Ray vs. Bèzier Surface Intersection

CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

https://cgg.mff.cuni.cz/~pepca/

Bicubic Bèzier patch

$$\mathbf{P}_{ij} = [x_{ij}, y_{ij}, z_{ij}]$$

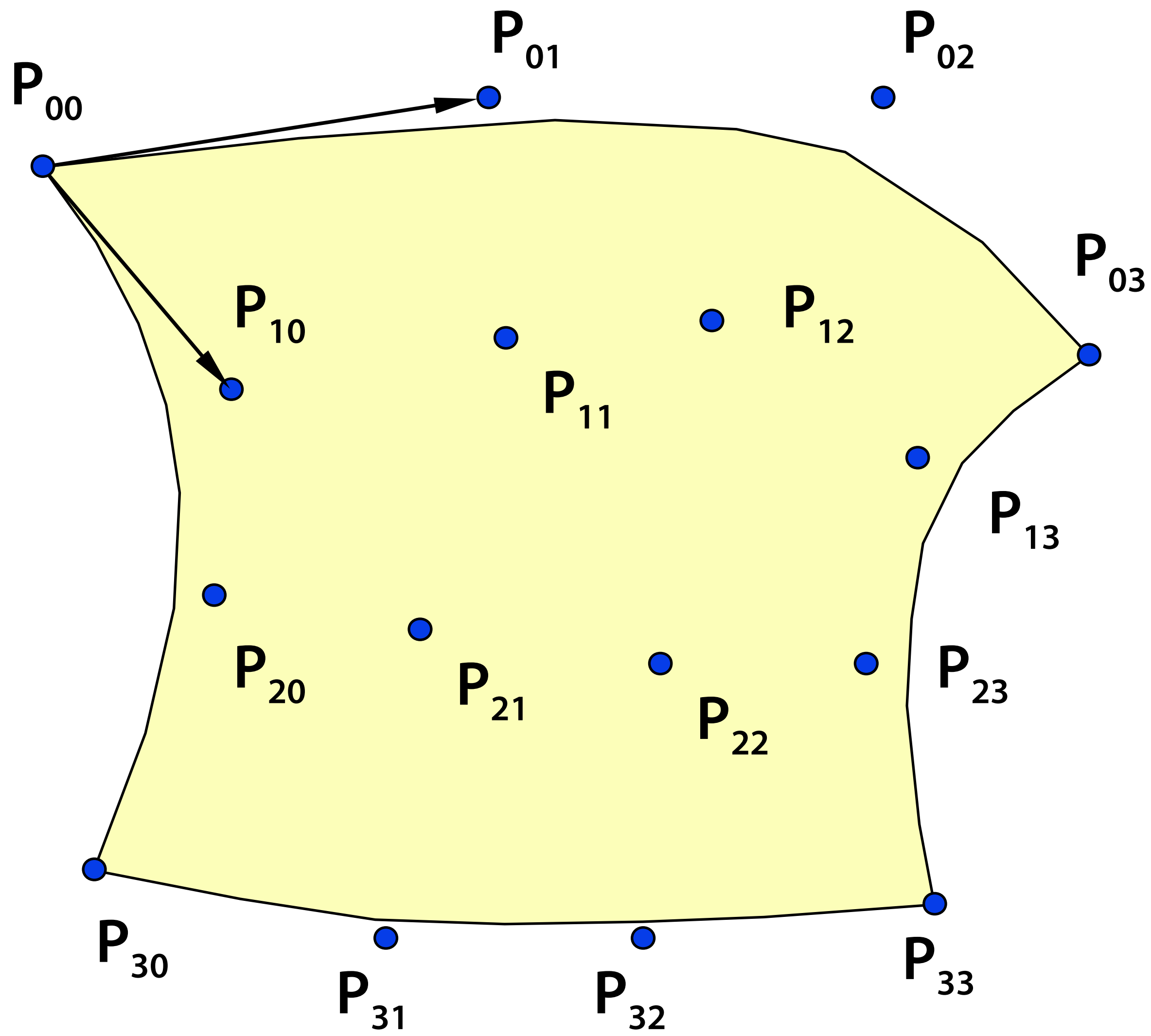
$$\mathbf{P} = [\mathbf{P}_{ij}]_{i,j=0}^{3}$$

$$\mathbf{P}(u, v) = \mathbf{B}(u)^T \cdot \mathbf{P} \cdot \mathbf{B}(v)$$

$$\mathbf{B}(t) = [B_k(t)]_{k=0}^{3}$$

$$B_k(t) = \binom{3}{k} t^k (1 - t)^{3-k}$$

Bernstein polynomials



Bernstein polynomials

$B_k(t)$ are **nonnegative cubic** polynomials

for $k = 0 \ldots 3$ and $0 \le t \le 1$

$\Sigma_k B_k(t) = 1$ for arbitrary t

- Cauchy's condition (affine invariance)

If $B_k(t)$ are used as weight coefficients (linear blending), result will be in a **convex hull** of input data (control polygon vertices in this case)

- $B_k(t)$ are blending coefficients of a convex combination

Ray vs. Bèzier patch intersection

After converting a bicubic Bèzier patch to implicit form we've got an **algebraic surface of the 18th degree!**

- **18th degree polynomial** to solve

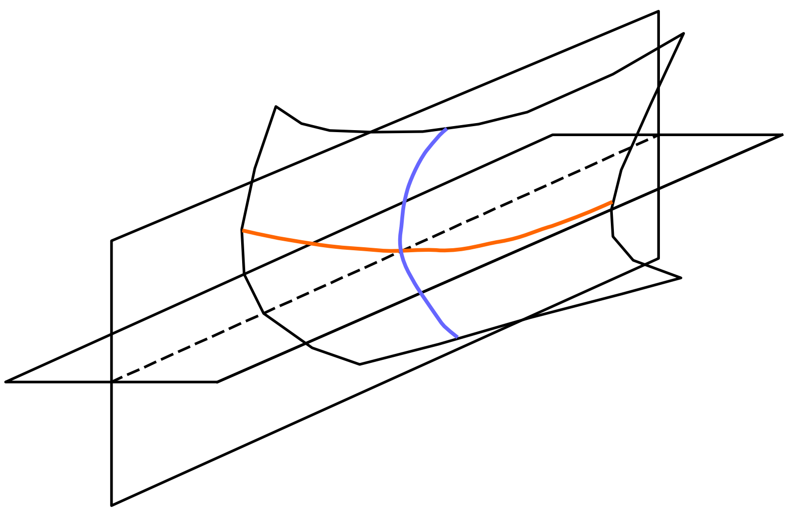
$\mathbf{B(u,v) = P_0 + t \cdot \vec{p}_1}$ is an algebraic system, three equations for three quantities: **t, u, v**

- can be solved using 3D **Newton iteration** (converges only in a relatively small interval)

Ray vs. Bèzier patch II

System of two algebraic equations for two quantities **u, v**

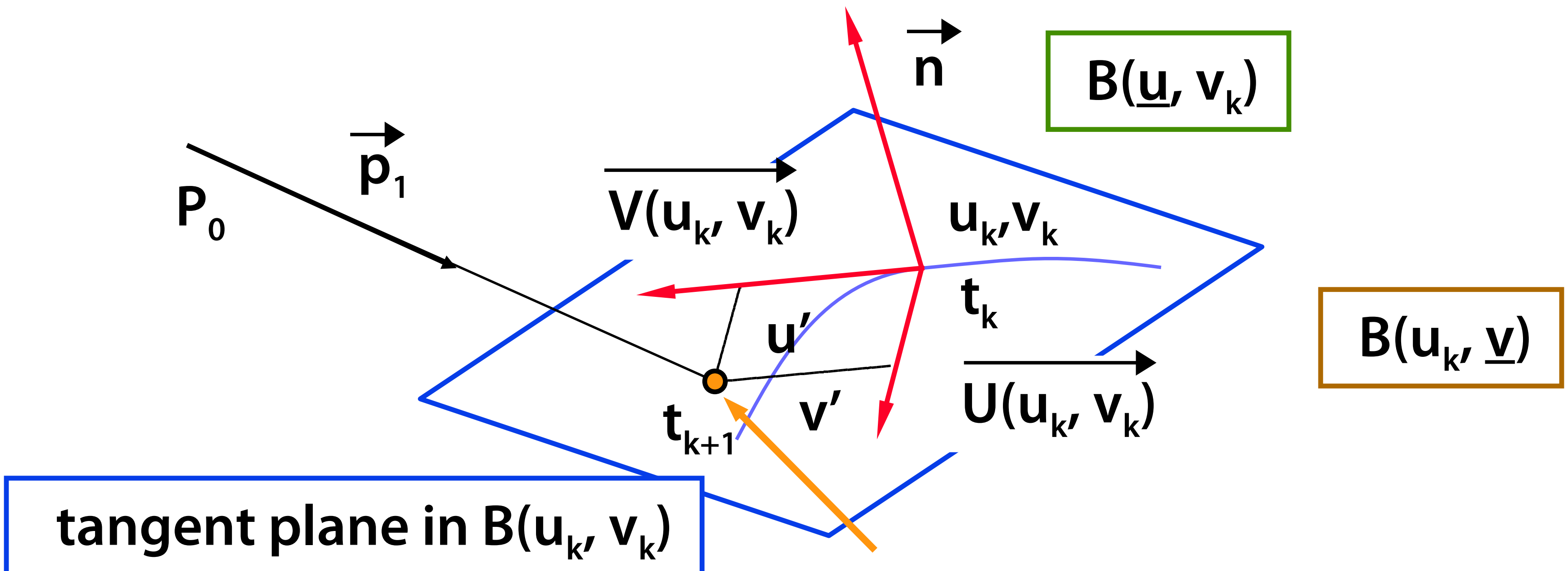
- **t** can be eliminated from the previous system
- let ray be **intersection of two planes**, planes vs. Bèzier patch are examined
- solution by a 2D **Newton iteration**



$$\mathbf{F_1(u, v) = 0}$$

$$\mathbf{F_2(u, v) = 0}$$

3D “Newtonian” iteration



Ray × tangent plane intersection: t_{k+1}, u', v'

$$V(u_k, v_k) = \frac{\partial B}{\partial v}(u_k, v_k)$$

$$U(u_k, v_k) = \frac{\partial B}{\partial u}(u_k, v_k)$$

$$u_{k+1} = u_k + u'$$
$$v_{k+1} = v_k + v'$$

Bèzier patch subdivision

One Bèzier patch **B(u,v)** **[$0 \le u, v \le 1$]** can be divided into four smaller ones

$B_{00}(u,v)$ [$0 \le u, v \le 1/2$]

$B_{01}(u,v)$ [$0 \le u \le 1/2, 1/2 \le v \le 1$]

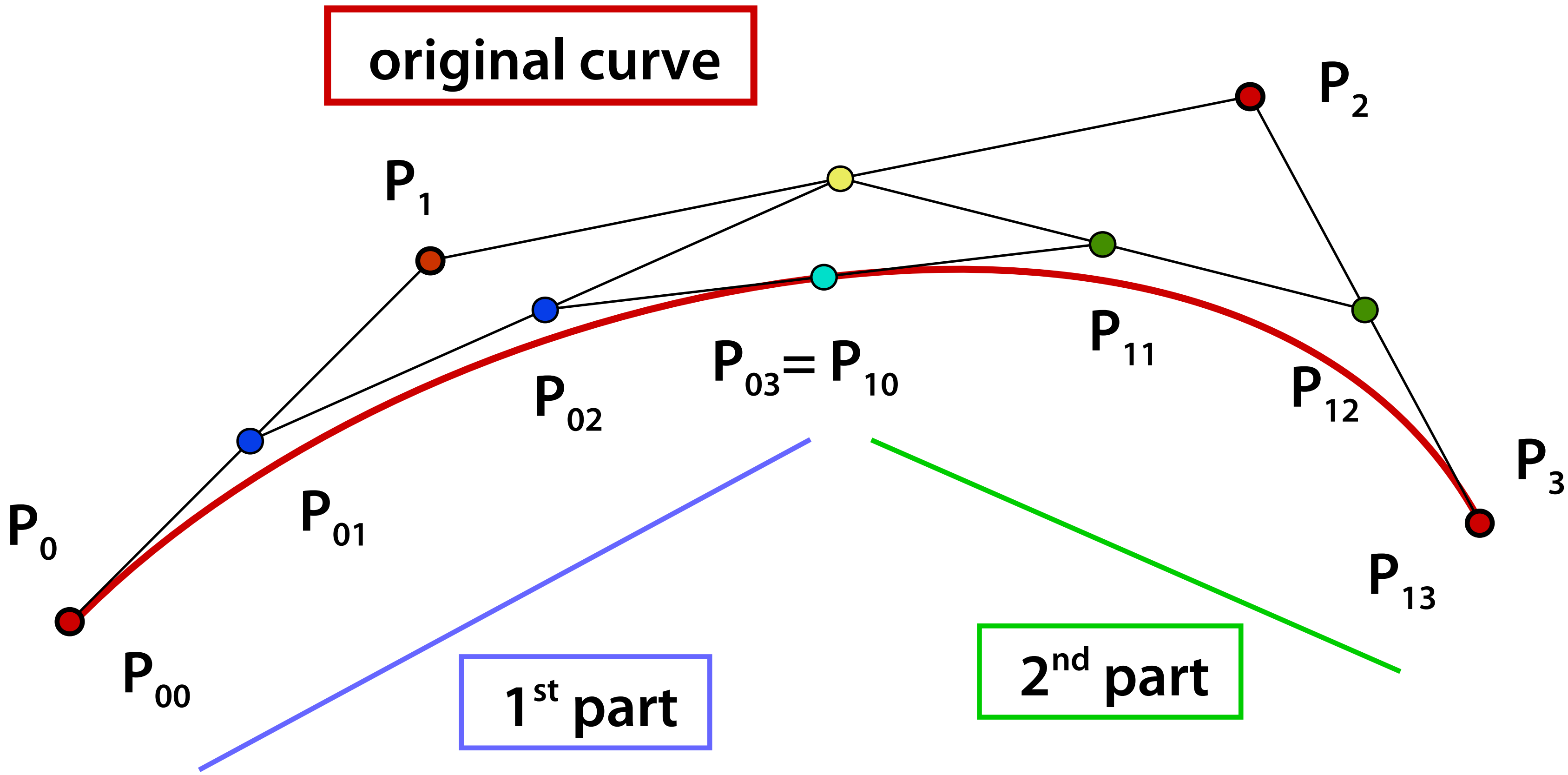
$B_{10}(u,v)$ [$1/2 \le u \le 1, 0 \le v \le 1/2$]

$B_{11}(u,v)$ [$1/2 \le u, v \le 1$]

New control points can be computed using recursive algorithm of **P. de Casteljau**

- only addition and dividing by two is used in this case!

De Casteljau subdivision (2D)



Algorithm ideas

We are looking for the **closest intersection** of the ray with the **set of Bèzier patches**

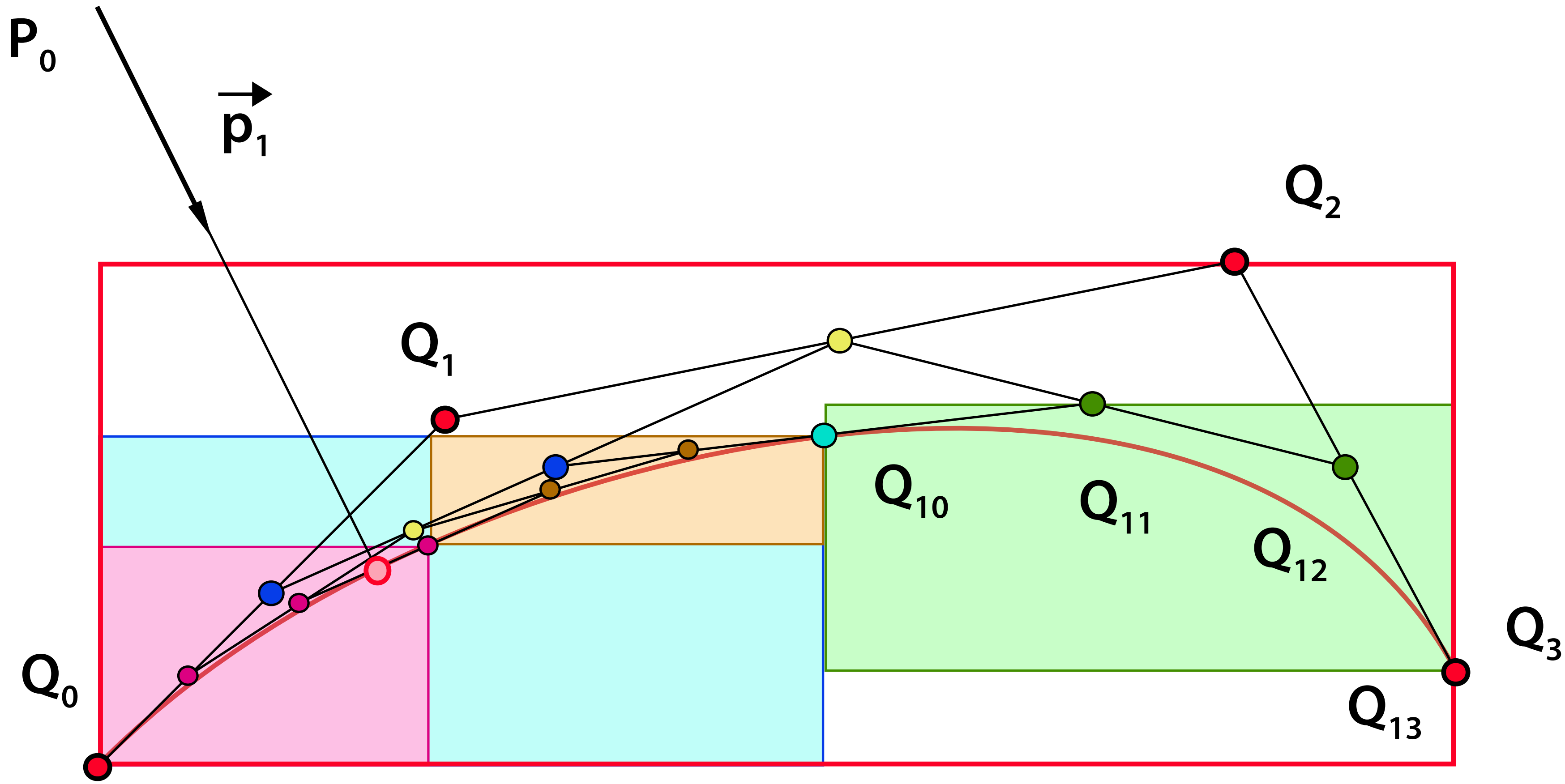
Every **Bèzier patch** lies inside a **convex hull** of its control points

- we will store **bounding box** for every patch (x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})

Relevant patch will be subdivided as long as it is intersected by a ray and too large to start the **Newtonian iteration** in it

- criterion = small **surface curvature**

Bounding boxes



Algorithm outline

❶ Intersected **bounding boxes** are maintained in the order of the intersection (**front-to-back**) … heap

❷ The closest bounding box is selected – if it has proper (low) curvature, the **Newtonian iteration** is started in it. If an actual intersection is found, it is placed into the result set

- the whole algorithm ends if the closest intersection is closer that the closest unprocessed patch (box)

❸ The closest patch with **high curvature** is divided into four parts, they are re-inserted into the list (heap)

- go back to ❷

Literature

A. Glassner: ***An Introduction to Ray Tracing***, Academic Press, London 1989, 99-102

J. Foley, A. van Dam, S. Feiner, J. Hughes: ***Computer Graphics, Principles and Practice***, 507-528