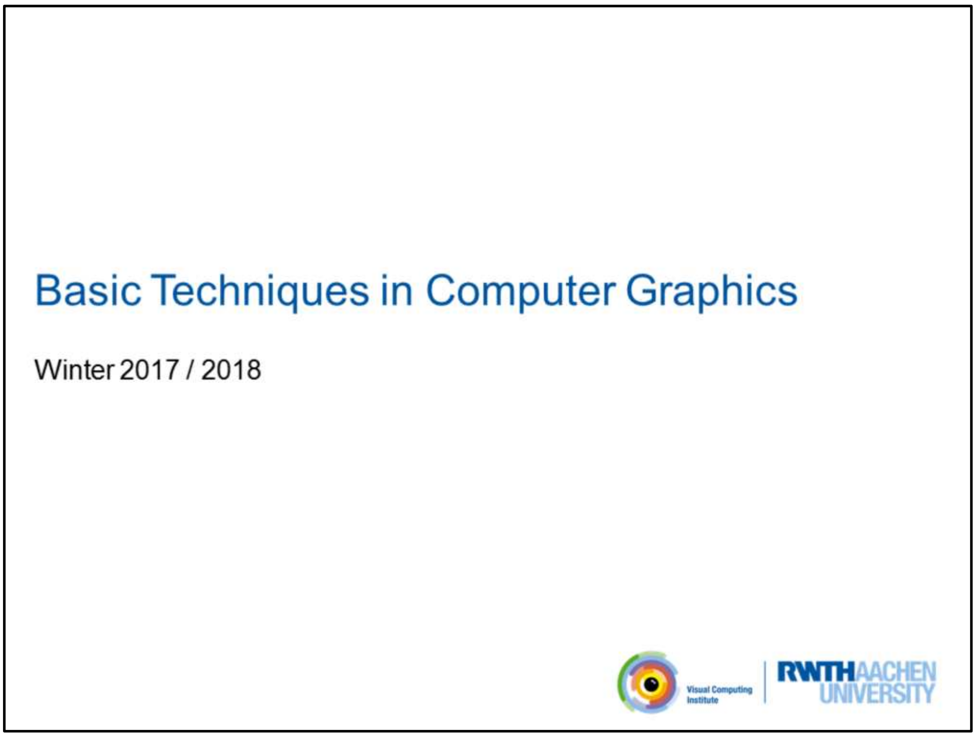


The slide comments are not guaranteed to be complete, they are no alternative to the lectures itself. So go to the lectures and write down your own comments!

Curves and Surfaces

- **Freeform curves**
- Freeform surfaces
- Mesh Subdivision

Functional Representation

$$f : \mathbb{R}^d \to \mathbb{R}^n$$

d = 1 ... univariate / curves
d = 2 ... bi-variate / surfaces
d = 3 ... tri-variate / volumes
d = 4 ... quad-variate / space-time
light fields

Functional Representation

$$f : \mathbb{R}^d \to \mathbb{R}^n$$

n = 1 ... scalar
n = 2 ... planar
n = 3 ... spatial
n = 4 ... homogeneous

Basic Arithmetics

- Polynomials: $f(t) \in \Pi^n$

Linear Structure

- Polynomials: $f(t) \in \Pi^n$
- Parametric curves $f : R \to R^3$
  - ***vector valued*** coefficients b_i
  - ***scalar valued*** basis functions F_i

$$f : \mathbb{R} \to \mathbb{R}^3, \;\; t \mapsto \sum_{i=0}^{n} b_i F_i(t)$$

Freeform Curves

Monomial basis

- No geometric meaning
- Sum of vectors, not points

$$f(t) = \sum_{i=0}^{n} b_i \cdot t^i$$

$\Rightarrow$ Find a better basis for Π^n

Freeform Curves

Bernstein basis

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

- Non negative

$$B_i^n(t) \geq 0, \;\; t \in [0,1]$$

- Partition of unity

$$\sum_{i=0}^{n} B_i^n(t) \equiv 1$$

- Recursion

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

$$B_0^0(t) \equiv 1 \;\wedge\; B_i^n(t) \equiv 0 \text{ for } i \notin \{0,\ldots,n\}$$

Bézier Curves

Bézier curves $$f(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

– Geometric meaning (affine combination)

de Casteljau Algorithm

Given: control points $b_0,\ldots,b_n$, parameter t

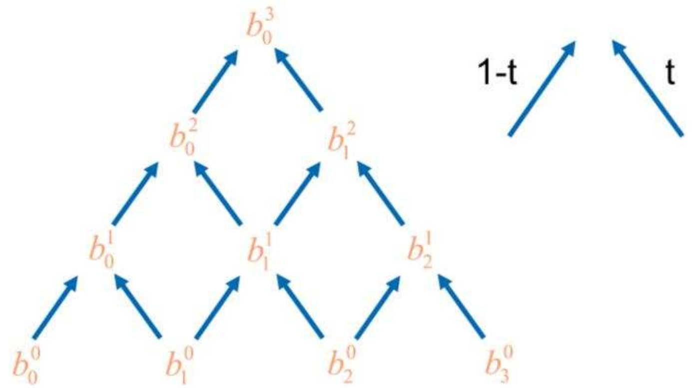
Init: $b_i^0(t) := b_i$

Recursion:

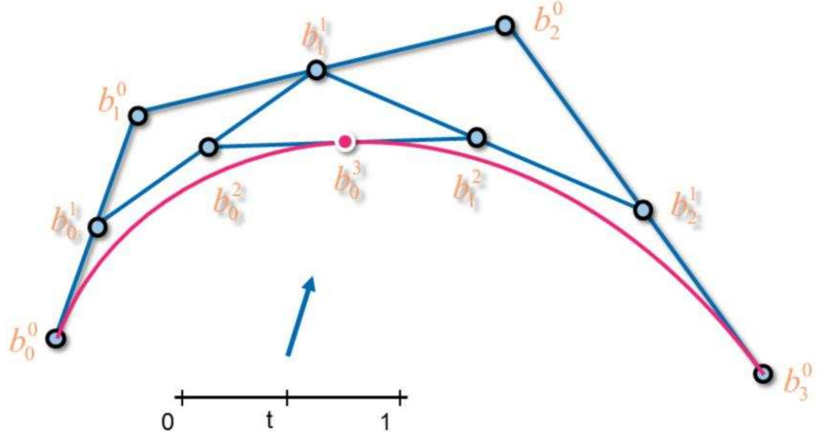
$$b_i^k(t) = (1-t)b_i^{k-1} + t b_{i+1}^{k-1} \quad \begin{array}{lcl} k & = & 1,\ldots,n \\ i & = & 0,\ldots,n-k \end{array}$$

$$\rightarrow \; b_0^n(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

de Casteljau Algorithm



de Casteljau Algorithm



Bézier Curves

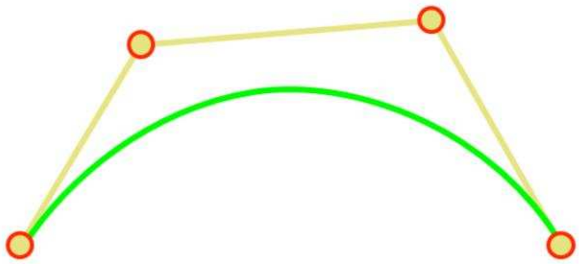
Bézier curves

$$f(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

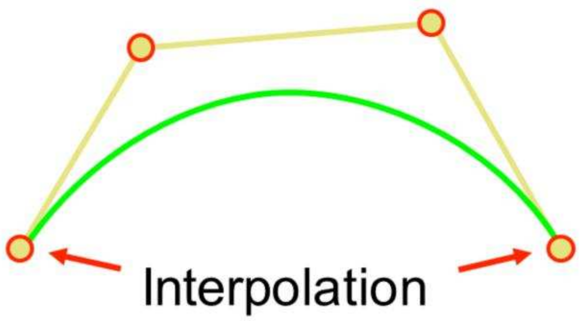
- Geometric meaning (affine combination)
- Affine invariance
- Convex hull
- Endpoint interpolation $f(0) = b_0 \wedge f(1) = b_n$
- Endpoint derivative

$$f'(0) = n(b_1 - b_0), \quad f'(1) = n(b_n - b_{n-1})$$

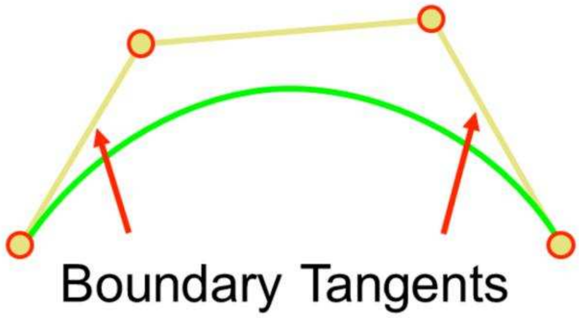
Properties

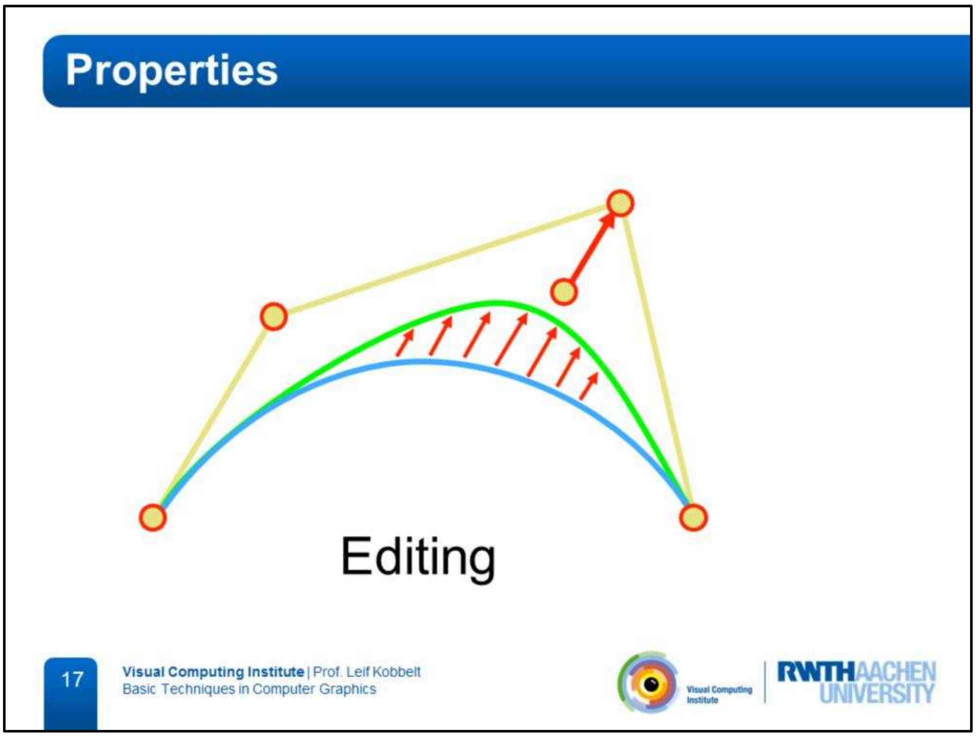


Properties



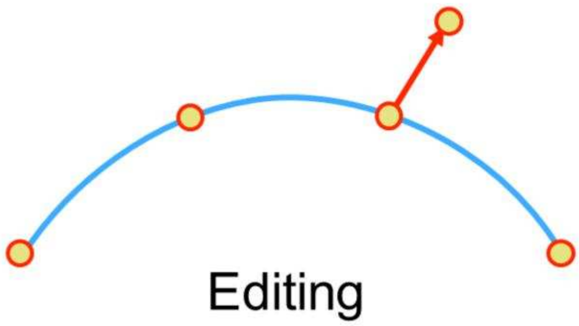
Properties

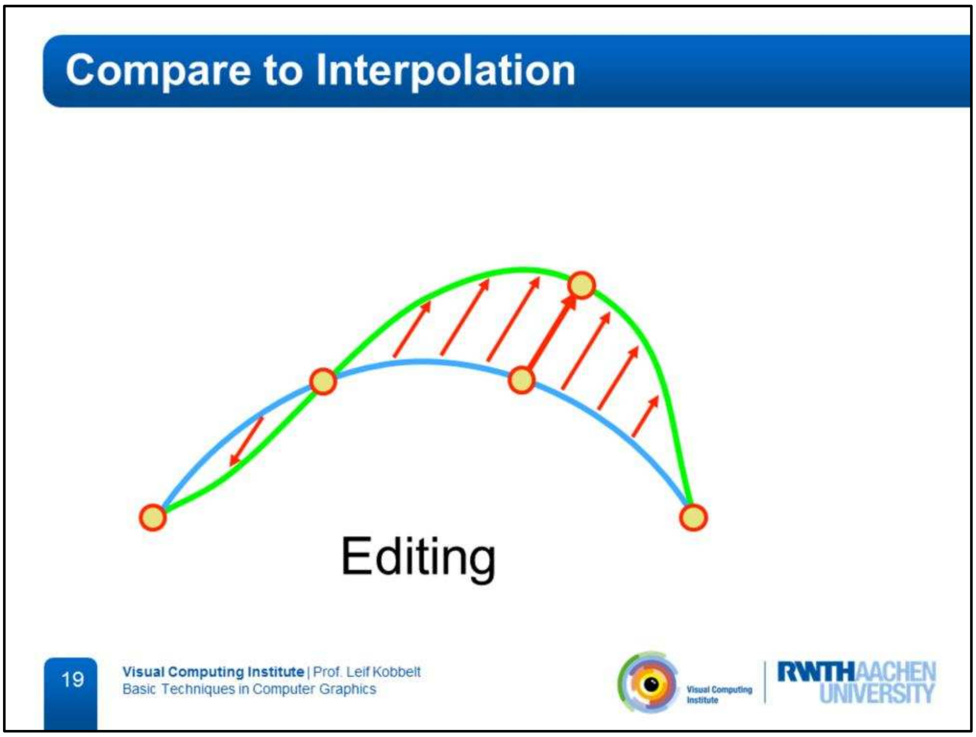




Moving one vertex of the control polygon will move the curve in that direction (only the distance is dependent of the position on the curve).

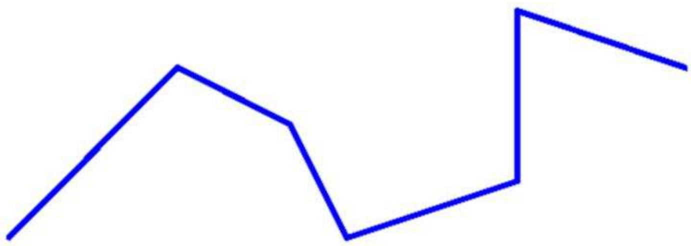
Compare to Interpolation



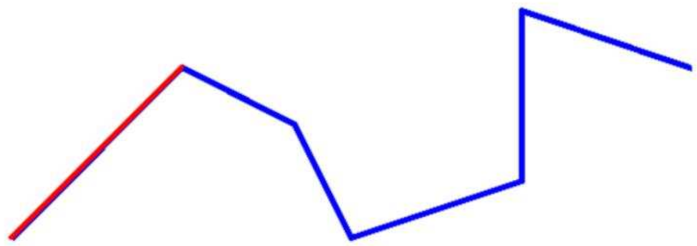


With curves that are interpolated between given points, on the pro side a point of the curve itself can be moved around, but on the down side the resulting curve can behave in an unexpected way (like moving in the opposite direction).

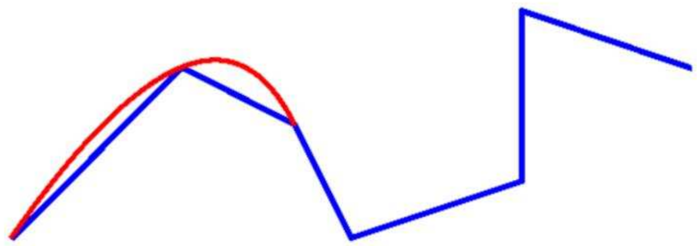
Stability



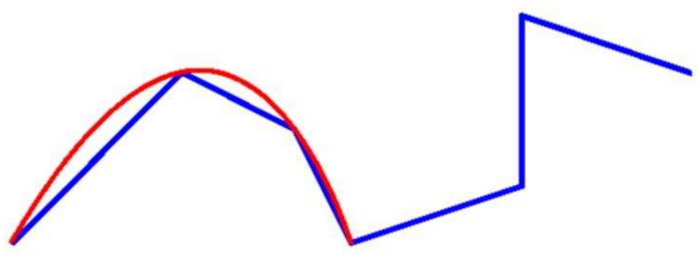
Stability



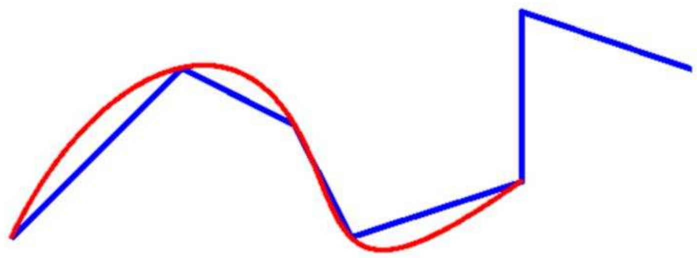
Stability



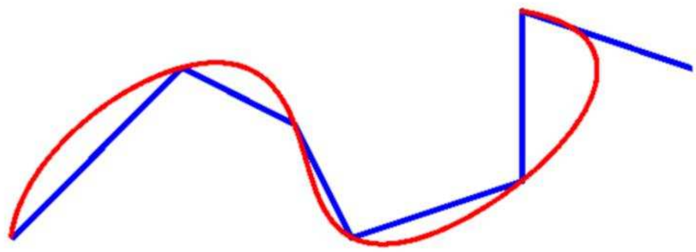
Stability



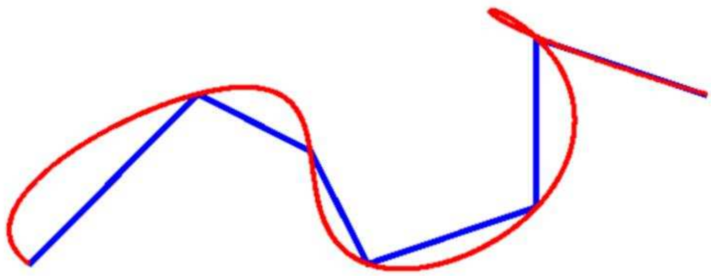
Stability



Stability

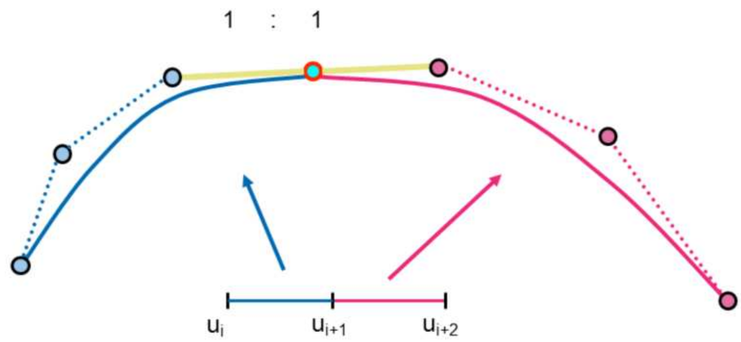


Stability



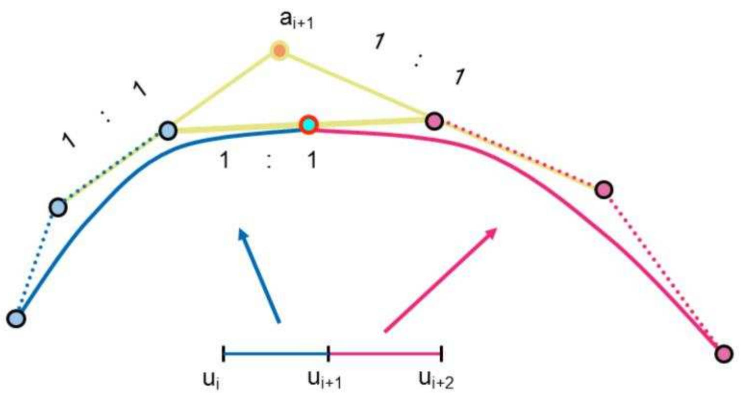
Splines: C^1 continuity

Control points: $c_n = d_0 = \frac{1}{2}(c_{n-1} + d_1)$



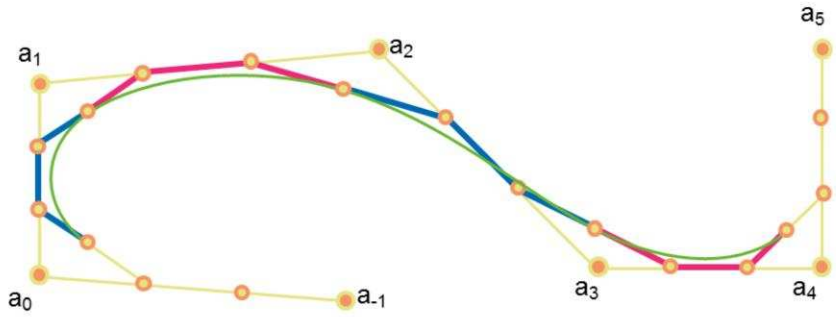
Splines: C^2 continuity

A-frame construction



C^2 Cubic Splines

Bézier control points defined by $a_{-1}, \ldots, a_{L+1}$



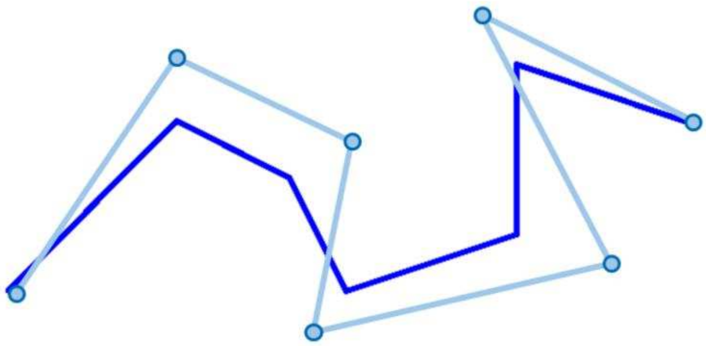
Interpolation

$$\begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & 1 & 4 & 1 & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ d_{i-1} \\ d_i \\ d_{i+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ p_{i-1} \\ p_i \\ p_{i+1} \\ \vdots \end{pmatrix}$$

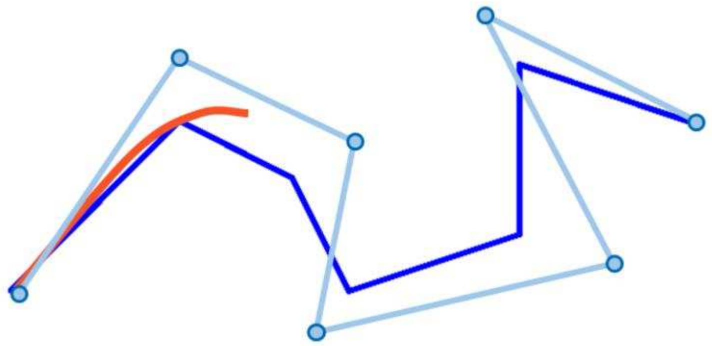
Interpolation

$$\begin{pmatrix} 1 & & & & & & \\ 1 & 4 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & 1 & 4 & 1 \\ & & & & & & 1 \end{pmatrix} \begin{pmatrix} \vdots \\ d_{i-1} \\ d_i \\ d_{i+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ p_{i-1} \\ p_i \\ p_{i+1} \\ \vdots \end{pmatrix}$$

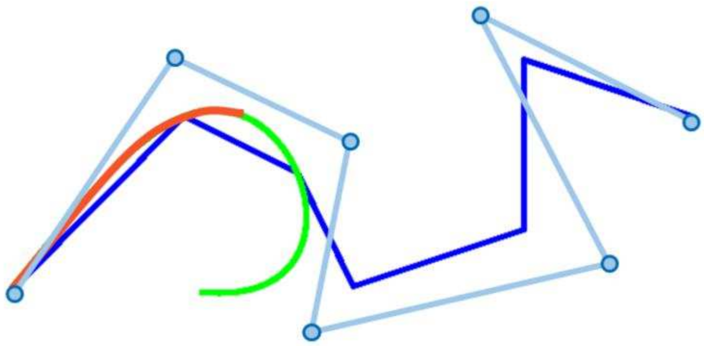
C^2 Cubic Splines



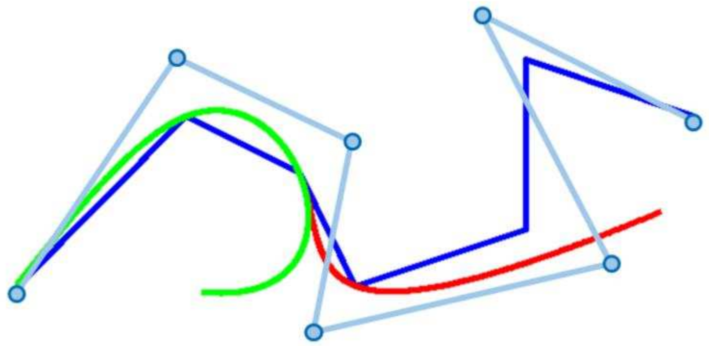
C^2 Cubic Splines



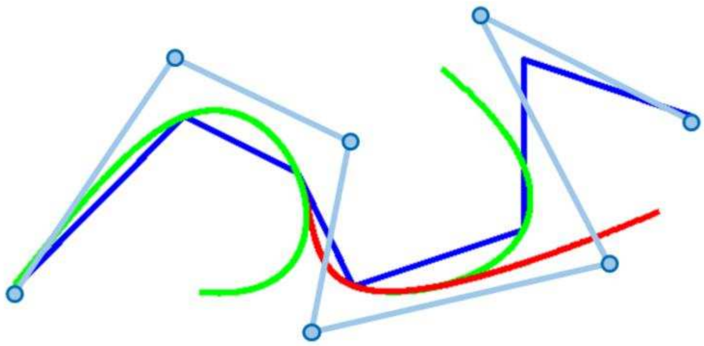
C^2 Cubic Splines



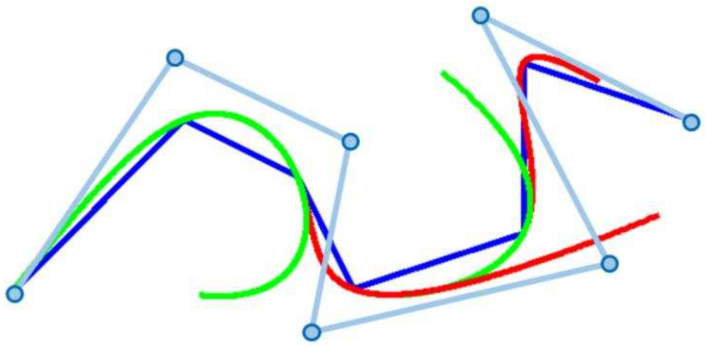
C^2 Cubic Splines



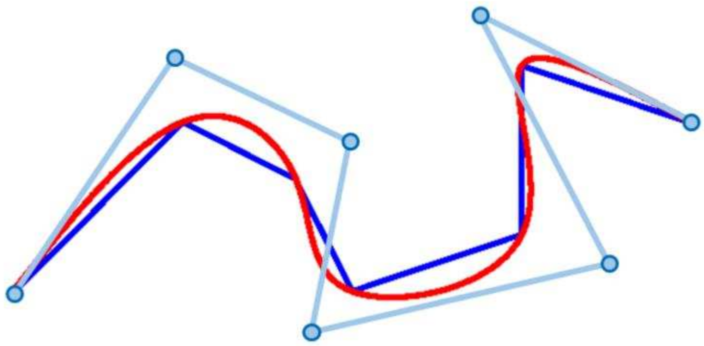
C^2 Cubic Splines



C^2 Cubic Splines



C^2 Cubic Splines



C^2 Cubic Splines

- Bézier control points defined by $a_0, \ldots, a_n$
- Affine invariance
- Convex hull
- Maximal smoothness
- Local control !

Curves and Surfaces

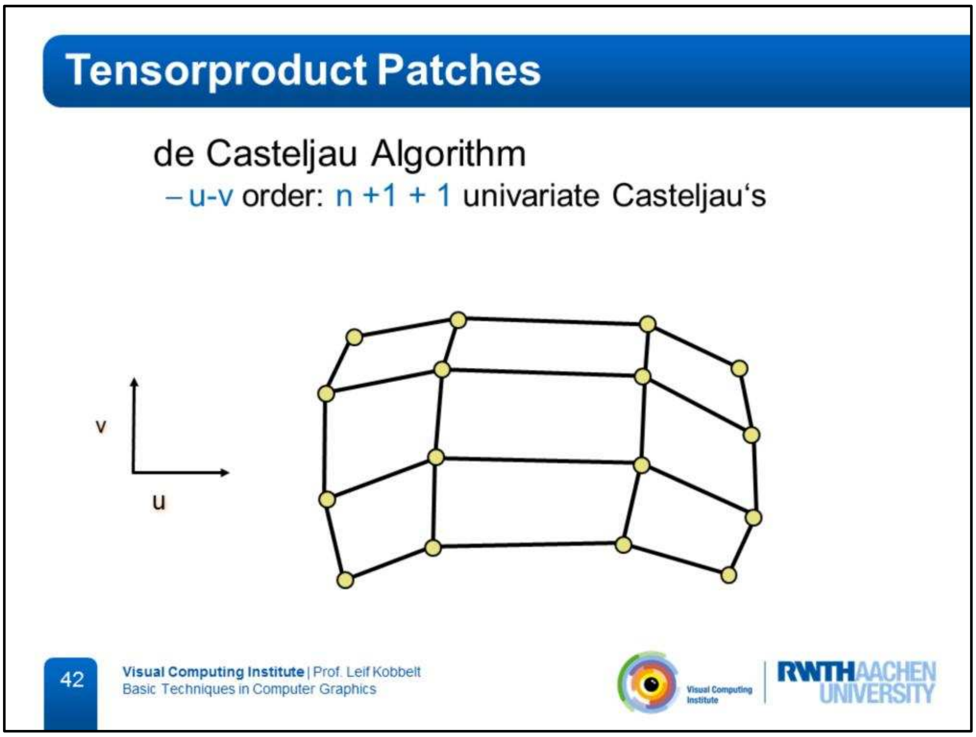
- Freeform curves
- **Freeform surfaces**
- Mesh Subdivision

Tensorproduct Patches

"Curve-valued curves"

- Bézier curve $$f(t) = \sum_{i=0}^{m} b_i B_i^m(t)$$
- Control points move on curves

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

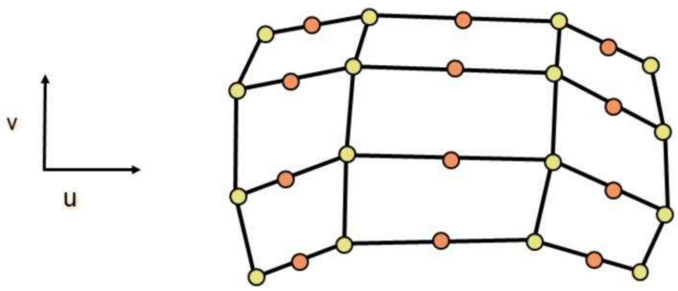


Run de Casteljau four times in one direction...

Tensorproduct Patches

de Casteljau Algorithm

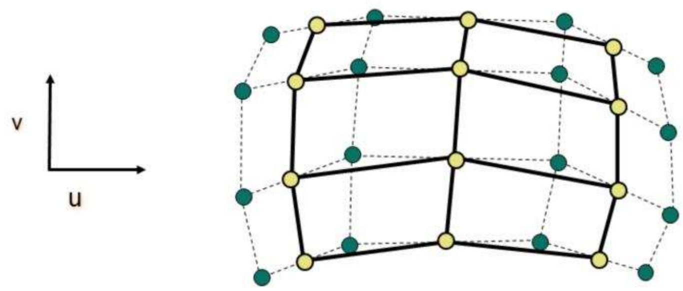
– u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

de Casteljau Algorithm

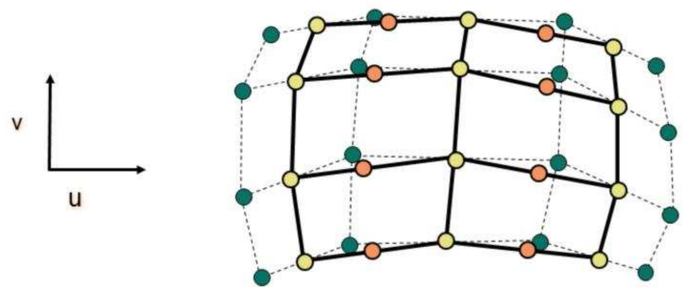
– u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

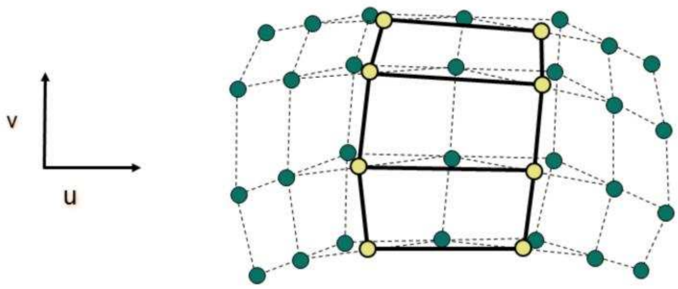
de Casteljau Algorithm

– u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

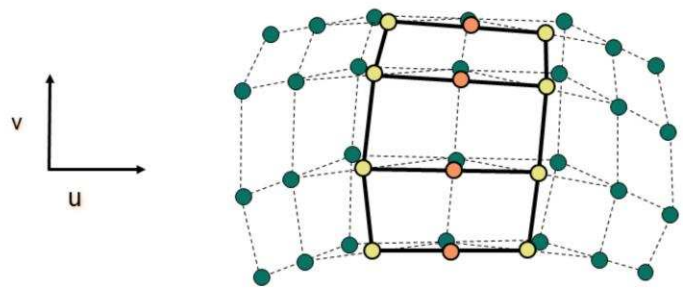
de Casteljau Algorithm
– u-v order: n +1 + 1 univariate Casteljau's

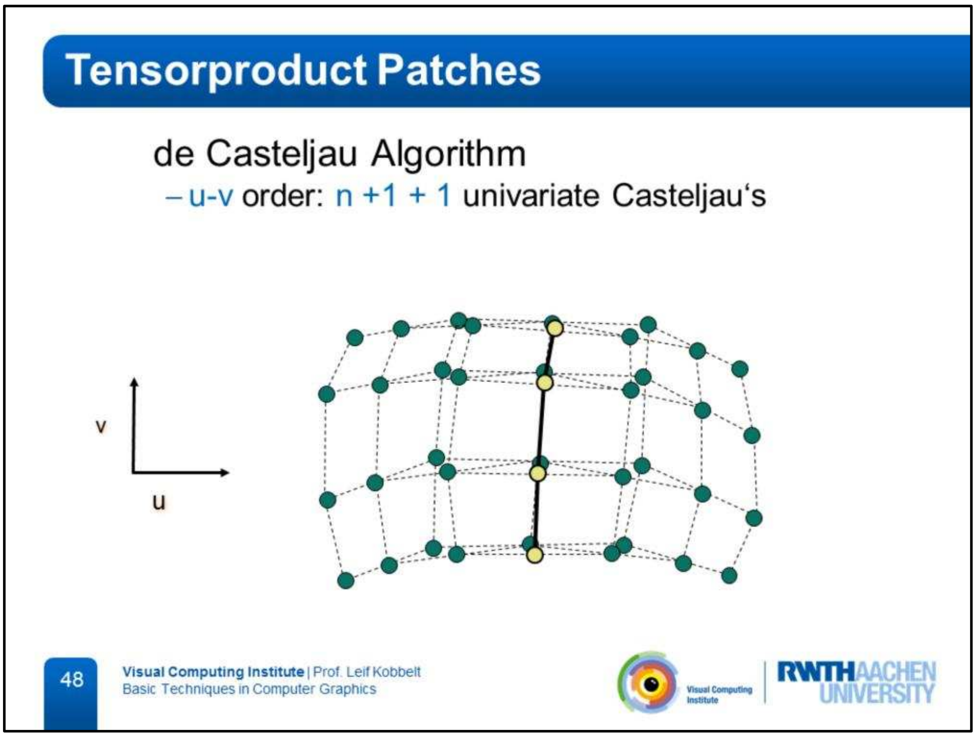


Tensorproduct Patches

de Casteljau Algorithm

– u-v order: n +1 + 1 univariate Casteljau's



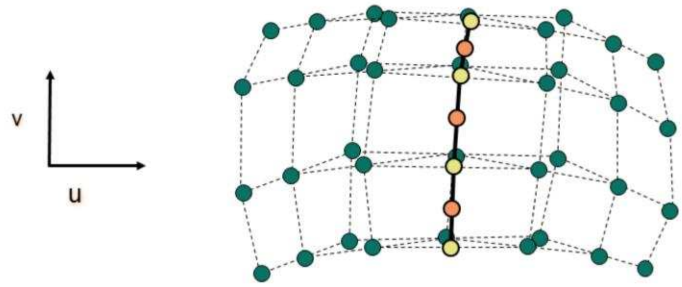


...use the four resulting points as start of a de Casteljau in the other direction.

Tensorproduct Patches

de Casteljau Algorithm

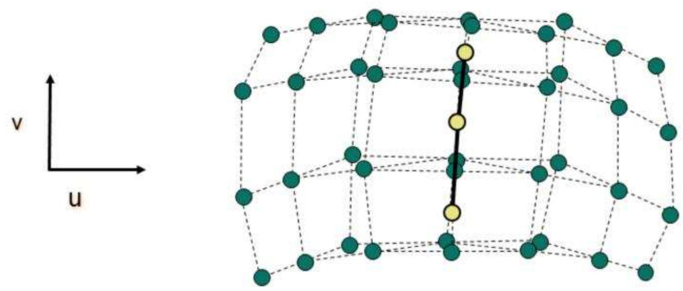
– u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

de Casteljau Algorithm

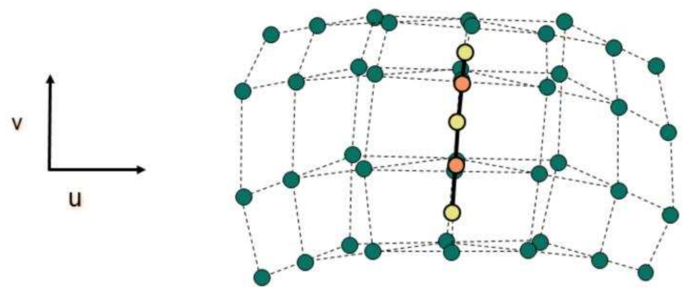
– u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

de Casteljau Algorithm

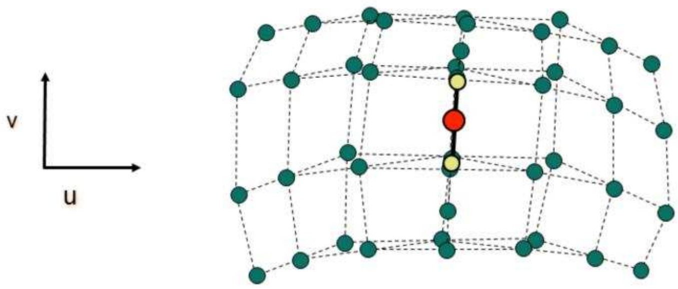
- u-v order: n +1 + 1 univariate Casteljau's



Tensorproduct Patches

de Casteljau Algorithm

– u-v order: n +1 + 1 univariate Casteljau's



The final point.

Tensorproduct Patches

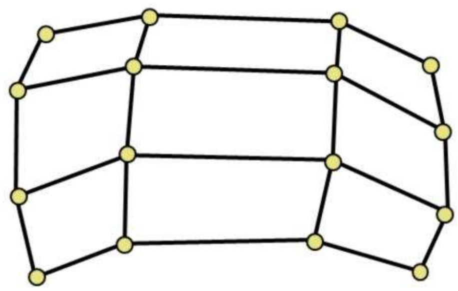
de Casteljau Algorithm

– u-v order: $n+1+1$ univariate Casteljau's
– v-u order: $m+1+1$ univariate Casteljau's

Tensorproduct Patches

de Casteljau Algorithm

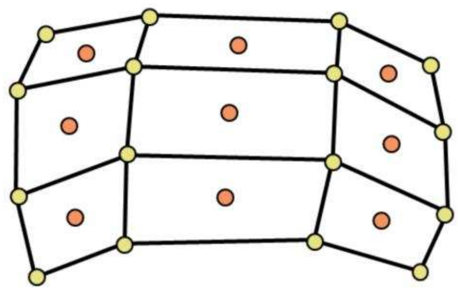
– direct, bilinear



Tensorproduct Patches

de Casteljau Algorithm

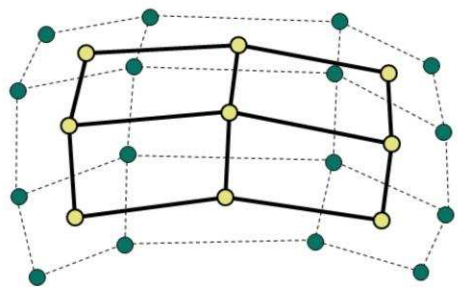
- direct, bilinear



Tensorproduct Patches

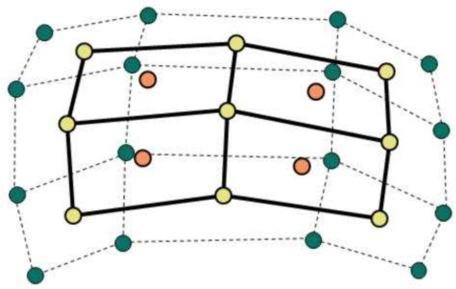
de Casteljau Algorithm

– direct, bilinear



Tensorproduct Patches

de Casteljau Algorithm
- direct, bilinear



Tensorproduct Patches

de Casteljau Algorithm

– direct, bilinear