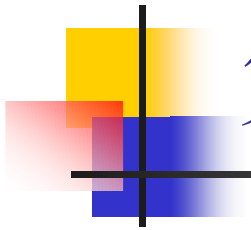
第7章 非真实感图形绘制

浙江大学CAD&CG国家重点实验室

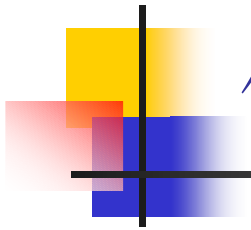
秦学英

2004年9月



第7章 非真实感图形绘制内容

- 什么是非真实感图形绘制?
- Toon 着色处理
- 轮廓边缘绘制 Silhouette Edge Rendering
- 其他风格
- 线条



什么是非真实感图形绘制 1

- Non-photorealistic Rendering (NPR)
  - 也叫Stylistic Rendering
  - 相对于真实感图形而言的

Using a term line ‘nonlinear science’ is like referring to the bulk of zoology as ‘the study of nonelephant animals’.

-Stanislaw Ulam

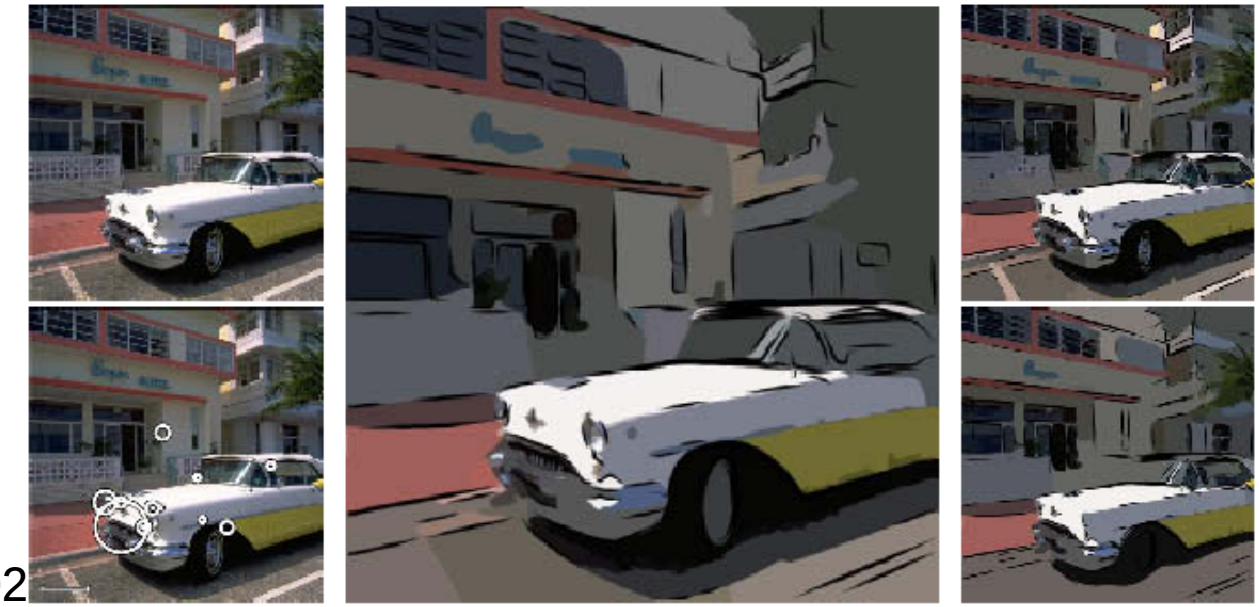
什么是非真实感图形绘制 2

- 真实感图形(Photo-realistic)试图生成拟似照片图像
- 非真实感图形(Nonphoto-realistic)与之相对，也称风格绘画

SIGGRAPH2002

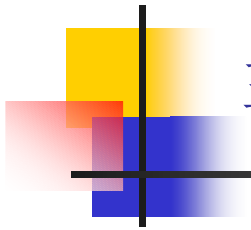
(a) (b) (c)

Figure 6: (a) A source image (1024 × 688) and fixations gathered by the eye-tracker; (b) the resulting line drawing ($c_{\text{scale}} = 0.05$, $l_{\text{min}} = 40$); (c) region boundaries before and after smoothing.



(a) (b) (c)

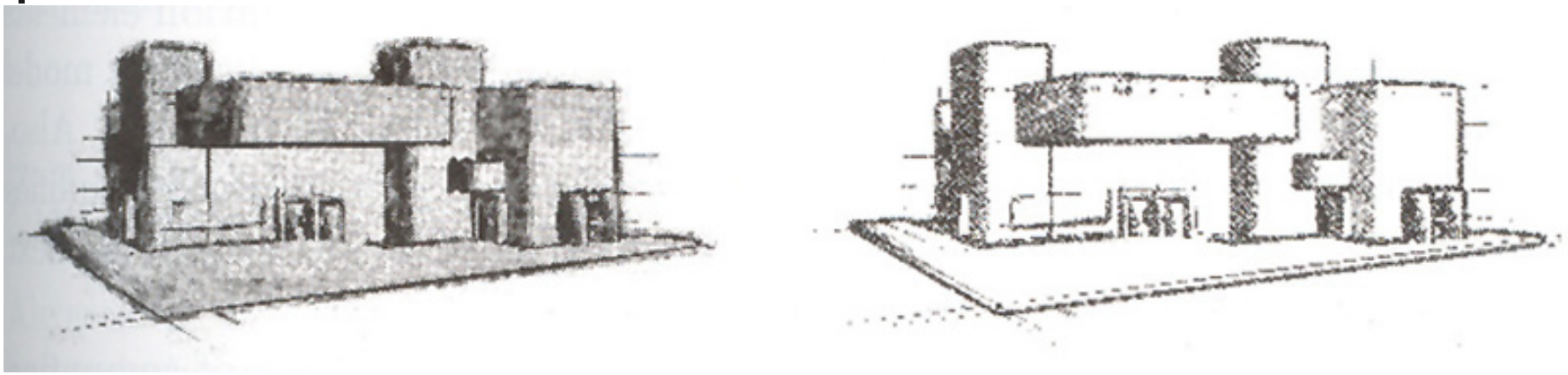
Figure 7: Comparison with and without eye-tracking data for the 768 × 768 image in (a). The drawing in (b) uses fixation data, and important details (as seen by the user) are retained ($e_{\text{outside}} = 40°$). The drawings in (c) instead use a constant eccentricity (3° on top, 12° on the bottom image) across the entire image so that no meaningful abstraction is performed. (All use $c_{\text{scale}} = 0.14$, $l_{\text{min}} = 15$.)



非真实感图形绘制的目标

- 技术性图解 Technical Illustration
  - 为了小车的商业性贩卖，其发动机的光眩图片会更有效
  - 为了修理发动机，显示一个发动机的结构，则线框图会更有效
- 显示绘画风格
  - 钢笔画
  - 水彩画
- 这一章的目标是概述NPR的实时绘制技术

什么是非真实感图形绘制 3

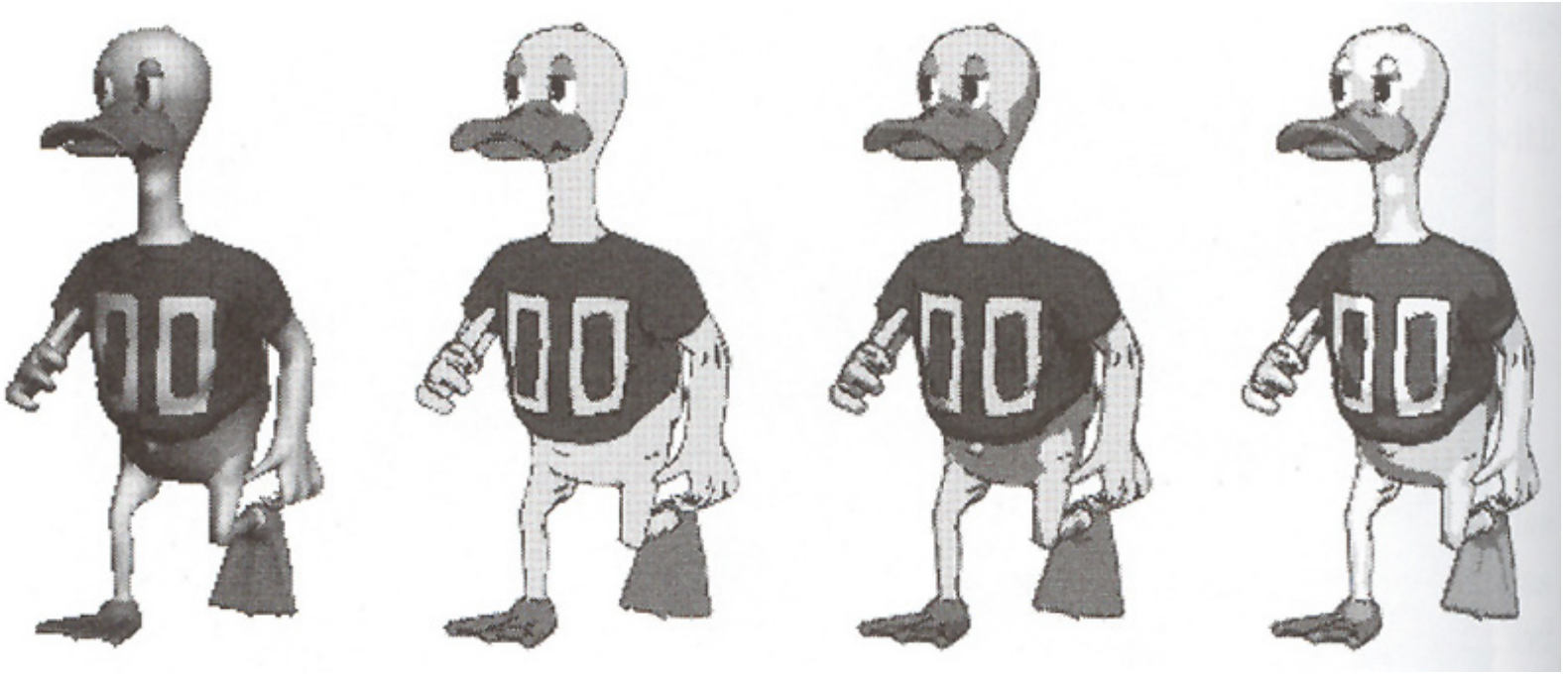


两种风格的非真实感图形：用Viewpoint LineStyle生成

Toon Shading

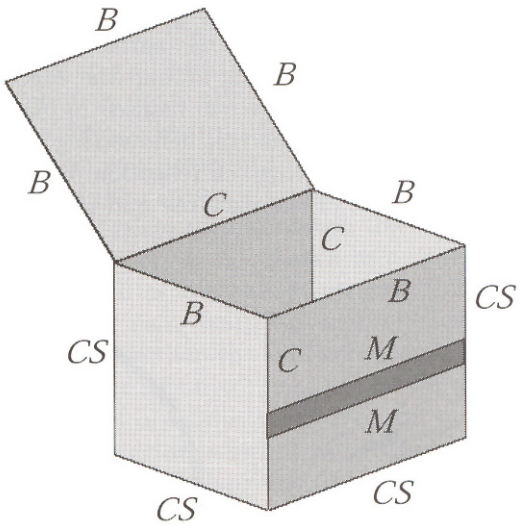
- McCloud in Understanding Comics:
  - Amplification through simplification

- 左图：Gouraud-shaded duck
- 其它：silhouettes rendered, with solid shading, diffuse two-tone shading, and specular/diffuse three-tone



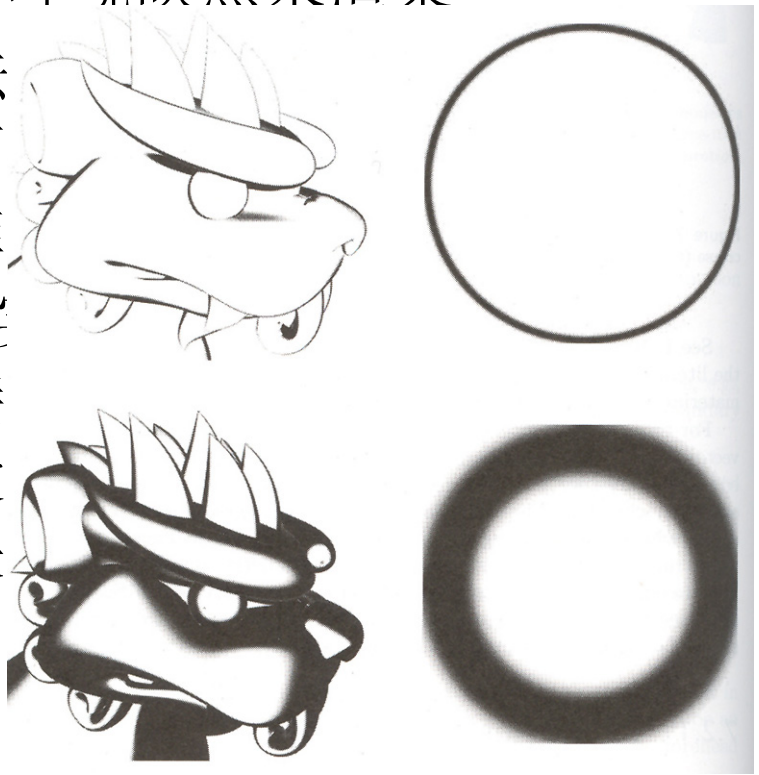
Toon Rendering的边界绘制风格

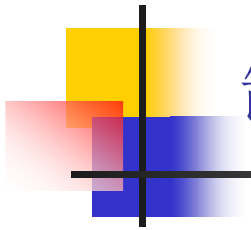
- boundary or border edge: 仅为一个多边形拥有的边界。因此，一个立体造型没有这样的边界
- crease or hard edge:为两个多边形共有的边界，且两个多边形的夹角（称为 dihedral angle) 大于一个阈值。这个阈值的参考值是60度。这种边界又分为ridge（脊）valley（谷）。
- material edge: 不同材质的两个多边形的公共边界。也可以是艺术家希望总是显示的线条。
- silhouette edge: 相对于某个方向而言，面向不同朝向的两个多边形的公共边界



7.2.1 面角轮廓surface Angle Silhouetting

- 轮廓线可以由视线方向和表面法向的内积判定
  - 当内积趋向于0时，就说明多边形靠近轮廓
  - 这种技术类似于用带黑边的环境映照来渲染
  - Marshall[521]实现了此算法
    有计算反射光的方向，取而
    点法向的内积，来决定一维
  - Everitt[215]用mipmap表来
    的用黑色绘制。当多边形接
    层，然后呈现黑色。由于没
    很尖锐，但其计算速度极快



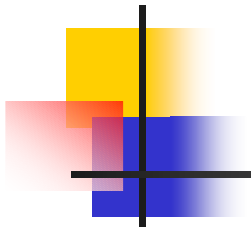


缺点

- 需要满足条件，即对表面法向与视线方向的内积与轮廓线的关系保持。因此，不是对所有的模型都效果很好，如立方体就会失败
- 轮廓线绘制宽度是变化的，其宽度依赖于表面的曲率
- 巨大的多边形在靠近边界时会全变为黑色，这往往不是我们需要的效果
- Wu在实现游戏Cel Damage时发现仅有1/4的模型是好的，其余都失败了

7.2.2 过程几何轮廓 Procedural Geometry Silhouette

- Basic Idea
  - 正常绘制front faces
  - 在绘制back faces时，使轮廓显示
- 几种方法：所有的方法都已先绘制front faces到Z-buffer,然后仅绘制back faces
  - 仅绘制back faces的边，而不是面[661,462]
    - 用偏移bias或7.4中的技术来使其可见
    - 适用于单像素宽度的轮廓线



- 如果要使线条更宽一些，可以用黑色绘制back faces
  - 必须要将Z-buffer中的值进行偏移，否则back faces将全不可见
  - 偏移值有几种选择：
    - 固定值
    - 与Z值具有非线性关系
    - 使用函数glPolygonOffset
  - 问题是所有的这些方法都不能使边的宽度相同，一个解决的办法是，考虑其相邻的面的法向

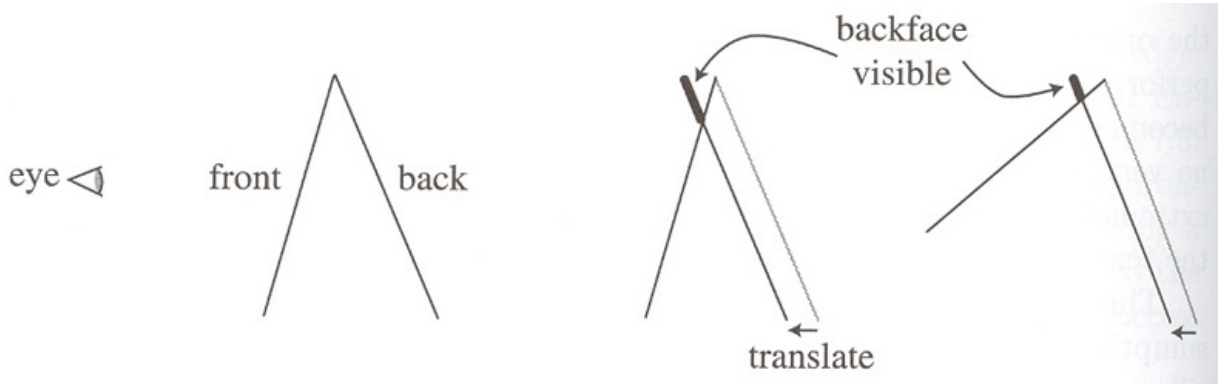


图7.6

- Back face的斜率可用来决定其偏移量。如图7.6示，其可见部分，即链条的粗细仍然与front face的夹角相关。
- 三角片的斜率和与视点的距离决定back face的偏移量，使线条的粗细正是所需要的[643,644]

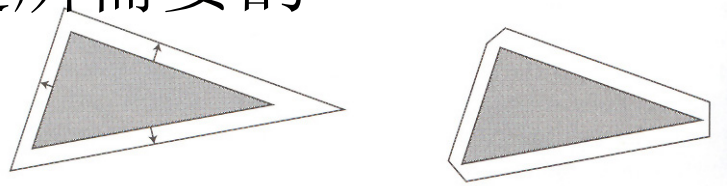


图7.7

绘制效果

后线条法
Think lines

Z值偏移
Z bias

多边形肥胖法
Fattened triangle algorithm

背面用线条绘制

其它方法 – shell technique

- 以上方法都是在多边形本身的平面上进行扩展，另一种方法就是，沿着顶点的平均法向，根据其历史点的远近，移动back face的顶点坐标[323]

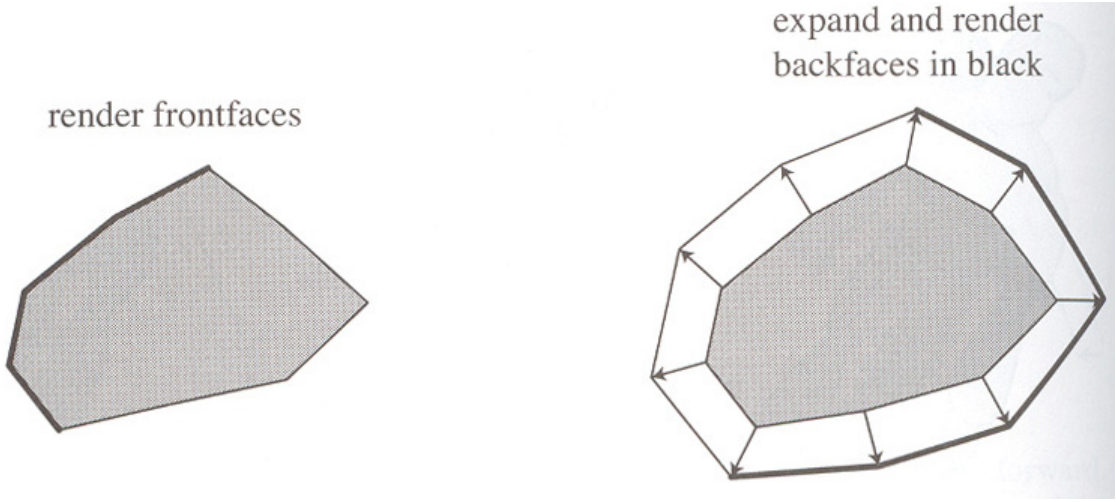
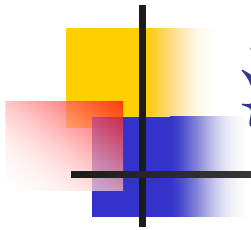


图7.9

绘制效果



潜在的缺陷

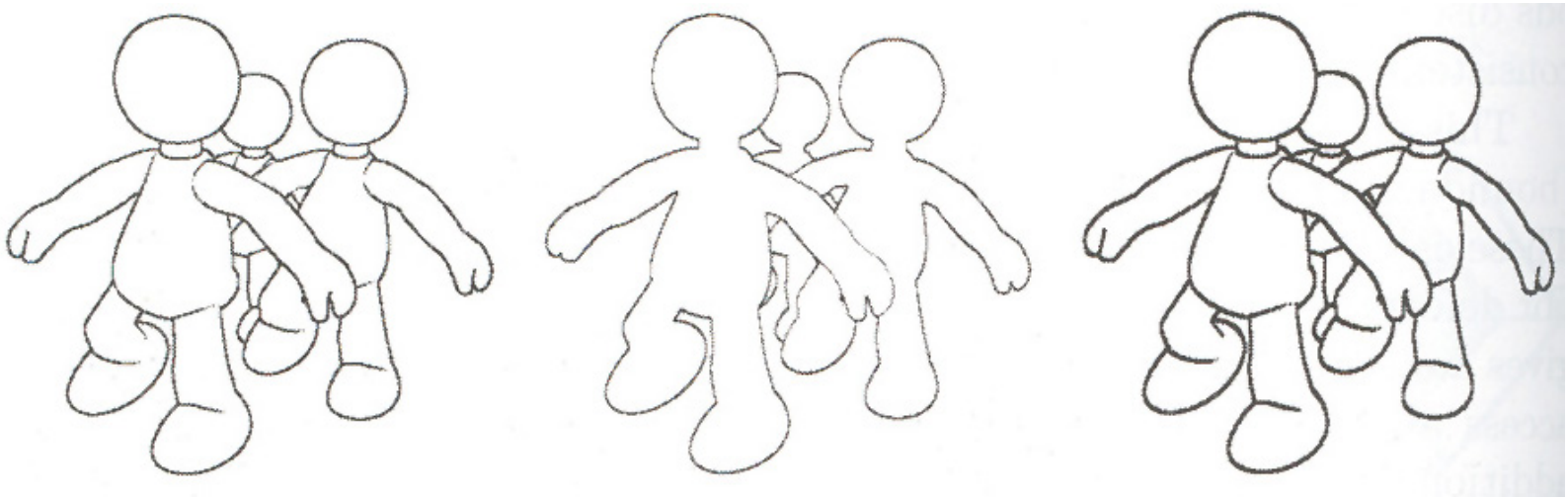
- 在某些情况下会形成空洞，因为扩展后的多边形并不真正形成一个封闭的壳[826]
- Shell and fatterning techniques 浪费资源：大量像素参与计算，但是仅有少量的像素真正绘制了。尽管对于现代的加速器而言，这还不是真正的问题。但是，fattening technique对于曲面不工作，而shell technique仅当曲面能沿法向向外平移时；Z-bias适用于所有曲面
- 边的各种效果绘制很困难，如半透明效果，反走样处理
- 这些方法的优点是，不需要边的连接信息。但是，在绘制之前，必须做预处理以保证多边形的连贯性

7.2.3 通过图像处理生成轮廓 Silhouetting by Image Processing

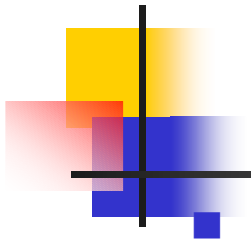
- NPR可以对基于各种缓冲区的信息进行处理而获得[668,163]。
  - 基于缓冲区的图像，与几何信息完全无关
  - 通过寻找Z-buffer中深度值的连贯性，可以找到大部分的轮廓线
- Card and Mitchell[112]的方法
  - 用pixel shader将场景的世界坐标下的法向和Z-深度值写到纹理中，法向作为法向图(normal map)写到颜色通道，Z值写到Alpha通道
  - 寻找轮廓线、边界和脊线：

绘制结果

- 例中法向图以足够找到所有的边



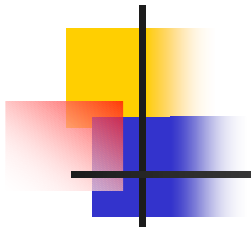
法向图　　Z-深度图　　合成图



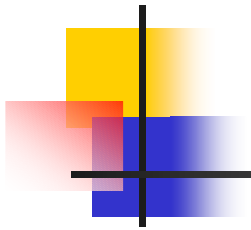
- Mitchell[557]描述了怎样用vertex 1.4 pixel shader来寻找边界，使此计算更加切实可行
- 优点：
  - 对任意类型几何物体有效，包括曲面
  - 多边形网格不需要连接，或保持连贯
  - 不需要使用CPU来创建边表
- 缺点：
  - 对于接近边缘的面，Z值比较的过滤器可能错误地探测到轮廓线
  - 如果Z值差异太小，有可能丢失轮廓线，如纸面[163]
  - 当场景的Z值范围太大时，降低了深度的分辨率，从而使轮廓线遗失[112]。

7.2.4 轮廓边缘检测 Silhouette Edge Detection

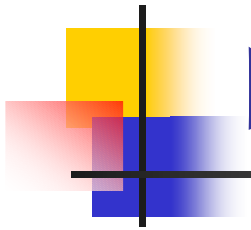
- 算法缺陷
  - 需要两遍绘制来获得轮廓
  - 对几何法(procedural geometry)而言，背面通道(back facing pass)需要处理比实际可见像素多得多的像素
  - 如果需要厚一点的边，可总问题都会发生，并且对风格缺乏控制
  - 图像方法也有相同或相似的问题
- 考虑直接绘制轮廓边：
  (n0.v>0)!=(n1.v<0)



- 标准方法
  - 对所有的物体边界实施上述检查[521]
- 改进：将平面内的边界剔除，因其不可能是轮廓边
- 决定轮廓边的有效方法[105]：
  - 检查面是朝向/背向视点，在其所有边表上标志，如果一条边被设定为front/back, 那么一定是边界。2bit都被设定的一定是边界
  - 如果没有boundary边的话，只需一个bit就够了
  - 避免了被两个面重用的边，其所在面的重复检测
  - 利用作背面剔除的判断，设定多边形的每条边界

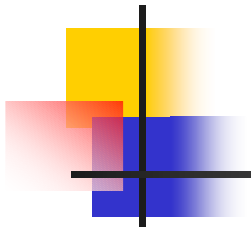


- 快速查找轮廓边的随机搜索方法
  - 先找到一条轮廓边，然后搜索与其顶点相连的边，直到所有轮廓边都被搜索到
  - 如果两个面间夹角很小，那么其公共边可能仍是轮廓边
  - 优点：不需要特别的数据结构，速度快(5倍)
  - 缺点：有时会丢失轮廓边

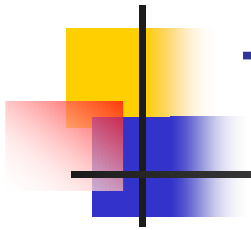


隐含轮廓边的探查

- 其瓶颈在于对无需的内存的访问
  - 对表面、边顶点同时进行排序是极为困难的[826]
- 使用vertex shader来工作[112]
  - 每一条边都被作为退化的四边形送进vertex shader，两个相邻顶点的法向也一并送入
  - 当一条边成为轮廓线后，四边形的点被删除，使其不再退化。即使其可见
  - 边界边(boundary edge)上也可以输入，其第二法向是面法向的反向。这样，这条边就总是被绘制
  - 缺点：大量增加了边的数量。如果网格进行非线性变换时，效果不好

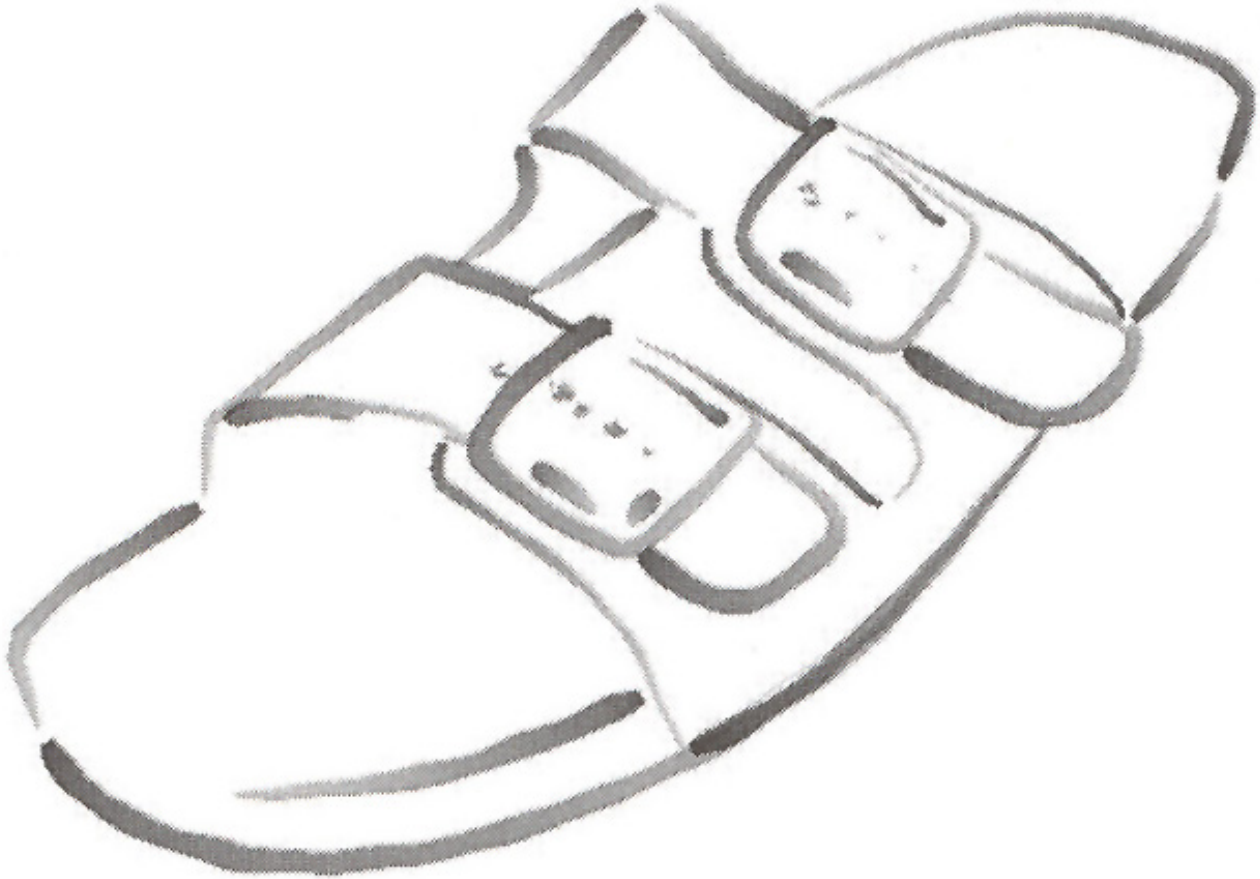


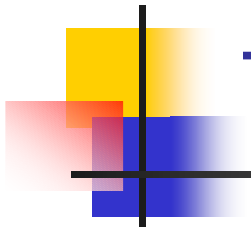
- 一旦轮廓线找到，边就被绘制。这些边可以使用：线绘制、带纹理的imposter 、或其他的任何方法
- 某种方式的偏移仍然是必需的，以保证它们画在面的前面，使其可见
- 其缺点是：这种方法强调了模型的多边形特征，即：使轮廓线很明显地由直线构成。[453]给出算法使其看起来像曲线。
- 另外，由于在CPU中多边形面的存在，对混合顶点的N-patch和其它加速生成的表面都不工作



7.2.5 混合轮廓线

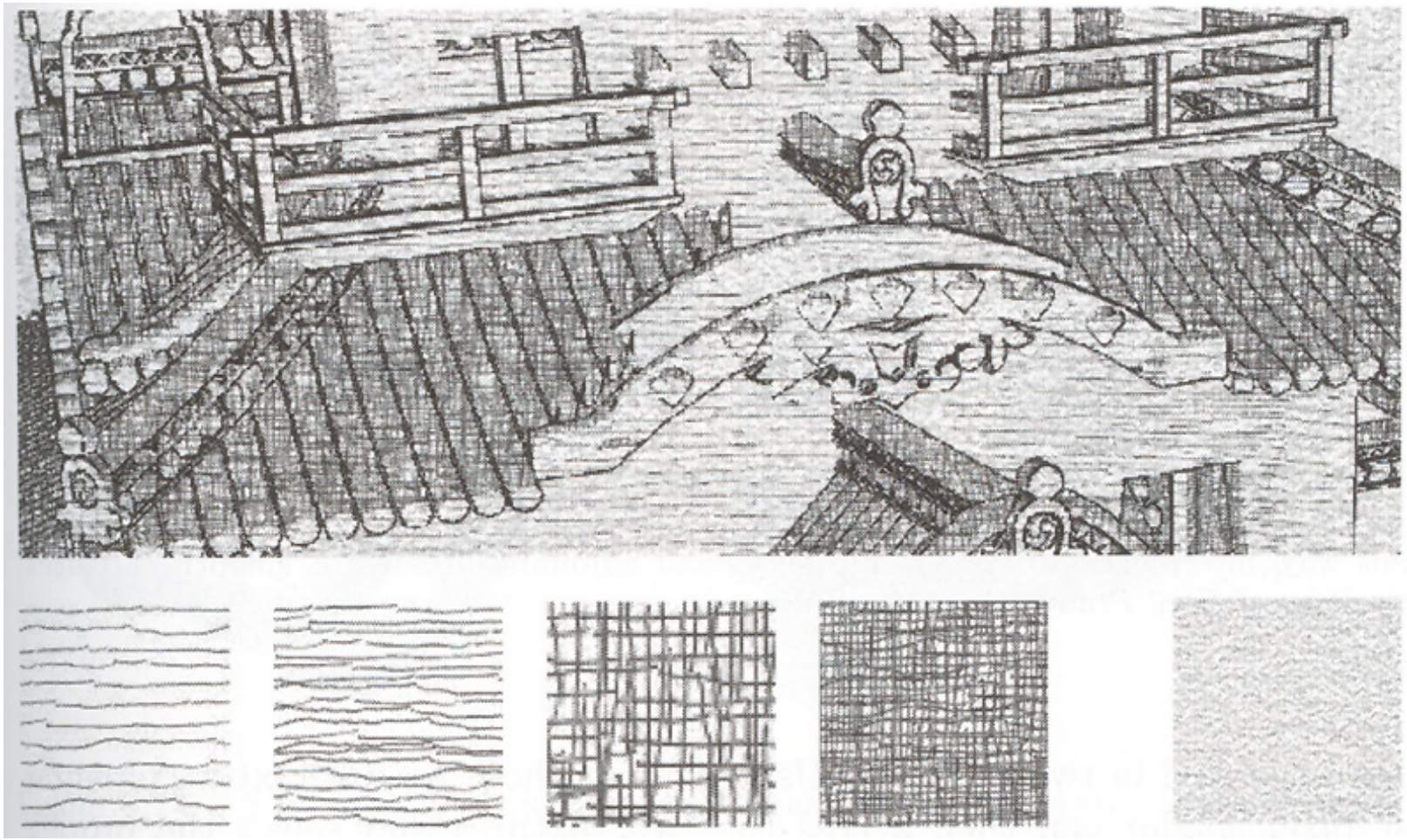
- 用图像与几何元素方法来获得轮廓线[589]
  - 先找到一轮廓线边列表
  - 给所有的三角形和轮廓边一个ID颜色，并画入
  - 读出这个buffer，这样所有的边表都有了
  - 可见边表在进行光滑等其他处理

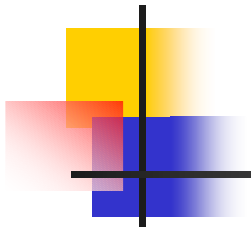




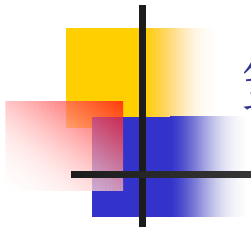
7.3 其它风格

- 除了toon rendering是一种受欢迎的风格，仍然有很多其它风格的绘制。NPR技术从编辑/修改(modifying)真实的纹理(realistic texture)[432,412,470],到设计算法以逐步产生帧到帧的装饰图案生成[402,518].在这一节，我们额要地介绍实时绘制的技术。其中有很多非交互的NPR技术，在有pixel shader和vertex shader的情况下，必将达到实时。



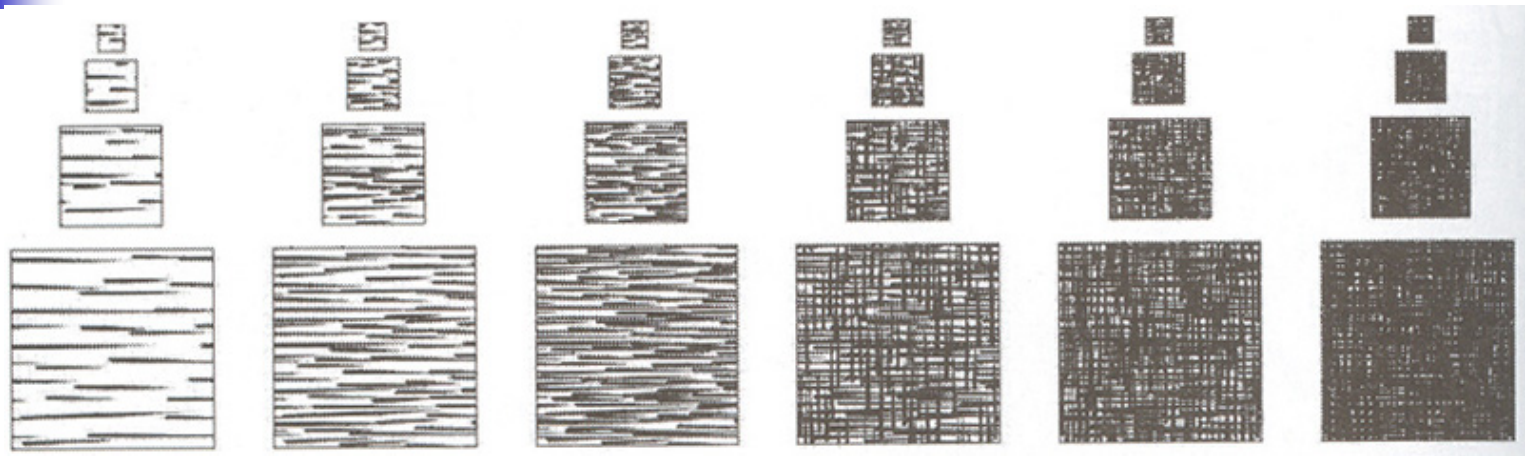


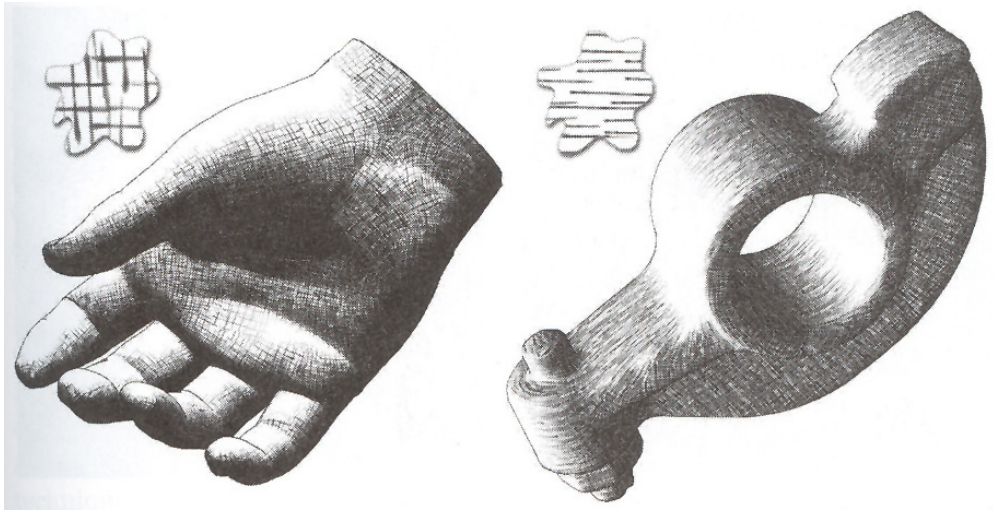
- 用不同的纹理来代替原图中的不同灰度的物体[453]
  - 纹理坐标是屏幕坐标，因而当视点移动时会发生漂移现象
  - 缺点是，如果一个多边形由两种纹理，必须对此多边形进行剖分
  - 可以采用多层纹理的方式避免这种缺陷[464]
  - 类似于7.1，修改alpha通道中的内容
- 对于灰色纹理[633],可以用纹理的不同通道来存储。比如，6幅不同的灰度图可以存在2个纹理中。这样多纹理通道可以选择合适的通道进行显示。通过pixel shader, 其显示不会浪费资源。

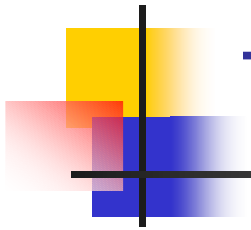


笔划stoke

- Praum *et al.*[633]
  - 将笔划texture(stroke texture)存于mipmap texture中。其基本思想是用mipmap中的texture来绘制笔划，这样会产生手绘的效果
- Cend&Mitchell[112]利用pixel shader给出了一个有效的实现方法
- Girshick et al. [259]讨论了principle curve方向线的方法
- 移植风格 (Greftal Style)[402，518]

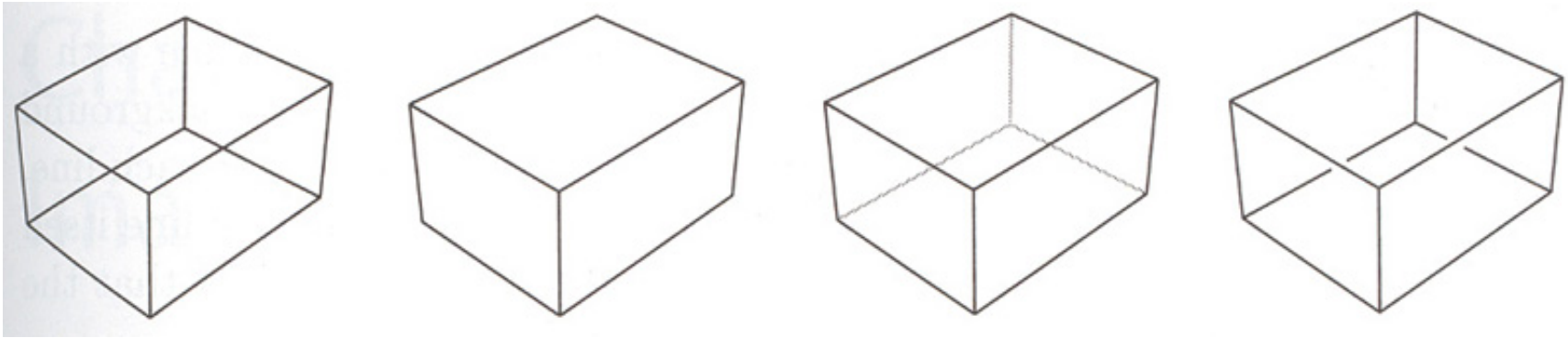






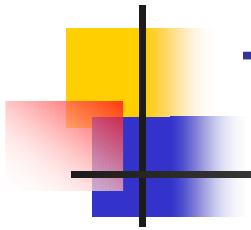
7.4 线条lines

- 边的辅助照明(Edge Higlighting)
  - 这是一种视点固定的算法，比如当光标经过一个物体的边时，将此边绘制为一种特殊颜色，如红色；当光标移开后，变为灰色
  - 这种方式在二维描画或仅对边绘制时很有效
- 多边形绘制
  - 在已绘好的多边形场景中，加绘边会更困难，因为会需要保证边一定在多边形的前面。对边进行一点偏移往往是有效的[56,536],但也有例外。



- 多边形绘制
  - glPolygonOffset()可以解决此问题
  - [356]中的方法可以避免平移
    - 在下述状态下填充多边形
      - 将z-buffer深度测试打开
      - 将z-buffer复位并关掉
      - 将color buffer打开
    - 将所有开关打开，再绘制一遍所有的边
    - 在下述状态下再填充一次多边形
      - 将所有z-buffer打开
      - 将color buffer关闭
    - 此方法看起来不错，但是对edge highlighting效果不好

- 正常绘制多边形，同时在stencil buffer中标出多边形的区域
- 绘制多边形的边。如果一个像素在stencil buffer中没有标出，那么正常地比较Z值，然后写入；反之，如果已标出，就不用比较，总是用边界的Z值代替
- 清除stencil buffer的值



7.4.3 隐藏线绘制

- 直观的方法是，用背景颜色对多边形进行填充，同时绘制边界。另外一个更快的方法使，将多边形画入Z-buffer，但是不在颜色buffer，然后再画出边界。第二种方法避免了不必要的颜色填充
- 隐藏线可以用灰色来绘制，用以区别实现，同时可见其形状
  - 先用灰色画出所有的线条
  - 在将多边形深度画入
  - 最后用深色画出可见线