Computer Graphics

Shi-Min Hu

Tsinghua University

Bézier Curves and Surfaces

- **Parametric curves and surface: Some Concepts**
- **Bézier Cuvres: Concept and properties**
- **Bézier surfaces: Rectangular and Triangular**
- **Conversion of Rectangular and Triangular Bézier surfaces**

Parametric curves and surface

- **In Graphics, we usually design a scene and then generate realistic image by using rendering equation.**
- **It's necessary to introduce geometric modeling for scene design.**
- **How to represent 3D shapes (models) in computer?**

History of Geometric Modeling

- Surface Modeling（曲面造型）:
  - In 1962, Pierre Bézier, an engineer of French Renault Car company, propose a new kind of curve representation, and finally developed a system UNISURF for car surface design in 1972.

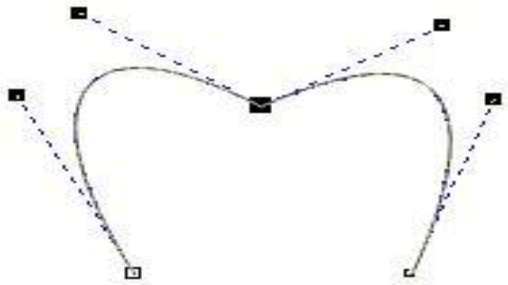
Bezier Curve / Pierre Bézier

<Back to Last Page> <Full Glossary>

Definition: A Bézier curve is a curved line or path defined by mathematical equations. It was named after Pierre Bézier, a French mathematician and engineer who developed this method of computer drawing in the late 1960s while working for the car manufacturer Renault. Most graphics software includes a pen tool for drawing paths with Bézier curves. (Continued below...)

The most basic Bézier curve is made up of two end points and control handles attached to each node. The control handles define the shape of the curve on either side of the common node. Drawing Bézier curves may seem baffling at first; it's something that requires some study and practice to grasp the geometry involved. But once mastered, Bezier curves are a wonderful way to draw!

Pierre Bézier was born September 1, 1910 and died November 25, 1999 at the age of 89. In 1985 he was recognized by ACM SIGGRAPH with a 'Steven A. Coons' award for his lifetime contribution to computer graphics and interactive techniques.



A Bezier curve with three nodes. The center node is selected and the control handles are visible.

- Solid Modeling（实体造型）:
  - In1973, Ian Braid of Cambridge University developed a solid modeling system for designing engineering parts.
  - Ian Braid presented in his dissertation “Designing with volumes”, this work being demonstrated with the BUILD-1 system.

SolidModeling.org

The website for the Solid Modeling Association.

Ian Braid, Alan Grayer and Charles Lang

The 2008 Pierre B閟ier Award Recipients

This group of three people made many fundamental contributions to practical solid modelling, and their work has had a profound influence on today's commercial solid modelling systems. They commenced working together in the CAD Group at the Computer Laboratory, Cambridge University. The Group was set up by Charles Lang under Prof Maurice Wilkes' direction in 1965 to undertake research on tools for building mechanical CAD/CAM systems, with an emphasis on software system components, computer graphics and computational geometry. Initial experiments in solid modelling were made in 1969. Also in 1969 Ian Braid joined the Group where, under Charles Lang's supervision, he developed the BUILD boundary representation modeller, the most advanced such system of its day. Whereas other systems used faceting to avoid the problems of calculating intersections between non-planar surfaces, the BUILD team tackled such problems head-on. Ian was awarded his PhD in 1973. Alan Grayer joined the group in 1971 and, also under Charles' supervision, developed algorithms for the automatic machining of prismatic parts modelled in BUILD. These were machined on a model making machine, built by the Group in 1971 following an inspirational visit to B閟ier at Renault in Paris. Alan was awarded his PhD in 1977. Ian then developed a completely new solid modeller, BUILD 2, which was a significant advance as it made a clear separation of geometry and topology in both its data structures and algorithms. This made it possible to implement generalised boolean operations and to systematically extend the geometric coverage and the functionality of the modeller with operations such as blending. Subsequently other PhD theses supervised by Ian and based on the BUILD modellers included Dimensions and Tolerances (Hillyard 1978), Feature Recognition (Kyprianou 1980), Automatic 2D and 3D Mesh Generation (W鯽denweber 1982) and Surface Intersections (Solomon 1986). These theses were some of the earliest

How to represent a curve?

- **There are three major types of object representation:**
  - Explicit representation: the explicit form of a curve in 2D gives the value of one variable, the dependent variable, in terms of the other, the independent variable. In x, y space, we may write

$$y = f(x)$$

  - For the line, we usually write $y = mx + h$

– Implicit representation: In two dimensions, an implicit curve can be represented by the equation

$$f(x, y) = 0$$

– For the line

$$ax + by + c = 0$$

– For the circle

$$x^2 + y^2 - r^2 = 0$$

- Parametric form: The parametric form of a curve expresses the value of each spatial variable for points on the curve in terms of an independent variable t, the parameter.
- In 3D, we have three explicit functions

$$x = x(t)$$

$$y = y(t)$$

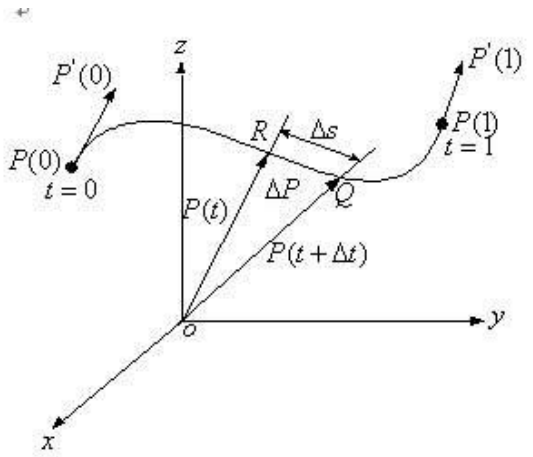
$$z = z(t)$$

– One of the advantages of the parametric form is that it is the same in two and three dimensions. In the former case, we simply drop the equation for z.（容易推广到高维）

– A useful representation of the parametric form is to visualize the locus of points (点的轨迹)

$$P(t) = [x(t), y(t), z(t)]^T$$

being drawn as t varies.



– We can think of the derivative

$$\frac{dP(t)}{dt} = \begin{bmatrix} \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \\ \frac{dz(t)}{dt} \end{bmatrix}$$

As the velocity with which the curve is traced out and points in direction tangent to the curve.

Parametric polynomial curves

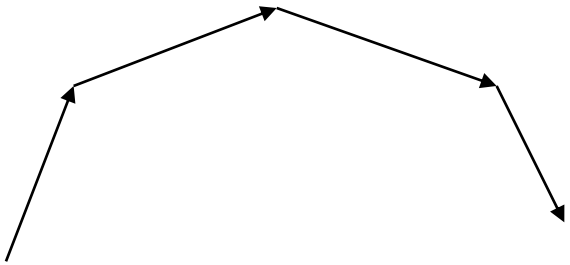
- Parametric curves are not unique. A given curve or surface can be represented in many ways, but we shall find that parametric forms in which the functions are polynomials in t.
- A polynomial parametric curve of degree 3 is of form

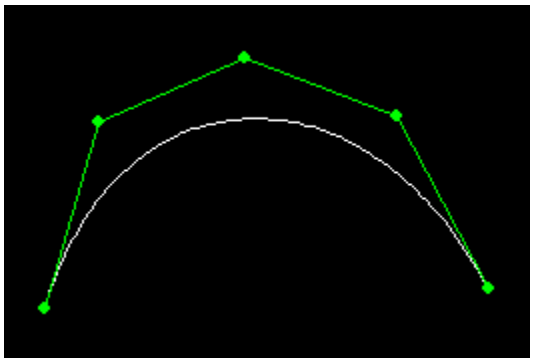
$$P(t) = at^3 + bt^2 + ct + d$$

- It was known as **Ferguson** curve, and was used for airplane design in earlier in USA.

- But Ferguson curve is not straightforward, even the coefficients a, b, c, d are given, it's still hard to imagine the shape of the curve.

Bézier Cuvres: Concept and properties





- Pierre Bézier proposed a method to represent a curve in terms of connected vectors.

$$V(t) = \sum_{i=0}^{n} f_{i,n}(t) A_i$$

$$f_{i,n}(t) = \begin{cases} 1, & i = 0, \\ \dfrac{(-t)^i}{(i-1)!} \dfrac{d^{i-1}}{dt^{i-1}} \left(\dfrac{(1-t)^{n-1} - 1}{t} \right) \end{cases}$$

- Curve can be designed interactively?
  - Curve are generated according to the control net (those connected vectors)
  - We may change vectors to modify the curve
  - See demo: curvesystem.exe

- However, such a definition is unintelligible
- 1972, Forrest published his famous paper in Computer Aided Design journal,
  - he pointed out that, Bézier curve can be defined in terms of points with help of Bernstein Polynomials.
  - Liang Youdong, Chang Gengzhe, Liu Dingyuan
  - P Bézier was passed away in 1999, CAGD published a Special issue for him in 2001

CAGD special issue

ELSEVIER

Computer Aided Geometric Design 18 (2001) 667–671

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/comaid

Conversion between triangular and rectangular Bézier patches

Shi-Min Hu

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China

Received September 2000; revised May 2001

In memory of P. Bézier

Abstract

This paper presents an explicit formula that converts a triangular Bézier patch of degree n to a degenerate rectangular Bézier patch of degree $n \times n$ by reparametrization. Based on this formula, we develop a method for approximating a degenerate rectangular Bézier patch by three nondegenerate Bézier patches; more patches can be introduced by subdivision to meet a user-specified error tolerance.

Keywords: Bézier surfaces; Degree elevation; Subdivision; Conversion

Definition

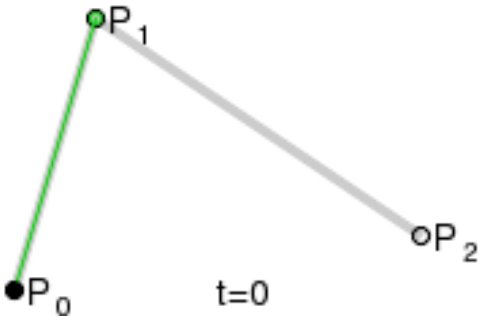
- Given control points $P_0, P_1, \ldots P_n$, Bezier curve can be defined as:

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), \qquad t \in [0,1]$$

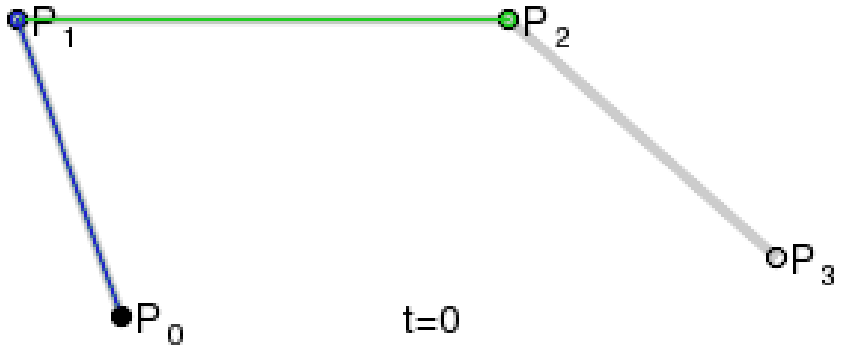
Where $B_{in}(t)$ is *i*-th *Bernstein polynomial* of degree n

$$B_{i,n}(t) = C_n^i t^i (1-t)^{n-i} = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

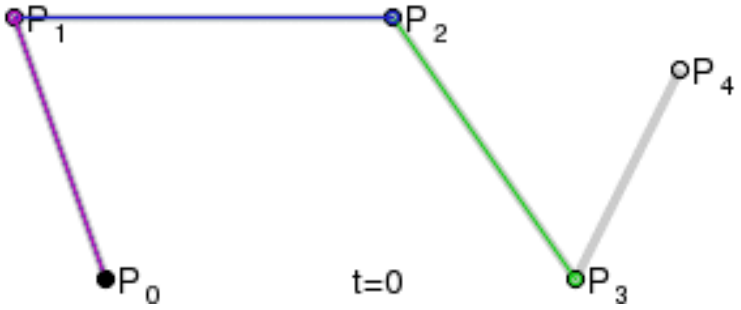
$$(i = 0,1,\ldots n)$$



Degree two



Degree three



Degree four

Three Bezier curves

Property of *Bernstein polynomial*

- Non-negative (非负)

$$B_{i,n}(t) = \begin{cases} = 0 & t = 0,1 \\ > 0 & t \in (0,1), i = 1,2,\cdots,n-1; \end{cases}$$

- End point

$$B_{i,n}(0) = \begin{cases} 1 & (i = 0) \\ 0 & otherswise \end{cases}$$

$$B_{i,n}(1) = \begin{cases} 1 & (i = n) \\ 0 & otherswise \end{cases}$$

- Unity

$$\sum_{i=0}^{n} B_{i,n}(t) \equiv 1 \qquad t \in (0,1)$$

  – Proof: According to *Binomial Theorem(二项式定理), we have*

$$\sum_{i=0}^{n} B_{i,n}(t) = \sum_{i=0}^{n} C_n^i t^i (1-t)^{n-i} = [(1-t)+t]^n \equiv 1$$

- Symmetry (对称)

$$B_{i,n}(1-t) = B_{n-i,n}(t)$$

– Proof:

$$B_{n-i,n}(t) = C_n^{n-i}[1-(1-t)]^{n-(n-i)} \cdot (1-t)^{n-i}$$

$$= C_n^i t^i (1-t)^{n-i} = B_{i,n}(1-t)$$

- Recursive (递归性)

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t),$$
$$(i = 0,1,\ldots,n)$$

  – This means that *Bernstein Polynomial* of degree n is a linear combination of two *Bernstein Polynomial* of degree n-1.

$$\begin{aligned} B_{i,n}(t) &= C_n^i t^i (1-t)^{n-i} = (C_{n-1}^i + C_{n-1}^{i-1}) t^i (1-t)^{n-i} \\ &= (1-t)C_{n-1}^i t^i (1-t)^{(n-1)-i} + tC_{n-1}^{i-1} t^{i-1} (1-t)^{(n-1)-(i-1)} \\ &= (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t) \end{aligned}$$

- Derivation (导数)

$$B'_{i,n}(t) = n[B_{i-1,n-1}(t) - B_{i,n-1}(t)],$$

$$i = 0,1,\cdots,n;$$

- Maximum
  - $B_{i,n}(t)$ has a unique local maximum on the interval [0,1] at x = i/n

- degree raising formula

$$(1-t)B_{i,n}(t) = (1-\frac{i}{n+1})B_{i,n+1}(t)$$

$$tB_{i,n}(t) = \frac{i+1}{n+1}B_{i+1,n+1}(t)$$

$$B_{i,n}(t) = (1-\frac{i}{n+1})B_{i,n+1}(t) + \frac{i+1}{n+1}B_{i+1,n+1}(t)$$

- Integral

$$\int_0^1 B_{i,n}(t) = \frac{1}{n+1}$$

Property of Bezier curve

- End point properties
  - Position of end point
    - According to the end position's property of *Bernstein polynomial*, We have

$$P(0) = P_0, \quad P(1) = P_n,$$

    - So the start point and end point of Bezier curve coincide with start point and end point of the control polygon.

– Tangent Vector
  - Since

$$P'(t) = n\sum_{i=0}^{n-1} P_i[B_{i-1,n-1}(t) - B_{i,n-1}(t)]$$

  - We have

$$P'(0) = n(P_1 - P_0), \quad P'(1) = n(P_n - P_{n-1})$$

  - This implies the tangent vector of the curve at the start and end points is the same with the direction of the first and last edges of the control polygon

– Second Derivative

$$P''(t) = n(n-1)\sum_{i=0}^{n-2}(P_{i+2} - 2P_{i+1} + P_i)B_{i,n-2}(t)$$

So we have

$$P''(0) = n(n-1)(P_2 - 2P_1 + P_0)$$

$$P''(1) = n(n-1)(P_n - 2P_{n-1} + P_{n-2})$$

By the Curvature formula, we have

$$k(0) = \frac{n-1}{n} \cdot \frac{|(P_1 - P_0) \times (P_2 - P_1)|}{|P_1 - P_0|^3} \qquad k(1) = \frac{n-1}{n} \cdot \frac{|(P_{n-1} - P_{n-2}) \times (P_n - P_{n-1})|}{|P_n - P_{n-1}|^3}$$

– Difference form of k-th Derivative

$$P^k(t) = \frac{n!}{(n-k)!}\sum_{i=0}^{n-k}\Delta^k P_i B_{i,n-k}(t) \qquad t \in [0,1]$$

- The high-order forward difference vector is recursively defined by low-order forward difference vector:

$$\Delta^0 P_i = P_i$$

$$\Delta^k P_i = \Delta^{k-1} P_{i+1} - \Delta^{k-1} P_i$$

- Symmetry(对称)
  - The curve with control points $P_i^* = P_{n-i}, (i = 0,1,...,n)$ remain the shape of the curve P(t), but with opposite direction.

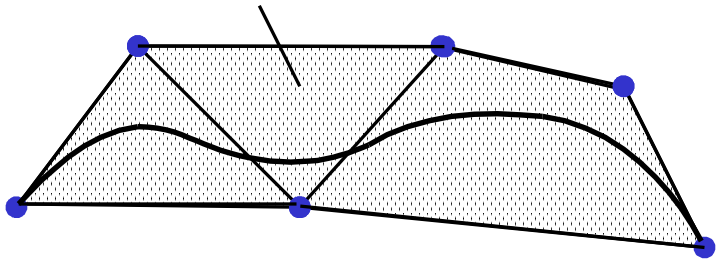
$$C^*(t) = \sum_{i=0}^{n} P_i^* B_{i,n}(t) = \sum_{i=0}^{n} P_{n-i} B_{i,n}(t) = \sum_{i=0}^{n} P_{n-i} B_{n-i,n}(1-t)$$

$$= \sum_{i=0}^{n} P_i B_{i,n}(1-t), \qquad t \in [0,1]$$

- Convex hull (凸包) Property
  - By $\sum_{i=0}^{n} B_{i,n}(t) \equiv 1$

    and

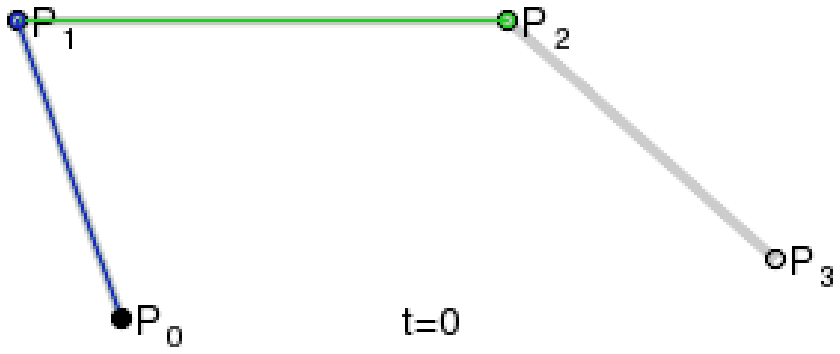
    $$0 \le B_{i,n}(t) \le 1 (0 \le t \le 1, i = 0,1,\cdots,n)$$
  - The curve P(t) is inside the convex hull of the control polygon



- Geometric invariance (几何不变性)
  - It means that some geometry property doesn't change with Coordinates varying.
  - The position and shape of Bezier curve are dependent on vertex P_i of control polygon, but not the Coordinates.

de Casteljau Algorithm

- In industry application, it's required to evaluate a point on the curve at parameter t.
- We didn't evaluate the value by the Bezier Curve Equation, but use a recursive algorithm, which is numerical stable.



- Please note

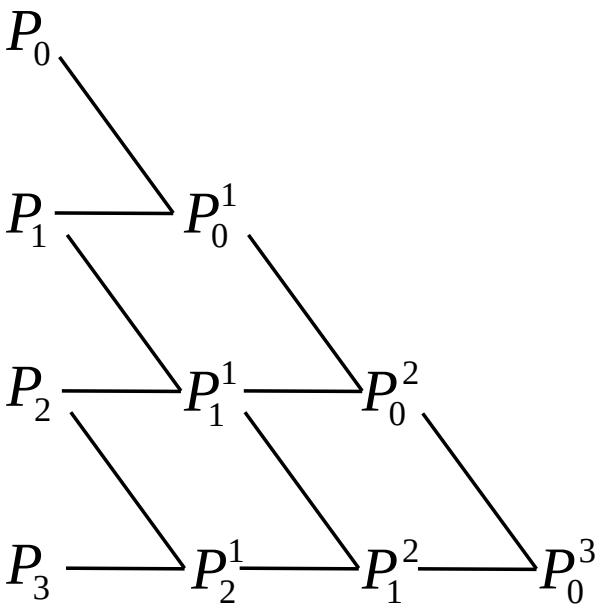
$$P(t)=\sum_{i=0}^{n}P_iB_{i,n}(t)=\sum_{i=0}^{n}P_i\left[(1-t)B_{i,n-1}(t)+tB_{i-1,n-1}(t)\right]$$

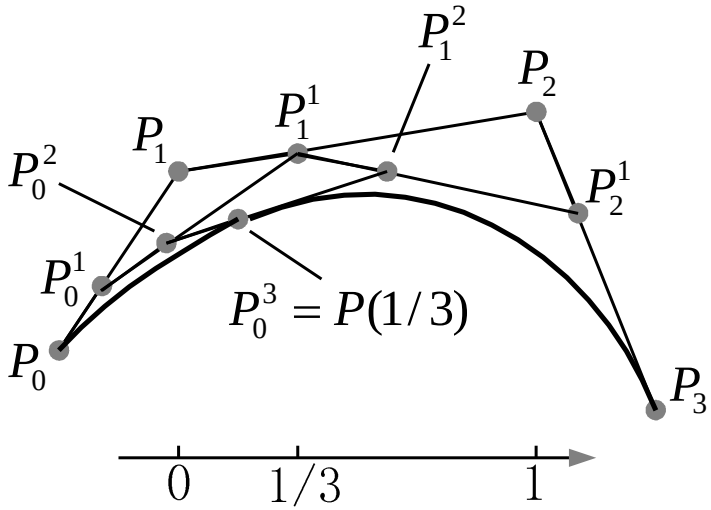
$$=\sum_{i=0}^{n}\left[(1-t)P_i+tP_{i+1}\right]B_{i,n-1}(t)$$

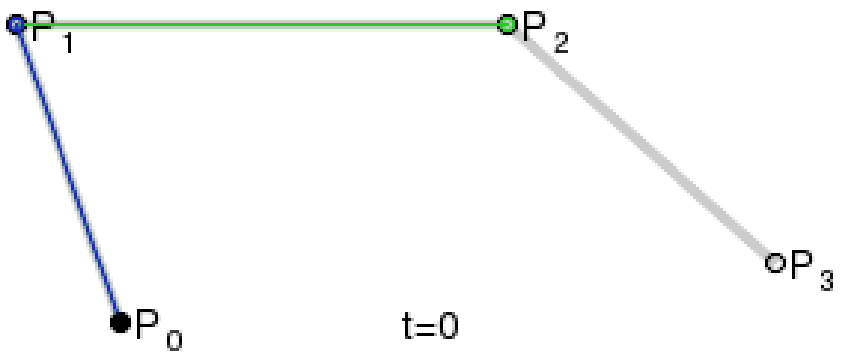
- We have the recursive formula for evaluation of Bézier curve by $P(t) = P_0^n$

$$P_i^k = \begin{cases} P_i & k = 0 \\ (1-t)P_i^{k-1} + tP_{i+1}^{k-1} & k = 1, 2, \ldots, n, \\ & i = 0, 1, \ldots, n-k \end{cases}$$

- When n = 3, the recursive procedure is illustrated by the following figure:




P_1^2
P_2
P_1^1
P_1
P_0^2
P_2^1
P_0^1
$P_0^3 = P(1/3)$
P_0
P_3
0
1/3
1


P 1
P 2
P 3
P 0
t=0

Geometric continuity

- In CAD applications, it's not encouraged to design a curve by high degree Bézier curve.
- It's common to use lower degree Bézier curves with smooth connections.
- Can we use traditional concept of continuity?

$$\Phi(t)=\begin{cases} V_0+\dfrac{V_1-V_0}{3}t, & 0\le t\le 1 \\ V_0+\dfrac{V_1-V_0}{3}+(t-1)\dfrac{2(V_1-V_0)}{3}, & 1\le t\le 2 \end{cases}$$

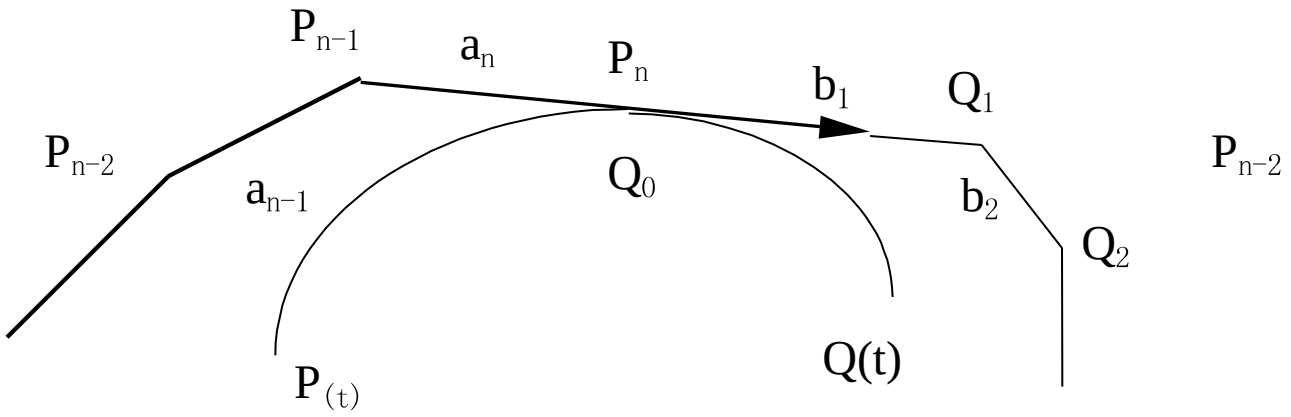
- Please note

$$\Phi'(1^-) = \frac{1}{3}(V_1 - V_0) \qquad \Phi'(1^+) = \frac{2}{3}(V_1 - V_0)$$

- This means the function is not continuous.
- But the function is actually a straight line.
- Such a fact implies traditional concept of continuous is not suitable for describing smoothness of shapes in CAD and Graphics.
- This is why we use “Geometric continuity”
- Liu （刘鼎元）and Liang (梁友栋)

- Given two Bezier Curves defined by control points $P_i(i=0,1,\cdots,n)$ and $Q_j(j=0,1,\cdots,m)$ respectively
- Two curves share an end point, and :

$$a_i = P_i - P_{i-1}, b_j = Q_j - Q_{j-1}$$



– G^0 **continuous** $P_n = Q_0$

– G^1 continuous

$$Q_0 - Q_1 = \alpha(P_n - P_{n-1})$$

and P_{n-1}, $P_n = Q_0$, Q_1 are collinear,

– G^2 continuous (curvature continuous)

$$Q_2 = \left(\alpha^2 + 2\alpha + \frac{\beta}{n-1} + 1\right)P_n - \left(2\alpha^2 + 2\alpha + \frac{\beta}{n-1}\right)P_{n-1} + \alpha^2 P_{n-2}$$

$$Q''(0) = \alpha^2 P''(1) + \beta P'(1)$$

Degree raising/elevation

- Degree raising means that adding control points to raise the degree of the Bézier curves, but the shape and direction of the curve remain unchanged.
  - Degree raising increases the flexibility of shape control.
  - After degree raising, control points are changed.
  - How to raising the degree of a polynomial? Bézier curves?

- Please note

$$P(t)=\sum_{i=0}^{n}P_iB_{i,n}(t)=\sum_{i=0}^{n}P_i\left((1-t)+t\right)B_{i,n}(t)$$

$$=\sum_{i=0}^{n+1}\left(\frac{n+1-i}{n+1}P_i+\frac{i}{n+1}P_{i-1}\right)B_{i,n+1}(t)$$

- We have the degree raising formula

$$P_i^*=\frac{i}{n+1}P_{i-1}+\left(1-\frac{i}{n+1}\right)P_i \qquad (i=0,1,\cdots,n+1)$$

- The above formula illustrates
  - The new control points are linear combination of the old control points with Coefficient i/(n+1).
  - The new control polygon is inside the convex hull of old control polygon.
  - The new control polygon is nearer to the curve.
- Demo: curve-system

Degree reduction

- Degree reduction is the opposite of degree raising
  - Can we reduce the degree of a polynomial without change of shapes?
  - Degree reduction is to find a curve defined by new control points with minimum error

$$P_i^* (i = 0,1,\cdots,n-1)$$

– Suppose P_i is result of degree raising P_i^*:

$$P_i = \frac{n-i}{n} P_i^* + \frac{i}{n} P_{i-1}^*$$

– We can get two recursive formula

$$P_i^{\#} = \frac{nP_i - iP_{i-1}^{\#}}{n-i} \qquad i = 0,1,\cdots,n-1$$

$$P_{i-1}^* = \frac{nP_i - (n-i)P_i^*}{i} \qquad i = n,n-1,\cdots,1$$

- Then we have two kinds of degree reduction schemes
  - Forrest (1972)

$$\hat{P}_i = \begin{cases} P_i^{\#}, & i = 0,1,\cdots,\left[\frac{n-1}{2}\right] \\ P_i^{*}, & i = \left[\frac{n-1}{2}\right]+1,\cdots,n-1 \end{cases}$$

  - Farin (1983)

$$\hat{P}_i = (1-\frac{i}{n-1})P_i^{\#} + \frac{i}{n-1}P_i^{*}$$

- Reference for accurate degree reduction
  - M. A. Watkins and A. J. Worsey, Degree reduction of Bézier curves, Computer Aided Design, **20**(7), 1988, 398-405
  - 胡事民、孙家广、金通光等，Approximate degree reduction of Bezier curves, Tsinghua Science and Technology, No.2, 1998, 997-1000. (was reported in national CAGD conference, 1993)
  - 雍俊海、胡事民、孙家广等，Degree reduction of B-spline curves, Computer Aided Geometric Design, 2001, Vol. 13, NO. 2, 2001, 117-127.

Bezier surface

- **Rectangular Bézier Surface**
  - Suppose $P_{ij}(0,1,\cdots,n;\ j=0,1,\cdots,m)$ is $(n+1)\times(m+1)$ control points, a degree $m\times n$ rectangular Bezier surface can be defined in the form of tensor product

$$P(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{ij}B_{i,m}(u)B_{j,n}(v) \qquad u,v\in[0,1]$$

where $B_{i,m}(u)=C_m^i u^i(1-u)^{m-i}$ and $B_{i,n}(v)=C_n^j v^j(1-v)^{n-j}$ are ***Bernstein polynomial*.**

– Bézier Surfaces by matrix representation

$$P(u,v)=\left[B_{0,n}(u),B_{1,n}(u),\cdots,B_{m,n}(u)\right]\begin{bmatrix}P_{00} & P_{01} & \cdots & P_{0m}\\ P_{10} & P_{11} & \cdots & P_{1m}\\ \cdots & \cdots & \cdots & \cdots\\ P_{n0} & P_{n1} & \cdots & P_{nm}\end{bmatrix}\begin{bmatrix}B_{0,m}(v)\\ B_{1,m}(v)\\ \cdots\\ B_{n,m}(v)\end{bmatrix}$$

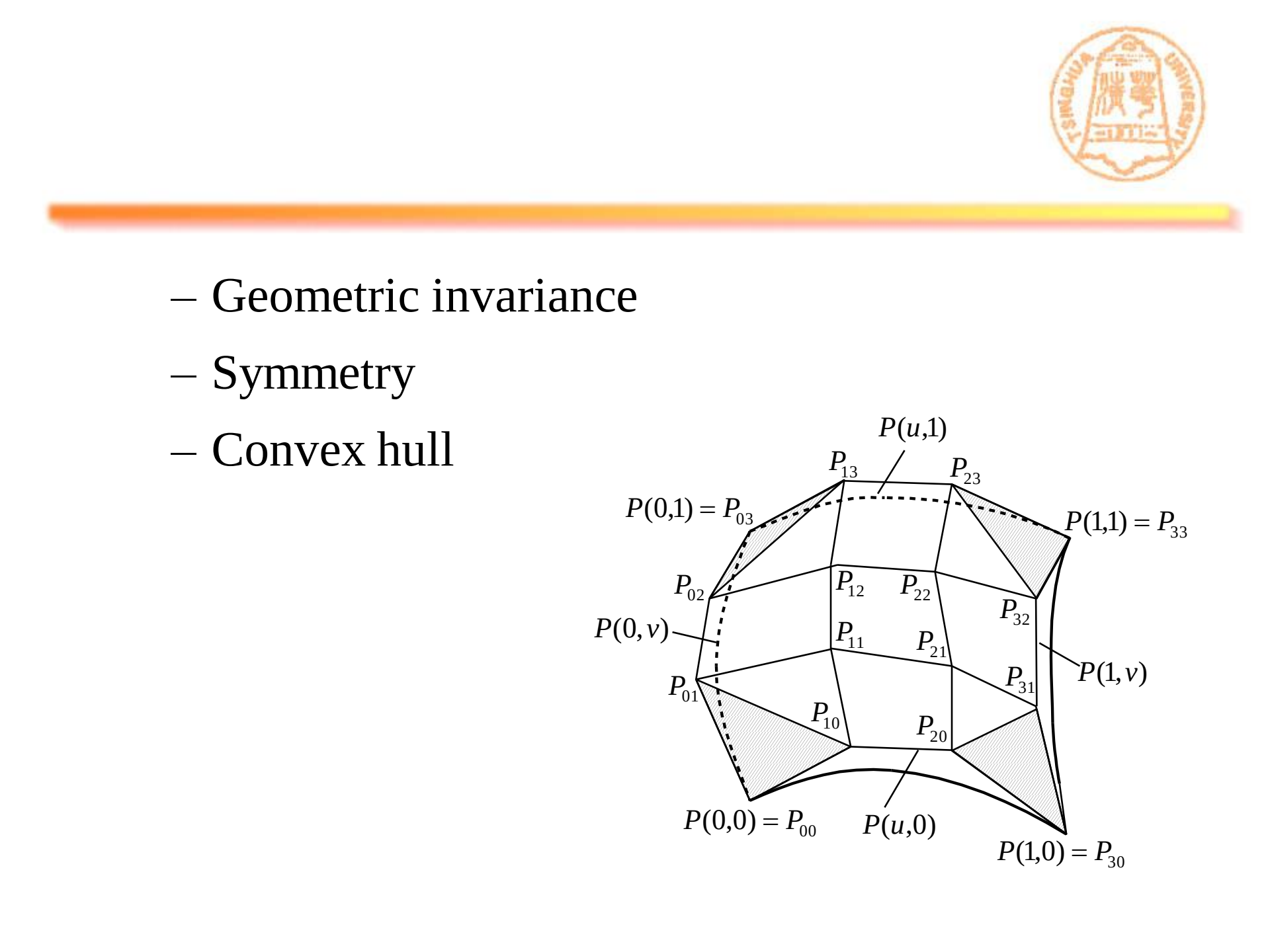
- Properties of rectangular
  - Bézier Surface hold similar properties of Bézier curves :
    - The four corners of control net are also the corners of the Bézier surface.

$$P(0,0) = P_{00} \quad P(1,0) = P_{m0} \quad P(0,1) = P_{0n} \quad P(1,1) = P_{mn}$$

    - The triangles

$$P_{00}P_{10}P_{10} \quad P_{on}P_{1n}P_{0,n-1} \quad P_{mn}P_{m,n-1}P_{m-1,n} \quad P_{m0}P_{m-1,0}P_{m1}$$

    give tangent plane at 4 corners.



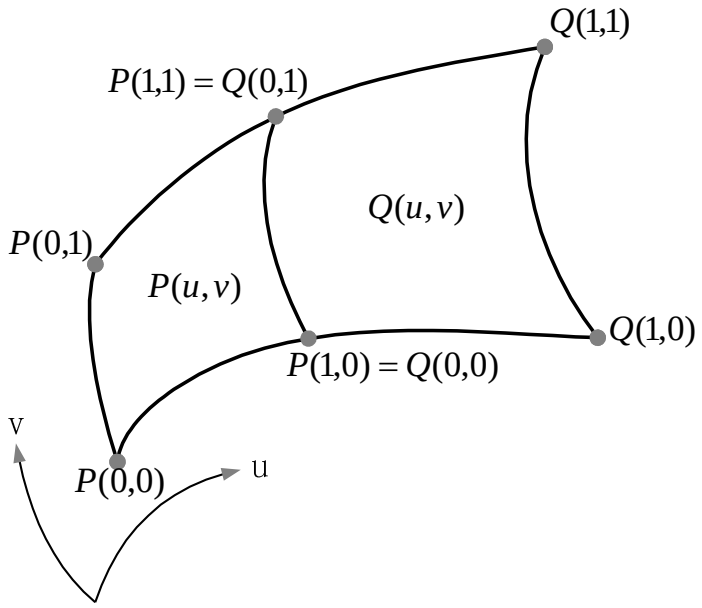
- Geometric invariance
- Symmetry
- Convex hull

- Geometric continuity
  - Given two degree m*n Bézier surfaces with control points P_{ij} and Q_{ij}:

$$P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} B_{i,m}(u) B_{j,n}(v)$$

$$Q(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} Q_{ij} B_{i,m}(u) B_{j,n}(v)$$

$$u, v \in [0,1]$$



– Conditions of G^0 continuous:

$$P(1,v) = Q(0,v)$$

i.e., $P_{ni} = Q_{0i}$, $(i = 0,1,\cdots,m)$

– Conditions of G^1 continuous:

$$Q_u(0,v) = \alpha(v)P_u(1,v) + \beta(v)P_v(1,v)$$

- de Casteljau algorithm
  - De Casteljau algorithm of Bézier curves can be extended for surfaces. Given control points $P_{ij}(i = 0,1,\cdots,m;\ j = 0,1,\cdots,n)$ and parameter (u,v). we have

$$P(u,v) = \sum_{i=0}^{m-k}\sum_{j=0}^{n-l} P_{i,j}^{k,l} B_{i,m}(u) B_{j,n}(v) = \cdots = P_{00}^{m.n}$$

$$u, v \in [0,1]$$

$P_{i,j}^{k,l}$ are defined by following recursive formulas

$$P_{i,j}^{k,l} = \begin{cases} P_{ij} & (k = l = 0) \\ (1-u)P_{ij}^{k-1,0} + uP_{i+1,j}^{k-1,0} & (k = 1,2,\cdots,m; l = 0) \\ (1-v)P_{0,j}^{m,l-1} + vP_{0,j+1}^{m,l-1} & (k = m, l = 1,2,\cdots,n) \end{cases}$$

Or

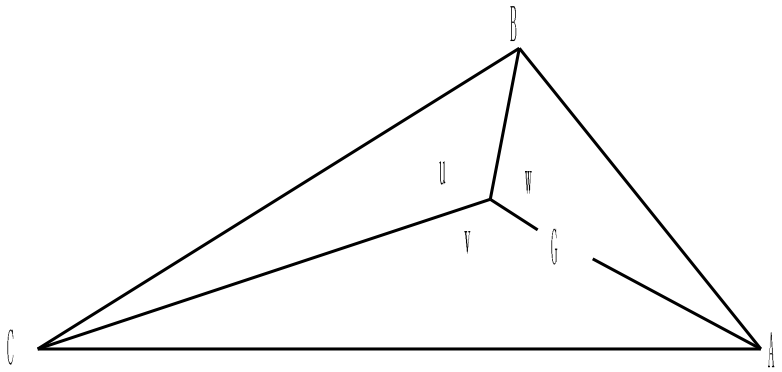
$$P_{ij}^{k,l} = \begin{cases} P_{ij} & (k = l = 0) \\ (1-v)P_{ij}^{0,l-1} + vP_{i,j+1}^{0,l-1} & (k = 0; l = 1,2,\cdots,n) \\ (1-u)P_{i0}^{k-1,n} + uP_{i+1,0}^{k-1,n} & (k = 1,2,\cdots,m; l = n) \end{cases}$$

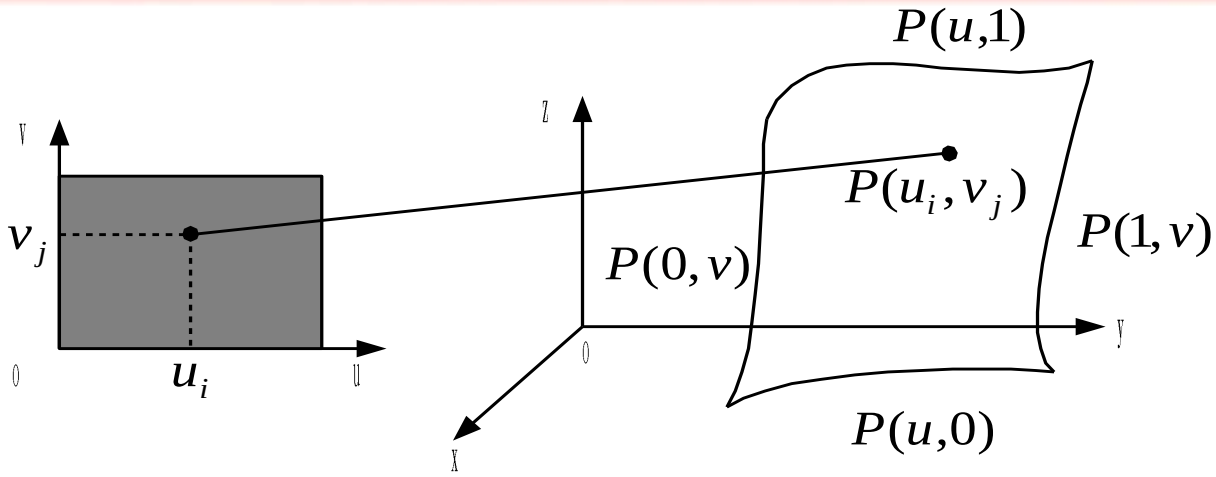
- Straightforward interpretation of de Casteljau algorithm

$$P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} B_{i,m}(u) B_{j,n}(v)$$

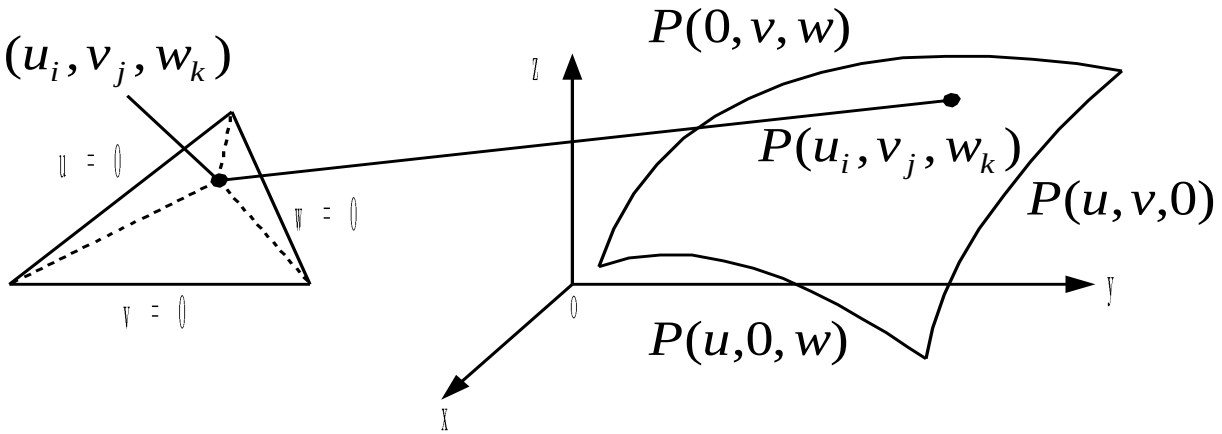
$$= \sum_{i=0}^{m}\left(\sum_{j=0}^{n} P_{ij} B_{j,n}(v) \right) B_{i,m}(u)$$

- **Triangular Bézier surfaces**
  - Triangular Bézier surfaces are defined over triangles, not squares.
  - Barycentric coordinates (u,v,w) are used for the definition of triangular Bézier surfaces .




P(u,1)
V
Z
v_j
$P(u_i, v_j)$
P(1,v)
P(0,v)
0
u_i
U
Y
0
X
P(u,0)

（a）矩 形 域 曲 面


(u_i, v_j, w_k)
P(0,v,w)
Z
u = 0
$P(u_i, v_j, w_k)$
P(u,v,0)
w = 0
v = 0
0
Y
P(u,0,w)
X

- Bernstein Base Function:

$$B_{i,j,k}(u,v,w) = \frac{n!}{i!\,j!k!}u^i v^j w^k \qquad u,v,w \in [0,1]$$

Where i+j+k=1 and i,j,k>=0.

– There are $\frac{1}{2}(n+1)(n+2)$ degree n Bernstein Base functions.

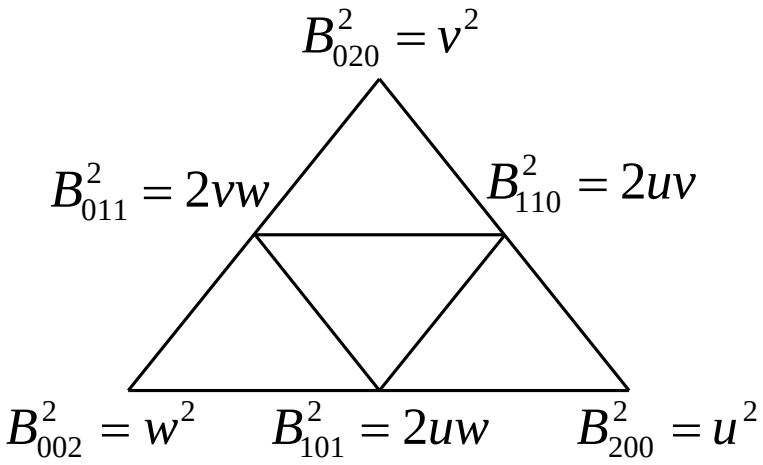


图 3.1.19 二次 Bernstein基的三角阵列

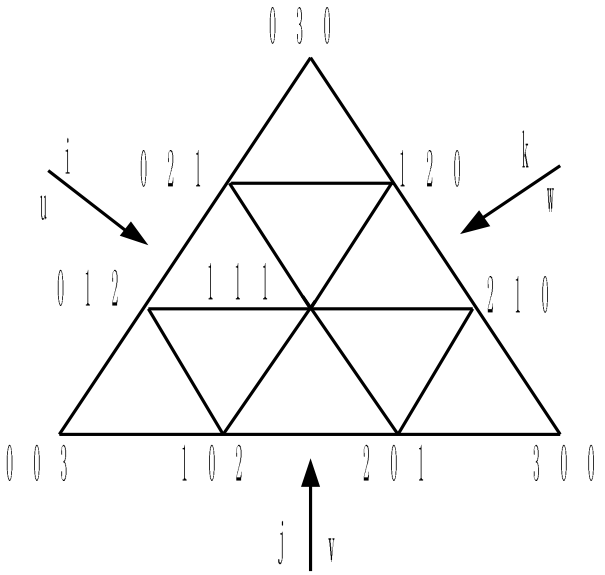


图 3.1.20 n=3时三角域各节点指标

– Properties of non-negative and unity

$$B_{i,j,k}^{n}(u,v,w) \geq 0$$

$$\sum_{i+j+k=n} B_{i,j,k}^{n}(u,v,w) = 1$$

– Recursive:

$$B_{i,j,k}^{n}(u,v,w) = uB_{i-1,j,k}^{n-1}(u,v,w) + vB_{i,j-1,k}^{n-1}(u,v,w) + wB_{i,j,k-1}^{n-1}(u,v,w)$$

- Definition of triangular Bézier surfaces

$$P(u,v,w) = \sum_{i+j+k=n} P_{i,j,k} B^n_{i,j,k}(u,v,w)$$

$$= \sum_{i=0}^{n} \sum_{j=0}^{n-i} P_{i,j,k} B^n_{i,j,k}(u,v,w)$$

$$u,v,w \geq 0, u+v+w = 1$$

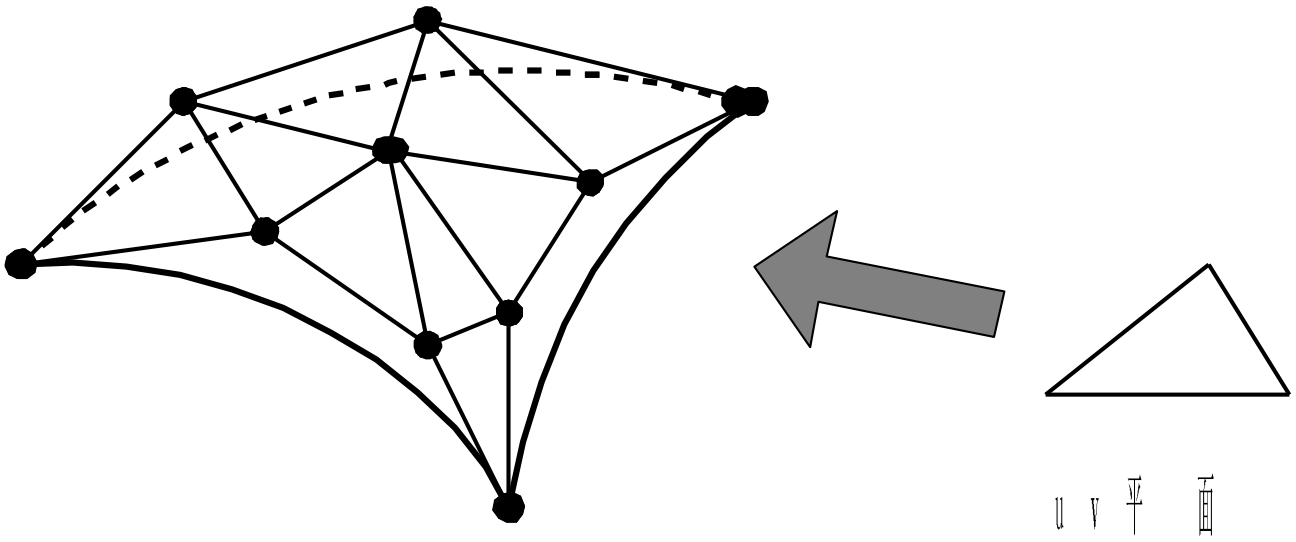
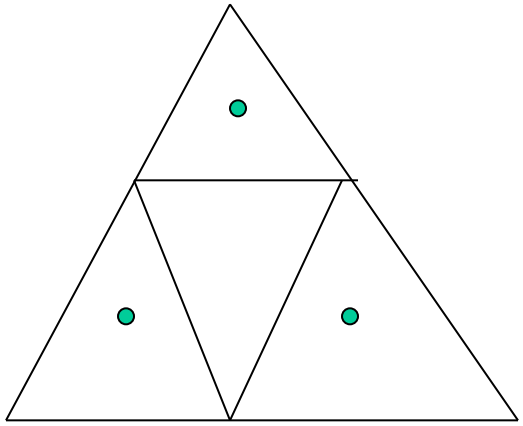
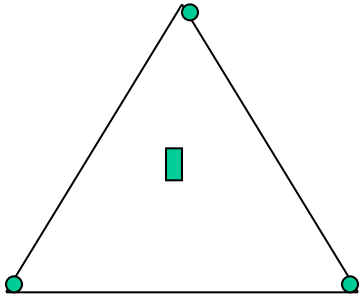


图 3.1.21 三边 Bezier曲面片

- de Casteljau Algorithm

$$P_{i,j,k} = uP_{i+1,j,k} + vP_{i,j+1,k} + wP_{i,j,k+1}$$



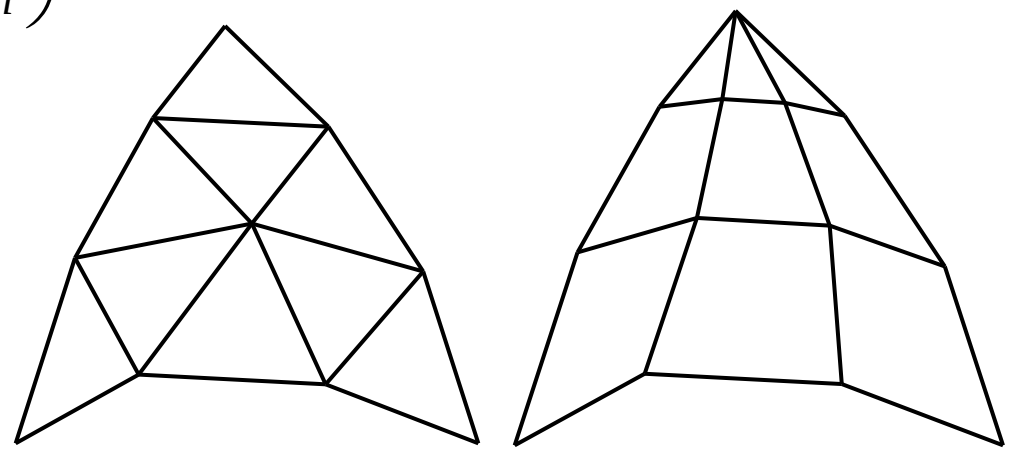


Conversion of Rectangular and Triangular Bézier surfaces

- Because the rectangular and the triangular patches use different base function, When they are used in the same CAD system, there will be some problem. So we introduce the conversion between them.
- A triangular surface can be represented as one degenerate rectangular surfaces or three non-degenerate rectangular surfaces

- To a degenerate rectangular surface

$$\begin{pmatrix} P_{i0} \\ P_{i1} \\ \vdots \\ P_{in} \end{pmatrix} = A_1 A_{2\cdots} A_i \begin{pmatrix} T_{i0} \\ T_{i1} \\ \vdots \\ T_{i,n-i} \end{pmatrix}, \quad i = 0,1,\cdots,n,$$

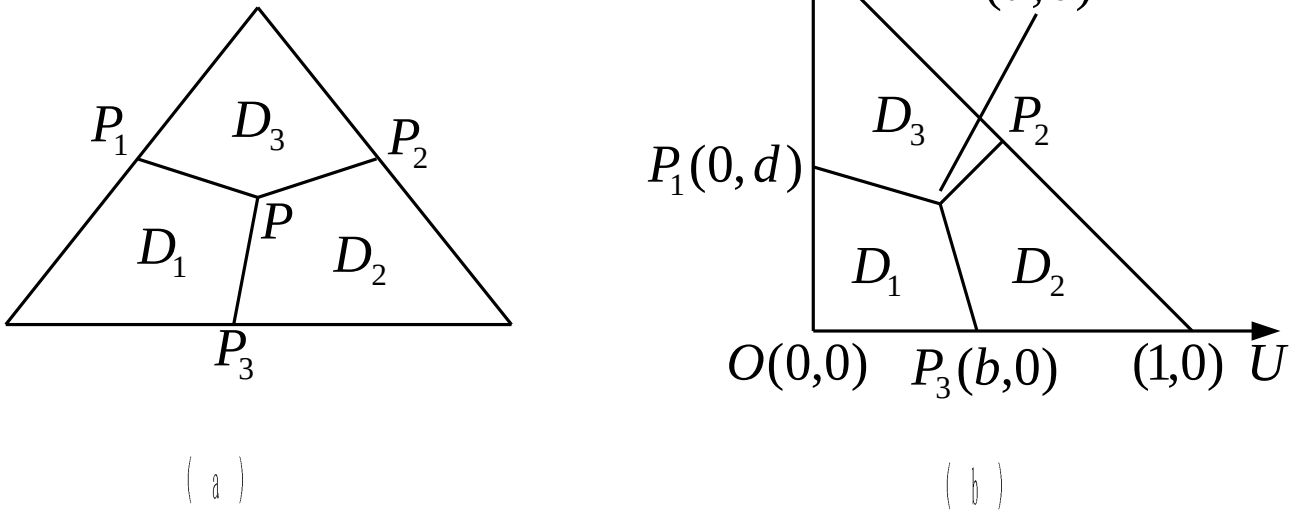


$A_i\ (i=0,1,\cdots n)$ is a degree raising operator:

$$A_k=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \dfrac{1}{n+1-k} & \dfrac{n-k}{n+1-k} & 0 & \cdots & 0 & 0 \\ 0 & \dfrac{2}{n+1-k} & \dfrac{n-k-1}{n+1-k} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \dfrac{n-k}{n+1-k} & \dfrac{1}{n+1-k} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}_{(k+1)\times k}$$

Shi-Min, Hu Conversion between triangular and rectangular Bezier patches, Computer Aided Geometric Design, 2001, 18(7), 667-671.

- To three rectangular surfaces
  - Domain decomposition



(a)　　(b)

- some operator
  - Identiy operator: $I : IT_{ij} = T_{ij}$
  - Shifting operator: $E_i : E_1 T_{ij} = T_{i+1,j}, E_2 T_{ij} = T_{i,j+1}$
  - Difference operator: $\Delta_i : \Delta_1 T_{ij} = T_{i+1,j} - T_{ij}, \Delta_2 T_{ij} = T_{i,j+1} - T_{ij}$
- With help of those operators, we can rewrite triangular Bézier surfaces as

$$T(u,v) = (uE_1 + vE_2 + (1-u-v)I)^n T_{00}$$
$$= (\Delta_1 u + \Delta_2 v + I)^n T_{00}$$

- The control points defined over D_1 can be obtained by

$$P_{ij} = \sum_{\substack{k=0 \\ k+l=j}}^{i} \sum_{l=0}^{n-i} \binom{i}{k} \frac{\binom{n-i}{l}}{\binom{n}{j}} Q_{kl}^{(i)} \quad 0 \leqslant i, j \leqslant n, \tag{5}$$

in which for $0 \leqslant i \leqslant n,\ 0 \leqslant k \leqslant i,\ 0 \leqslant l \leqslant n-i,$

$$Q_{kl}^{(i)} = (aE_1 + cE_2 + (1-a-c)I)^k (bE_1 + (1-b)I)^{i-k} (dE_2 + (1-d)I)^l T_{00} \tag{6}$$

CAGD 1996

ELSEVIER

Computer Aided Geometric Design 13 (1996) 219–226

COMPUTER AIDED GEOMETRIC DESIGN

Conversion of a triangular Bézier patch into three rectangular Bézier patches [1]

Shi-Min Hu

Department of Applied Mathematics, Zhejiang University, Hangzhou, 310027, People's Republic of China

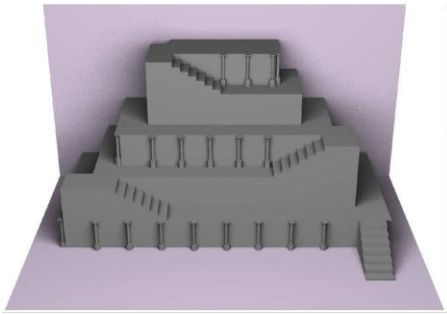
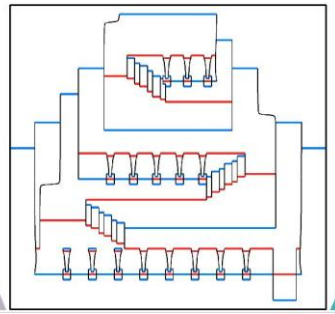
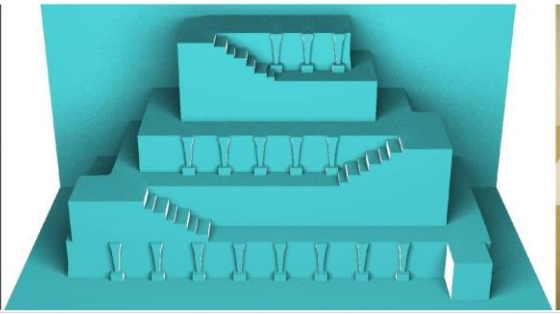
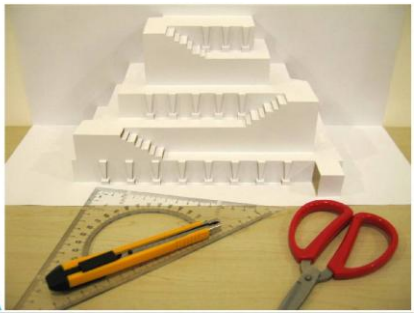
Received April 1994; revised February 1995

Abstract

In this paper, we give an explicit formula for converting a triangular Bézier patch into three non-degenerate rectangular Bézier patches of the same degree. The use of certain operators simplifies the formulation of such a decomposition. The formula yields a stable recursive algorithm for computing the control points of the rectangular patches. We also illustrate the formula with an example.

ACM SIGGRAPH

- **Popup: Automatic Paper Architectures from 3D Models (李先颖、沈超慧)**
  - Given a 3D architectural model with user-specified backdrop and ground (left), our algorithm automatically creates a paper architecture approximating the model (mid-right, with the planar layout in mid-left), which can be physically engineered and popped-up (right).

- **Finding Approximately Repeated Scene Elements for Image Editing**（程明明、张方略）
  - We propose a novel framework where simple user input in the form of 8 scribbles are used to guide detection and extraction of such repeated 9 elements

Thank You!