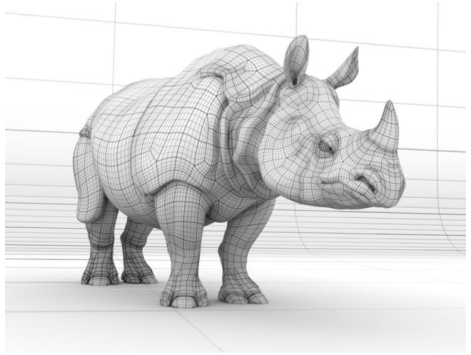
计算机图形学

胡事民

清华大学计算机科学与技术系

网格 (Mesh) 的细分、简化和分割

- 网格相关基本概念
- 网格细分 (Subdivision)
- 网格简化 (Simplification)
- 网格分割 (Segmentation)



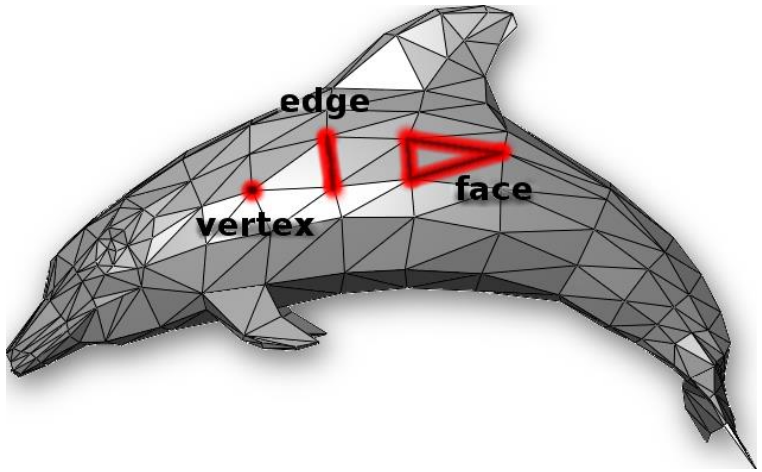
网格相关基本概念

- 网格的描述
  - 一系列的面片 $F = (f_1, f_2, \dots, f_n)$
    - 每一个面片都是三角形
  - 一系列的顶点 $V = (v_1, v_2, \dots, v_n)$
  - F中的每个面片是V中顶点的序列组

    e.g. $f_1: (v_1, v_2, v_3)$,

    $f_2: (v_4, v_5, v_6), f_3: (v_7, v_8, v_9)$

网格是计算机图形学中三维模型和场景的主要表示方式

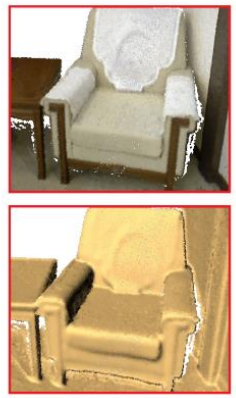


三维网格数据的来源

- **模型产生途径：**
  - 直接在几何文件中输入
  - 程序建模（Procedure Modeling）：通过程序代码进行创建, 比如L-系统，分形几何
  - 运用建模软件创建，比如3DS MAX / MAYA
  - 使用3D扫描仪（3D Scanner）获取真实模型

三维数据的来源

- **模型产生途径：**
  - 照片测量法 (Photogrammetry)：基于照片进行三维重构（Reconstruction），传统视觉方法
  - 基于照片测量法获取RGBD数据：微软Kinect



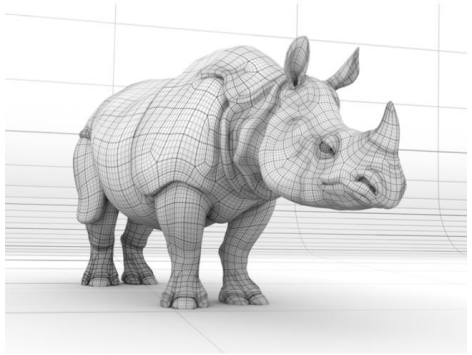
新方向：基于网格表示的几何处理

- 真实感的**实时绘制**问题变的日益突出，为了加速实时绘制，学术界开始研究网格模型的简化。
- 1992年开始，面片**简化、压缩、信号处理、纹理合成**成为SIGGRAPH会议的一个热门话题；后来人们关注**拓扑修复、细节建模、动画变形**等；从而演化为**数字几何处理**这一研究方向。
  - **Eurographics Symposium on Geometry Processing**

    from 2003

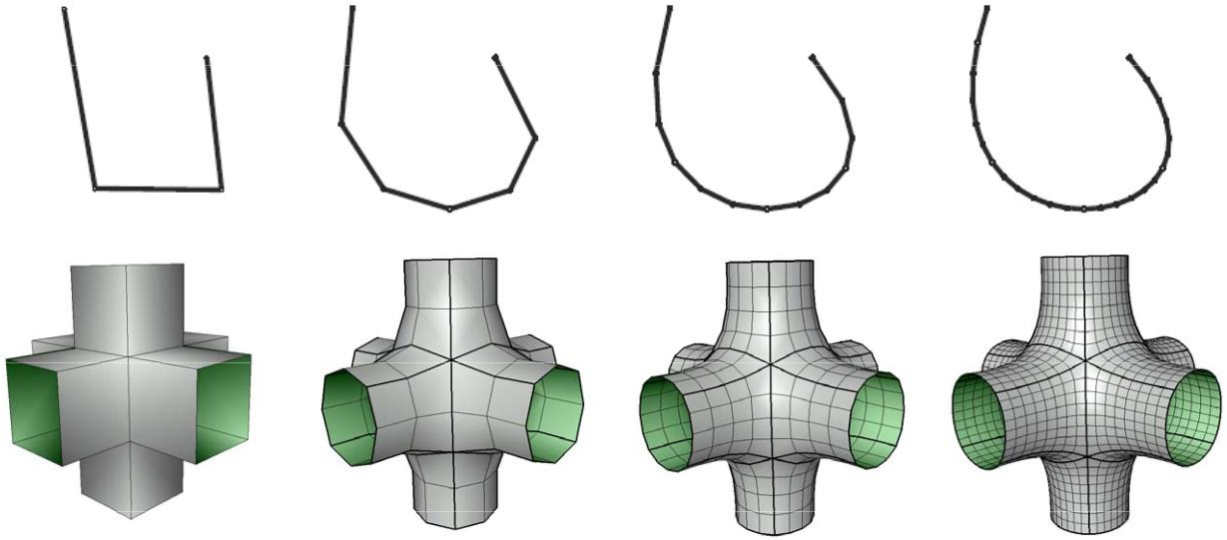
网格 (Mesh) 的细分、简化和参数化

- 网格相关基本概念
- 网格细分 (Subdivision)
- 网格简化 (Simplification)
- 网格分割 (Segmentation)



网格细分

- 一个细分的例子



细分 (Subdivision)

- 细分 (Subdivision)
  - Catmull and Clark, Doo and Sabin 在1978发表的论文标志着模型表面细分的开始。现在，细分已大量应用于电影作品中。
  - 细分的简略描述
    - 对于一个给定的原始网格，进行精细地改进产生更光滑的效果
  - 细分人物 （下一页）

- 细分人物谱
  - Edwin Catmull： Pixar & Disney, 图灵奖
  - Jim Clark:
    - Silicon Graphics
    - Netscape
  - **Charles Loop**
  - **Leif Kobbelt**
  - Tony Derose
  - Malcolm Sabin

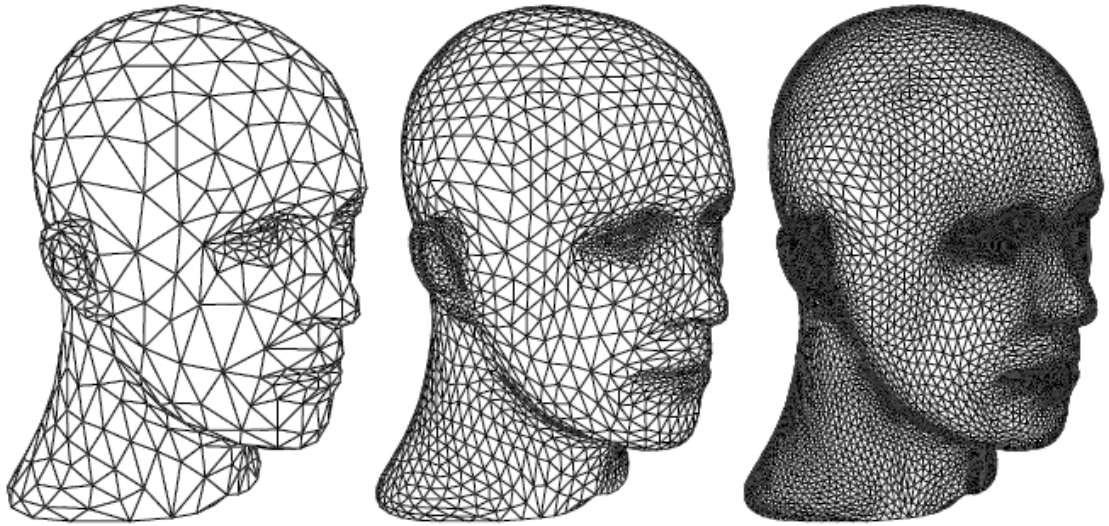
细分 (Subdivision)

- 一维细分：例子

  - 最左面的4个点由平直线段相连.
  - 右边一个改进版本: 3个新的点被插入.
  - 经过进一步的两轮细分,曲线已经变得相当光滑.

细分 (Subdivision)

- 细分的其他例子



三个连续的细分精炼结果. 最左边是最初的网格. 根据特定的规则，每一个三角形被分割成四个小的三角形 (中间). 再次进行前面的分割得到最右边的结果.

细分 (Subdivision)

- 细分(Subdivision)
  - 表面细分可以被看成一个两阶段过程. (最初的网格被称作控制网格).
    - 第一步：称作细化阶段, 创建新顶点并与先前顶点相连产生新的、更小的三角形
    - 第二步：称作平滑阶段, 计算新顶点的位置.
  - 这两步的细节决定了不同的**细分方案**. 在第一步中, 一个三角形可以不同的形式进行分割; 在第二步中, 新顶点的位置可以不同的方式插值产生.

细分方案 (Subdivision Schemes)

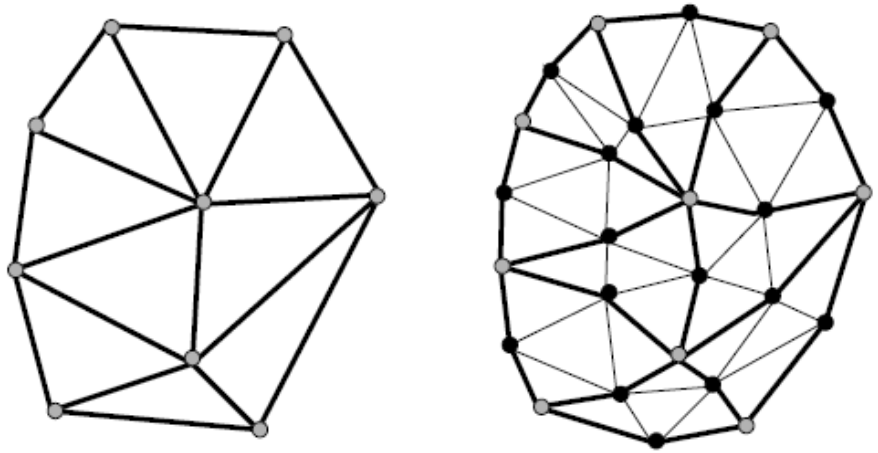
- 我们将介绍两个细分方案
  - Loop细分
  - $\sqrt{3}$细分 (Kobbelt)

Loop细分 (Loop Subdivision)

- Loop细分
  - Loop 细分是第一个基于三角网格的细分方案.
  - 它更新每个已有的顶点并对每条边创建一个新的顶点，然后每个三角形被分割成四个新的三角形. 因此，经过n步细分, 一个三角形被分割成4^n个三角形.

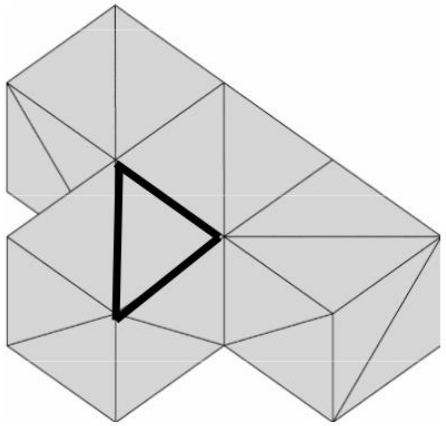
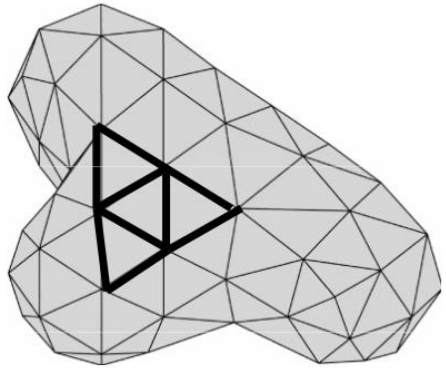
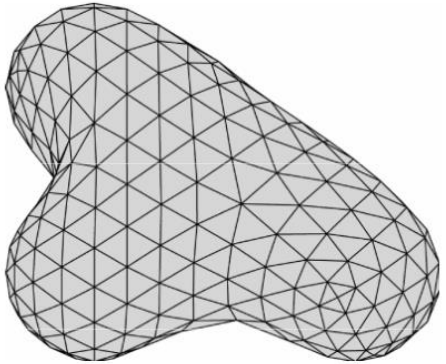
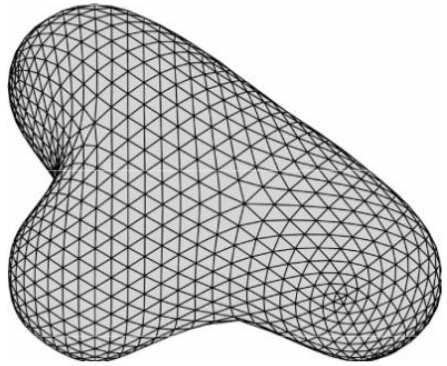
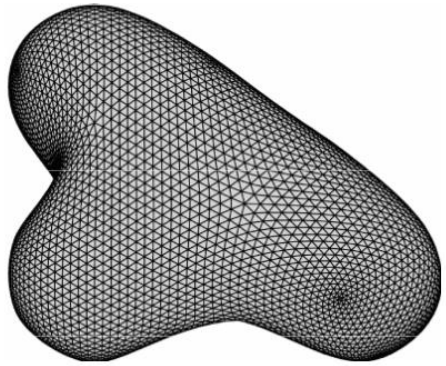
Loop细分(Loop Subdivision)

- Loop细分(Loop Subdivision)



Loop细分的例子. 新的顶点以黑色显示. 每条边中加入一个新的顶点, 新的顶点被连接起来创建四个新的三角形以替换掉原先的三角网格.

Loop细分(Loop Subdivision)

Loop细分(Loop Subdivision)

- 细分规则 (Subdivision rule)
  - 关注一个已存在的顶点p^k, 其中 k是当前已进行的细分步数。按此记法， p^0即表示控制网格的顶点。一般地，细分过程可以表示为

$$p^0 \to p^1 \to p^2 \to p^3 \ldots$$

  - 如果p^k的度为 n，那么p^k有n个相邻顶点，记做$p_i^k, i \in \{0,1,\ldots,n-1\}$

Loop细分(Loop Subdivision)

- 细分规则 (Subdivision rule)
  - Loop细分的规则：

    第一个公式是将每个顶点p^k升级为p^{k+1}的准则，第二个公式是在边$p^k p_i^k$上创建新顶点p_i^{k+1}的准则.

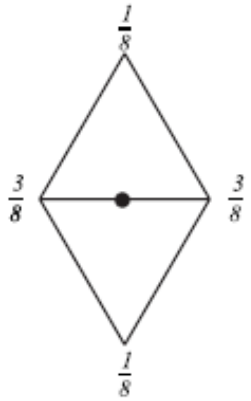
$$p^{k+1} = (1-n\beta)p^k + \beta(p_0^k + \ldots + p_{n-1}^k)$$

$$p_i^{k+1} = \frac{3p^k + 3p_i^k + p_{i-1}^k + p_{i+1}^k}{8}, i = 0, \ldots, n-1$$

β 是 n 的函数 $\beta(n) = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right)$

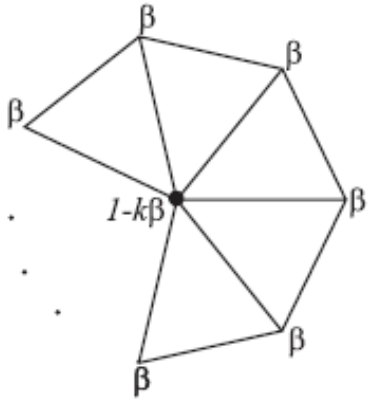
Loop细分(Loop Subdivision)

- 细分规则 (Subdivision Rule)
  - 上述公式图示：



在边上加入一个新顶点

$$p_i^{k+1} = \frac{3p^k + 3p_i^k + p_{i-1}^k + p_{i+1}^k}{8}, i = 0, \ldots, n-1$$



对顶点的更新调整

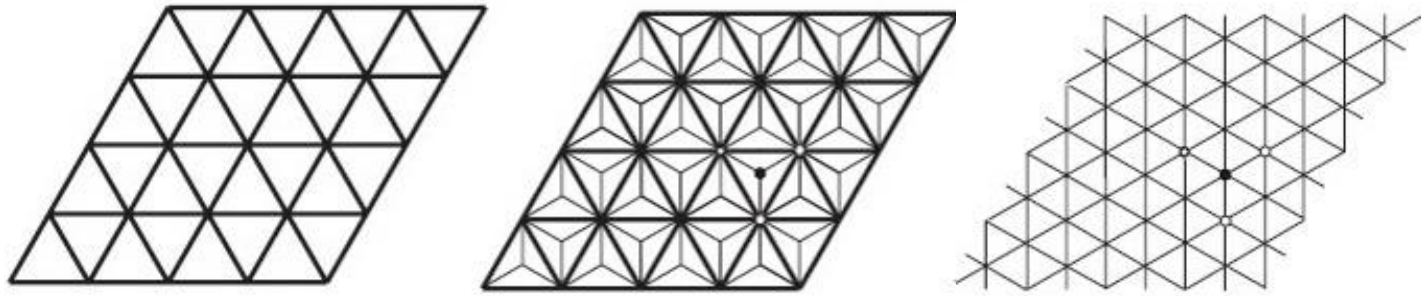
$$p^{k+1} = (1 - n\beta)p^k + \beta(p_0^k + \ldots + p_{n-1}^k)$$

$\sqrt{3}$细分($\sqrt{3}$-Subdivision)

- 细分$\sqrt{3}$
  - Loop细分将每个三角形分割成4个新的小三角形, 因此以$4^n m$的比率创建新的三角形, 其中 m 是控制网格中的三角形数, n是细分的步数.
  - 能否将一个三角形分为三个子三角形，构造细分格式？

$\sqrt{3}$细分($\sqrt{3}$-Subdivision)

- 细分($\sqrt{3}$-Subdivision)
  - 能否将一个三角形分为三个子三角形，构造细分格式？
  - 为了得到更均匀分布的三角形, 每一个原始三角形都被翻转以使之连接两个相邻的中间顶点而不是连接两个已经存在的顶点



$\sqrt{3}$ 细分的过程

√3细分(√3-Subdivision)

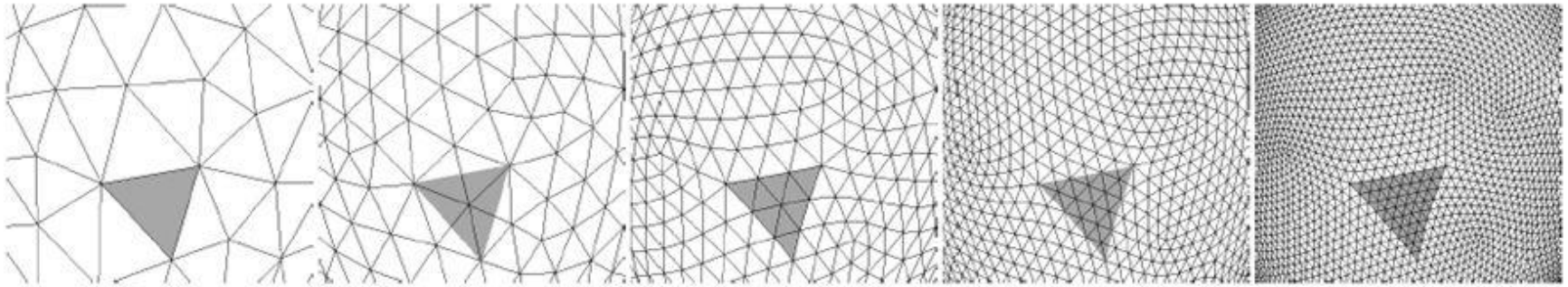
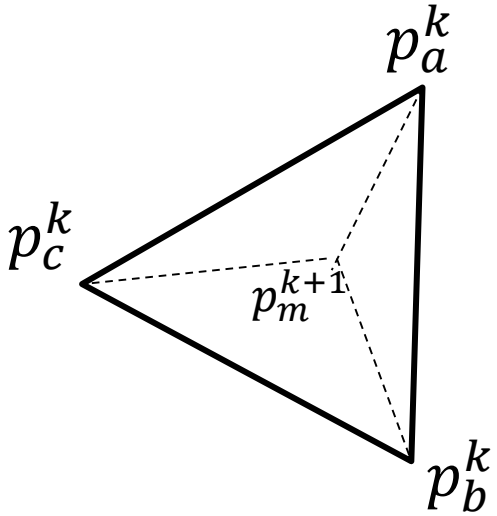


Figure 3: *The √3-subdivision generates semi-regular meshes since all new vertices have valence six. After an even number 2k of refinement steps, each original triangle is replaced by a regular patch with 9^k triangles.*

√3细分(√3-Subdivision)

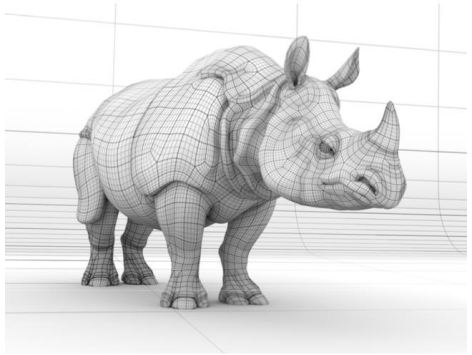
- 细分规则(Subdivision rule)
  - 在下面第一行公式中, p_m表示中间顶点,由计算三角形三个顶点p_a, p_b, p_c的平均值得来
  - 对于每一个存在的顶点, p^k用第二行公式来做更新,其中p_i^k表示p^k的相邻顶点.
  - n是p^k的度，k是细分步数.

$$\begin{cases} p_m^{k+1} = (p_a^k + p_b^k + p_c^k)/3 \\ p^{k+1} = (1-n\beta)p^k + \beta\sum_{i=0}^{n-1} p_i^k \\ \beta(n) = \dfrac{4-2\cos(2\pi/n)}{9n} \end{cases}$$



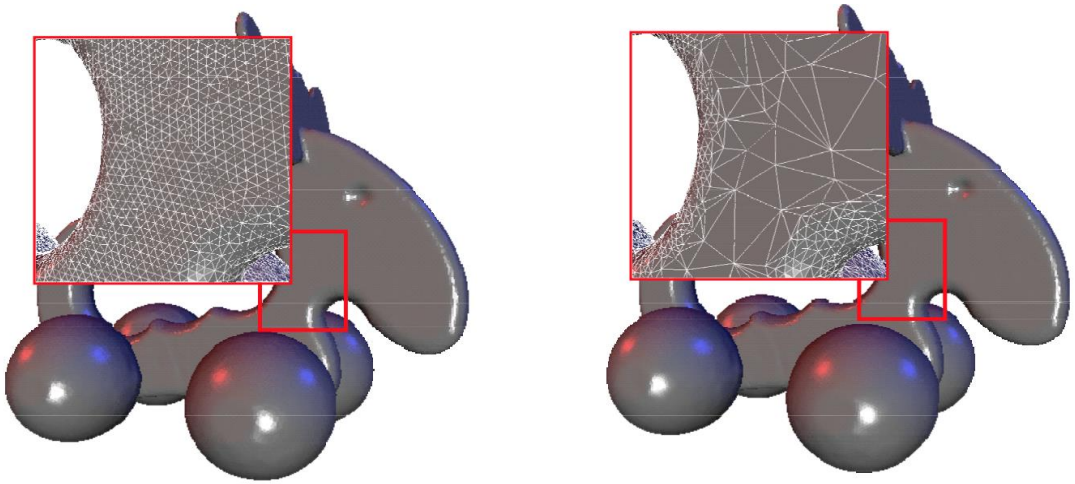
网格 (Mesh) 的细分、简化和分割

- 网格相关基本概念
- 网格细分 (Subdivision)
- 网格简化 (Simplification)
- 网格分割 (Segmentation)



简化 (Simplification)

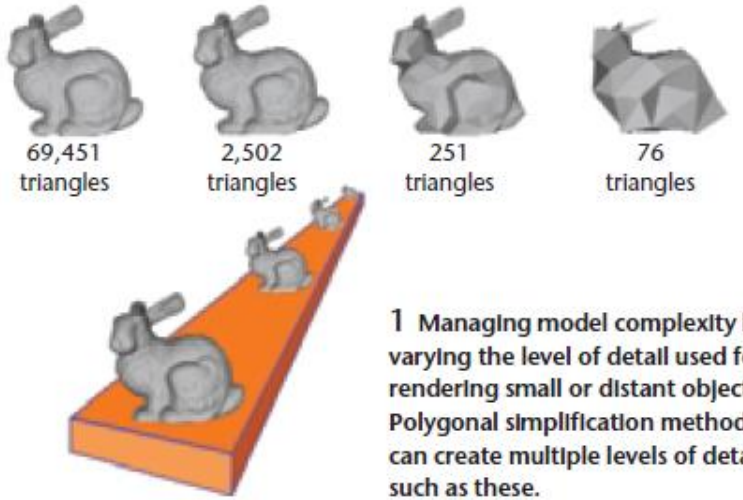
- 网格简化的概念
  - 用一些相对简单但维持几何特征的表示来近似给定的网格
  - 移除几何冗余
    - 如，一个有很多共面小三角形的平坦区域，将这些三角形融合成大的多边形能降低模型的复杂度.
    - 降低存储或传输的规模.



简化(Simplification)

- 提高运行性能
  - 简化对于高效的渲染编辑具有重要作用
  - 产生场景中物体的层次细节 (Level Of Details).

    较远的物体用较低的层次细节呈现，较近的物体用较高的层次细节呈现.



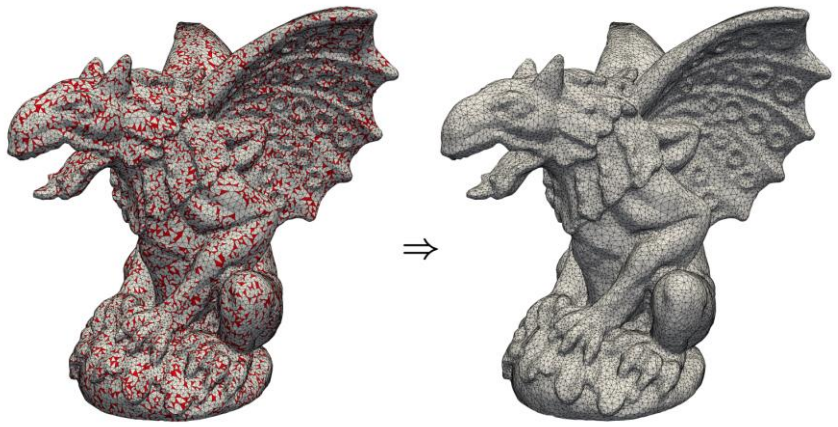
1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

简化(Simplification)

- 如何简化?
  - 几乎所有简化技术都使用下面四种基本技巧的组合或者变形:
    - 采样（Sampling）
    - 自适应细分（Adaptive subdivision）
    - 去除（Decimation）
    - 顶点合并（Vertex merging）.

简化 – 采样 (Sampling)

- 采样算法通过选取模型表面的点简单地对模型进行几何取样.
- 这类方法难于精确地获取高频特征，通常在没有尖角的光滑表面上取得最好的效果.
- 后续的改进：2001年 – 特征敏感的距离度量

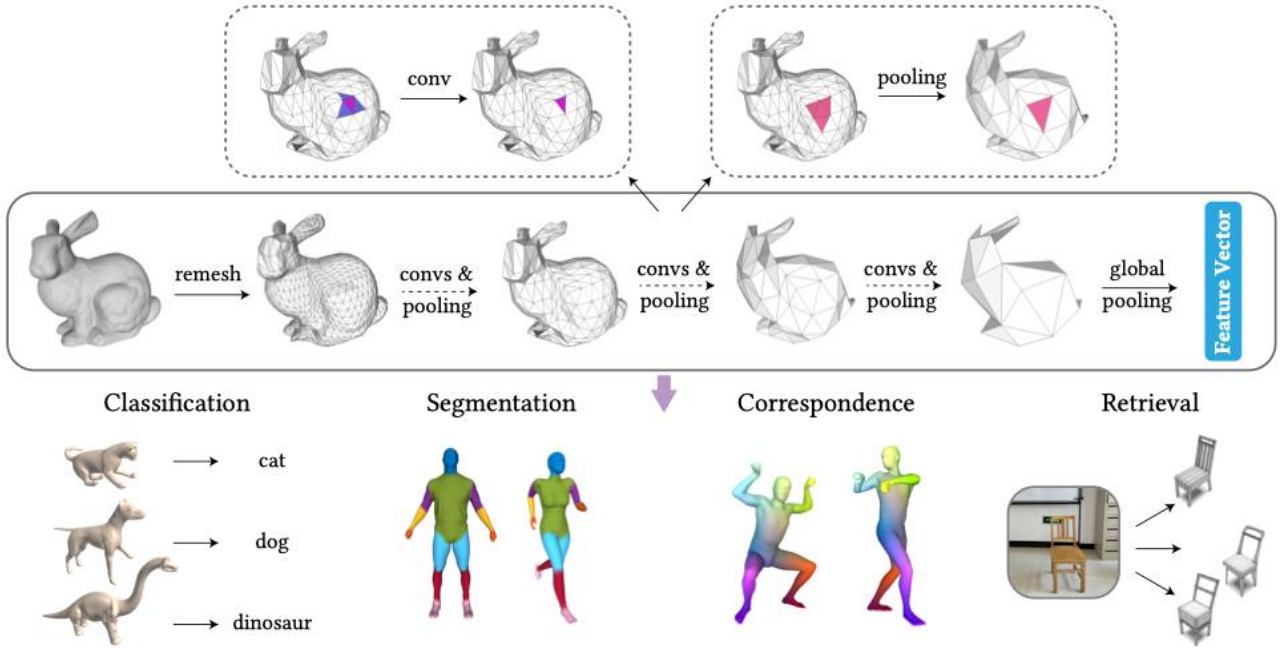


特征敏感的网格重剖 (Feature sensitive Re-meshing)

简化 – 自适应细分 (Adaptive subdivision)

- 自适应细分算法通过寻找一个可以递归细分的基底网格来近似最初的模型.
- 该算法在基底网格易于获取的情况下能取得最好效果。例如,对于地形模型是一个典型四边形的情形.
- 后续的应用：

  基于细分的卷积神经网络



简化 – 去除 (Decimation)

- 去除方法迭代地移除网格上的顶点或面片，并三角化每步移除后留下的孔洞.
- 这类方法相对简单、易于编程实现并且运行效率高。这类算法尤其善于处理剔除几何冗余，比如大量共面的多边形的情形。

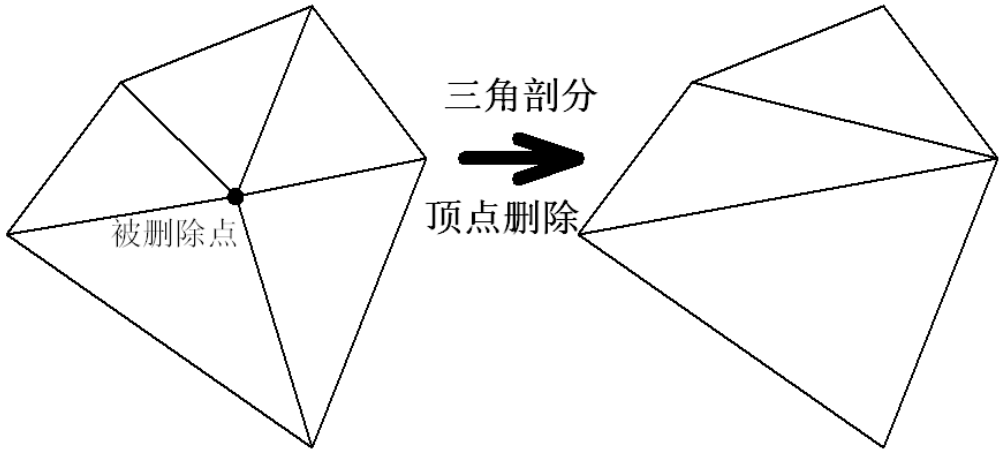
简化 – 顶点合并 (Vertex merging)

- 顶点合并方案通过将断裂三角化模型的两个或更多顶点合并为一个顶点，转而可以使之与其他顶点合并。该算法需要采用多种技术来决定以何种顺序合并顶点。能够修改拓扑并且聚合物体。
- 边坍塌算法 (Edge-collapse algorithms), 一种总是将共边的顶点合并的算法, 易于保持局部拓扑。

网格化简的基本操作（1）

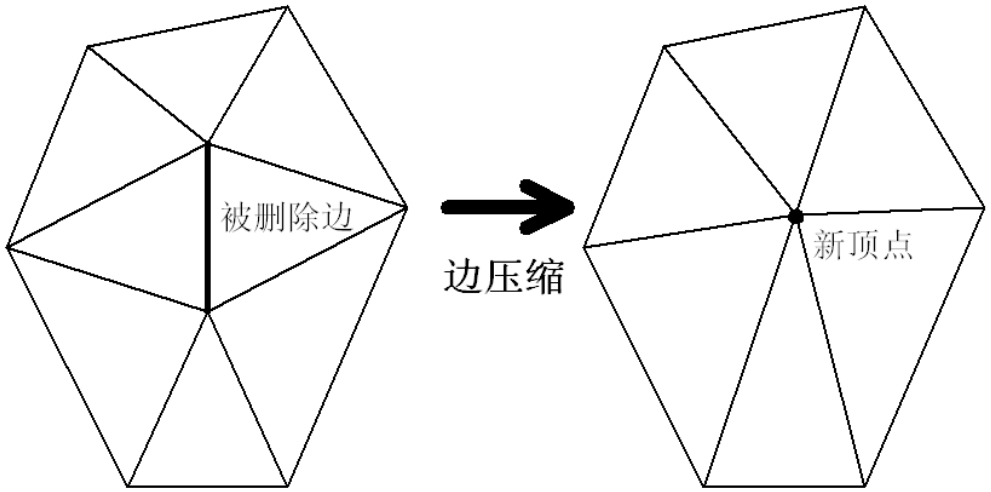
- 三种不同的基本化简操作：

1. **顶点删除操作：删除网格中的一个顶点，然后对它的相邻三角形形成的空洞作三角剖分**



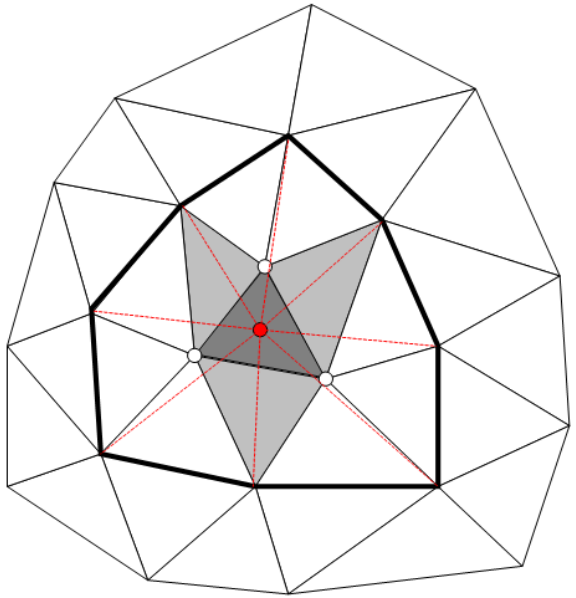
网格化简的基本操作（2）

2. **边压缩操作：网格上的一条边压缩为一个顶点，与该边相邻的两个三角形退化**



网格化简的基本操作（3）

3. **面片收缩操作：网格上的一个面片收缩为一个顶点，该三角形本身和与其相邻的三个三角形都退化**



Jian-Hua Wu, Shi-Min Hu, Chiew-Lan Tai and Jia-Guang Sun, An effective feature-preserving mesh simplification scheme based on face constriction, Pacific Graphics 2001, 12-21, Tokyo

基于长方体滤波的多面体简化

- **1993年，Rossignac和Borrel提出一个实用的、能实时建立LOD模型的多面体简化算法。**

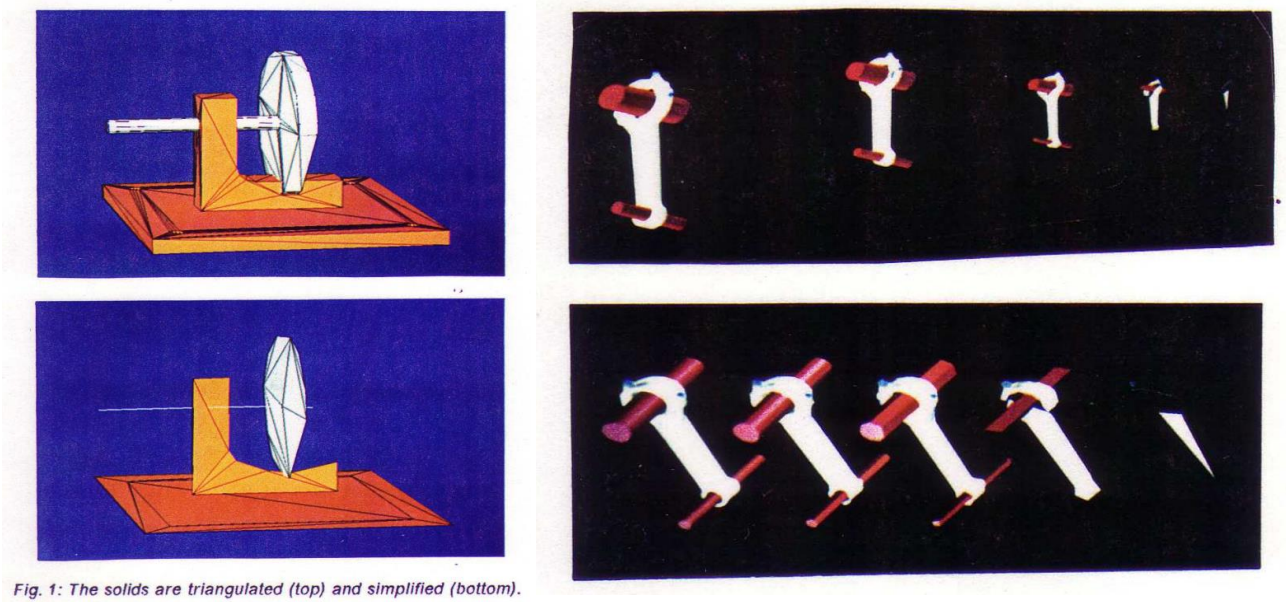
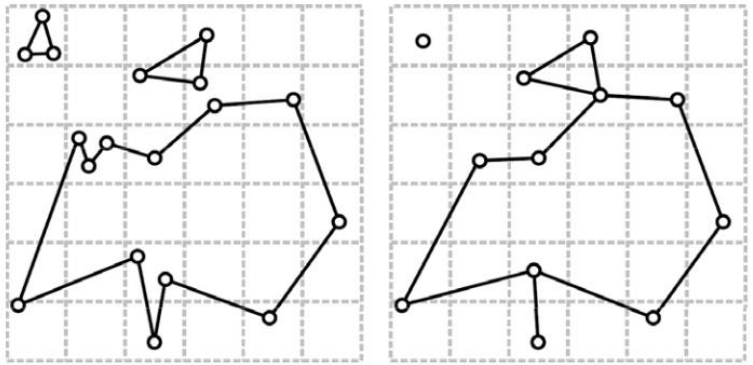


Fig. 1: The solids are triangulated (top) and simplified (bottom).

长方体滤波的步骤（1）

- 给定一个多面体 M，记 K 为其拓扑，假设 M 已三角化。算法首先建立 M 的长方体包围盒，并将该包围盒所包围的空间均匀剖分成一系列的小长方体子空间，然后采用各长方体子空间对景物顶点进行聚类合并，位于同一长方体空间的顶点被归于同一类 (Cluster)。



长方体滤波的步骤（2）

- 最后属于同一类的顶点被合并为一代表点，而这些代表顶点为原多面体所示景物的重新采样。基于原多面体的拓扑结构和这些这些采样点可重新产生一多面体。所得到的多面体即为保持一定层次细节的模型。原多面体的包围盒剖分生成的子空间越小，所得到的层次模型就越逼近于原多面体。

长方体滤波的缺点

- 方法的主要缺点是顶点合并导致了一些**重要高频细节的丢失**。
- 参考文献
  - Rossignac J, Borrel P, Mutli-Resolution 3D approximations for rendering complex scences, in Modeling in Computer Graphics, edit by B Falcidieno and T L Kunii, spring-Verleg, 1993, pp 455-465.

顶点删除技术

- **想法: 设法减少景物表面的采样点数目**
  - 假设景物表面已离散为一系列三角形，顶点删除算法首先从原始模型的顶点集中删除**一些不重要的顶点**，同时从其面片集中删除与这些顶点相连的所有面片。经上述操作后在原景物表面上留下了一些空洞，算法再对这些空洞进行局部三角剖分。
  - 如何确定哪些顶点不重要？局部判别准则！

局部判别准则

- 基于相邻面片和边界的局部平坦性原则
  - Schroeder W, Zarge J A, and Lorensen W E, Decimation of triangle meshes, Computer Graphics, 1992, 26-2, 65-70.
- 采用等距面来限定简化模型顶点的变化范围
  - Cohen J, Varshney A, Manocha D, and Turner D, Simplification Envelopes, Computer Graphics, 1996, 119-128.

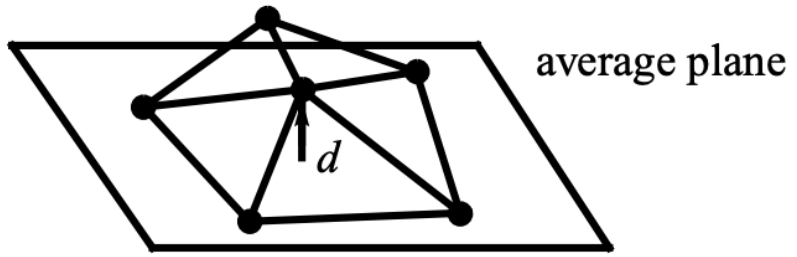
Schroeder的局部判别准则

该判别准则分两种情况：

1） 对网格内部顶点v，记其周围相邻面片集为S，则该点的平坦性标准由下述的距离来描述：

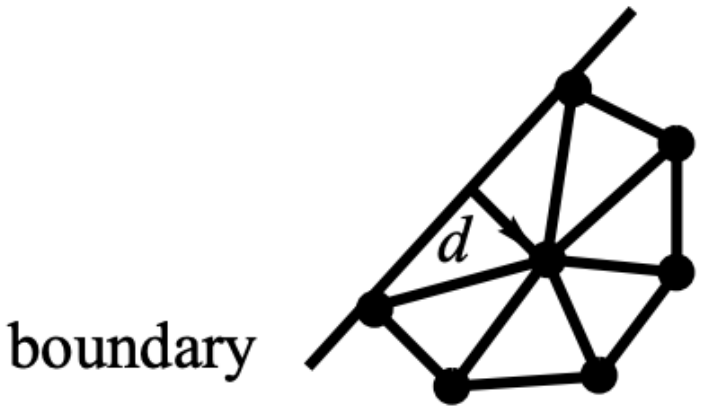
$$d = \left|\mathbf{N}\cdot(v-C)\right|$$

其中N为向量 $\dfrac{\sum_{f\in S}\mathbf{n}(f)A(f)}{\sum_{f\in S}A(f)}$ 的单位向量，$C=\dfrac{\sum_{f\in S}c(f)A(f)}{\sum_{f\in S}A(f)}$ ，



这里$A(f), c(f), \boldsymbol{n}(f)$ 分别为三角面片的面积、中心和法向量。

2） 对边界顶点 v，记与它相邻的两个边界顶为 v_1，v_2，则其平坦性标准定义为 v 到 v_1 与 v_2 连线的距离。



Cohen 的局部判别准则

- 多面体的表面包络（Surface Envelope）：

  多边形网格表面P可看作一张分片线性参数曲面。

$$r(u,v) = (r_x(u,v), r_y(u,v), r_z(u,v))$$

  其单位法向量为

$$\mathbf{n}(u,v) = (n_x(u,v), n_y(u,v), n_z(u,v))$$

表面包络（Surface Envelope）

- 对给定的$\varepsilon > 0$，P的三维ε等距面定义为

$$r^{\varepsilon}(u,v) = r(u,v) + \varepsilon \mathbf{n}(u,v)$$

近似地定义原始多边形网格P沿其正、负法向的ε等距面$P(+\varepsilon)$和$P(-\varepsilon)$。

ε等距面$P(+\varepsilon)$和$P(-\varepsilon)$上对应顶点v_i^+，v_i^-及其法向量可分别表示为

$$v_i^+ = v_i + \varepsilon \mathbf{n}_i \qquad v_i^- = v_i - \varepsilon \mathbf{n}_i \qquad \mathbf{n}_i^+ = \mathbf{n}_i^- = \mathbf{n}_i$$

上述方法生成的$P(\pm\varepsilon)$可能出现自交现象。

简化包络（Simply Envelope）

- 为了避免自交，我们需要缩小某些顶点扩张的ε范围，计算新的ε'
- 解析法：
  - 现来考察P上的任一三角形$\Delta v_1 v_2 v_3$
  - 对P上每个与三角形$\Delta v_1 v_2 v_3$不相邻的三角面片Δ_j，判别Δ_j是否本柱体相交；可计算得到q_j到$\Delta v_1 v_2 v_3$的距离δ_j
  - 最终的ε'取所有不相邻三角面片Δ_j所计算出的δ_j中的最小值的
- 我们把这种方式计算出的非自交包络叫做简化包络。只处于简化包络中，就说明本次简化得到的新几何和原有过ε

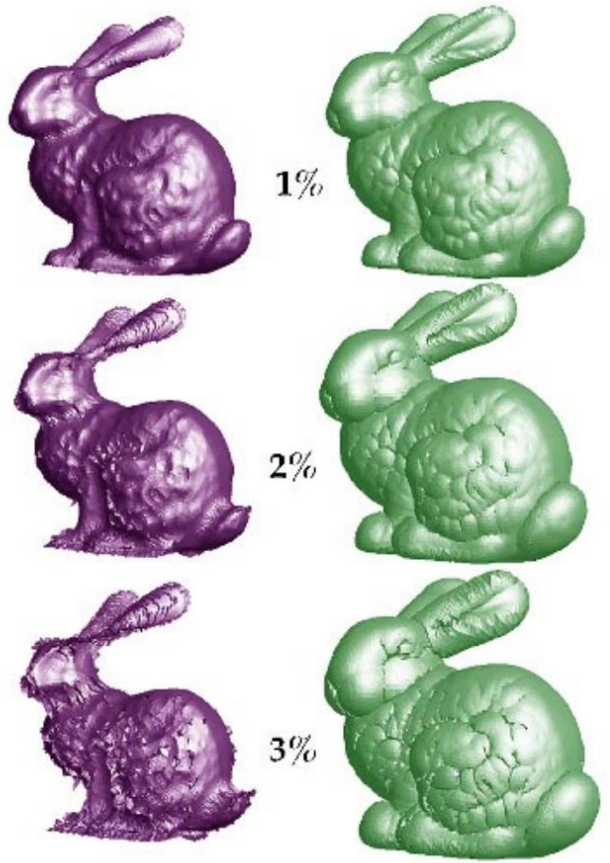


Figure 6: *Simplification envelopes for various* ϵ

Cohen的简化算法整体流程

- 采用贪婪搜索策略，将原表面P的所有顶点列入待处理的顶点队列。
- 对当前待处理顶点队列中的一顶点，算法尝试从P上删除该顶点及与该顶点直接相邻的三角面片，并试图用递归的三角剖分方法来填补顶点删除后在表面P上形成的空洞。

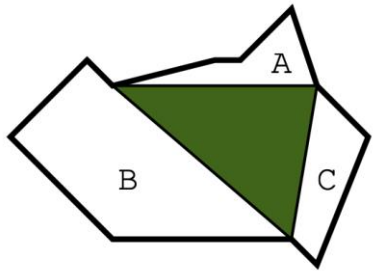
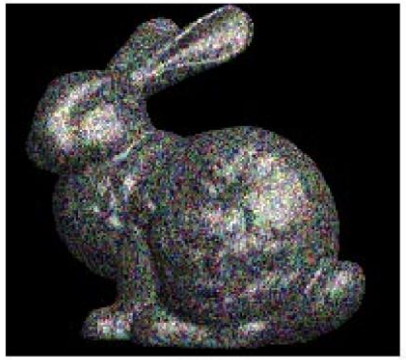


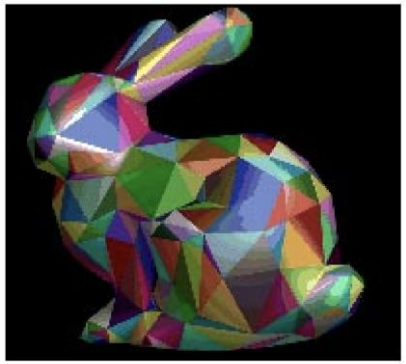
Figure 7: *Hole filling: adding a triangle into a hole creates up to three smaller holes*

Cohen的简化算法整体流程

- 若填补之后的空洞处几何形状完全位于P的简化包络内，则从当前队列中删除该顶点，简化表面P的模型并重构原来与该顶点相连接的各顶点的拓扑关系。否则，该顶点从当前待处理队列中退出，表面P保持不变。重复，直到待处理顶点队列变为空。



(a) bunny model: 69,451 triangles



(a) $\epsilon = 1\%$, 575 triangles

渐进的网格简化技术

- 渐进网格算法中，任一网格 $\hat{\mathrm{M}}$ 均可表示为基本网格及 n个逐步细化网格 $M^i, \{i = 1, 2, \ldots, n\}$ 的变换，且有 $\hat{M} = M^n$
- 一张网格 M 可定义为1个二元组 (K, V)，其中 K 是一个单纯复形(simplicial complex)，它表示了 M 的顶点、边和面的邻接关系； $V = \{v_i \in R^3 | i = 1,2, \ldots n\}$ 是 M 的顶点位置向量集，它定义了网格 M 在 R^3 中的形状

- 单纯复形 K 由顶点集{i=1,2,…,m}及称之为单形的非空子集组成
  - 0-单形 $\{i\} \in K$，即为顶点
  - 1-单形 $\{i, j\} \in K$，即为一条边
  - 2-单形$\{i, j, k\} \in K$，即为一个面

拓扑实现的概念

- 值得注意的是，单纯复形 K 并不包含点集 $\{i = 1,2, \ldots n\}$的所有子集，仅包含了构造网格 M 所有面、边、顶点的子集
- 为在结构上刻划单纯复形，我们引进拓扑实现(topological realization) $|K|$ 的概念

- 若将顶点 $\{i\}(i = 1,2,\ldots,m)$ 看成为 R^m 中的基向量，$e_i =$

$|K|$ K

$$|K| = \bigcup_{s \in K} |s|$$

几何实现的概念

- 为此我们记$\phi_V = \phi_{|K|}$，称为$\Delta v_1 v_2 v_3$在R^3中的几何实现(geometric realization)
- 若 $\phi_V(|K|)$ 不自交，则 ϕ_V 为1-1映射。此时，ϕ_V为一嵌入映射，即对$\forall p \in \phi_V(|K|)$，存在唯一m维向量b $\in |K|$，使得p $= \phi_V(b)$，我们将 b 称为 p 关于单纯复形的重心坐标向量(barycentric coordinate vector) 。事实上，b 可表示为:

$$b = \sum_{i=1}^{m} b_i e_i$$

容易知道，当 M 为一三角片网格时，$\phi_V(|K|)$ 上的任一点的重心坐标向量 b 中至多只有三个分量非零

显式能量函数度量

- 有了上述定义，Hoppe采用显式能量函数$E(M)$来度量简化网格与原始网格的逼近度([HOPP96])：

$$E(M)=E_{dist}(M)+E_{spring}(M)+E_{scalar}(M)+E_{disc}(M)$$

- 其中$E_{dist}(M)$为 M 的距离能量，定义为点集 $X=\{x_{1,\cdots,}x_n\}$ 到网格 M 的距离平方：

$$E_{dist}(M)=\sum_{i=1}^{n}d^2(x_i,\phi_V(|K|))$$

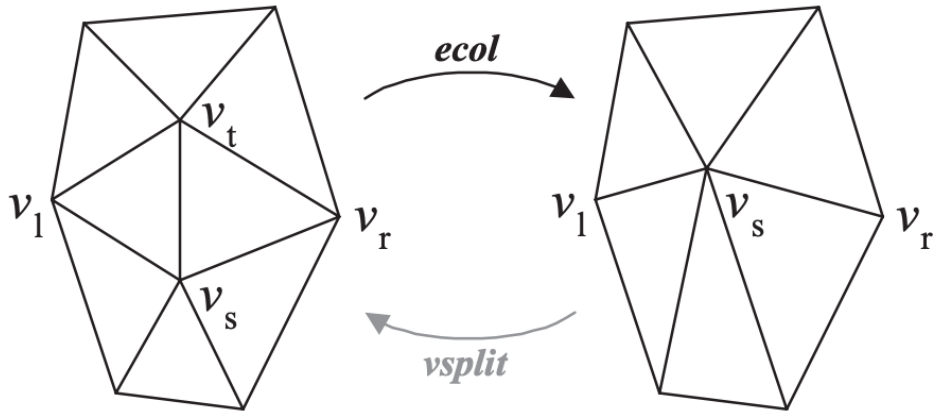
- E_{spring} 为 M 的弹性能量，这相当于在 M 的每条边上均放置一条弹性系数为 k 的弹簧，即

$$E_{spring}(M) = \sum_{\{i,j\} \in K} k \left\| v_i - v_j \right\|^2$$

- $E_{scalar}(M)$ 度量 M 的标量属性的精度，而 $E_{disc}(M)$ 则度量了 M 上视觉不连续的特征线（如边界线、侧影轮廓线等）的几何精度。

边收缩变换

- Hoppe利用边收缩变换(edge collapse transformation)来逐步迭代计算上述能量的优化过程。
- 下图给出了一个边收缩变换的过程，它将该边的两个端点 (v_s, v_t) 收缩为一个顶点 v_s 。经过这个变形后，其相邻两个面 $\{v_s, v_t, v_l\}$ 和 $\{v_t, v_s, v_r\}$ 均退化为一条边。



- 因此，初始网格$\hat{M} = M^n$可经过n组的边收缩变形后简化为M^0：

$$\hat{M} = M^n \xrightarrow{ecol_{n-1}} \cdots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0$$

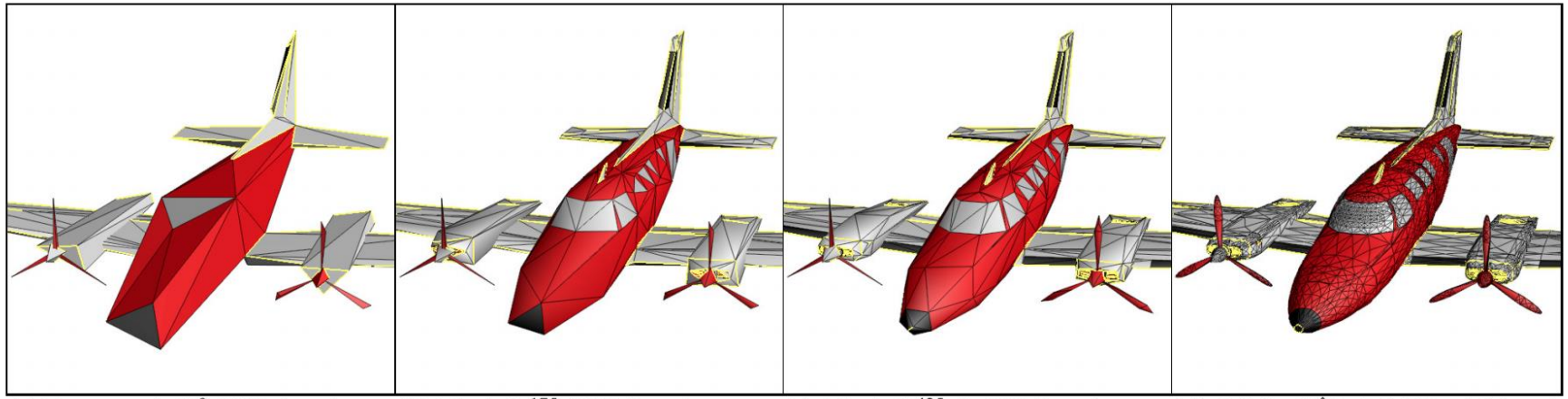
顶点分裂变换

- 顶点分裂变换(vertex split transformation)
  - 由于边收缩变换 *ecol* 是一可逆变换，我们称其逆变换为顶点分裂变换
- Hoppe利用刚刚提到的能量函数来选择这些变换，使变换前后两网格间的能量差 ΔE 达到最小

$$E(M) = E_{dist}(M) + E_{spring}(M) + E_{scalar}(M) + E_{disc}(M)$$

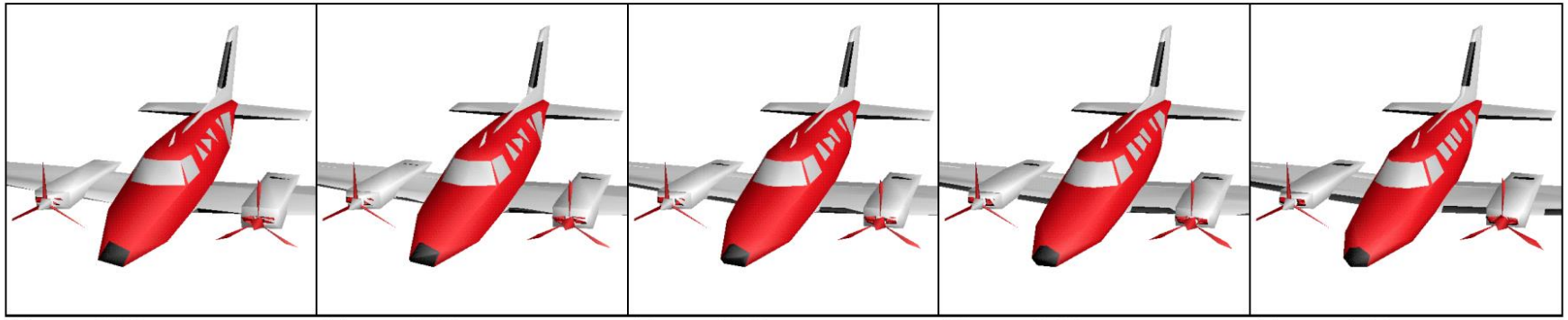
- 参考文献
  - Hoppe H, DeRose T, Duchamp T, Mesh Optimization, Computer Graphics, 1993,27-4,19-26.
  - Hoppe H, Progressive Meshes, Computer Graphics, 1996, 30-4, 99-108.

渐进网格简化：效果



(a) Base mesh M^0 (150 faces) (b) Mesh M^{175} (500 faces) (c) Mesh M^{425} (1,000 faces) (d) Original $\hat{M}=M^n$ (13,546 faces)

Figure 5: The PM representation of an arbitrary mesh $\hat{M}$ captures a continuous-resolution family of approximating meshes $M^0 \ldots M^n=\hat{M}$.

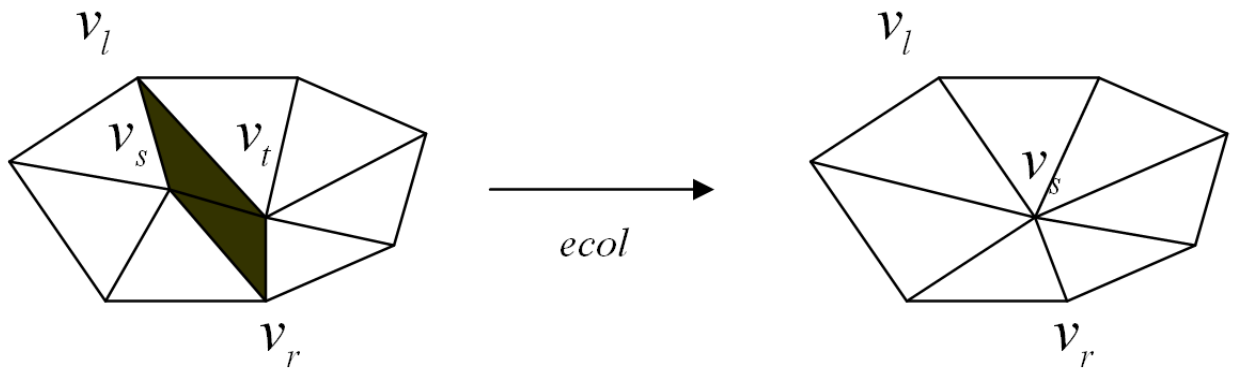


(a) $\alpha = 0.00$ (b) $\alpha = 0.25$ (c) $\alpha = 0.50$ (d) $\alpha = 0.75$ (e) $\alpha = 1.00$

Figure 6: Example of a geomorph $M^G(\alpha)$ defined between $M^G(0)\doteq M^{175}$ (with 500 faces) and $M^G(1)=M^{425}$ (with 1,000 faces).

基于二次误差度量的简化技术

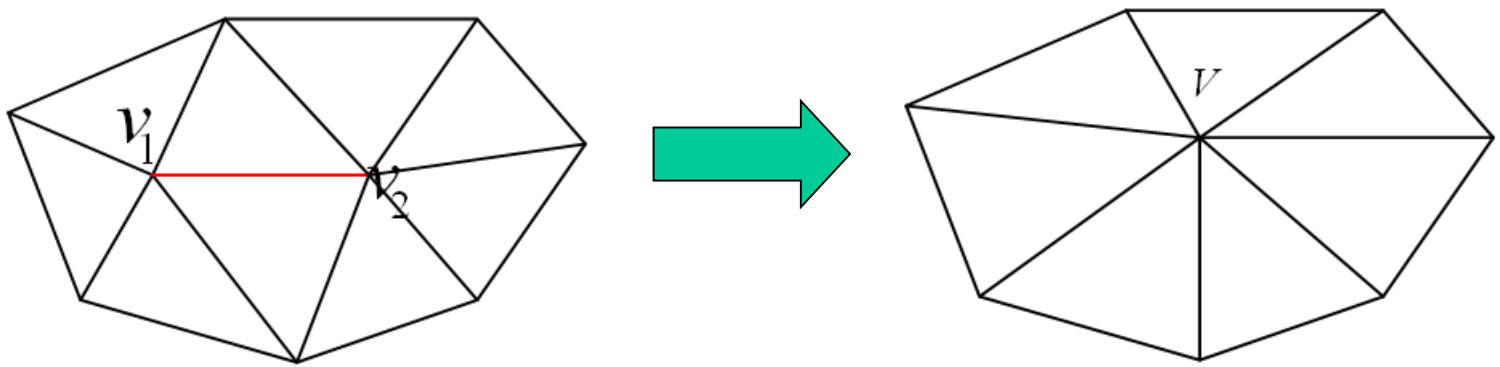
- Hoppe的边收缩(edge collapse)操作可推广为一般的顶点合并变换来描述$(v_1,v_2)\rightarrow v$，其含义是将场景中的两个顶点v_1,v_2 移到一新的位置 v，将连向 v_1,v_2 的所有边都连向 v，并删除所有退化的边和面片。



点对合并（1）

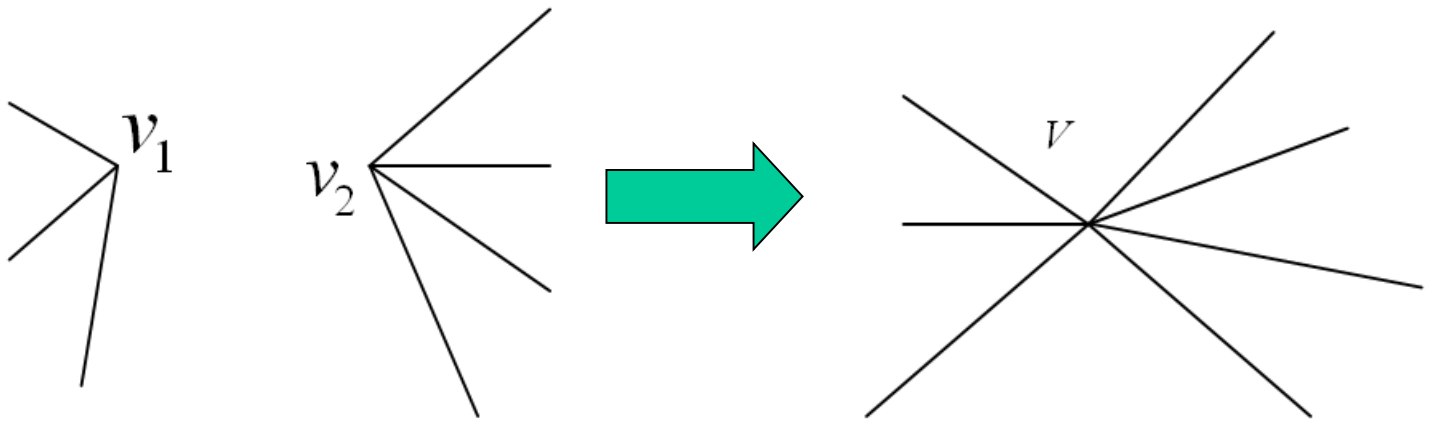
- 点对(v_1, v_2)合并的原则

  （1）v_1, v_2为某一表面上的相邻点，即 v_1, v_2 为一条边



点对合并（2）

（2）$\|v_1 - v_2\| < t$， t 为用户给定的阈值参数



二次误差度量

- Garland和Heckbert引进了二次误差度量来刻划每一个顶点移动后引起的误差。对表面上的每一个顶点v_a均有许多三角面片与之相邻，记 $plane(v_a)$ 为这些三角形所在的平面方程所构成的集合，即

$$plane(v_a)=\left\{(a,b,c,d)\,\middle|\,\begin{array}{c}ax+by+cz+d=0,(a^2+b^2+c^2=1)\\ \text{is coefficients of the adjacent plane of } v_a\end{array}\right\}$$

- 参考文献

Garland M., Heckbert P S., Surface simplification using quadric error matrix, Computer Graphics, 1997, 209-216.

- 则我们采用如下的二次函数来度量 v 移动到 v_a 时产生的误差：

$$\Delta(v_a \to v) = \sum_{p \in plane(v_a)} (pv^T)^2 \tag{1}$$

其中 v=(x, y, z, 1) 为齐次坐标，展开上式得到

$$\Delta(v_a \to v) = \sum_{p \in plane(v_a)} (pv^T)^2 = v\left(\sum_{p \in plane(v_a)} K_p\right)v^T = vQ(v_a)v^T \tag{2}$$

$$Q(v_a) = \sum_{p \in plane(v_a)} K_p$$

- （2）式中

$$K_p = p^T p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \qquad (3)$$

- 这样，对每一顶点 v_a ，在预处理时，我们均可按上述方法计算矩阵 $Q(v_a)$，进而就可对其移动进行误差度量了。但由于每次合并时，需同时移动两点，故必须考虑同时移动多个顶点后形成的误差

多点移动的误差

- Garland和Heckbert简单地采用加法规则来刻划多点移动而形成的误差，对点对合并$(v_1, v_2) \to v$，其误差为

$$\Delta(v) = \Delta(v_1 \to v) + \Delta(v_2 \to v) = v(Q(v_1) + Q(v_2))v^T = vQv^T$$

其中$Q = Q(v_1) + Q(v_2)$

因而，应选取$\boldsymbol{v}$使误差达到最小

- 由极值的性质知，v满足系统方程：

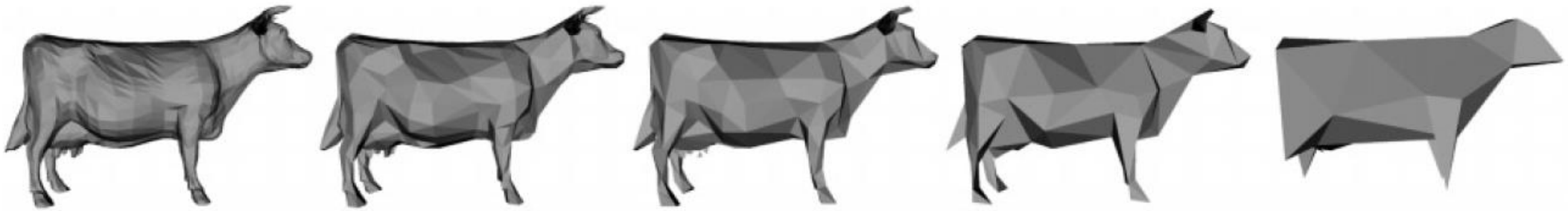
$$\frac{\partial \Delta(v)}{\partial x}=\frac{\partial \Delta(v)}{\partial y}=\frac{\partial \Delta(v)}{\partial z}=0$$

即

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

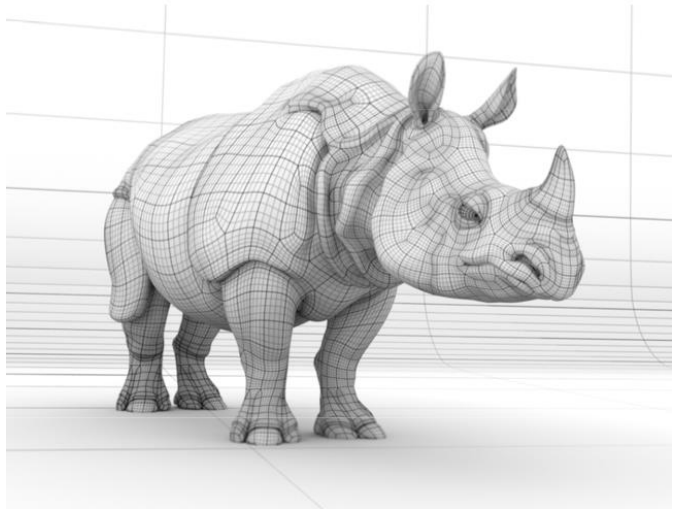
- 若上式有唯一解，则其解为v的最优解；否则，用伪逆技术求v
- 若伪逆技术失败，则简单地选取v为v_1, v_2或$\frac{v_1+v_2}{2}$中的任何一个

方法效果: Garland的牛



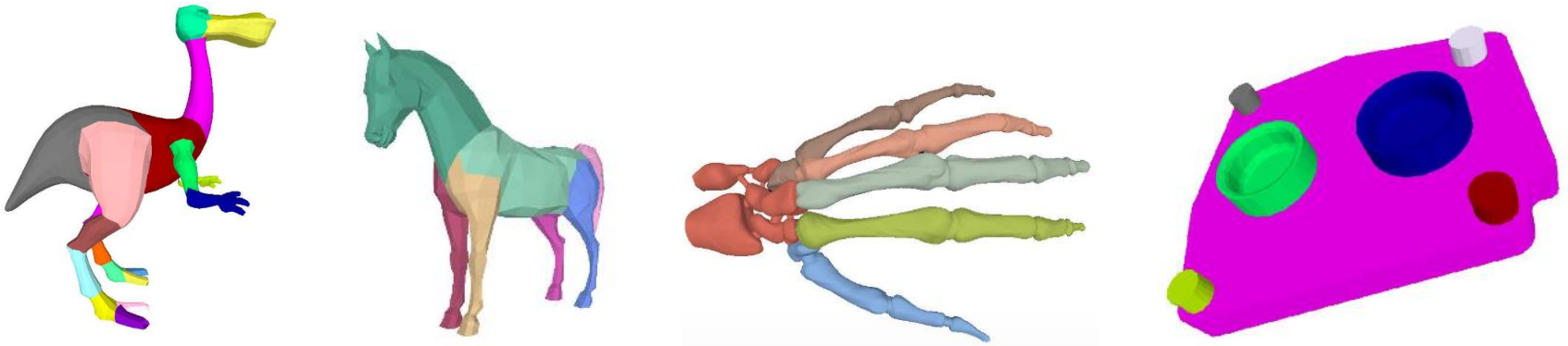
网格 (Mesh) 的细分、简化和分割

- 网格相关基本概念
- 网格细分 (Subdivision)
- 网格简化 (Simplification)
- 网格分割 (Segmentation)



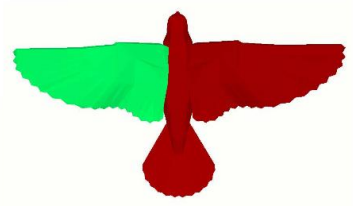
网格分割

- 将输入网格分割成小片（Patch），每个小片具有一定语义/几何信息。
- 应用：形状分析检索、骨架提取、碰撞检测、模型压缩简化等

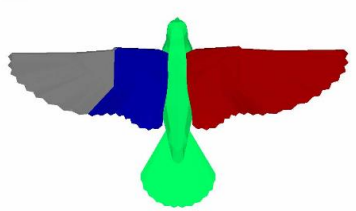


网格分片：定义

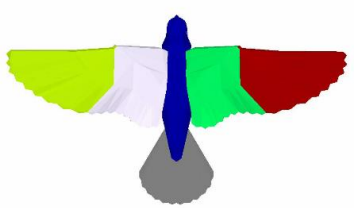
- k路分解：
  - 将原网格S分解成k个小片S1,S2,...,Sk，保证这些小片每个都是连通的且互不相交
- 0-1分解：k路分解中k=2的情况
- 层次化分解：



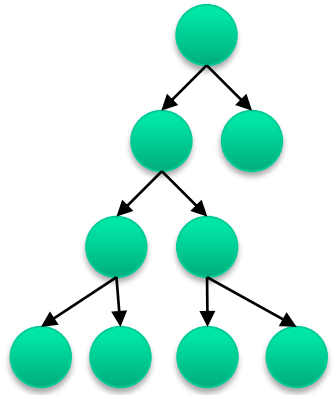
(a) first level



(b) second level



(c) third level



本节将主要介绍算法：

- Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts
  - Sagi Katz, Ayellet Tal. SIGGRAPH 2003
  - 引用量：928

我们首先介绍Katz的0-1分解方法，再拓展到k路分解

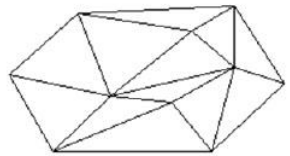
0-1分解：算法主要步骤

1. 生成带权对偶网格，及任意两个面片的最短距离。
2. 初始化2个种子。
3. (模糊聚类)使用种子更新面片概率，再更新种子。
4. 对最终的模糊区域运用最小割算法。

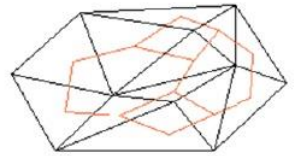
算法主要步骤

1. 生成带权对偶网格，及任意两个面片的最短距离。
2. 初始化2个种子。
3. (模糊聚类)使用种子更新面片概率，再更新种子。
4. 对最终的模糊区域运用最小割算法。

带权对偶网格

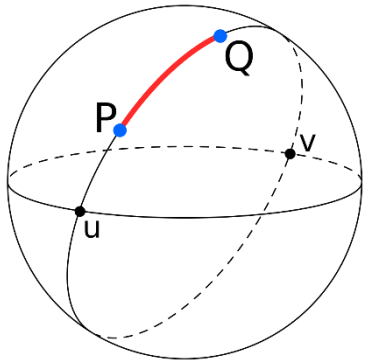


Simplex Mesh 2D



Dual Mesh

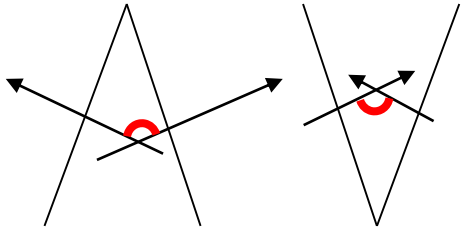
- 对偶网格 (计算几何)：
  - Mesh：顶点 – 边
  - Dual Mesh : 面被看做顶点 – 边翻转
- 测地线距离：
  - Geodesic distance : 沿模型表面的最短行进距离。
- 三维网格中边的凹凸性：
  - Convex: > 180°
  - Concave: < 180°
  - Co-planar: 180°



权值计算

- 对于相邻的面片f_i和f_j：
  - 角距离：$Ang\_Dist(\alpha_{ij}) = \eta(1-\cos\alpha_{ij})$.
    - 其中η的定义如右图所示
  - 测地线距离：$Geod(f_i, f_j)$
  - 最终权值为两个距离的加权和：

$$Weight(dual(f_i), dual(f_j)) = \\ = \delta \cdot \frac{Geod(f_i, f_j)}{avg(Geod)} + (1-\delta) \cdot \frac{Ang\_Dist(\alpha_{ij})}{avg(Ang\_Dist)}.$$



- 任意两个面的距离Dist(f_i, f_j)定义为对偶图中从f_i走到f_j的最短路径上的权值Weight之和

算法主要步骤

1. 生成带权对偶网格，及任意两个面片的最短距离。
2. 初始化2个种子：**选择距离最远的两个面片**
3. (模糊聚类)使用种子更新面片概率，再更新种子。
4. 对最终的模糊区域运用最小割算法。

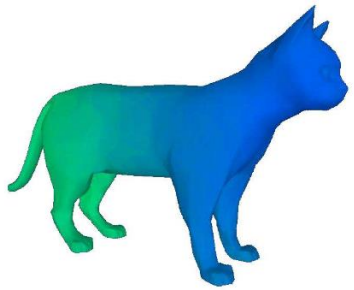
算法主要步骤

1. 生成带权对偶网格，及任意两个面片的最短距离。
2. 初始化2个种子。
3. (模糊聚类)使用种子更新面片概率，再更新种子。
4. 对最终的模糊区域运用最小割算法。

模糊聚类

- 模糊聚类 Fuzzy Clustering
  - 每个面片并非直接给出聚类标签，而是给出每个标签的从属概率
- 使用迭代的方式最小化如下能量：

$$F = \sum_{p} \sum_{f} probability(f \in patch(p)) \cdot Dist(f, p).$$



- 采用迭代方式：

$$REP_A = \min_f \sum_{f_i} (1 - P_B(f_i)) \cdot Dist(f, f_i)$$

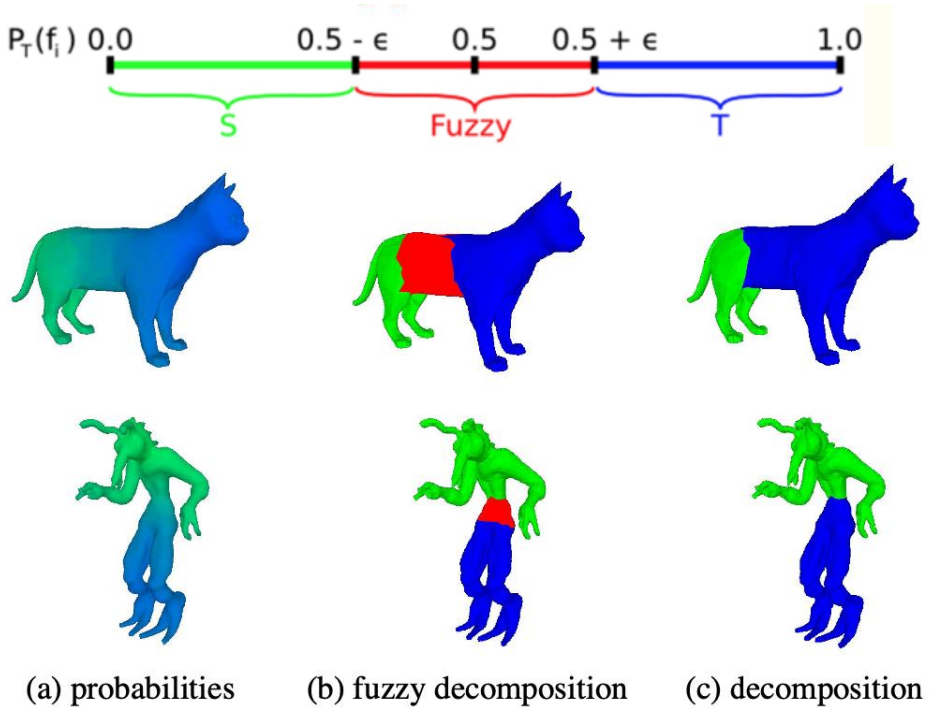
$$REP_B = \min_f \sum_{f_i} P_B(f_i) \cdot Dist(f, f_i).$$

$$P_B(f_i) = \frac{a_{f_i}}{a_{f_i} + b_{f_i}} = \frac{Dist(f_i, REP_A)}{Dist(f_i, REP_A) + Dist(f_i, REP_B)}.$$

更新种子点位置　　更新面片概率

确定模糊区域

- 根据上一步得出的概率，找出模糊区域进行进一步精化



(a) probabilities (b) fuzzy decomposition (c) decomposition

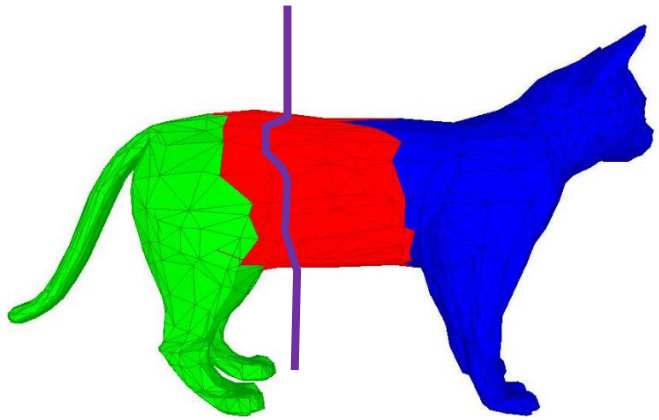
Figure 3: Binary decomposition

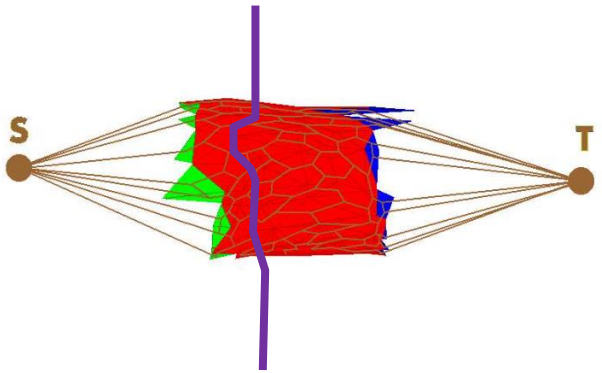
算法主要步骤

1. 生成带权对偶网格，及任意两个面片的最短距离。
2. 初始化2个种子。
3. (模糊聚类)使用种子更新面片概率，再更新种子。
4. 对最终的模糊区域运用最小割算法。

最小割算法

- 目标：在模糊区域中找到一条割线，从而分明地划清两个区域
- Graph中S到T的最大流值 = 最小割值
  - 算法举例：A new approach to the maximum flow problem





最小割算法：形式化定义

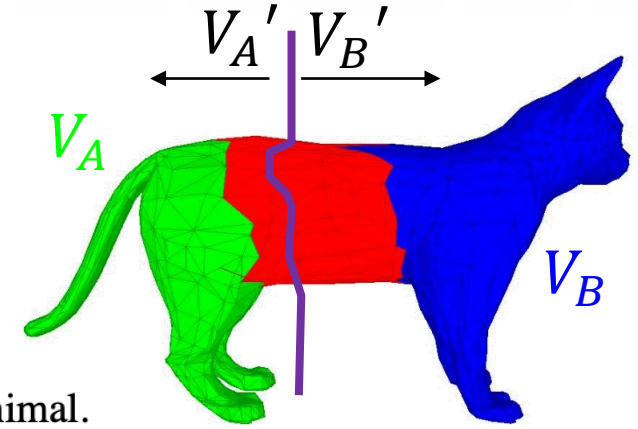
- 我们要求解的数学问题如下

$$(1)\quad V = V_{A'} \cup V_{B'}$$

$$(2)\quad V_{A'} \cap V_{B'} = \phi$$

$$(3)\quad V_A \subseteq V_{A'}, \qquad V_B \subseteq V_{B'}$$

$$(4)\quad weight(Cut(V_{A'}, V_{B'})) = \sum_{u \in V_{A'}, v \in V_{B'}} \omega(u, v) \quad \text{is minimal.}$$



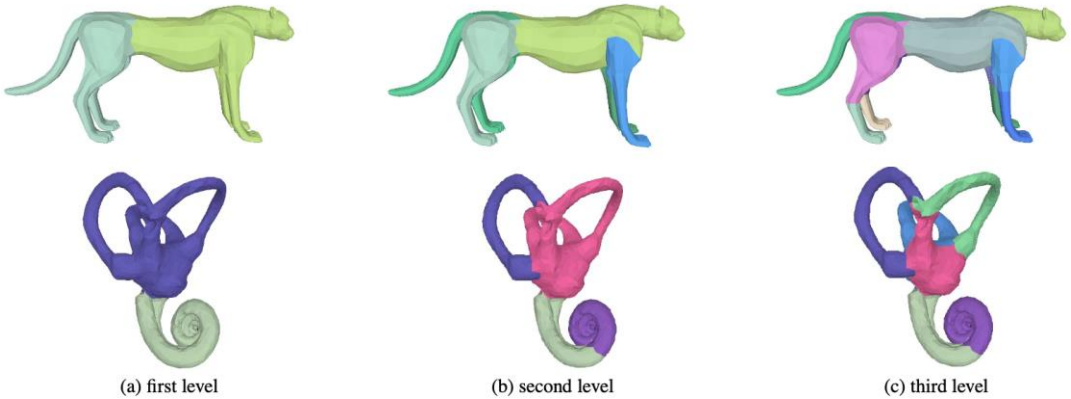
- 两个面片之间的权值/容量定义如下：

$$Cap(i,j) = \begin{cases} \frac{1}{1+\frac{Ang\_Dist(\alpha_{ij})}{avg(Ang\_Dist)}} & \text{if } \{i, j \neq S, T\} \\ \infty & \text{else} \end{cases}$$

- 该问题可以使用网络流等数值方法求解（习题课）

0-1分解 到 k路分解

- 0-1分解仅仅将网格分成两个部分，k路分解将其分成k个部分：
  - 如何确立初始的k个种子点？
  - 如何计算每个面片属于每个分片的概率？
  - 如何确定k？



(a) first level (b) second level (c) third level

Figure 8: Hierarchical k-way decompositions of a cheetah and an inner part of a human ear

k路分解

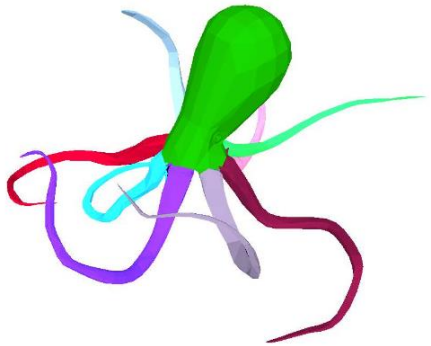
- 初始种子点选取：
  - 采用迭代式方法，第一个种子点选为距离其他所有面的距离之和最小的面片（这样是为了选择网格的“主体”区域）
  - 之后逐一添加剩下的种子点，使得第k个添加的种子点与前k-1个种子点距离的最小值最大
- 概率计算（拓展0-1分解的定义即可）：
  - 面片f_i属于区域p_j的概率为：

$$P_{p_j}(f_i) = \frac{\frac{1}{Dist(f_i, REP(p_j))}}{\sum_l \frac{1}{Dist(f_i, REP(p_l))}}$$

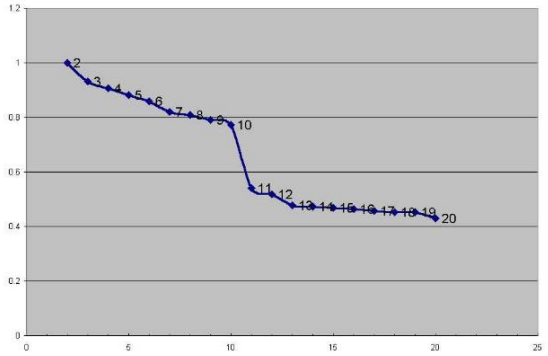
k路分解：确定k

- 采用G函数一阶导数的极值进行确定k，依据是寻找使得面片距离下降最多的分类种数

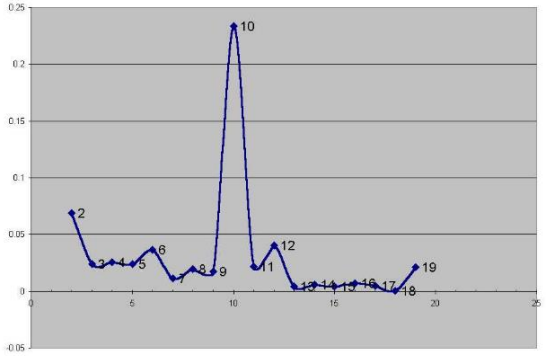
$$G(k) = min_{i<k}(Dist(REP_k, REP_i)).$$



(a) object



(b) function G



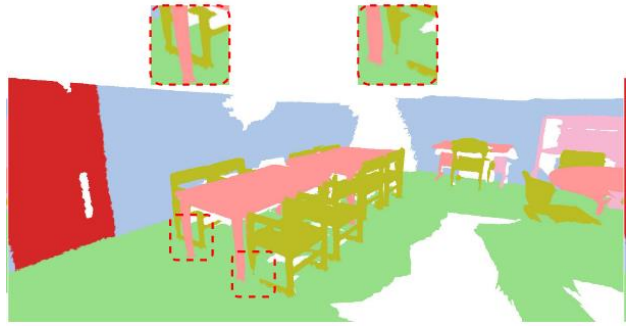
(c) first derivative of G

- G函数一定是单调递减的，我们希望能以尽可能少的分片数达到较好的分割效果

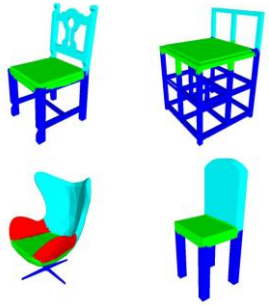
展望：深度学习时代的网格分割

- 随着深度学习技术的发展，通过大规模的数据学习，现有算法能够有效利用数据集中的语义、几何先验知识对三维数据进行更有效的分割。
- 此外，由于网格本身的不均匀性，研究者们也常常选用点云、体素、隐式函数场等方式对三维数据进行表示，并将分割结果投影到三维网格上，完成最终的分割任务。

MeshCNN
SIGGRAPH 2019



Voxel-Mesh Net
ICCV 2021



BAE-Net
ICCV 2019

展望：基于细分的网格卷积网络(SubdivNet)

三角卷积

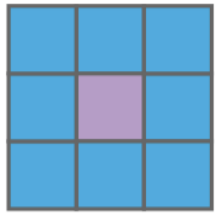
a).

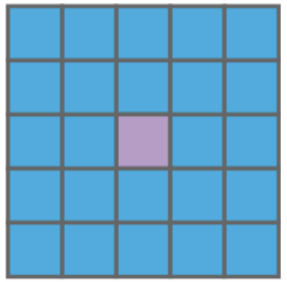
k=3, d=1

k=5, d=1

图像卷积

b).





k=3, d=1

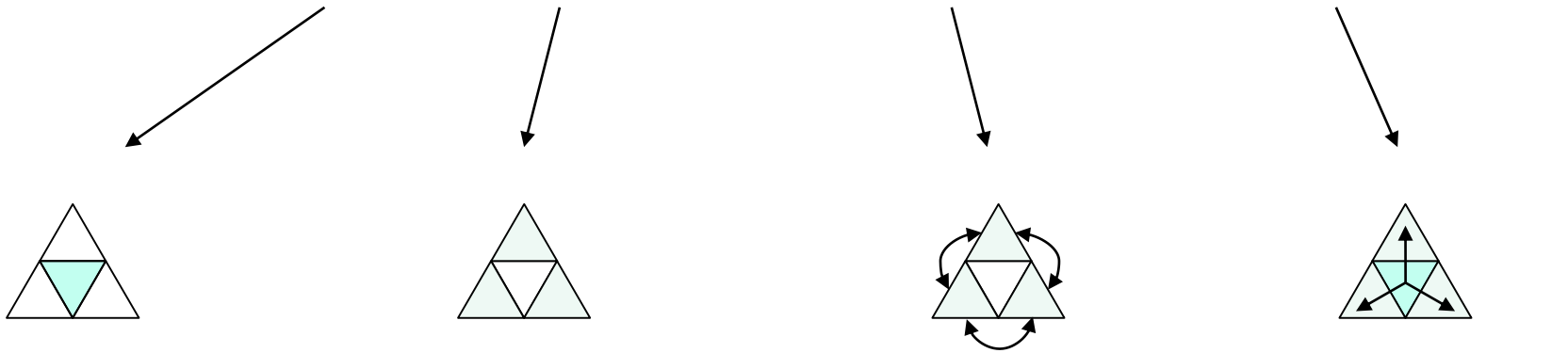
k=5, d=1

展望：基于细分的网格卷积网络(SubdivNet)

三角面片没有顺序，因此需要排列不变性:

构造顺序无关的中间特征，再进行加权

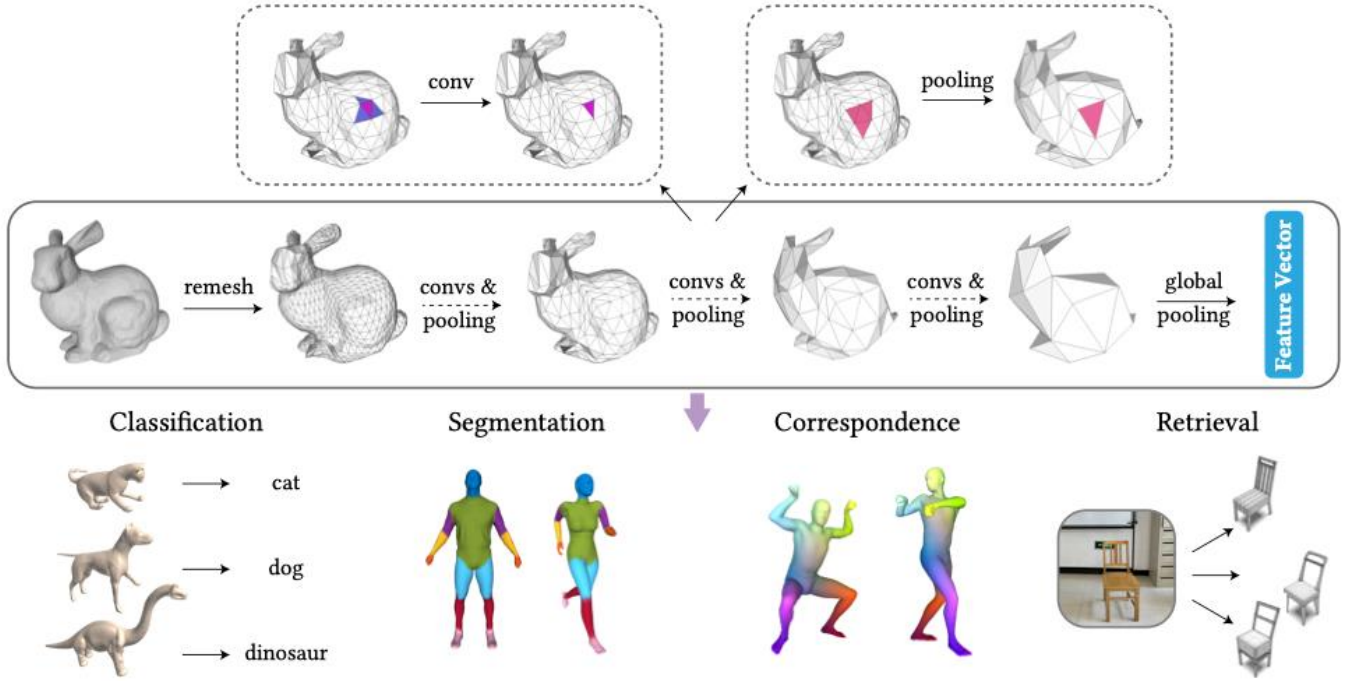
$$\mathrm{Conv}(f_i) = w_0 e_i + w_1 \sum_{j=1}^{n} e_j + w_2 \sum_{j=1}^{n} |e_{j+1} - e_j| + w_3 \sum_{j=1}^{n} |e_i - e_j|$$



中心特征　　邻域特征的和　　邻域差分的和　　中心与邻域的差的和

展望：基于细分的网格卷积网络(SubdivNet)

- SubdivNet：提供了简单、自然的三角网格上的卷积、池化方法



Hu, S. M., Liu, Z. N., Guo, M. H., Cai, J. X., Huang, J., Mu, T. J., & Martin, R. R., Subdivision-based mesh convolution networks. ACM Transactions on Graphics, 2022, 41(3), 1-16.

谢谢！

公众号：图形学与几何计算

清华大学图形学实验室