2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

第三节 参数曲线曲面的显示

多项式值的计算

- 显示多项式曲线的最简单方法就是计算出多项式上许多点的值，从而形成一条近似的折线
- 对于曲面，可以形成近似的三角网格或者四边网格
- 应用Horner方法计算多项式在一点的值
  - $p(u)=c_0+u(c_1+u(c_2+uc_3))$
  - 对于三次的情形，只需要三次乘法

有限差分

- 对于等间距的$\{u_k\}$，如下定义有限差分

$$\Delta^{(0)} p(u_k) = p(u_k)$$

$$\Delta^{(1)} p(u_k) = p(u_{k+1}) - p(u_k)$$

$$\Delta^{(m+1)} p(u_k) = \Delta^{(m)} p(u_{k+1}) - \Delta^{(m)} p(u_k)$$

- 对于n次多项式，n阶差分的结果是一个常数

建立有限差分表格

$p(u)=1+3u+2u^2+u^3$

t	0	1	2	3	4	5
$\mathbf{p}$	1	7	23	55	109	191
$\Delta^{(1)}\mathbf{p}$	6	16	32	54	82	
$\Delta^{(2)}\mathbf{p}$	10	16	22	28		
$\Delta^{(3)}\mathbf{p}$	6	6	6			

求出下一个值

从底部开始，依次向上可以求出多项式的新值

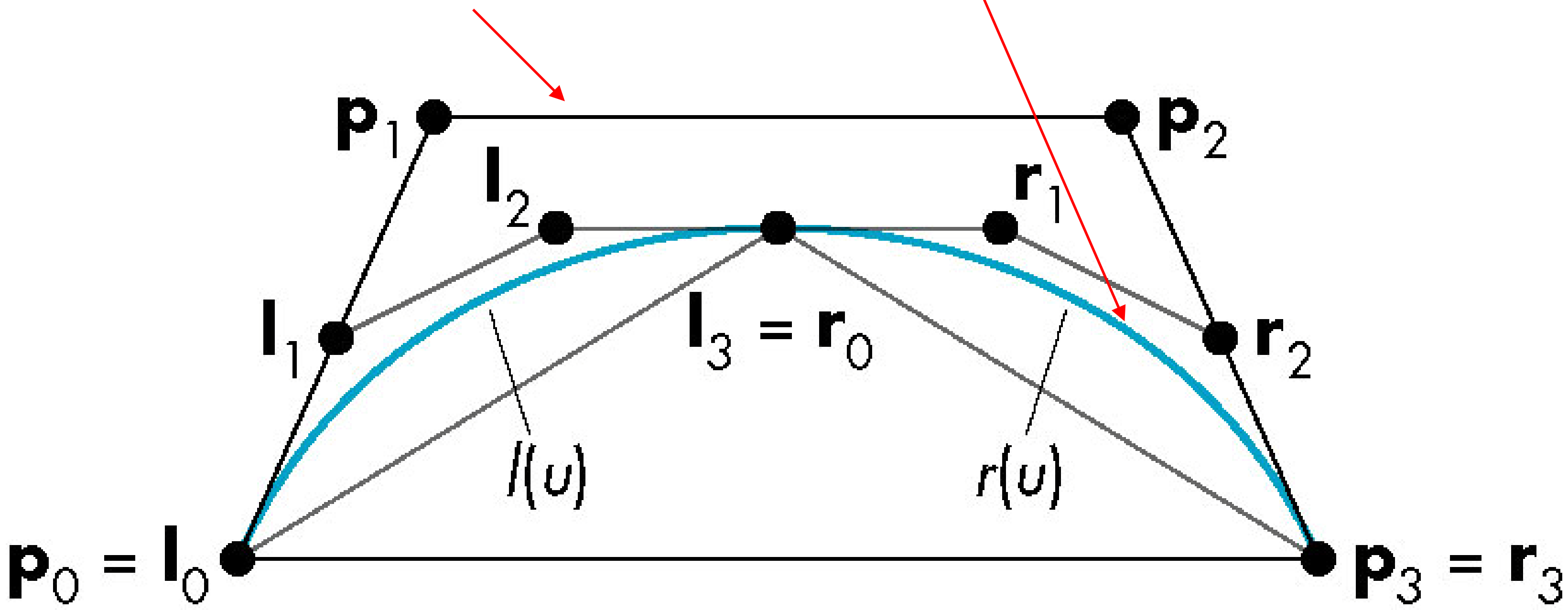
t	0	1	2	3	4	5
$\mathbf{p}$	1	7	23	55	109	191
$\Delta^{(1)}\mathbf{p}$	6	16	32	54	82	
$\Delta^{(2)}\mathbf{p}$	10	16	22	28		
$\Delta^{(3)}\mathbf{p}$	6	6	6			

de Casteljau算法

- 可以应用Bézier曲线的凸包性质给出一个有效的递推方法显示曲线
  - 只需要用到控制顶点的仿射组合
- 基于想法“任何多项式以及多项式的任何一部分都可以在正确选择了控制顶点后转化为一个Bézier多项式”

分割三次Bézier曲线

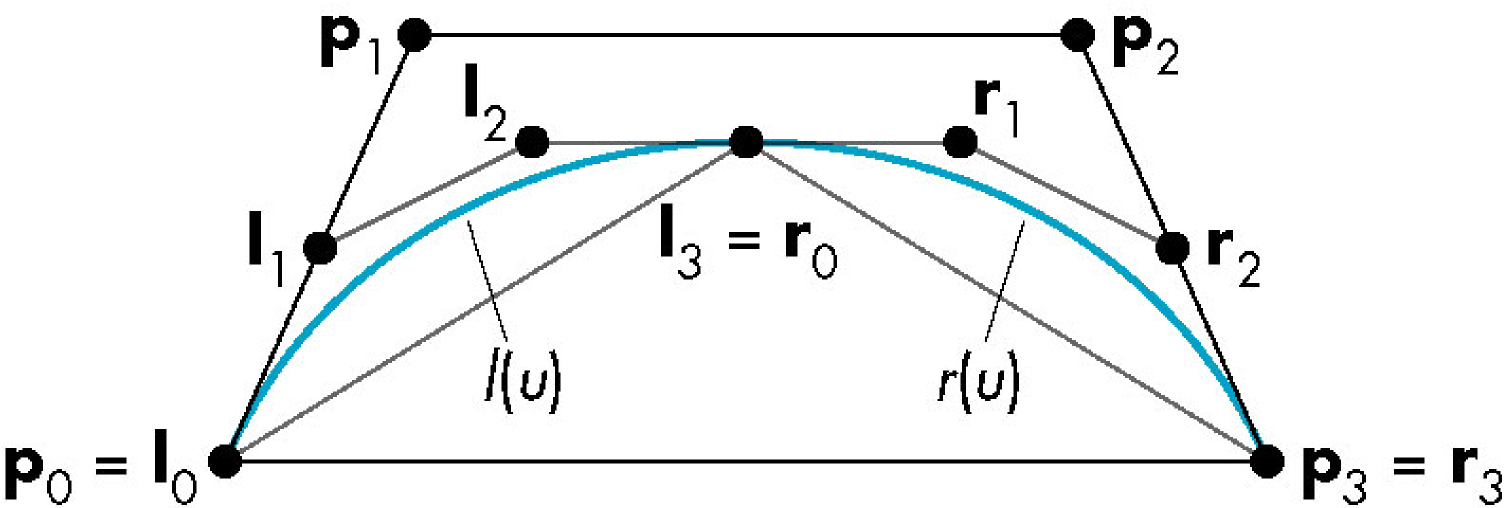
p_0, p_1, p_2, p_3 确定了一条三次Bézier多项式以及它们的凸包



考虑左半部分l(u)和右半部分 r(u)

l(u)和r(u)

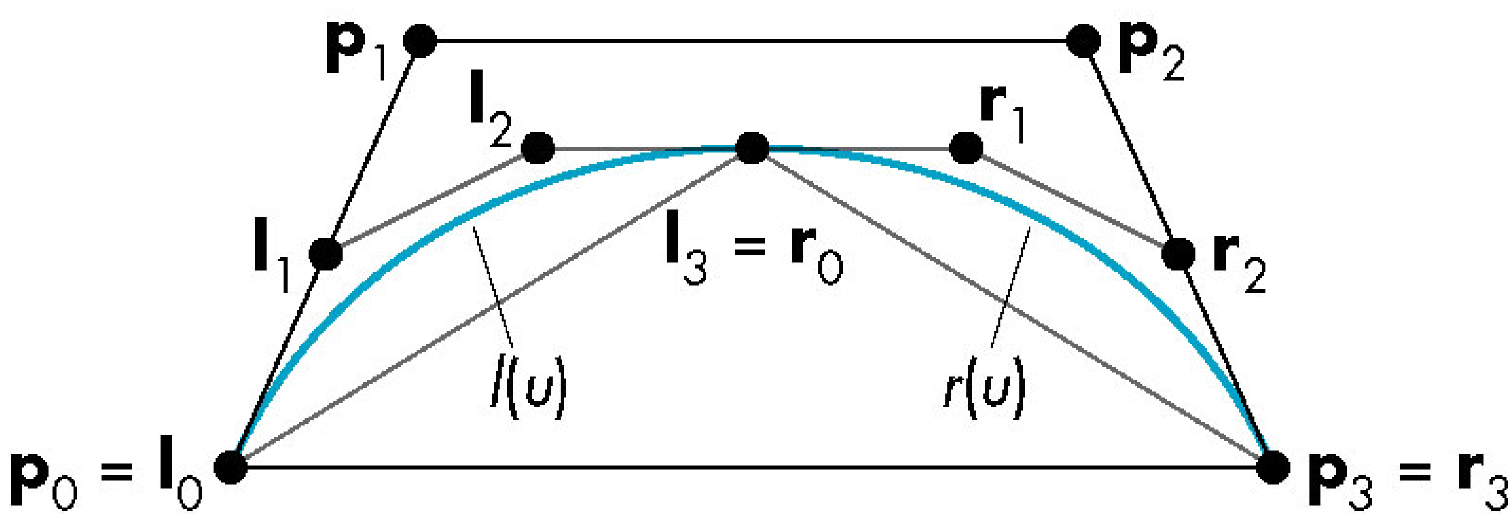
由于l(u)和r(u)也是Bézier曲线，我们可以求出两组控制顶点 $\{l_0, l_1, l_2, l_3\}$ 与 $\{r_0, r_1, r_2, r_3\}$，它们定义的这两条曲线



凸包

$\{l_0, l_1, l_2, l_3\}$ 与 $\{r_0, r_1, r_2, r_3\}$分别有自己的凸包，它们比$\{p_0, p_1, p_2, p_3\}$所确定的凸包更靠近曲线p(u).这条性质称为变差减缩(variation diminishing)

由从l_0到l_3 $(=r_0)$ 再到 r_3各点确定的折线是 p(u)的一个近似。重复这个过程可以得到更好的近似



公式

从Bézier方程$p(u)=\mathbf{u}^T M_B \mathbf{p}$开始

l(u) 需要插值p(0) 和 p(1/2)

$$l(0) = l_0 = p_0$$
$$l(1) = l_3 = p(1/2) = 1/8(p_0 + 3p_1 + 3p_2 + p_3)$$

为了匹配斜率，考虑到l(u) 与 r(u)分别只走了p(u)一半的路程

$$l'(0) = 3(l_1 - l_0) = p'(0) = 3/2(p_1 - p_0)$$
$$l'(1) = 3(l_3 - l_2) = p'(1/2) = 3/8(-p_0 - p_1 + p_2 + p_3)$$

对 r(u) 成立类似的公式

有效形式

$l_0 = p_0$

$r_3 = p_3$

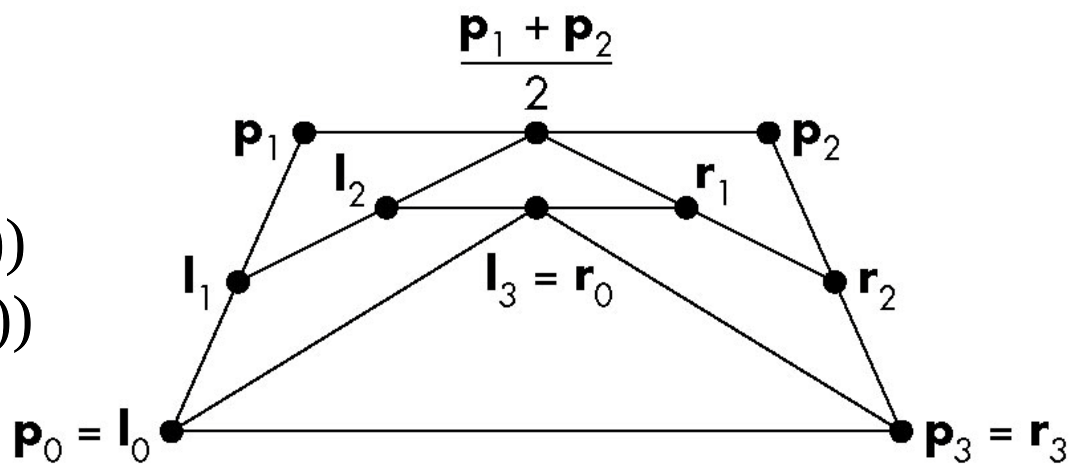
$l_1 = \frac{1}{2}(p_0 + p_1)$

$r_1 = \frac{1}{2}(p_2 + p_3)$

$l_2 = \frac{1}{2}(l_1 + \frac{1}{2}(p_1 + p_2))$

$r_1 = \frac{1}{2}(r_2 + \frac{1}{2}(p_1 + p_2))$

$l_3 = r_0 = \frac{1}{2}(l_2 + r_1)$



只需要移位与加法

所有曲线都是Bézier曲线

- 如果能找到给定多项式曲线表示为Bézier曲线所需要的控制顶点，那么就可以应用递推方法显示它
- 假设p(u)是以插值曲线的形式给出，数据点为q，

$$p(u)=\mathbf{u}^{T}\mathbf{M}_{I}\mathbf{q}$$

- 那么存在Bézier控制点p使得

$$p(u)=\mathbf{u}^{T}\mathbf{M}_{B}\mathbf{p}$$

- 列出方程并求解，可得$\mathbf{p} = \mathrm{M}_{B}^{-1}\mathrm{M}_{I}\,\mathbf{q}$

矩阵

插值形式到Bézier形式

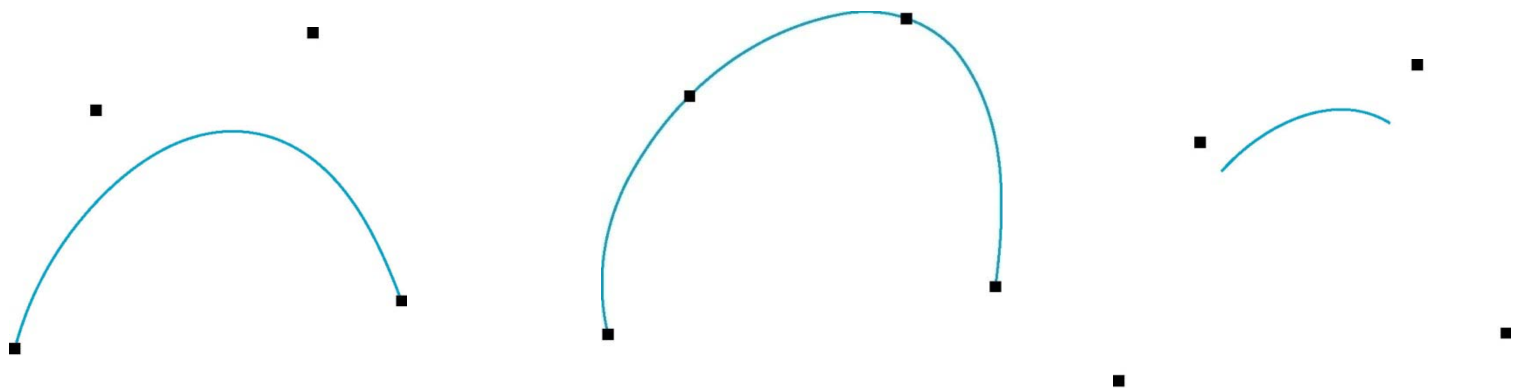
$$\mathbf{M}_B^{-1}\mathbf{M}_I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B-样条到Bézier形式

$$\mathbf{M}_B^{-1}\mathbf{M}_S = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}$$

示例

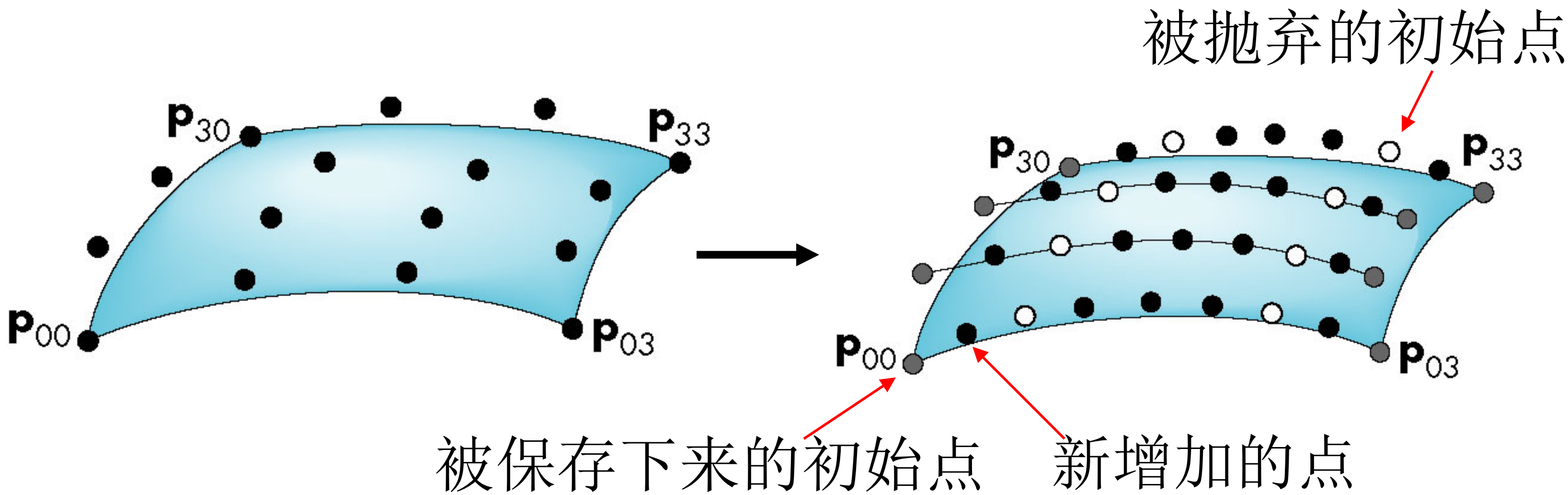
生成这些曲线的初始数据是相同的，但生成时都是把它们转化为Bézier控制顶点，然后采用Bézier递推方法



Bézier　插值　B样条

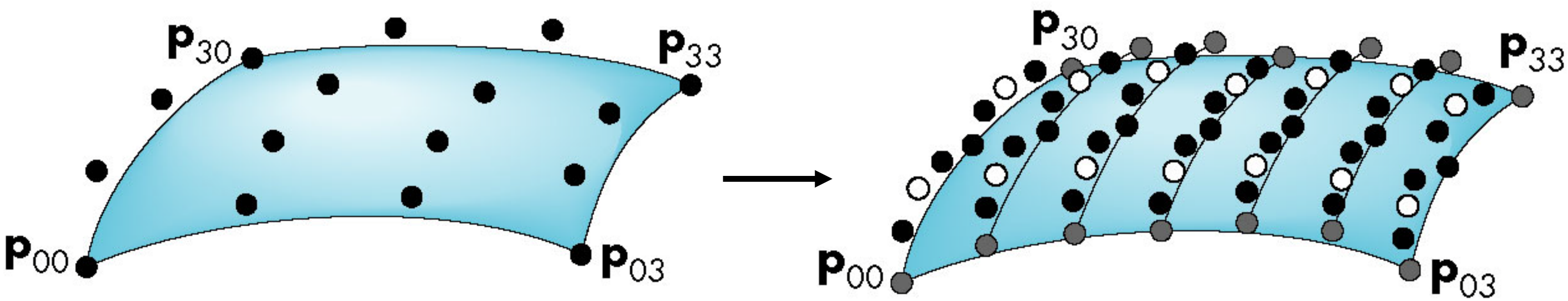
曲面

- 由于Bézier曲面片中常数参数u（或v）对应的曲线也是Bézier曲线，因此可以应用递推方法到曲面上
- 然后沿u方向细分
  - 这个过程得到新的点
  - 原来的某些控制顶点被抛弃

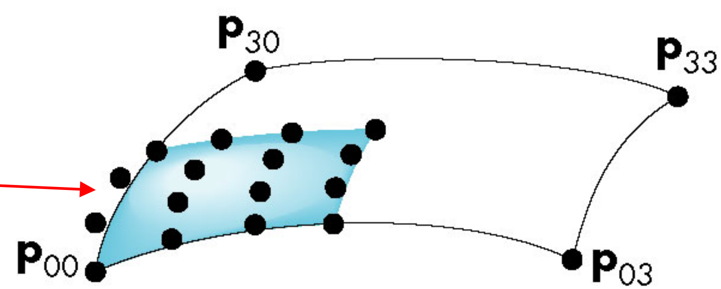


第二次细分

- ● 由细分生成的新点
- ○ 在细分后被抛弃的旧点
- ● 在细分后被保留的旧点



生成1/4曲面片创建了16个点



法向计算

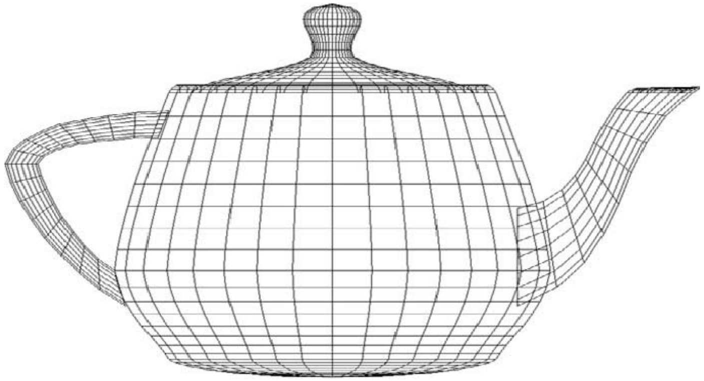
- 在显示的时候，为了进行明暗理，我们需要法向量
  - 可以从参数方程中计算

$$\mathbf{n} = \frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

  - 可以用角顶点确定
  - OpenGL可以自动计算

Utah茶壶

- 计算机图形学中最著名的数据集之一
- 通常用到的是由363个三维顶点定义的32张Bézier曲面片



几何建模参考书籍

- Gerald E. Farin. Curves and Surfaces for CAGD, 5th Ed. The Morgan Kaufmann, 2001.
- Josef Hoschek and Dieter Lasser. Fundamentals of Computer Aided Geometric Design. 1996.
- A. J. P. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, C. Galbraith. Implicit Curves, and Surfaces: Mathematics, Data Structures and Algorithms. Springer. 2009.
- J. Bloomenthal, C. Bajaj, J. Blinn, M. P. Cani-Gascuel, A. Rockwood, B. Wyvill, G. Wyvill. Introduction to Implicit Surfaces. Morgan Kaufmann Publishers. 1997.
- J. Warren, H. Weimer. Subdivision Methods for Geometric Design A Constructive Approach. Morgan Kaufmann Publishers. 2002.
- J. Peters, U. Reif. Subdivision Surfaces. Springer. 2008.
- M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Levy. Polygon Mesh Processing. A K Peters. 2010.
- M. Gross, H. Pfister. Point-Based Graphics. Morgan kaufmann Publishers. 2007.

Thanks for your attention!