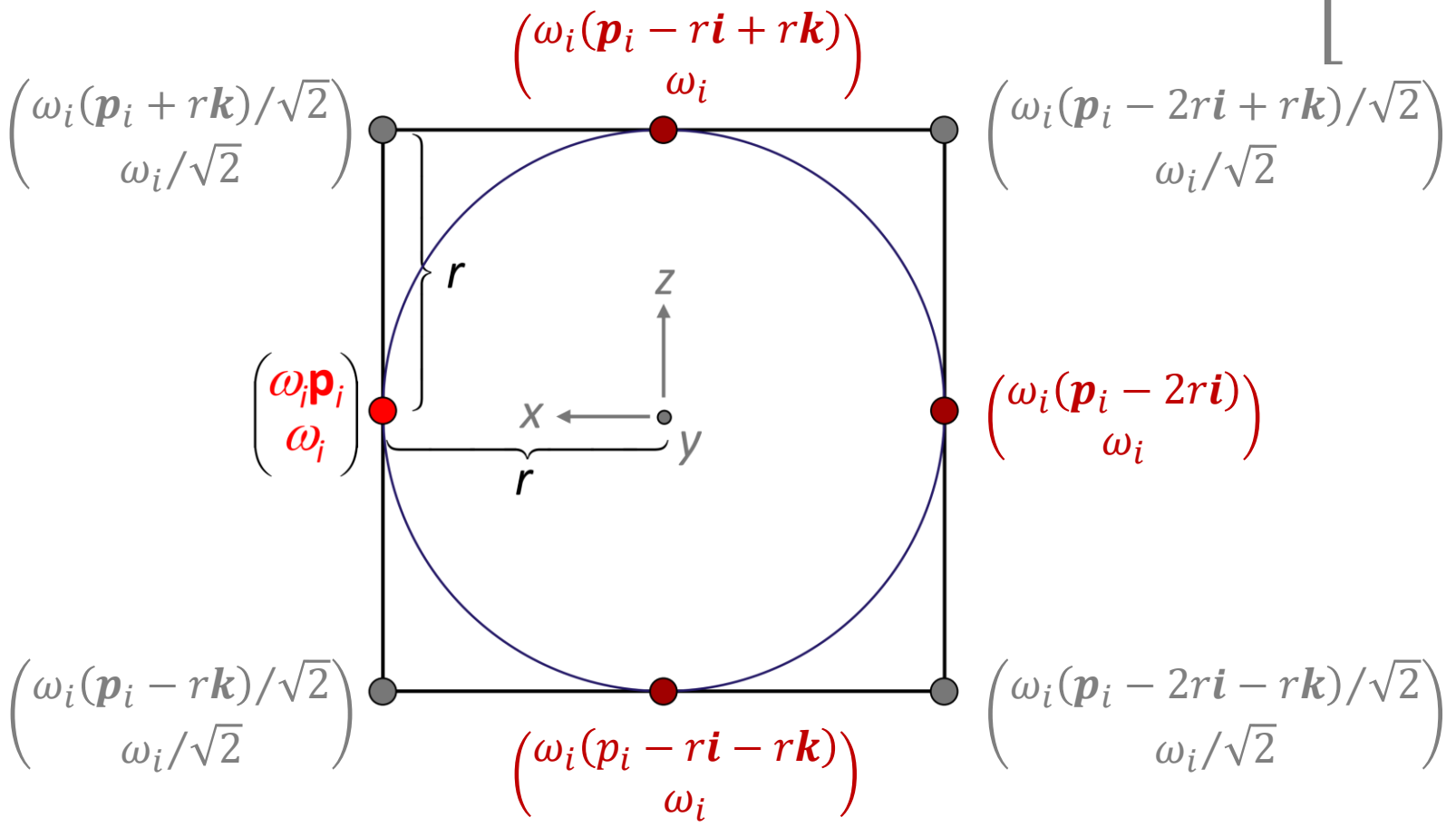
$$\left[\boldsymbol{i} := \begin{pmatrix}1\\0\\0\end{pmatrix}, \boldsymbol{k} := \begin{pmatrix}0\\0\\1\end{pmatrix}\right]$$



Surface of Revolution

Making one point rotate around the y-axis:

$$\left[\boldsymbol{i} \coloneqq \begin{pmatrix}1\\0\\0\end{pmatrix}, \boldsymbol{k} \coloneqq \begin{pmatrix}0\\0\\1\end{pmatrix}\right]$$



Remark

What we get:

- We obtain 4 segments, i.e. 4 patches for each Bezier 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

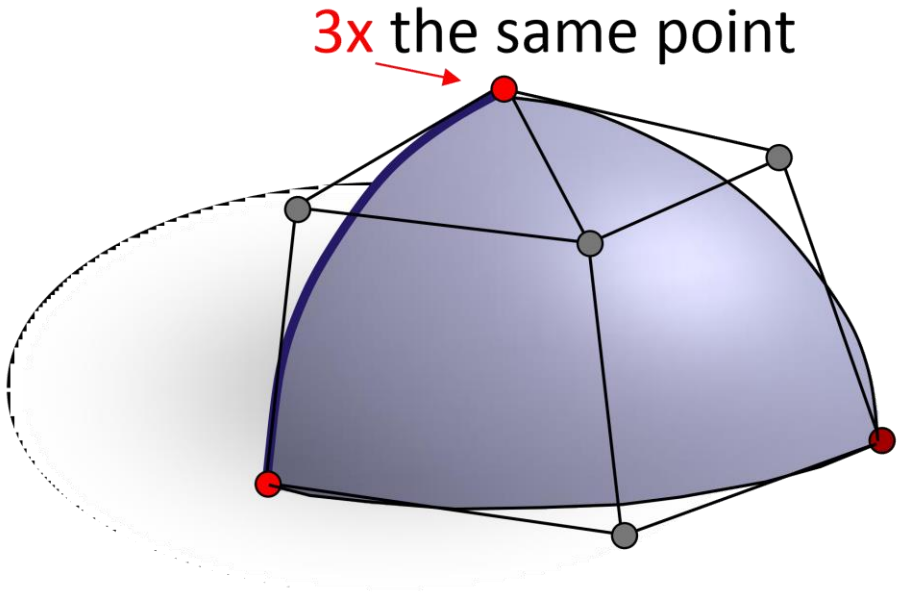
Benefit

With this construction, it is straightforward to create:

- Spheres
- Tori
- Cylinders
- Cones

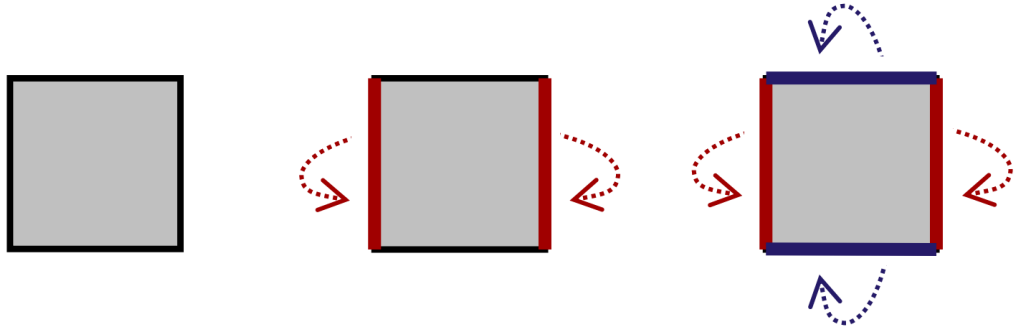
And affine transformations of these (e.g. ellipsoids)

Parametrization Restrictions



Remaining problem:

- The sphere and the cone are not regularly parametrizied (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

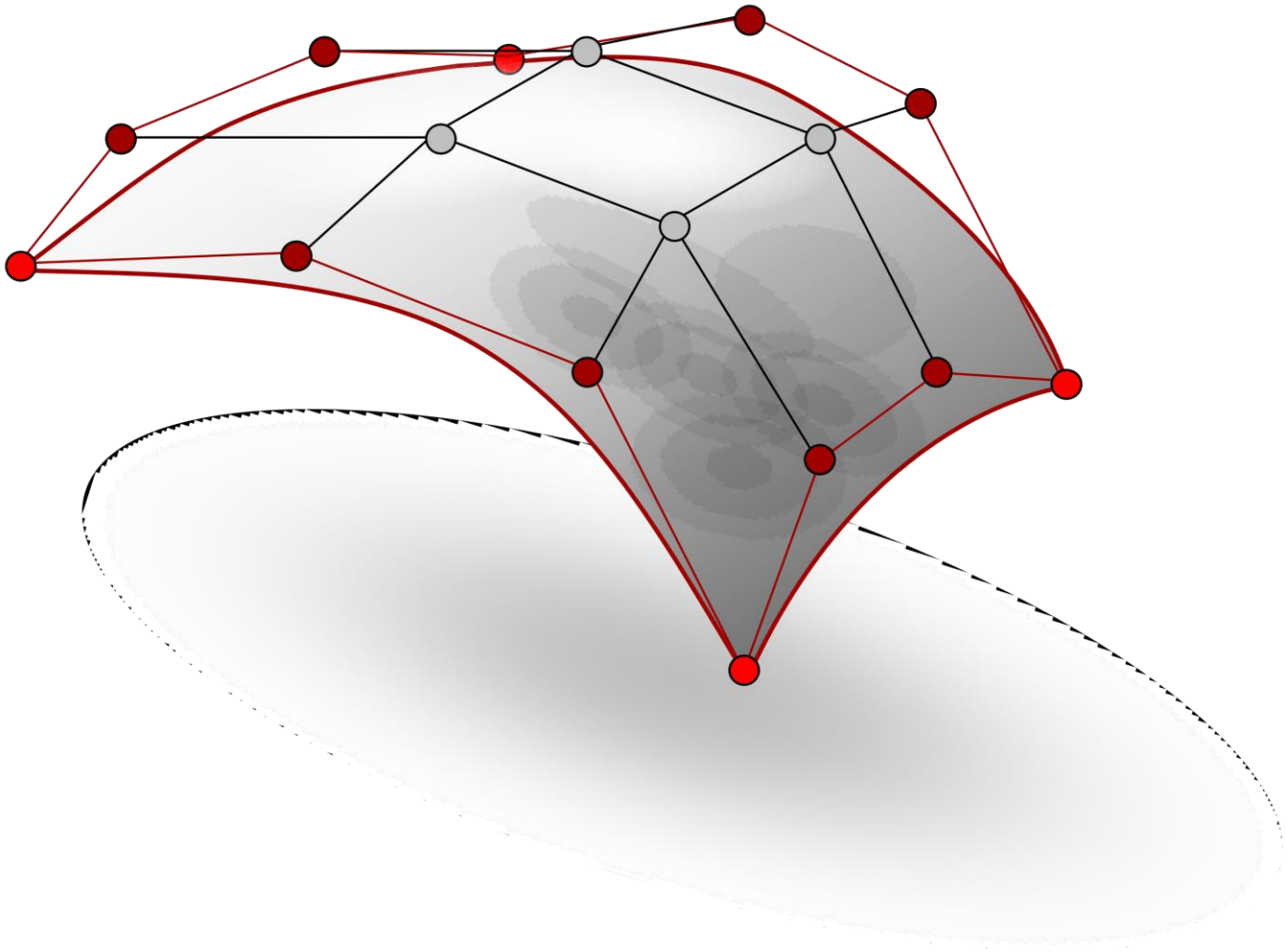
Consequence:

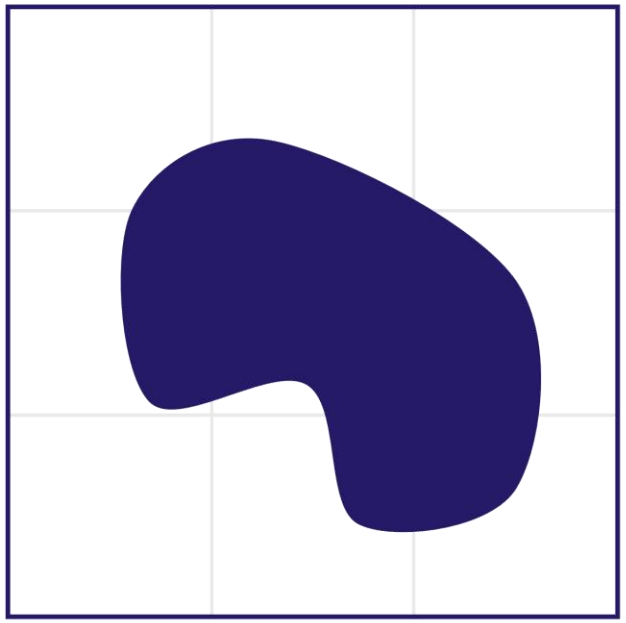
- We need more control over the parameter domain
- One solution is *trimming* using *curves on surfaces (CONS)*
- Standard tool in CAD: *trimmed NURBS*

Basic idea:

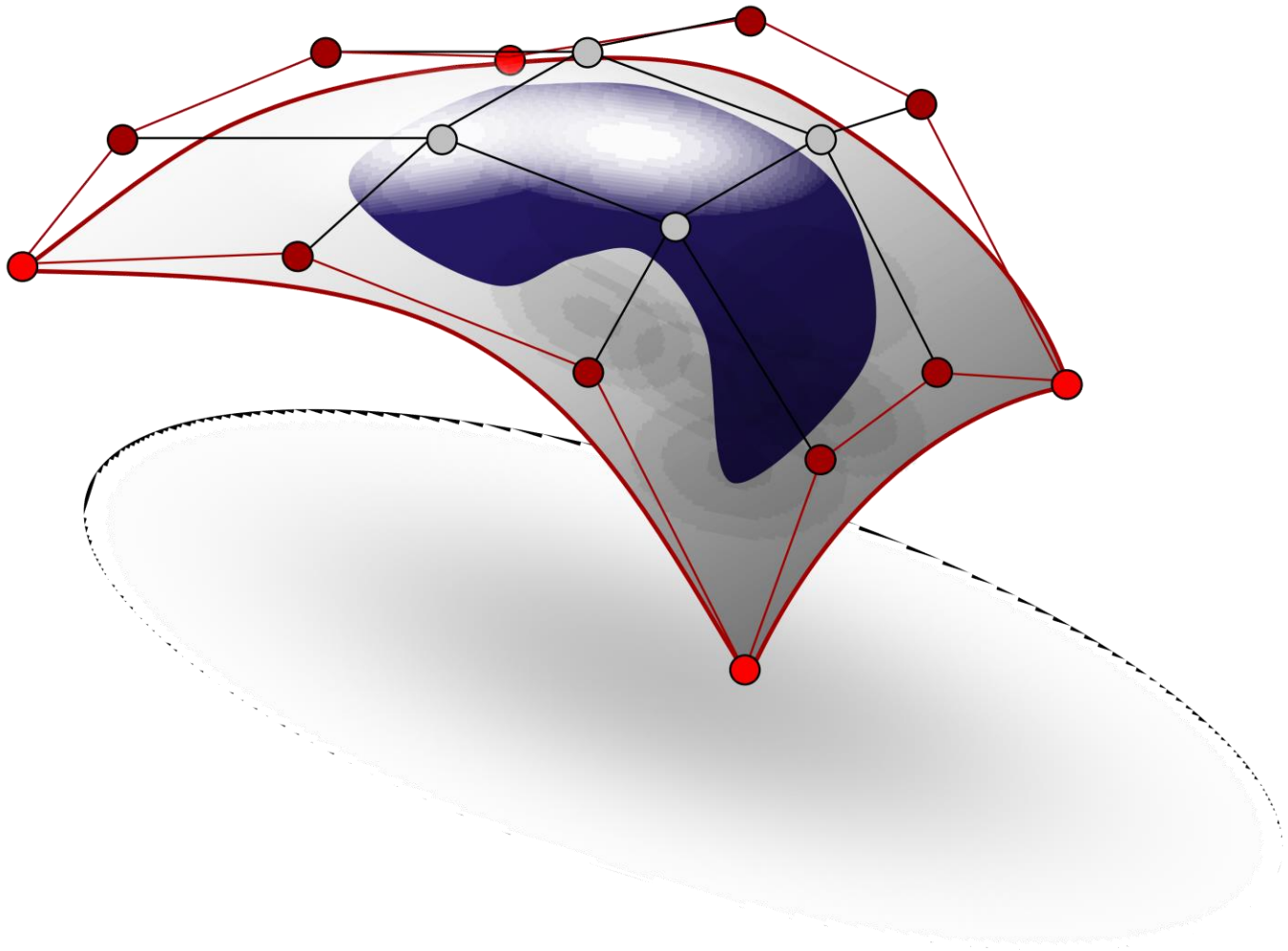
- Specify a curve in the parameter domain that encapsulates one (or more) pieces of area
- Tessellate the parameter domain accordingly to cut out the trimmed piece (rendering)

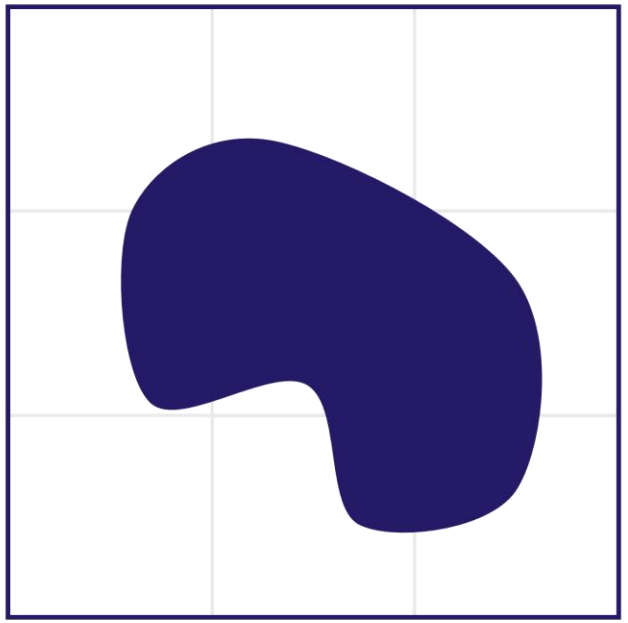
Curves-on-Surfaces (CONS)



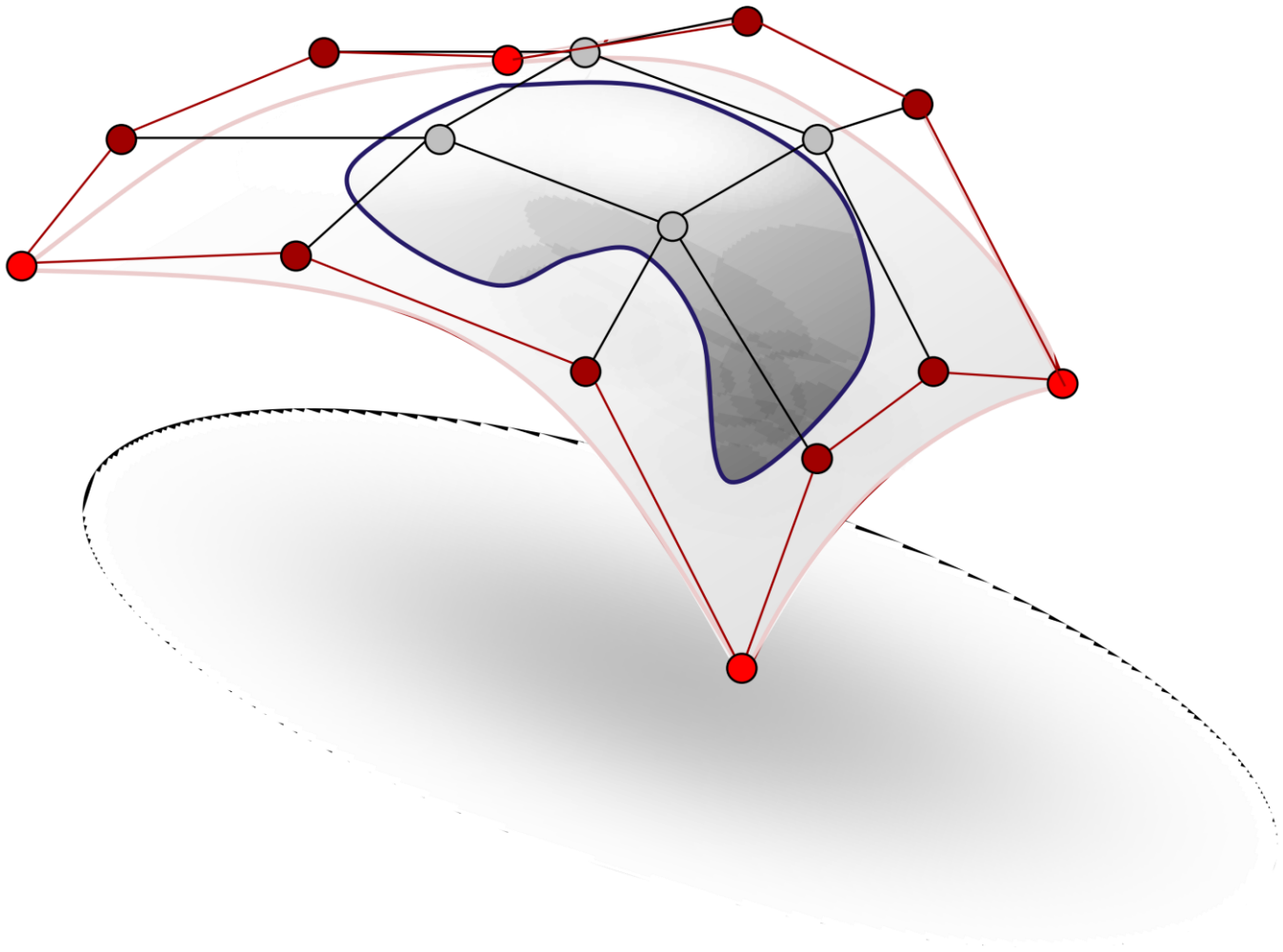


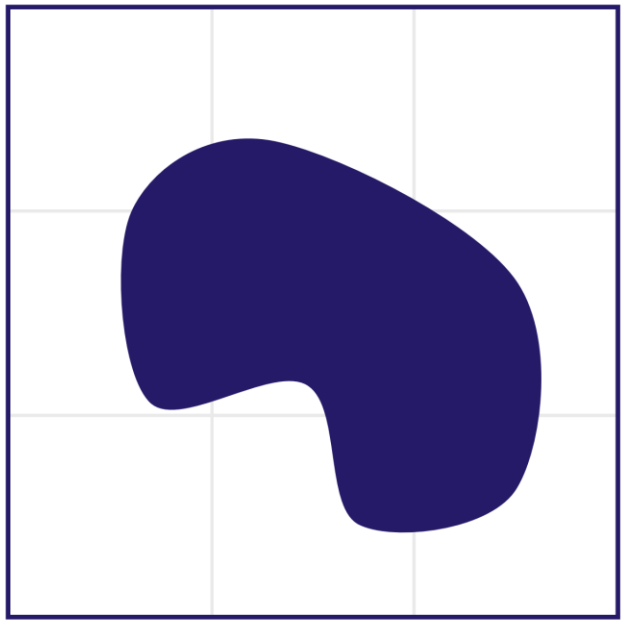
Curves-on-Surfaces (CONS)





Curves-on-Surfaces (CONS)

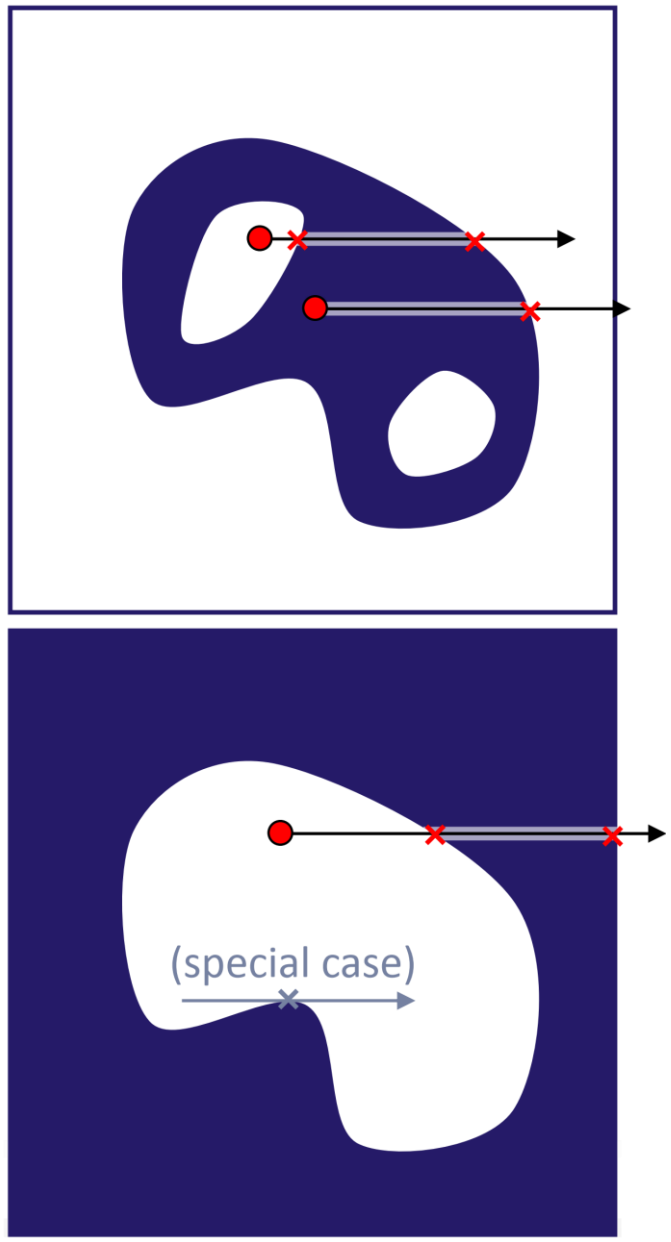




General Shapes

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

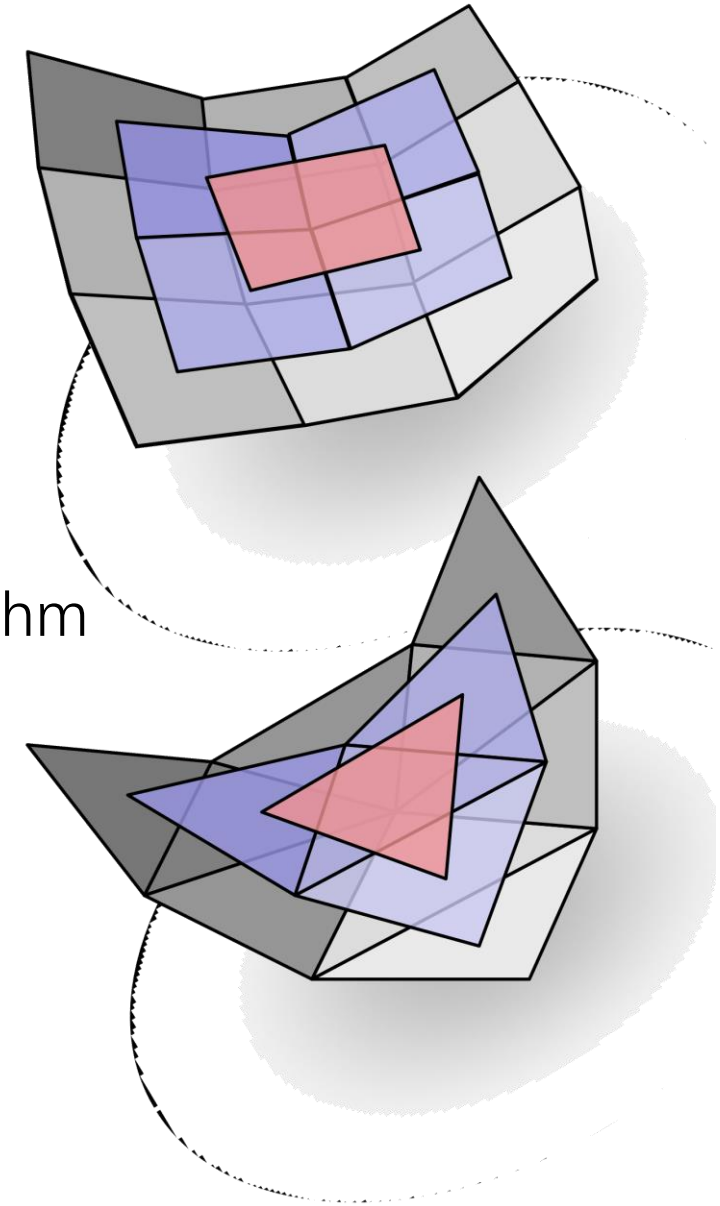


Total Degree Surfaces

Bezier Triangles

Alternative surface definition: Bezier 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 Bezier 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:

  $\boldsymbol{F}: \mathbb{R}^{n} \rightarrow \mathbb{R}^{m}$

  $\boldsymbol{f}: \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{m}$

  points as arguments

- Required Properties:
  - Diagonality: $\boldsymbol{f}(\boldsymbol{t}, \boldsymbol{t}, \ldots, \boldsymbol{t}) = \boldsymbol{F}(\boldsymbol{t})$
  - Symmetry: $\boldsymbol{f}(\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_d) = \boldsymbol{f}(\boldsymbol{t}_{\pi(1)}, \boldsymbol{t}_{\pi(2)}, \ldots, \boldsymbol{t}_{\pi(d)})$
    for all permutations of indices π
- Multi-affine: $\sum \alpha_k = 1$

$$\Rightarrow \boldsymbol{f}\left(\boldsymbol{t}_1, \ldots, \sum \alpha_k \boldsymbol{t}_i^{(k)}, \ldots, \boldsymbol{t}_d\right)$$
$$= \alpha_1 \boldsymbol{f}\left(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_i^{(1)}, \ldots, \boldsymbol{t}_d\right) + \cdots + \alpha_n \boldsymbol{f}\left(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_i^{(n)}, \ldots, \boldsymbol{t}_d\right)$$

Example

Example: bivariate monomial basis

- In powers of (u, v):

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

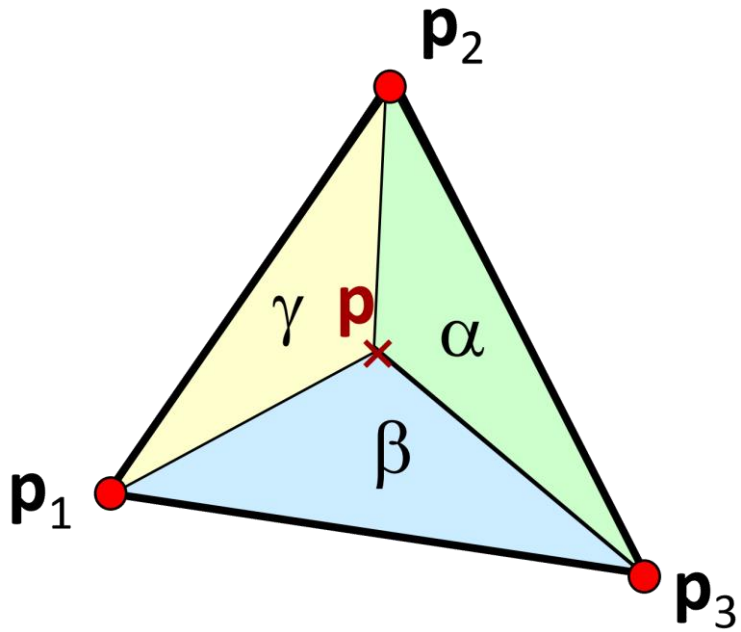
- Blossom form: multilinear in (u_1, u_2, v_1, v_2)

  $B = \{1,$

  $\frac{1}{2}(u_1 + u_2), \frac{1}{2}(v_1 + v_2),$

  $u_1 u_2, \frac{1}{4}(u_1 v_1 + u_1 v_2 + u_2 v_1 + u_2 v_2), v_1 v_2\}$

Barycentric Coordinates



Barycentric Coordinates:

- Planar case:

  Barycentric combinations of 3 points

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

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

- Area formulation

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

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

Barycentric Coordinates

Barycentric Coordinates:

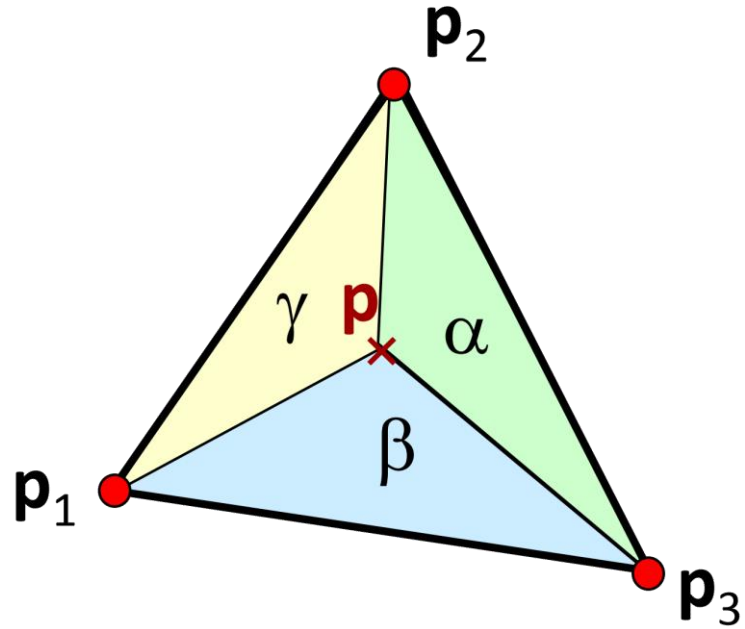
- Linear formulation:

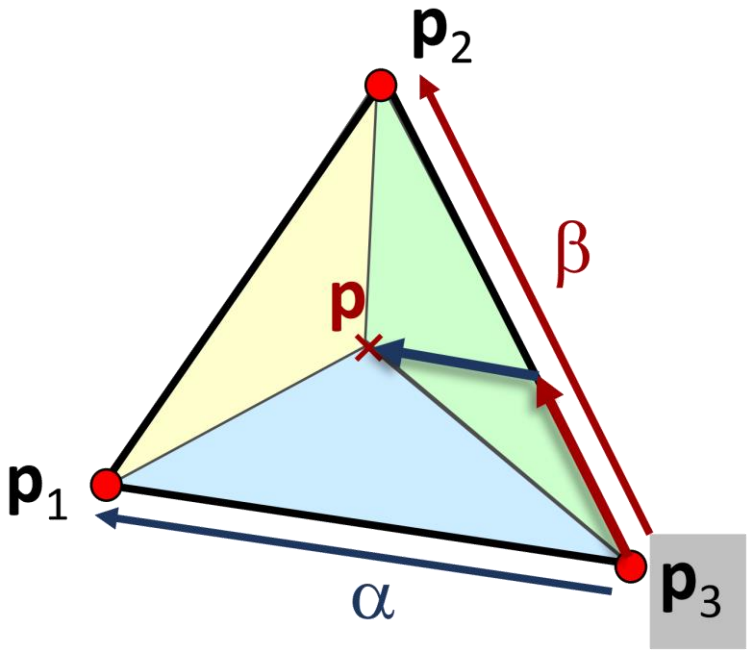
$$\boldsymbol{p} = \alpha\boldsymbol{p}_1 + \beta\boldsymbol{p}_2 + \gamma\boldsymbol{p}_3$$

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

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

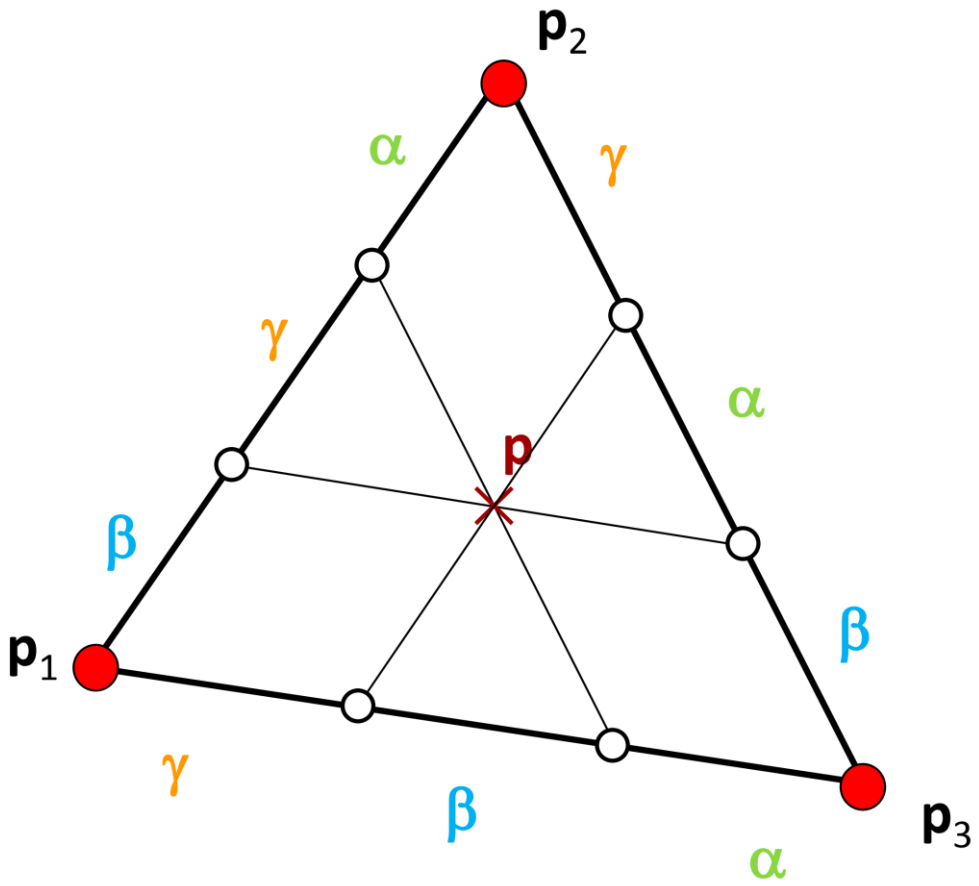
$$= \boldsymbol{p}_3 + \alpha(\boldsymbol{p}_1 - \boldsymbol{p}_3) + \beta(\boldsymbol{p}_2 - \boldsymbol{p}_3)$$

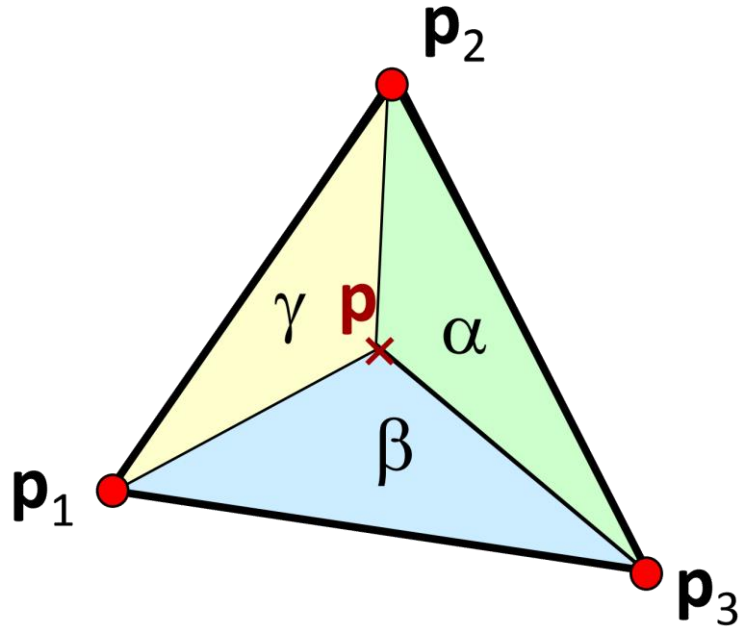




Barycentric Coordinates

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





Bezier Triangles: Overview

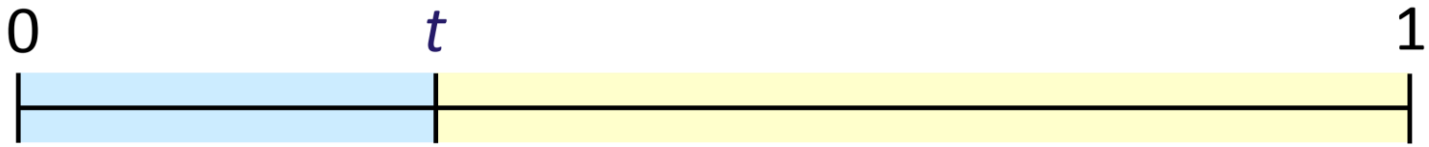
Bezier Triangles: Main Ideas

- 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: $\boldsymbol{p}(\underbrace{a, \ldots, a}_{i}, \underbrace{b, \ldots, b}_{j}, \underbrace{c, \ldots, c}_{k})$
- De Casteljau Algorithm: compute polynomial values $p(x, \ldots, x)$ by barycentric interpolation

Plugging in the Barycentric Coord's

Analog: 2D curves in barycentric coordinates

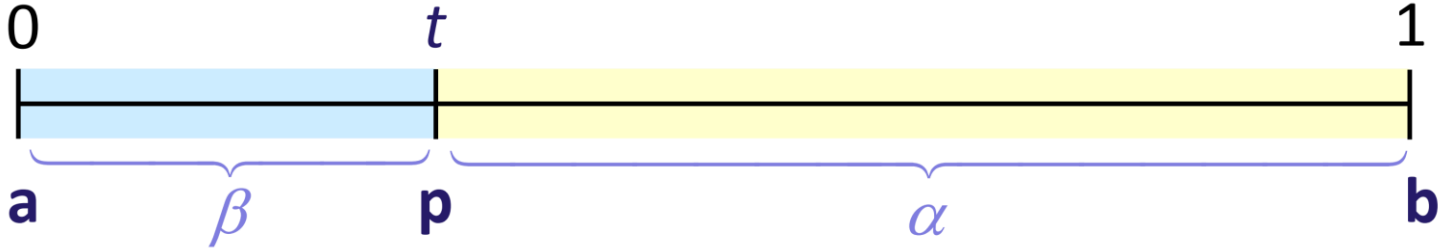
- Barycentric coordinates for 2D curves:



Plugging in the Barycentric Coord's

Analog: 2D curves in barycentric coordinates

- Barycentric coordinates for 2D curves:



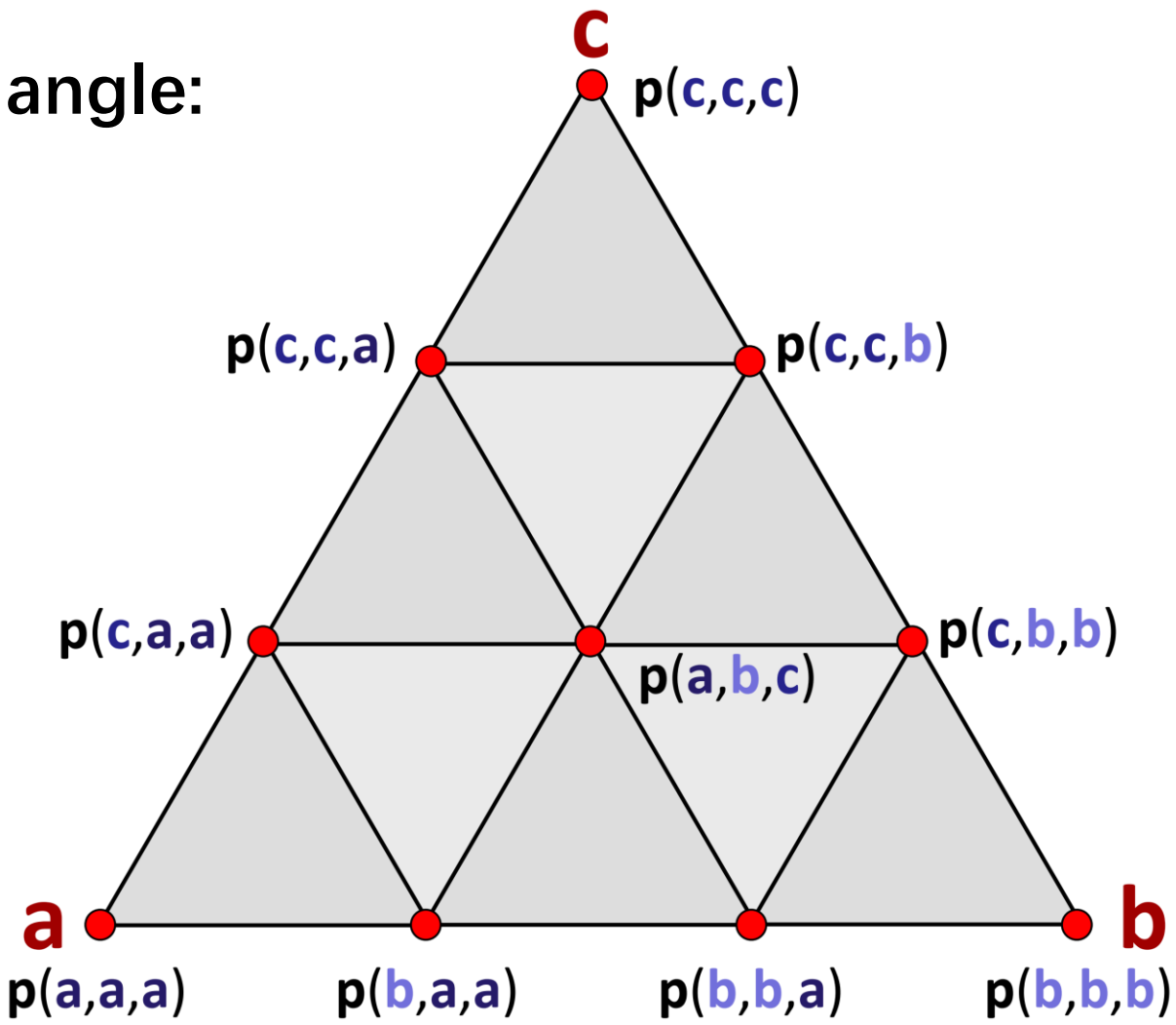
- $p = \alpha a + \beta b, \quad \alpha + \beta = 1$
- Bezier splines:

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

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

Example

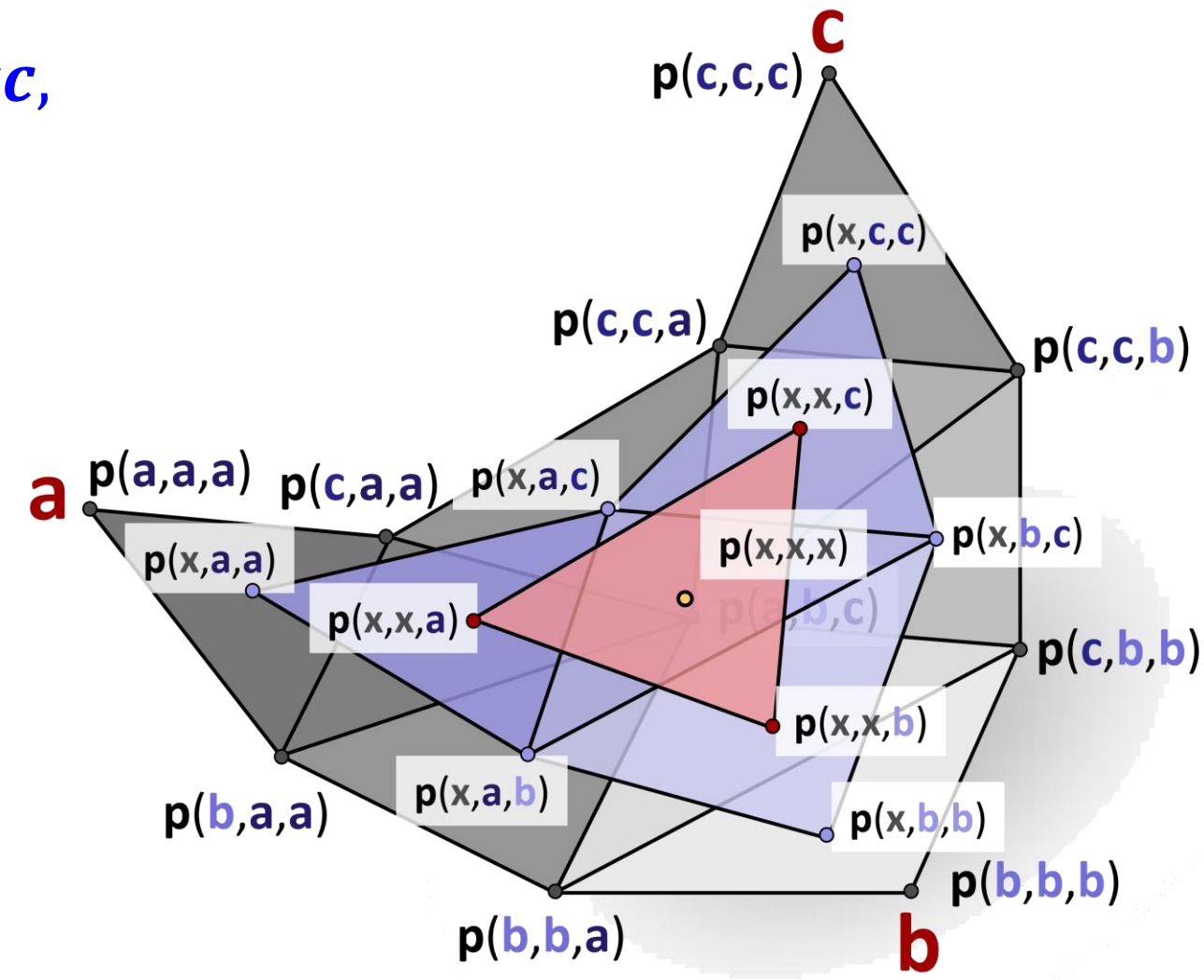
Cubic Bezier Triangle:



De Casteljau Algorithm

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

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



Bernstein Form

Writing this recursion out, we obtain:

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

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

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

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

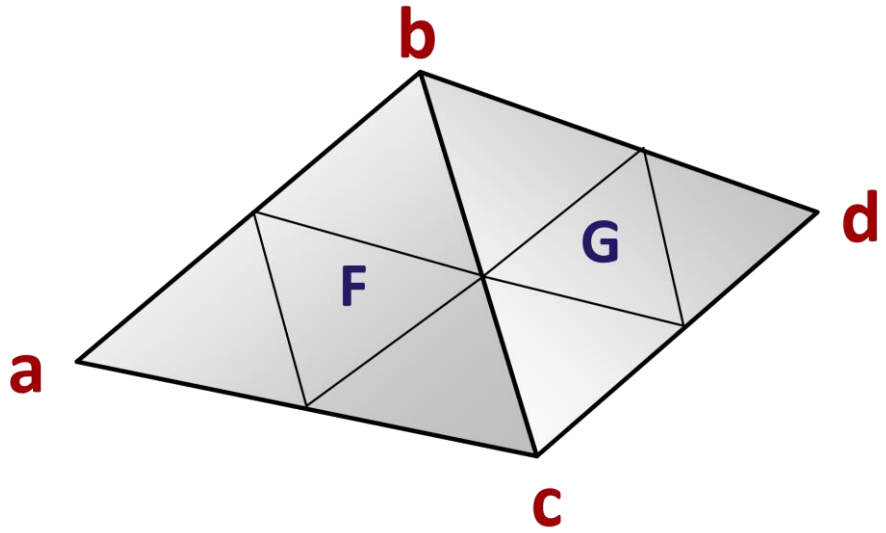
Continuity

We need to assemble Bezier 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

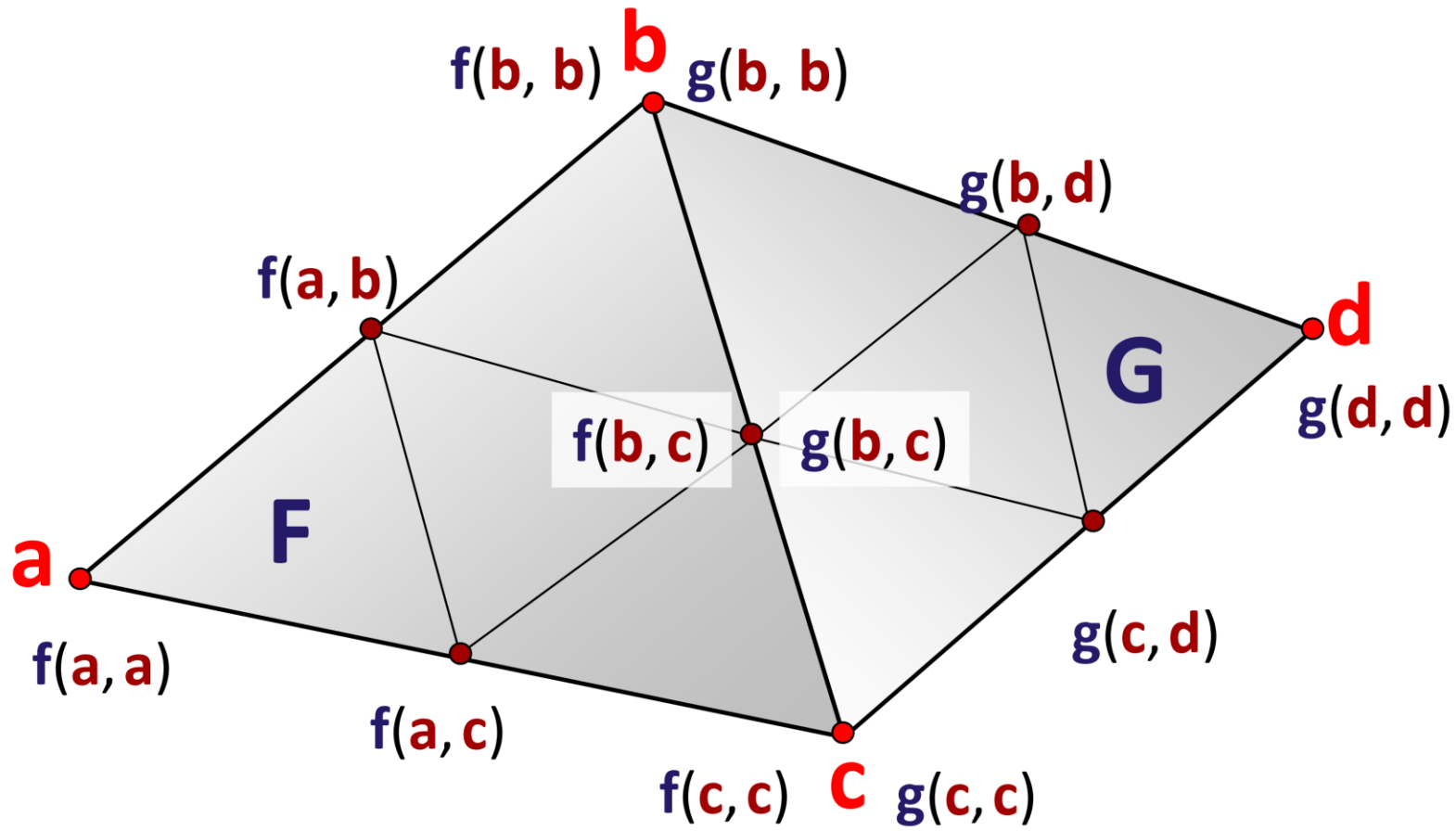
Situation:



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

Continuity

Situation:

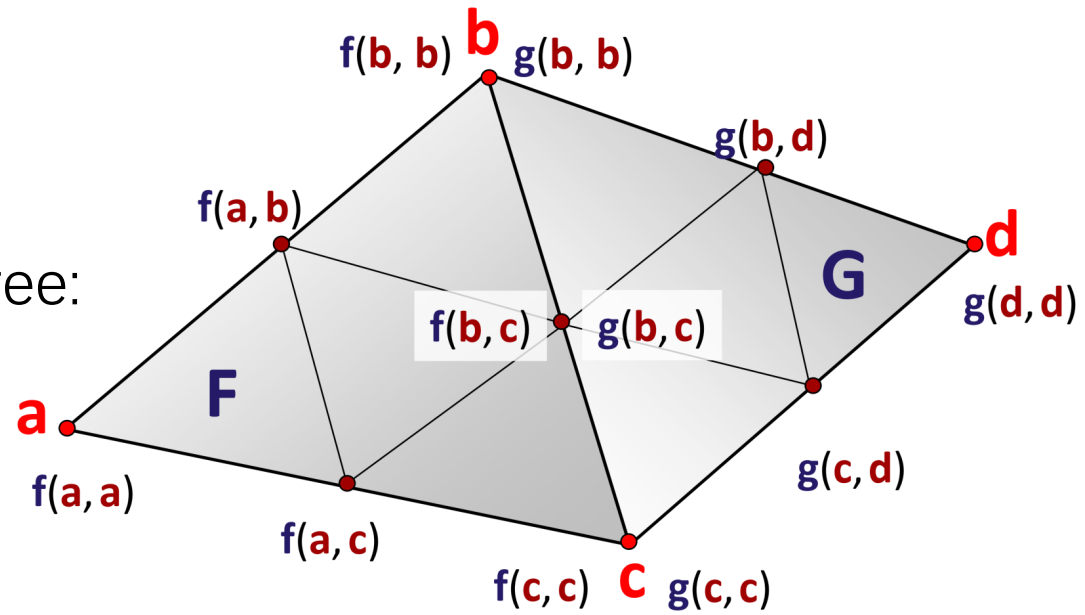


Continuity

C^0 continuity:

- The points on the boundary have to agree:

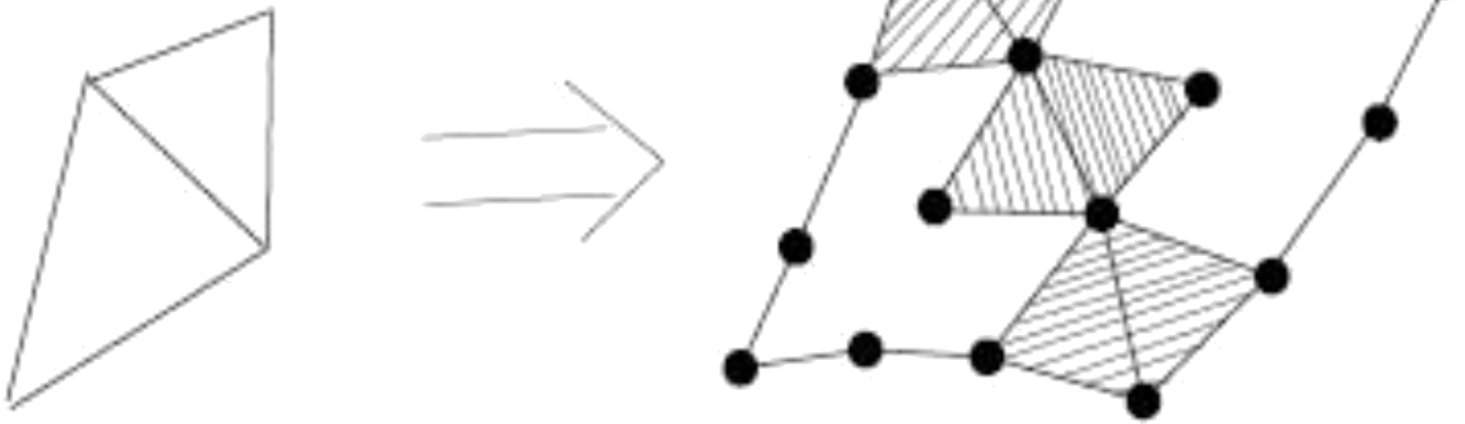
$$\boldsymbol{f(b,b) = g(b,b)}$$
$$\boldsymbol{f(b,c) = g(b,c)}$$
$$\boldsymbol{f(c,c) = g(c,c)}$$



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

$$\boldsymbol{f(x,x)} = \beta\boldsymbol{f(b,x)} + \gamma\boldsymbol{f(c,x)}$$
$$= \beta^2\boldsymbol{f(b,b)} + 2\beta\gamma\boldsymbol{f(b,c)} + \gamma^2\boldsymbol{f(c,c)}$$
$$\| \qquad\qquad \| \qquad\qquad \|$$
$$\boldsymbol{g(b,b)} \qquad \boldsymbol{g(b,c)} \qquad \boldsymbol{g(c,c)}$$
$$= \beta^2\boldsymbol{g(b,b)} + 2\beta\gamma\boldsymbol{g(b,c)} + \gamma^2\boldsymbol{g(c,c)}$$
$$= \beta\boldsymbol{g(b,x)} + \gamma\boldsymbol{g(c,x)} = \boldsymbol{g(x,x)}$$

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:

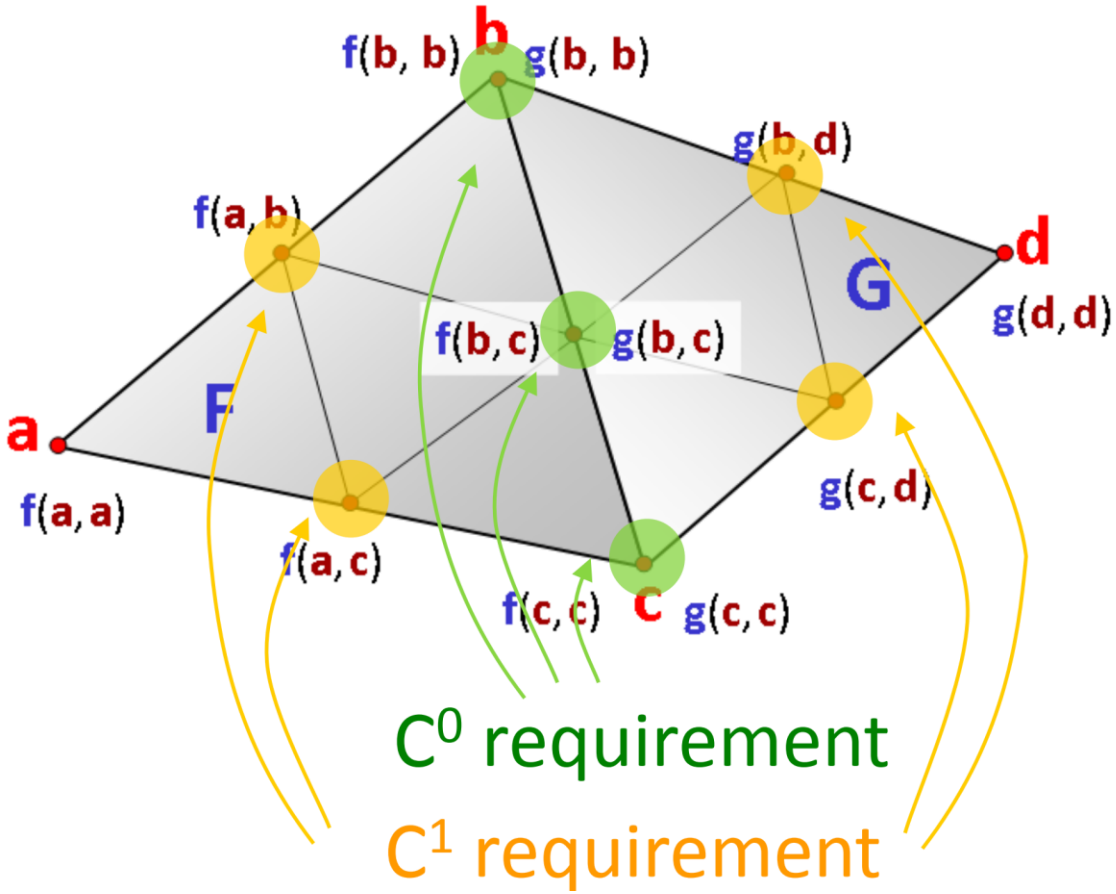
  The blossoms have to agree partially:

$$f(a,b) = g(a,b)$$

$$f(b,d) = g(b,d)$$

$$f(a,c) = g(a,c)$$

$$f(c,d) = g(c,d)$$



Continuity

C^1 Continuity: Proof

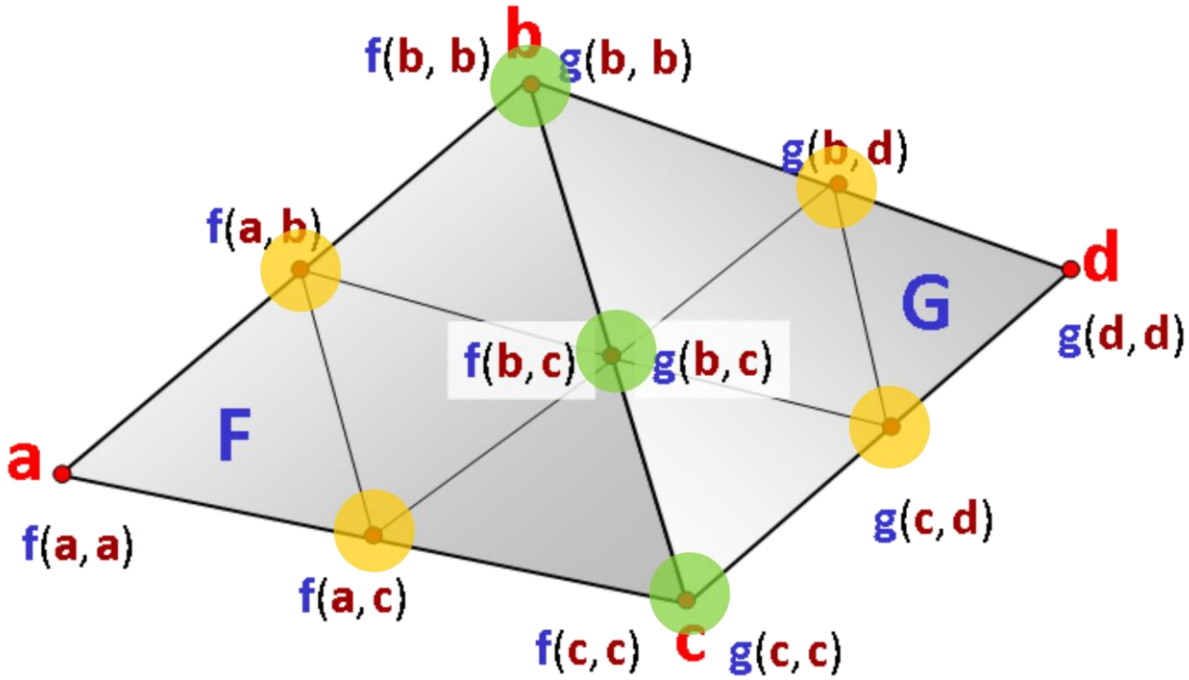
- Derivatives:

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

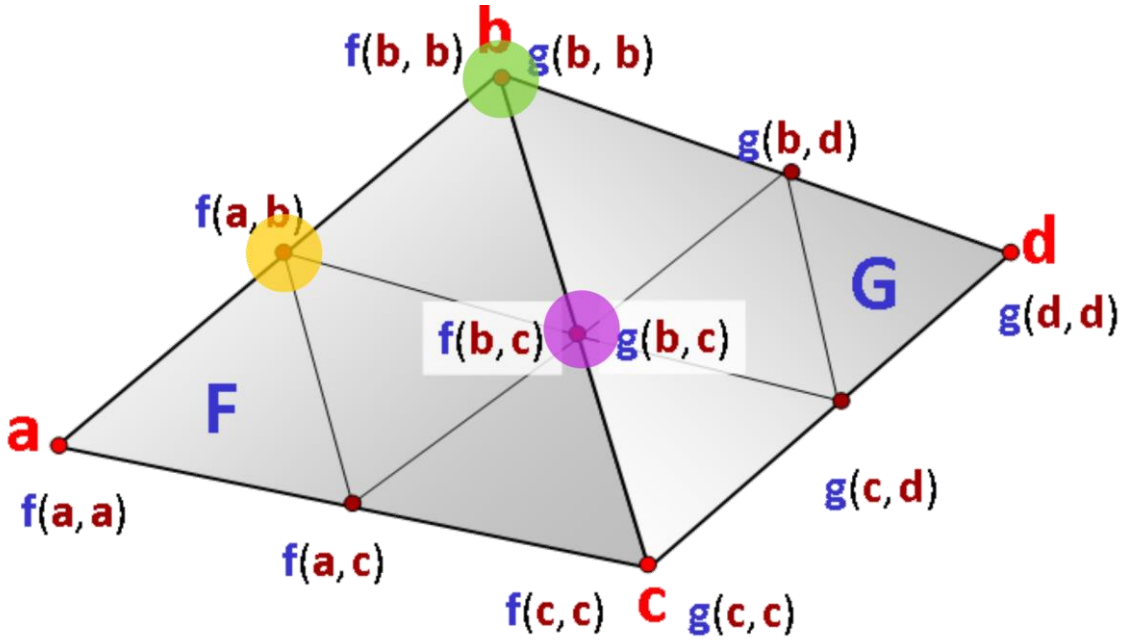
  (similar to the curve case)
- C^1-Continuity:

  $$\forall \boldsymbol{x} \in \mathbb{R}^3 : \boldsymbol{f}(\boldsymbol{p}, \boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{p}, \boldsymbol{x})$$
- We have to show

  $$\forall \boldsymbol{x} \in \mathbb{R}^3 : \begin{cases} \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{b}, \boldsymbol{x}) \\ \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{c}, \boldsymbol{x}) \end{cases}$$
- $\Rightarrow C^1$ continuity follows for all boundary points (by interp.)



Continuity



C^1 **Continuity:** Proof

- So we have to show

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

- Proof:

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

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

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

C^0 C^0

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

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

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

(same for the other conditions)

Continuity

So what does this mean?

- The blossoms have to agree partially:

$$f(a,b) = g(a,b)$$
$$f(b,d) = g(b,d)$$
$$f(a,c) = g(a,c)$$
$$f(c,d) = g(c,d)$$

- The points must be coplanar (with edge points):
  $f(a,b), g(b,d), g(b,b), g(b,c)$
- The points in F must be affine images of the points in G