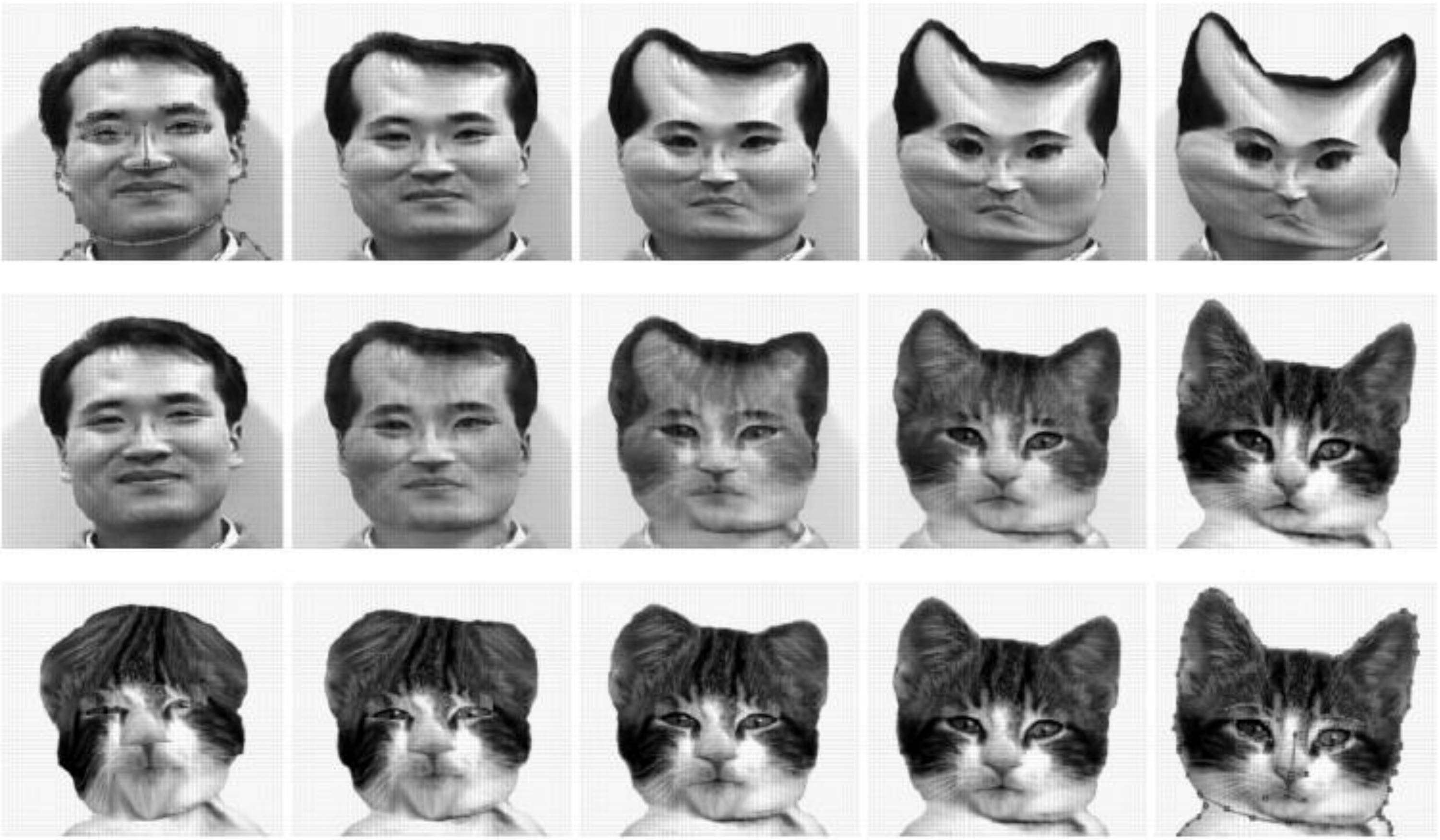
第六章 Morphing和空间变形动画

Morphing

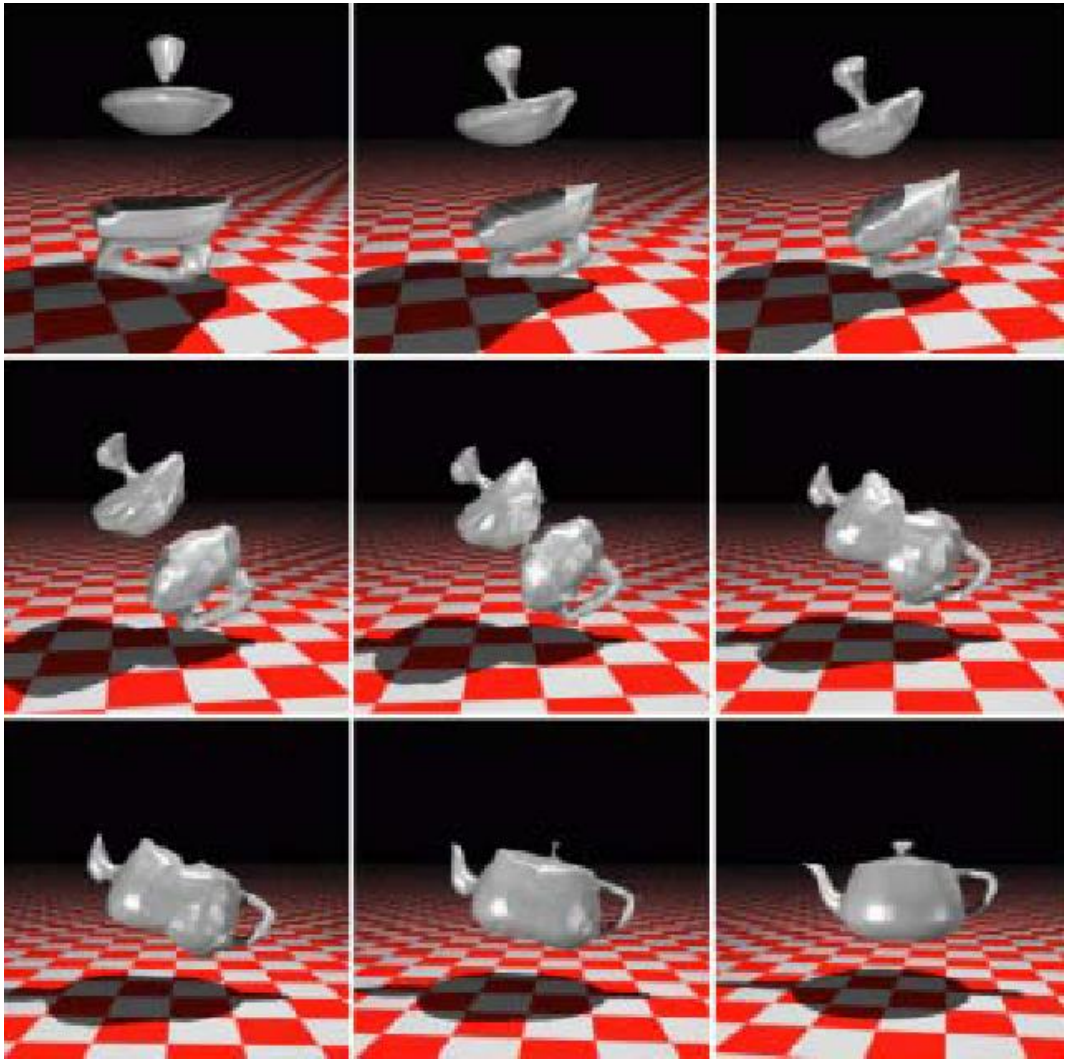
- 形状渐变(形状过渡)
  - 将一给定的源对象S光滑地变换到目标对象T
  - 中间对象既有 S 的特征,又有 T 的特征
  - S 和 T 可以具有不同的拓扑
  - 2D 对象<---> 数字图像
  - 3D 对象<---> 几何模型

Image Morphing

Image Morphing

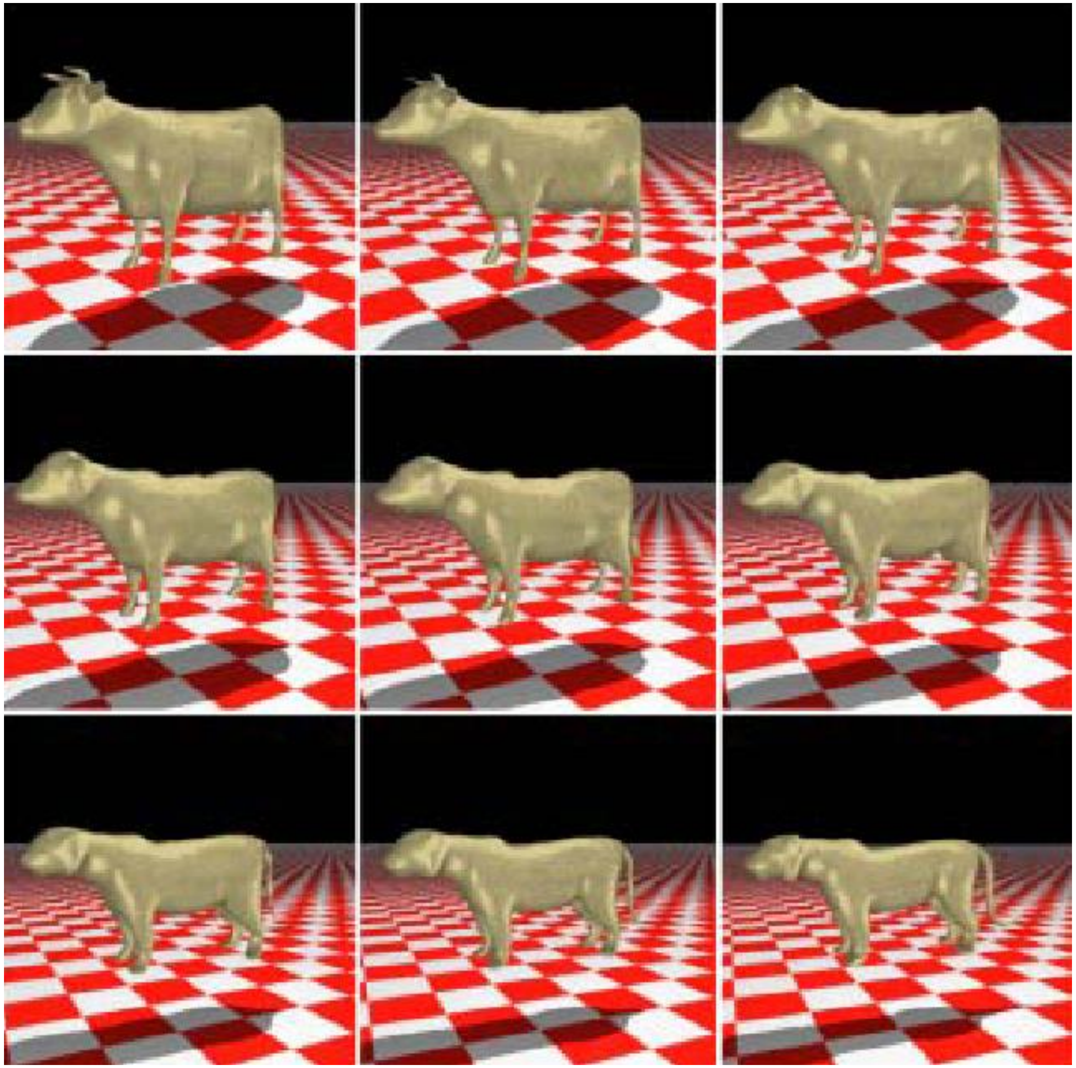


3D Morphing



3D Morphing

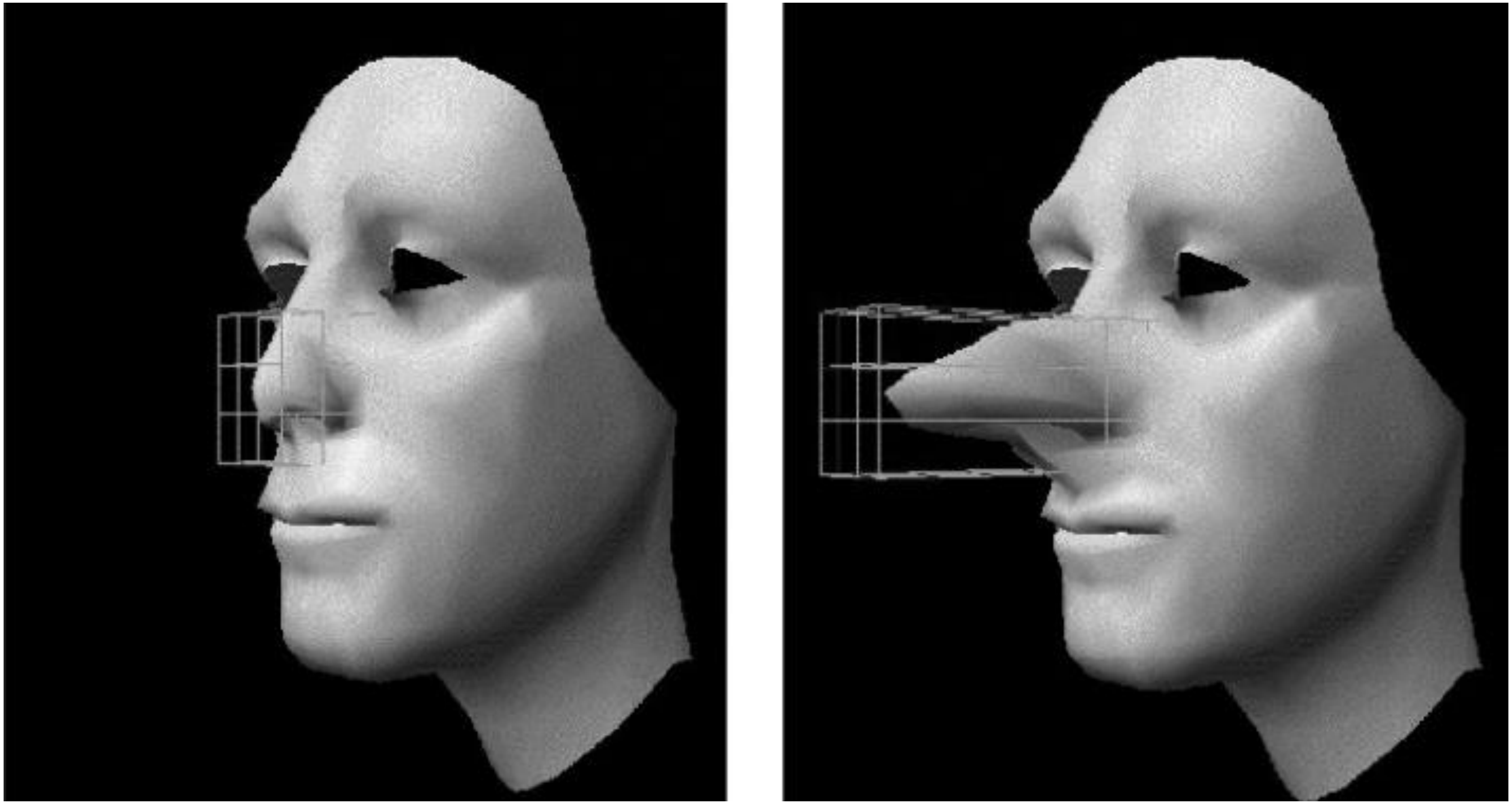
next section



Deformation---空间变形

- 将单个几何对象的形状作某种扭曲，变形， 使它变换到动画师所需要的形状
- 变换过程中，几何对象的拓扑关系保持不变
- 空间变形通常用较少的点去控制较多的点，要求提供的方法直观，简单，方便，控制灵活

Deformation



第六章 Morphing和空间变形动画

Image Morphing

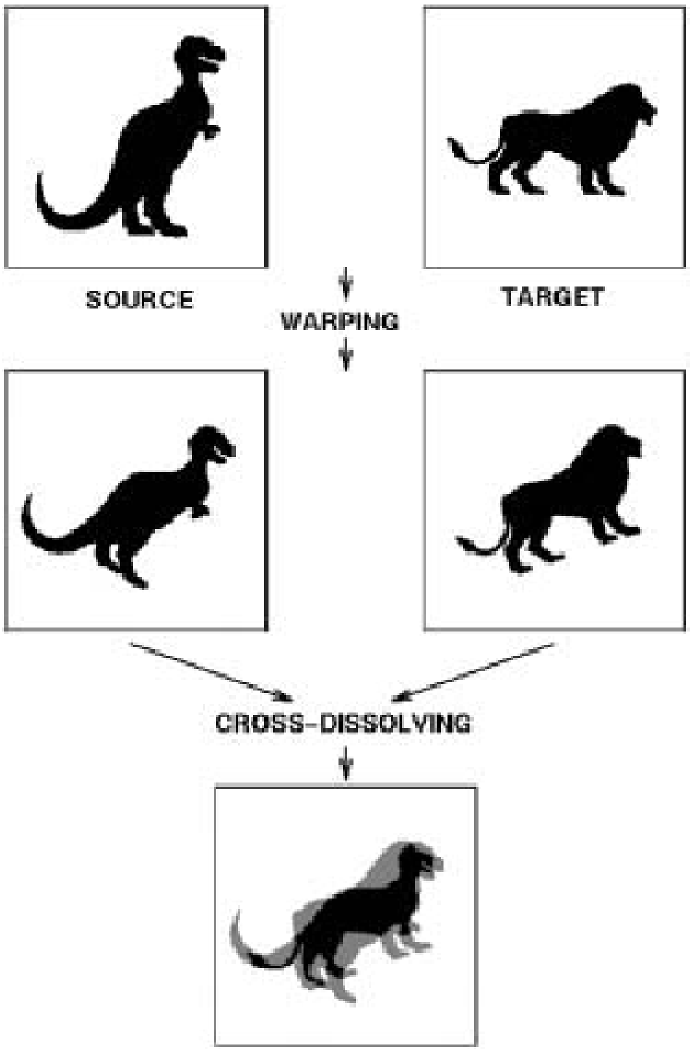
Image Morphing---图像的自然渐变

- 把一幅数字图像以一种自然流畅的，戏剧性的，超现实主义的方式变换到另一幅数字图像
- 由morphing生成的图像序列中，前面部分象源图像，后面部分象目标图像，中间部分既象源图像又象目标图像
- 可以表达特殊的视觉效果
  - 广告中的返老还童效果
  - 影视中后期处理的一种手段

Image Morphing

1. 几何变换(warping)
   - ❖ 通过简单的几何元素(网格节点, 线段...)建立图像特征之间的对应关系
   - ❖ 由特征对应关系计算几何变换
     - 定义了两幅图像象素之间的几何对应关系
2. 交融(cross-dissolve)
   - ❖ 一幅图像淡出时, 另一幅图像淡入

Image Morphing 的过程



未经过几何对齐,直接使用交融技术的结果

Go

第六章 Morphing和空间变形动画

基于网格的图像Morphing

基于网格的图像Morphing

- 基于网格的图像渐变是图像morphing中最早的方法
- 采用的网格通常为<u>双三次样条曲面</u>
- 应用于多部电影

  -<<Indiana Joes and the last crusade>> 1988年

  -<<Willow>>

  - <<The Abyss>>

基于网格的图像Morphing

I_s: 源图像 I_D:目标图像

M_s:源图像I_s上放置的曲面网格, 它确定了控制顶点的坐标

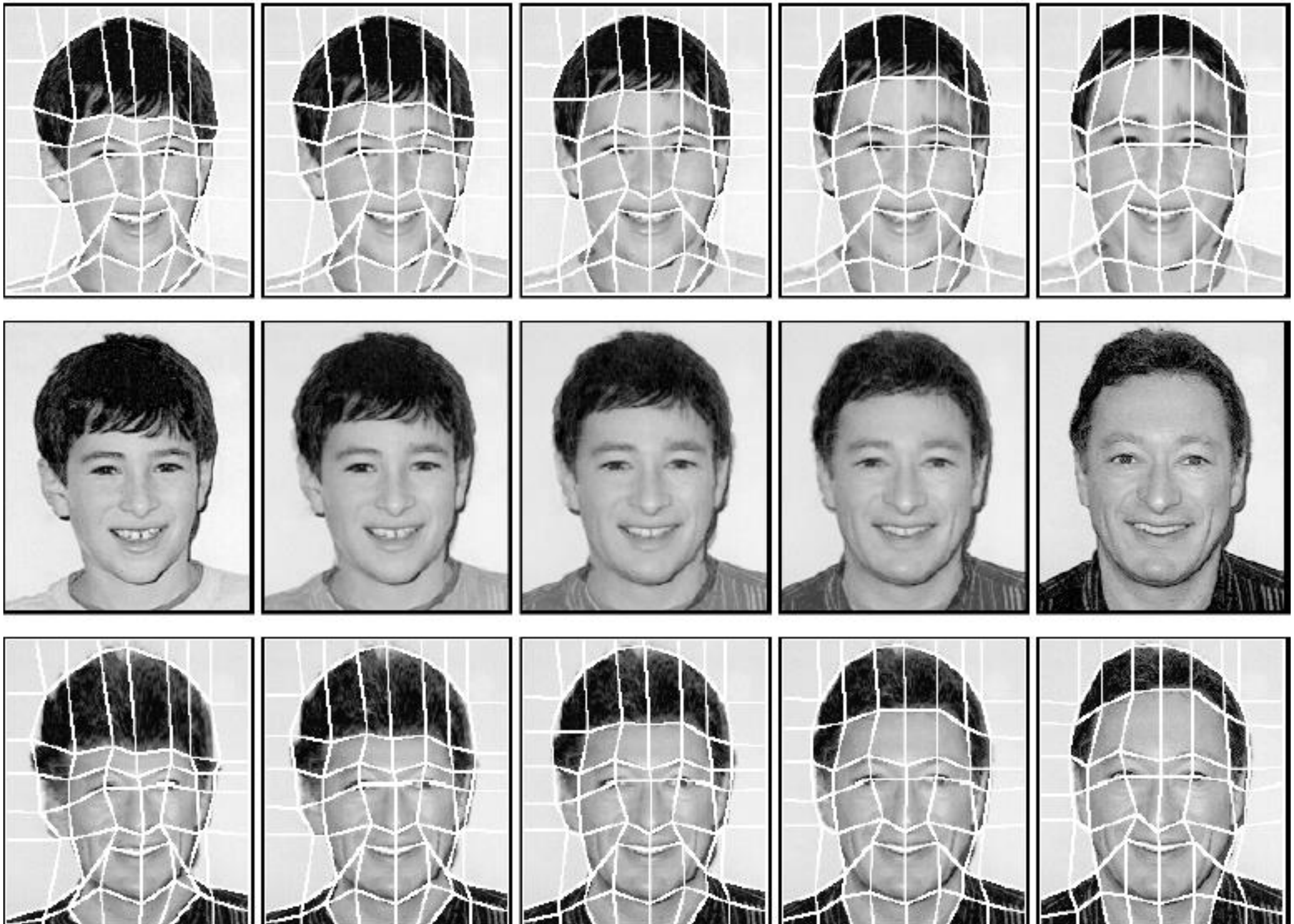
M_D:目标图像I_D上放置的曲面网格,它指定了M_s在目标图像的对应点

M_s和M_D的拓扑关系相同, 用来定义把源图像上的所有点映射到目标图像的空间变换

网格的边界通常与图像的边界重合

基于网格的图像Morphing

Back



基于网格的图像Morphing

- 生成Morphing的中间帧图像的步骤:
  1. 线性插值网格M_s和M_D，得到网格M
  2. 应用由网格M_s和M定义的变换，使源图像I_s扭曲变形到I_0.
  3. 应用由网格M_D和M定义的变换，使目标图像I_s扭曲变形到I_1.
  4. 对图像I_0和I_1进行线性插值，得到中间帧图像

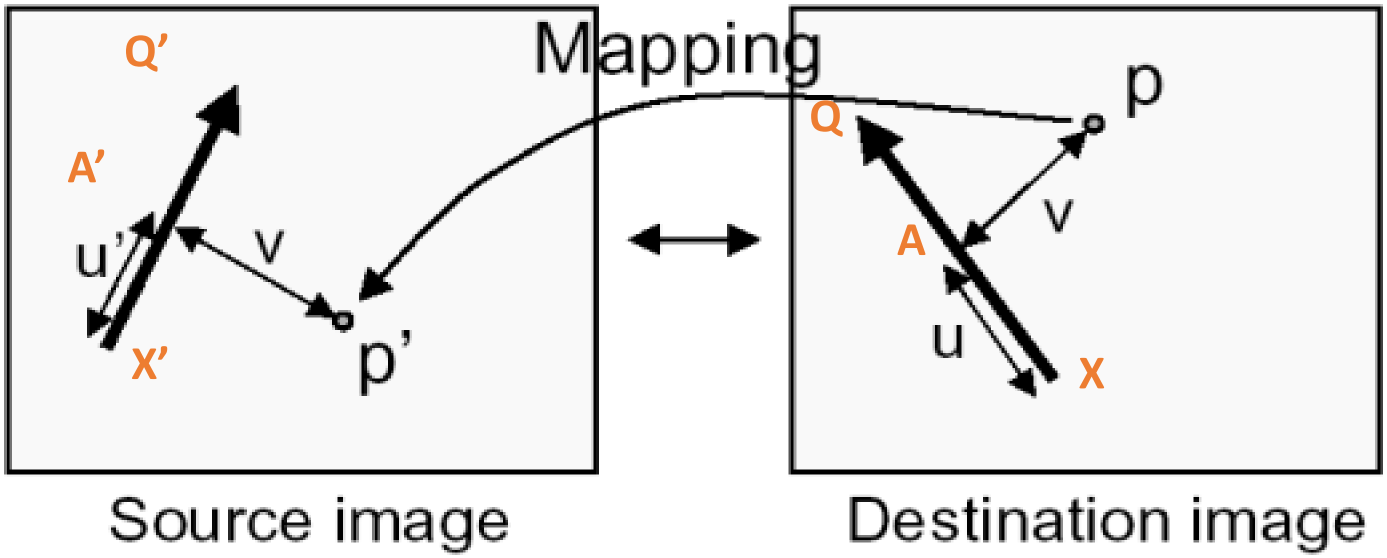
第六章 Morphing和空间变形动画

基于线对的图像Morphing

基于线对的图像Morphing方法

1. 几何变换（warping）
   - 用户利用线对交互地指定特定图像特征
   - 在源图像I_s上定义一条有向直线段,在目标图像I_D中定义一条对应的有向直线段, 这对直线段可以定义一个由源图像到目标图像的一个映射
2. 交融（cross-dissolve）

基于单线对的图像Warping方法



Jump

基于单线对的图像Warping方法

$$u = \frac{\| XA \|}{\| XQ \|} = \frac{(P - X).(Q - X)}{\| Q - X \|^2}$$

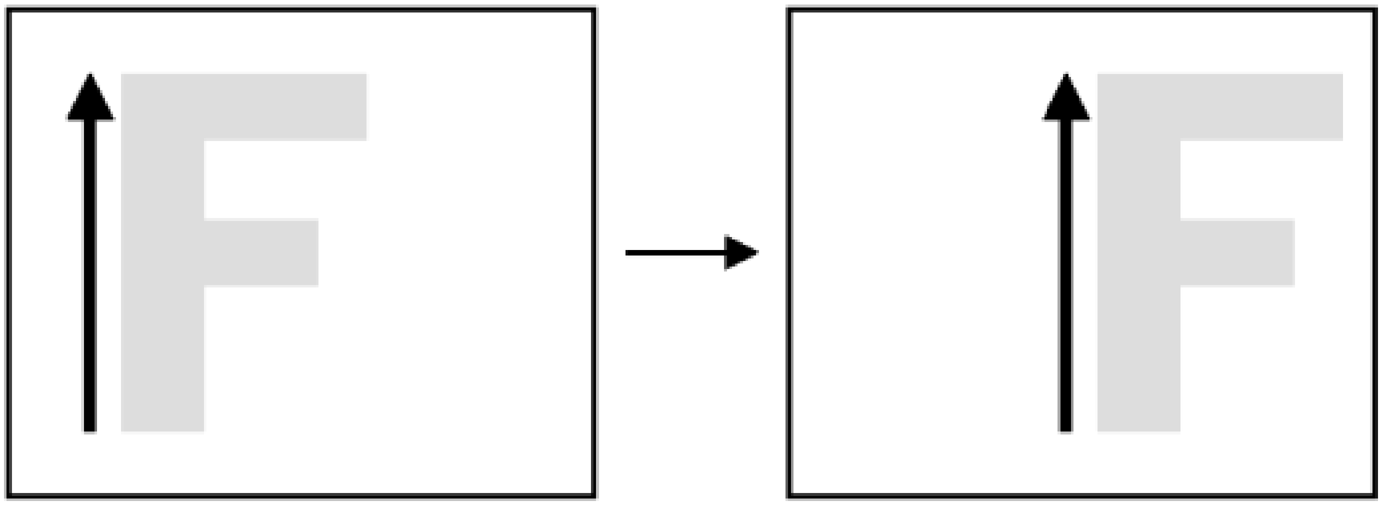
$$v = AP = (P - X).\frac{Perpendicular(Q - X)}{\| Q - X \|}$$

$$P' = X' + u(Q' - X') + v\frac{Perpendicular(Q' - X')}{\| Q' - X' \|}$$

基于单线对的图像Warping方法

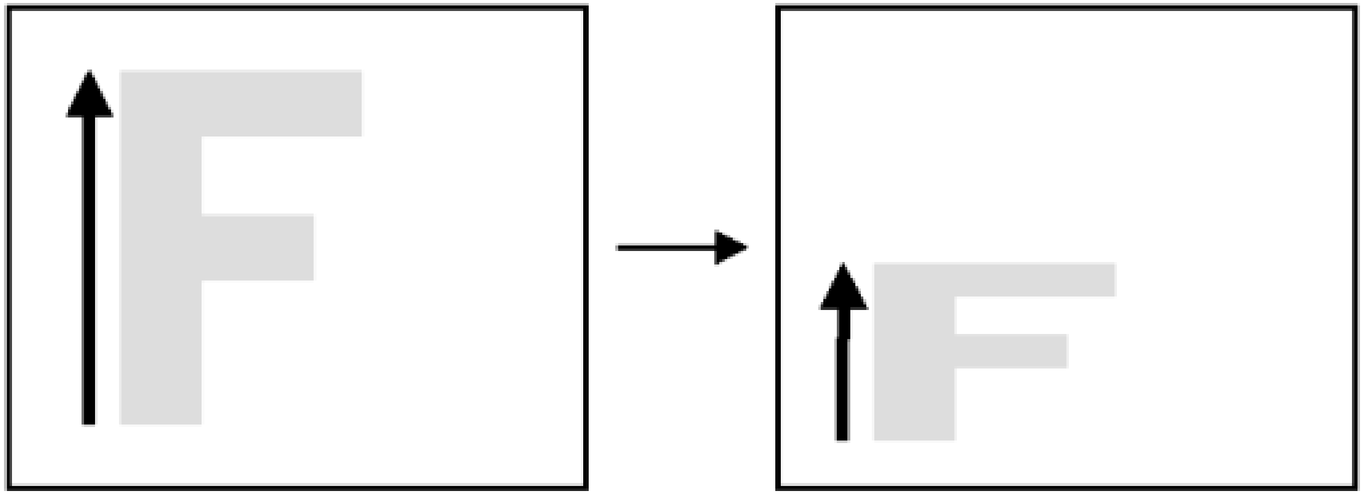
- 指定一对直线段时，图像变换的过程
  - 对于目标图像I_D中任何一象素P， 计算u, v值，之后计算P’象素， 最后将源图像在P’象素处的颜色值赋予目标图像在P象素的颜色值.
  - 一对直线间的变换实际上是一个由旋转，平移和比例变换复合成的变换

基于单线对的图像Warping方法



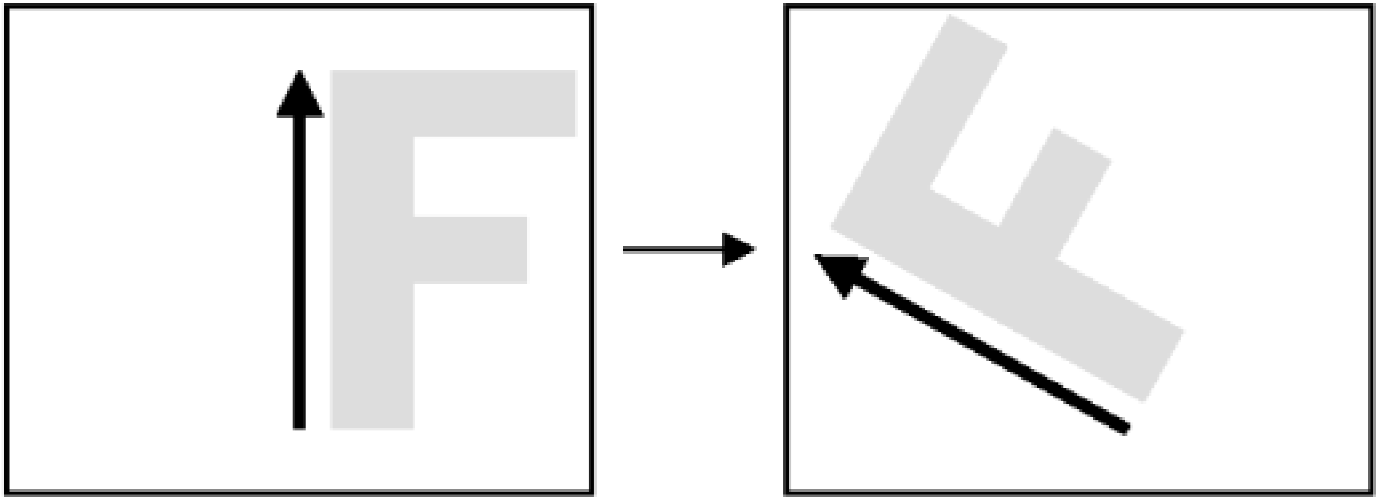
平移

基于单线对的图像Warping方法



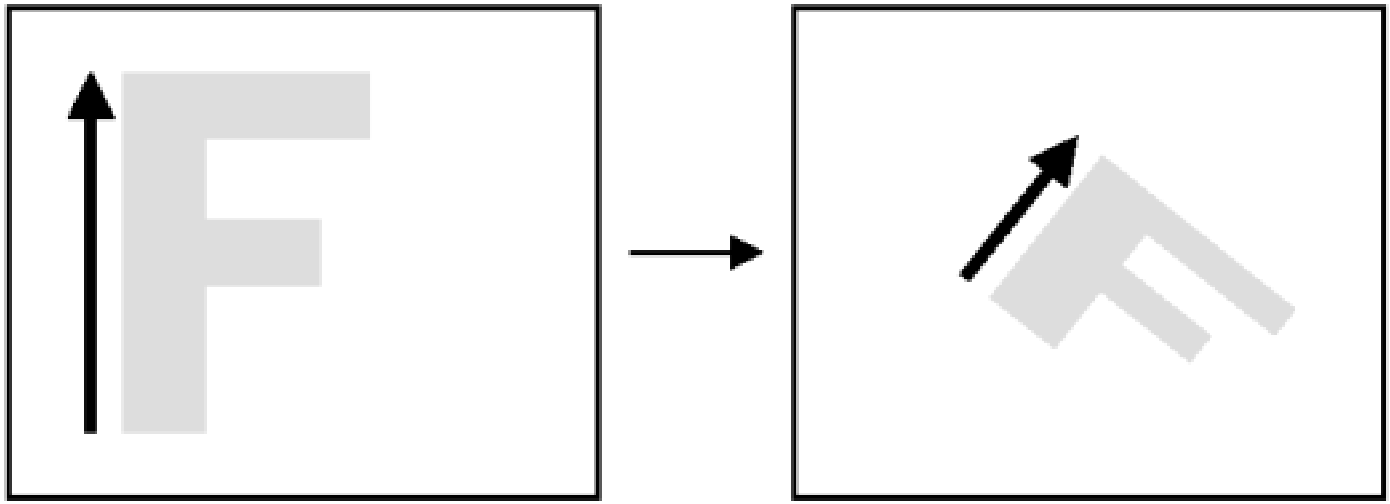
比例缩放

基于单线对的图像Warping方法



旋转

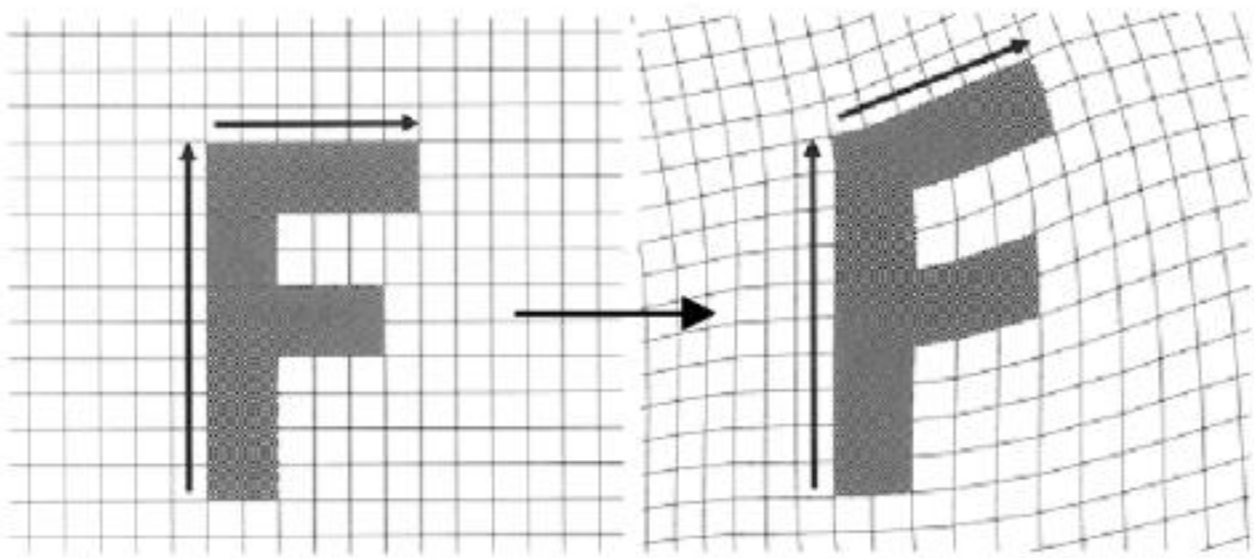
基于单线对的图像Warrping方法



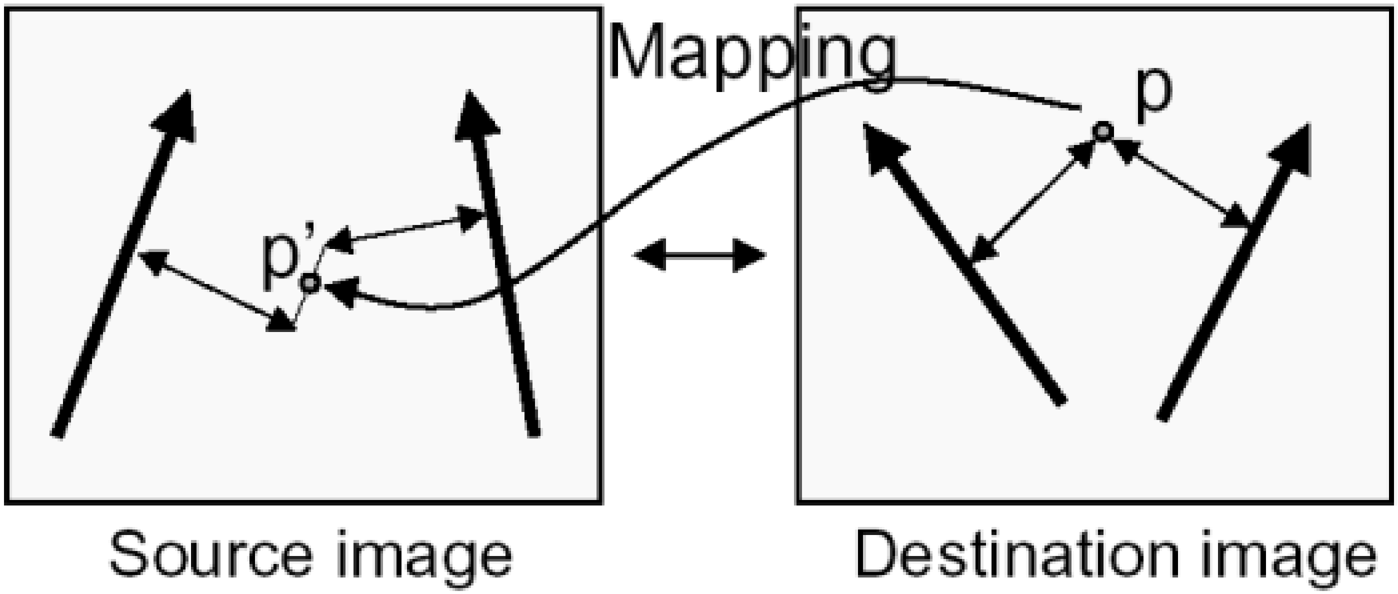
一般情况下对应于一个相似变换

基于多线对的图像Warrping方法

- 使用每对直线对定义的点的加权组合，得到一些比较复杂的几何变换效果



基于多线对的图像Warrping方法



通过加权平均得到P'

基于多线对的图像Warrping方法

设P为目标图像中的一个象素,对于每一对直线,都有一个与之对应的象素P_i. 令$D_i=P_i-P$, 则Di为第I对直线变换所引起的象素位置的偏移. 对所有偏移量进行加权平均,得到总的偏移D:

$$D = \frac{\sum_{i=1}^{n} \omega_i D_i}{\sum_{i=1}^{n} \omega_i}$$

P在源图像中的对应点P'为: P'=P+D

基于多线对的图像Warrping方法

$$\omega_i = \left[\frac{length[i]^p}{a + dist[i]} \right]^b$$

Length[i]是线段X_iQ_i的长度

示意图

当$0 \le u_i \le 1$时，dist[i]的值等于$|v_i|$

当$u_i < 0$时，dist[i]的值等于P到X_i的距离

当$u_i > 1$时，dist[i]的值等于P到Q_i的距离

基于多线对的图像Warrping方法

- 参数a用来控制映射的精确程度
  - 若a取较大值，控制精度变差
- 参数b决定了随距离dist[i]的变大，不同直线相对强度的减弱程度.
  - b的取值一般基于0.5到2之间
- 参数p决定了直线的长短对映射的影响.
  - 若p=0,则长短直线的地位相同
  - 若p等于1,则长直线比短直线有更大的权因子
  - p的取值范围一般为0到1

```
WarpImage(Image, L'[…], L[…])
begin
    foreach destination pixel p do
        psum = (0,0)
        wsum = 0
        foreach line L[i] in destination do
            p'[i] = p transformed by (L[i],L'[i])
            psum = psum + p'[i] * weight[i]
            wsum += weight[i]
        end
        p' = psum / wsum
        Result(p) = Image(p')
    end
end
```

基于多线对的图像Morphing方法

1. 在源图像I_S和目标图像I_D中**定义**控制变形的**对应直线对**
2. 通过插值得到中间图像I的控制直线

   -对直线段端点进行线性插值

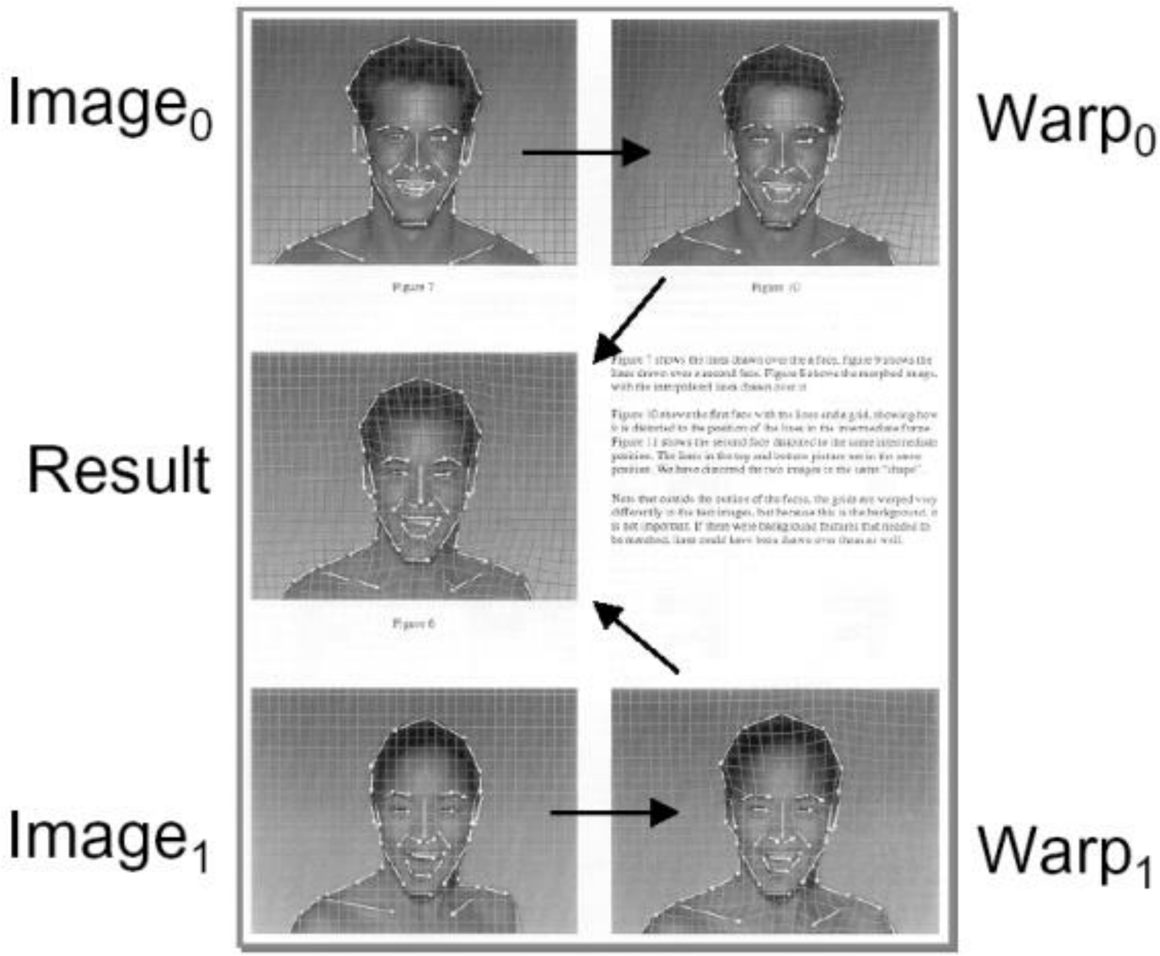
   - 对于旋转的直线对会得到缩短的插值直线段

   -对直线的中点,朝向和长度进行插值
3. 根据Is和I中的直线对变换得到一幅渐变图像I_0; 同样, 根据I_D和I得到一幅渐变图像I_1
4. 对I_0和I_1进行交融处理得到中间的渐变图像I

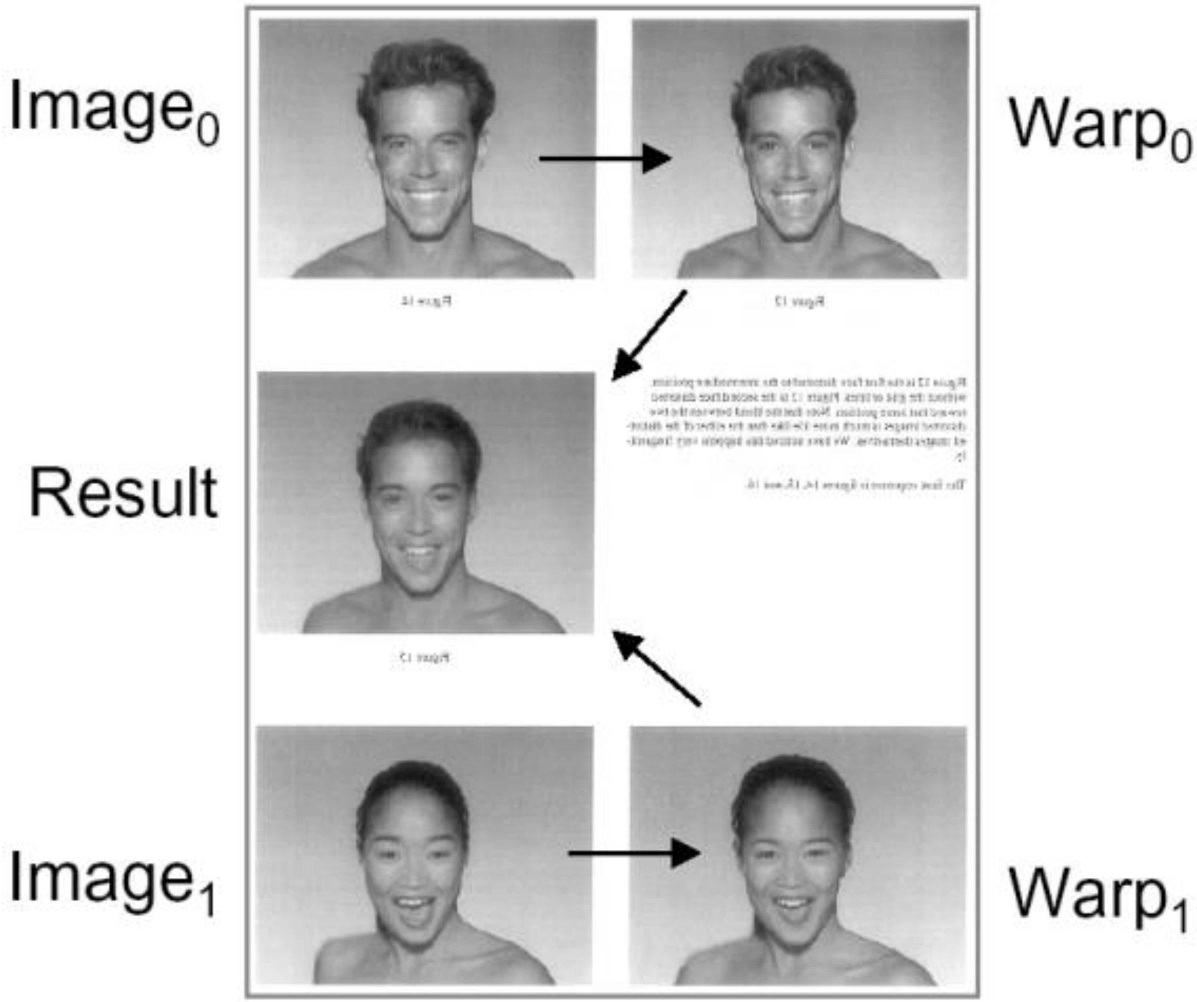
基于多线对的图像Morphing方法

```
GenerateAnimation(Image_0, L_0[…], Image_1, L_1[…])
begin
    foreach intermediate frame time t do
        for i = 1 to number of line pairs do
            L[i] = line t-th of the way from L_0 [i] to L_1 [i]
        end
        Warp_0 = WarpImage(Image_0, L_0, L)
        Warp_1 = WarpImage(Image_1, L_1, L)
        foreach pixel p in FinalImage do
            Result(p) = (1-t) Warp_0 + t Warp_1

    end
end
```



Image$_0$
Warp$_0$
Result
Image$_1$
Warp$_1$



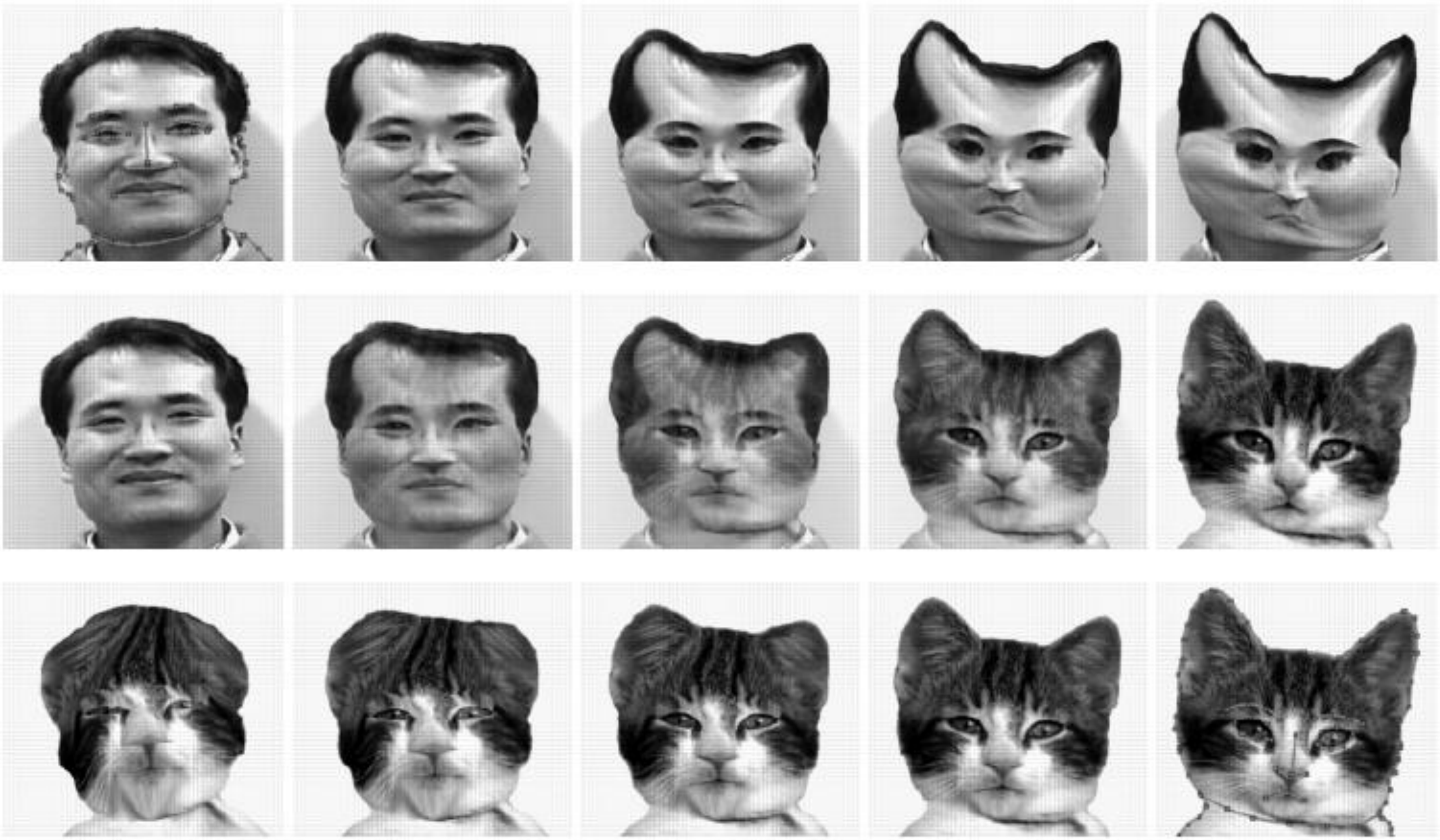
$Image_0$
$Warp_0$
Result
$Image_1$
$Warp_1$


Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full
Active Camera
Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full
Active

基于多线对的图像Morphing方法

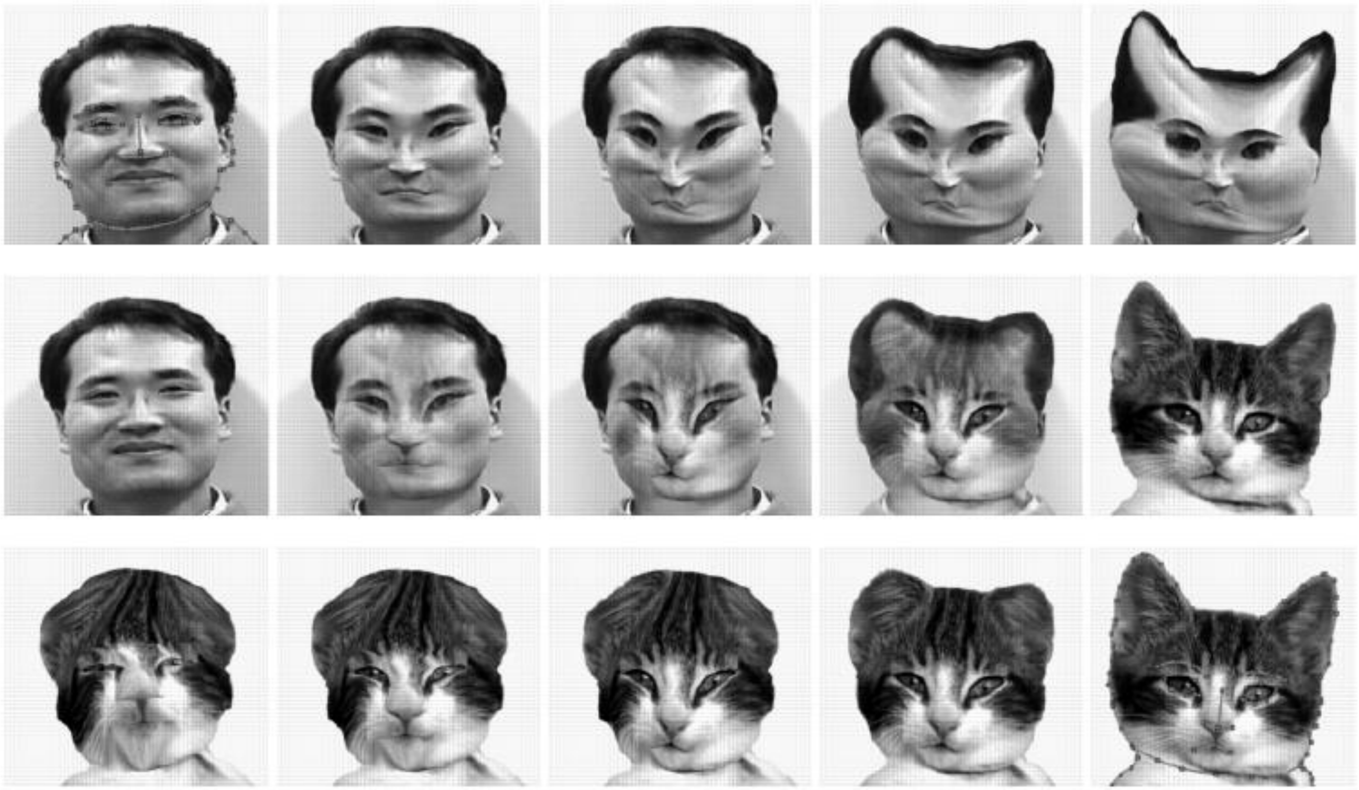
基于线对的方法的优点是直观，缺点是有可能生成一些意料之外的图像

图像Morphing中的过渡控制



均匀过渡：中间帧图像各部分之间的过渡速度相同

图像Morphing中的过渡控制



非均匀过渡: 使用非线性函数决定图像扭曲和颜色插值的速度,得到一些戏剧性视觉效果

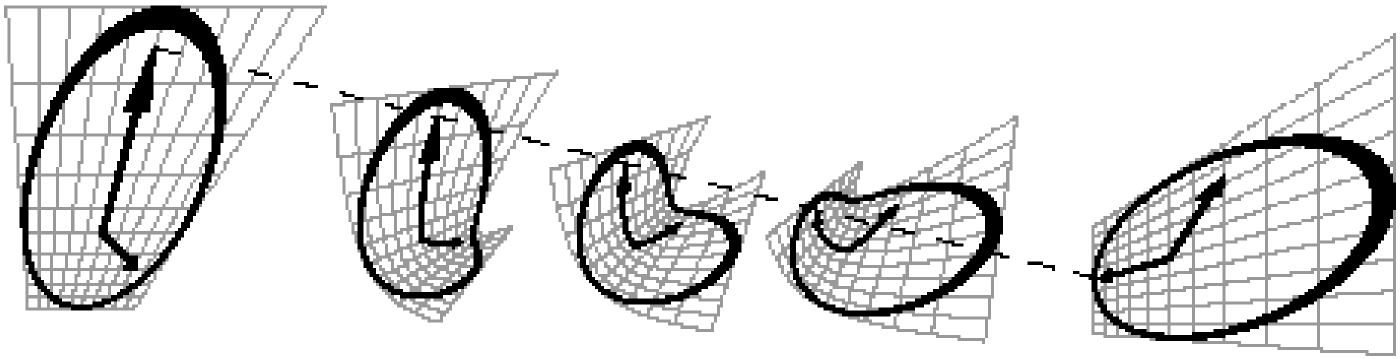
第六章 Morphing和空间变形动画

视域Morphing

视域Morphing

- 一般的图像morphing技术没有考虑视点的变化, 不能保证得到的结果是自然的
  - 如果源图像和目标图像是同一个物体在不同视点下的图像，使用一般的图像Morphing技术不能保证中间帧是该物体在新视点下的投影图像,而是一种会带来形状畸变的变换.例如,一条直线在morphing后可能变成一条曲线，导致不自然的图像过渡

Example: A Clock



形状扭曲的morphing过程

虚线为同一特征的线性路径

视域Morphing

- 同时插值几何, 颜色和视点, 可以产生类似三维的视觉效果

视域Morphing

- 计算视域Morphing需要的信息

1. 同一三维物体或者场景在两个不同视点的投影图像I_0和I_1
2. 两幅图像象素之间的对应关系

   ---可以由一般的图像Morphing技术得到

3. 两个视点的投影矩阵

   ---可以通过预先知道的一些图像点的3D位置信息计算得到投影矩阵

视域Morphing的三个步骤

1. 前置变形（Prewarp）

   ❖ H_0为表示I_0图像对应的成像平面在世界坐标系中的位置和方向的3*3矩阵

   $$I_0^{'} = H_0^{-1} I_0$$

   $$I_1^{'} = H_1^{-1} I_1$$

2. Morphing

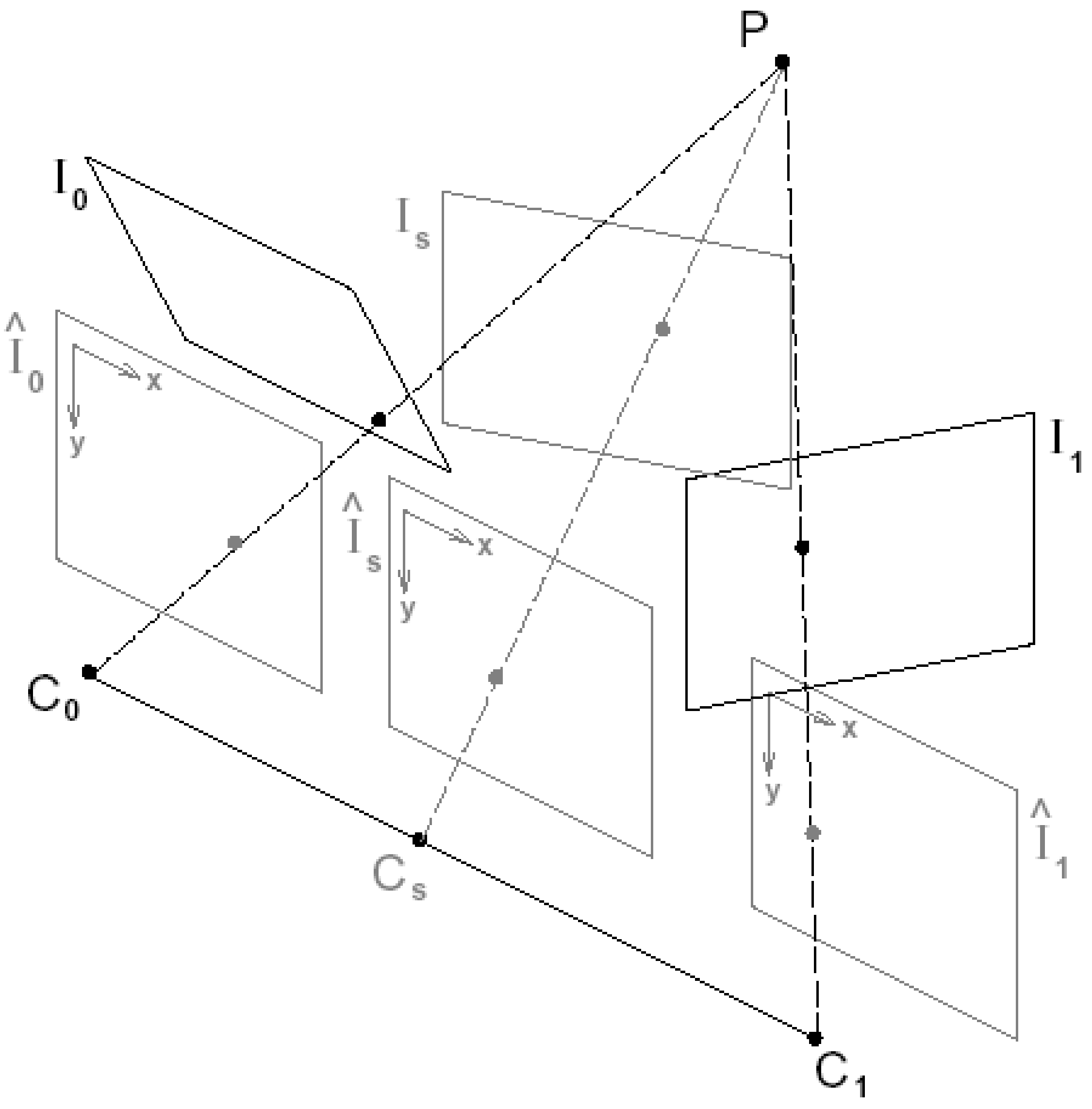
   ❖ 线性插值图像I_0'和I_1'相应点的位置和颜色得到 I_t'

3. 后置变形（Postwarp）

   ❖ 对图像I_t'应用变换H_t生成图像I_t

   ❖ H_t是H_0和H_1插值得到的3*3矩阵

视域Morphing的三个步骤



视域Morphing的三个步骤

- 前置变形在不改变摄像机中心的情况下,使图像I_0'和I_1'对应的象平面平行
- Morphing 过程使摄像机的中心移至C_t
- 后置变形使象平面变换至新的位置和方向

$\mathcal{I}_0$ $\mathcal{I}_{0.25}$ $\mathcal{I}_{0.5}$ $\mathcal{I}_{0.75}$ $\mathcal{I}_1$

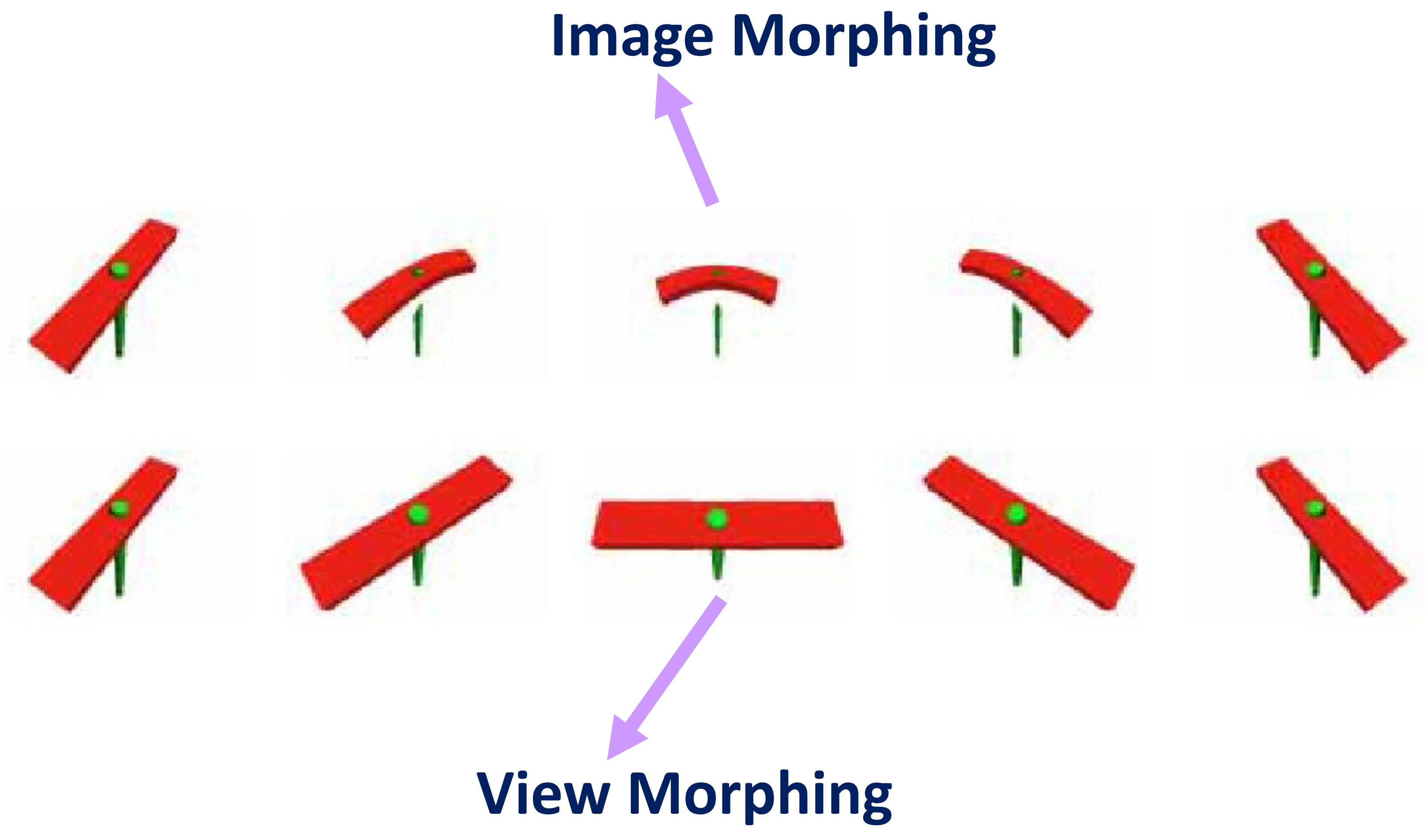


Image Morphing
View Morphing

第六章 Morphing和空间变形动画

3D Morphing

3D Morphing

- 三维Morphing是指的是将一个三维物体光滑连续地变换为另一个三维物体
- 3D Morphing 比 Image Morphing技术要复杂得多, 但能生成更加逼真更加生动的特技效果
- 3D Morphing得到的中间结果是物体的3D模型, 所以3D Morphing的结果与视点和光照参数无关,可以用不同的摄像机角度和光照条件重新渲染

3D Morphing

- 当两个物体的顶点数和拓扑结构都相同时, 只需要对对应顶点进行插值便可以实现3D Morphing的过程

3D Morphing

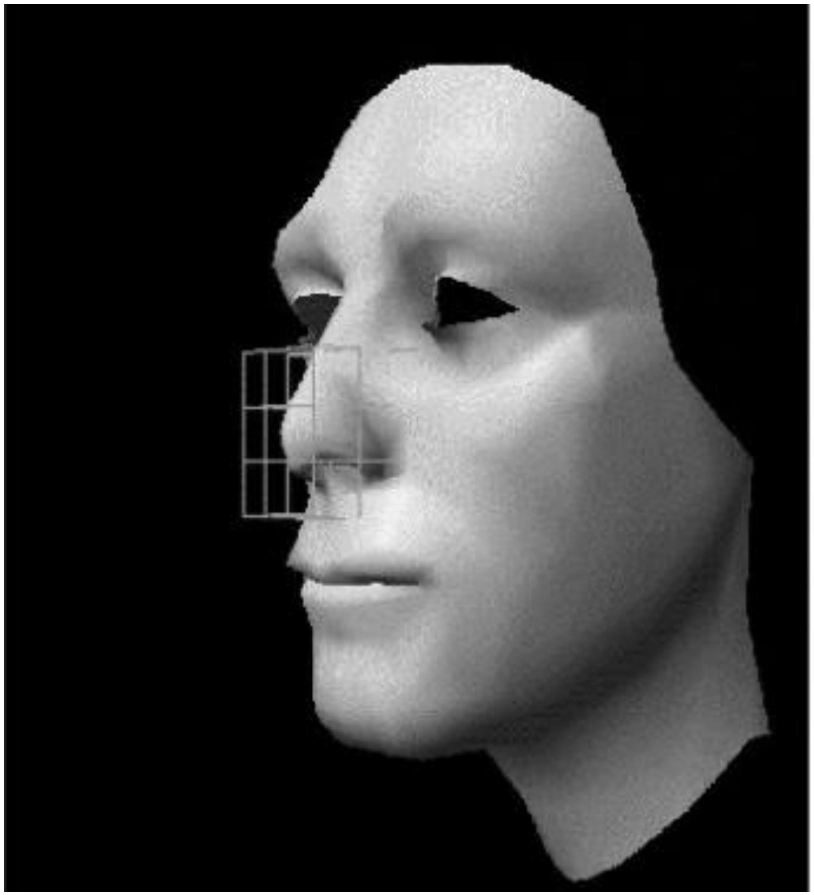
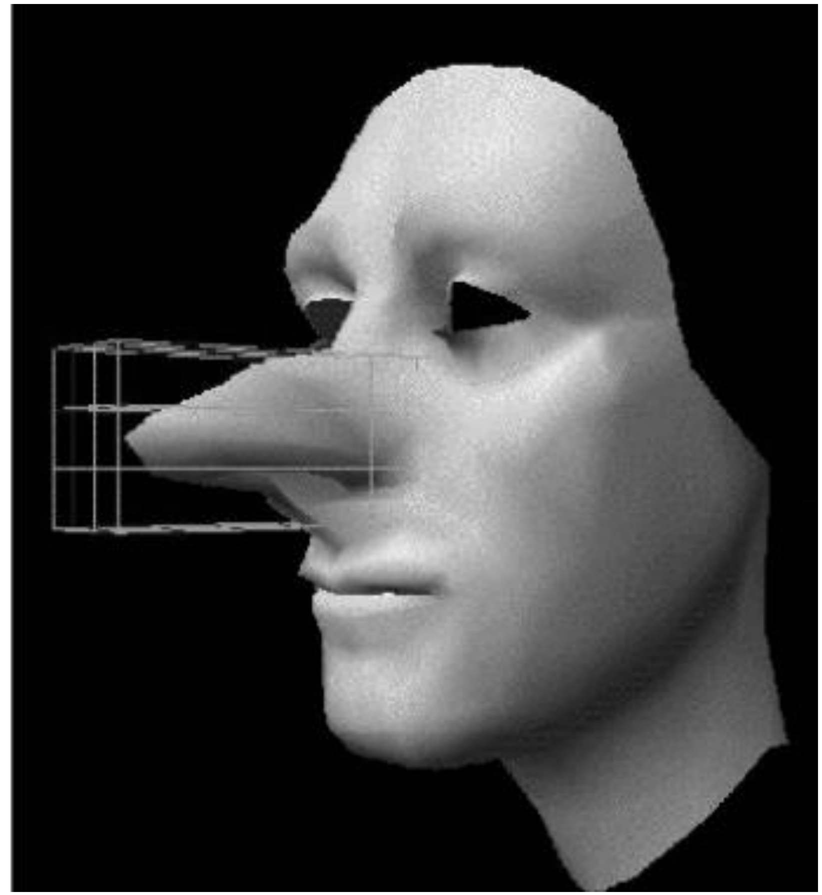
- 源物体和目标物体的顶点数和拓扑结果不相同
  - 基于星形物体的多面体Morphing
  - 基于体表示的三维Morphing

第六章 Morphing和空间变形动画

空间变形(Deformation)

空间变形(Deformation)

- 将单个几何对象的形状作某种扭曲，变形，使它变换成所需要的形状
- 空间变形过程中，几何对象的拓扑关系保持不变
- 属于针对动画的造型问题，将造型和动画有机地结合起来

空间变形(Deformation)

- 与物体表示<u>有关</u>的变形
  - 针对物体的某种具体的表示方式
    - 多面体, 参数曲面
- 与物体表示<u>无关</u>的变形
  - 既可以用于多面体表示的物体,也可以用作参数曲面表示的物体

空间变形(Deformation)

- 整体和局部变形方法
- 自由变形方法
- 轴变形方法
- 基于约束的变形

整体和局部变形方法

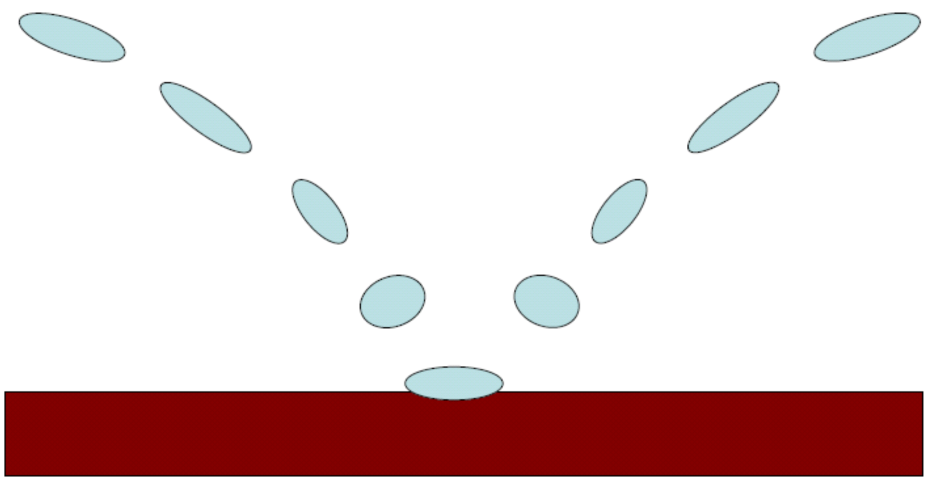
- 借鉴CSG（ Constructive Solid Geometry ）表示方法的思想；
- 变形对象定义在局部空间；
- 把整体和局部变形定义为变形算子的组合：
  - Twisting，bending，tapering等
- 复合变换产生复杂的形状；

整体和局部变形方法

- 非均匀尺度变换算子：

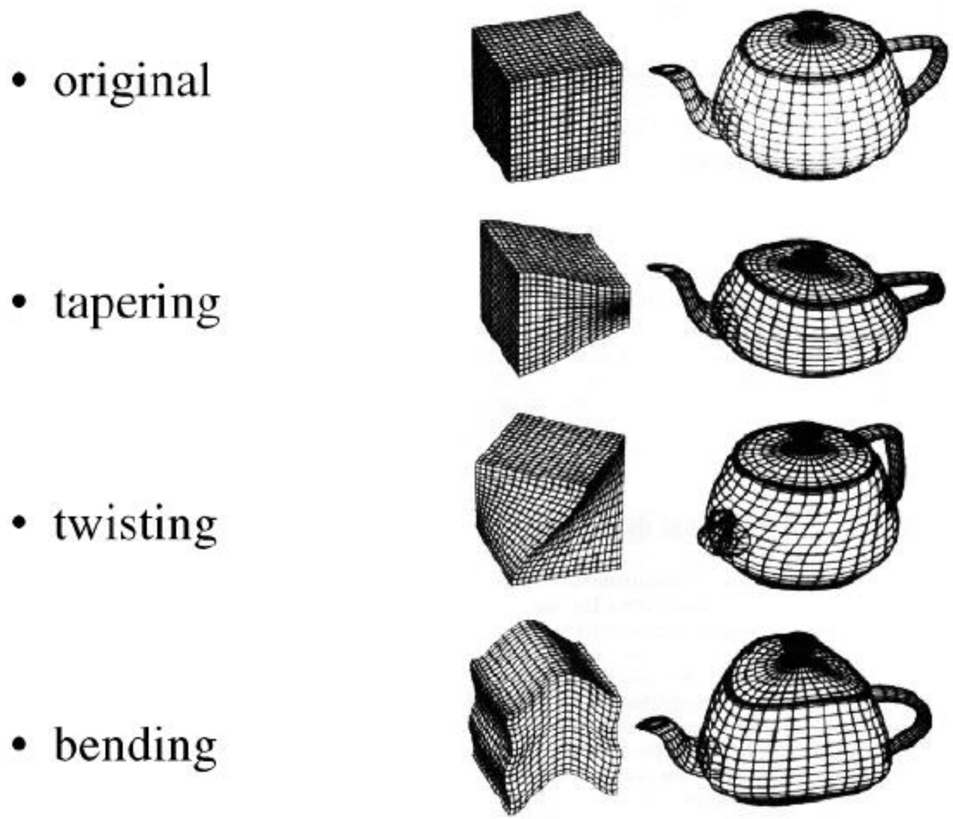
$$\begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- 非均匀尺度变换：



整体和局部变形方法

- 非线性整体变形：



整体和局部变形方法

- 整体变形矩阵为坐标的函数：

$$\begin{pmatrix} f(x,y,z) & g(x,y,z) & \cdots \\ \cdots & & \\ & & \\ & & \end{pmatrix}$$

Original Object　　Tapered Object

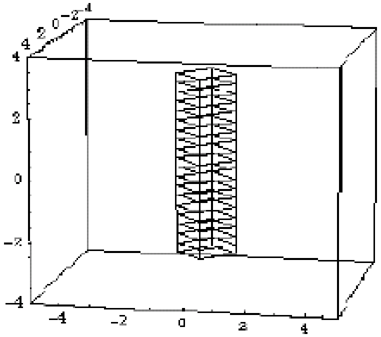
$$x' = x$$
$$y' = f(x)$$

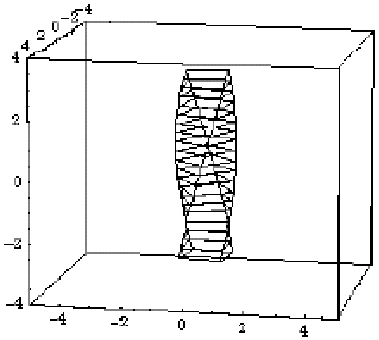
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & f(x) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

整体和局部变形方法

- 螺旋算子:

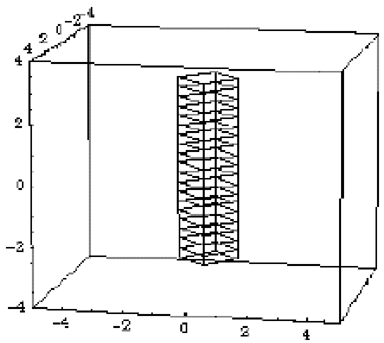
$$x' = x*\cos(f(y)) - z*\sin(f(y))$$
$$y' = y$$
$$z' = x*\sin(f(y)) + z*\cos(f(y))$$

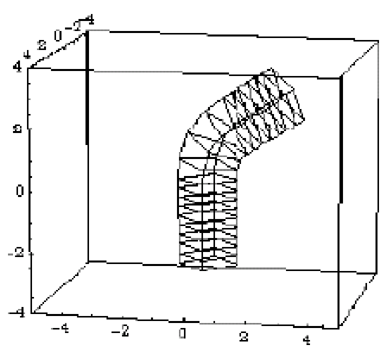




整体和局部变形方法

- 整体和局部变形方法：
- 弯曲算子：





y_0 - center of bend
1/k - radius of bend
y_{min}:y_{max} - bend region

$$\hat{y} = \begin{cases} y_{min} & y \le y_{min} \\ y & y_{min} < y < y_{max} \\ y_{max} & y \ge y_{max} \end{cases}$$

$$\theta = k \cdot (\hat{y} - y_0)$$
$$C_\theta = \cos\theta$$
$$S_\theta = \sin\theta$$

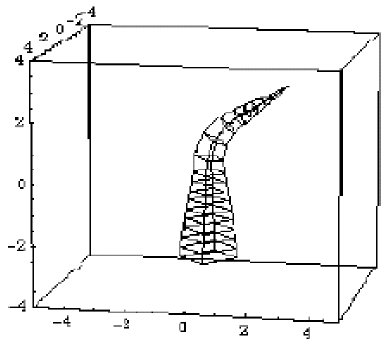
$$x' = x$$

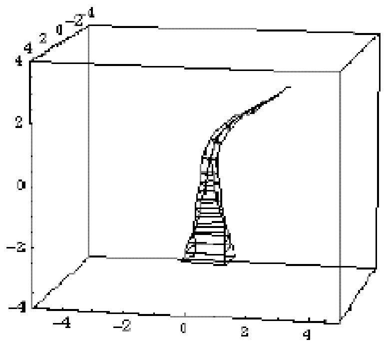
$$y' = \begin{cases} -S_\theta \cdot z - \frac{1}{k} + y_0 & y_{min} \le y \le y_{max} \\ -\left(S_\theta \cdot \left(z - \frac{1}{k}\right)\right) + y_0 + C_\theta \cdot (y - y_{min}) & y < y_{min} \\ \left(-\left(S_\theta \cdot \left(z - \frac{1}{k}\right)\right) + y_0 + C_\theta \cdot (y - y_{max})\right) & y > y_{max} \end{cases}$$

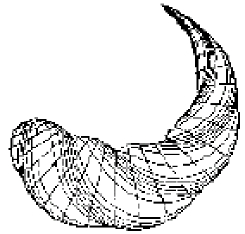
$$z' = \begin{cases} -C_\theta \cdot z - \frac{1}{k} + \frac{1}{k} & y_{min} \le y \le y_{max} \\ -\left(C_\theta \cdot \left(z - \frac{1}{k}\right)\right) + \frac{1}{k} + S_\theta \cdot (y - y_{min}) & y < y_{min} \\ \left(-\left(C_\theta \cdot \left(z - \frac{1}{k}\right)\right) + \frac{1}{k} + S_\theta \cdot (y - y_{max})\right) & y > y_{max} \end{cases}$$

整体和局部变形方法

- 算子的复合：



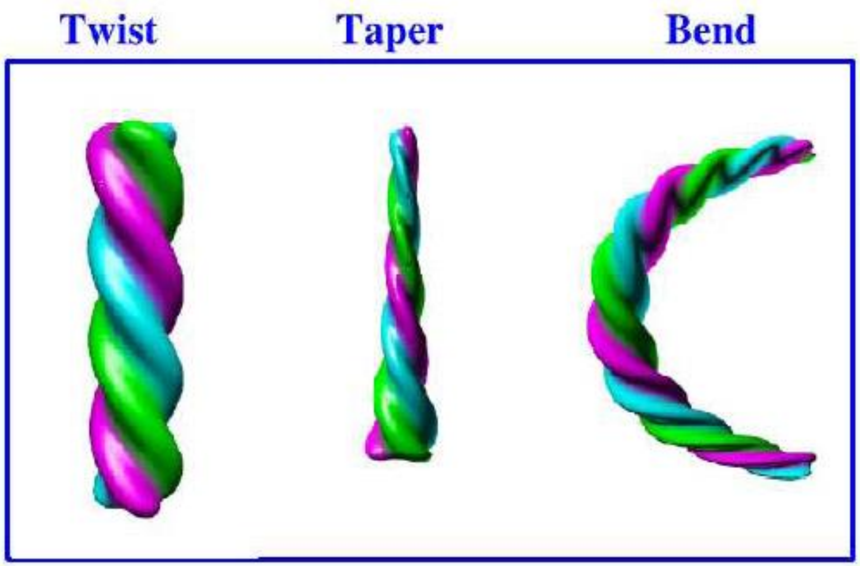




整体和局部变形方法

- 应用于隐函数曲面；

$$F_{total}(P)=\Sigma\ c_i F_i(|P-Q_i|)$$



$$F_{total}(P)=\Sigma\ c_i F_i(|w(P)-Q_i|)$$

整体和局部变形方法

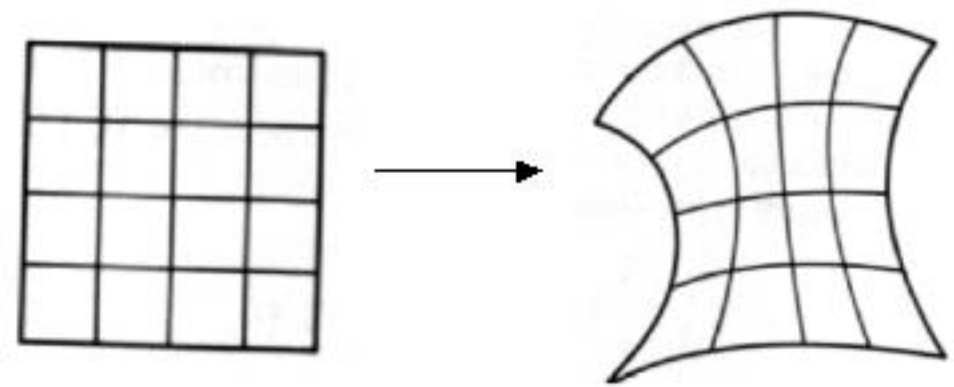
- 推广了传统的造型运算，可以生成许多传统造型方法难以生成的形体；
- 变形后物体的法向量可以用原物体的法向量和变换矩阵解析求得；

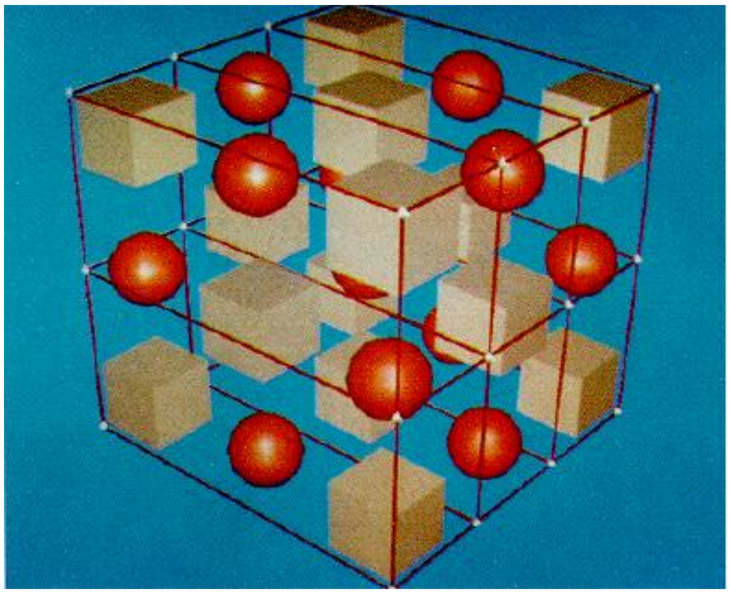
空间变形(Deformation)

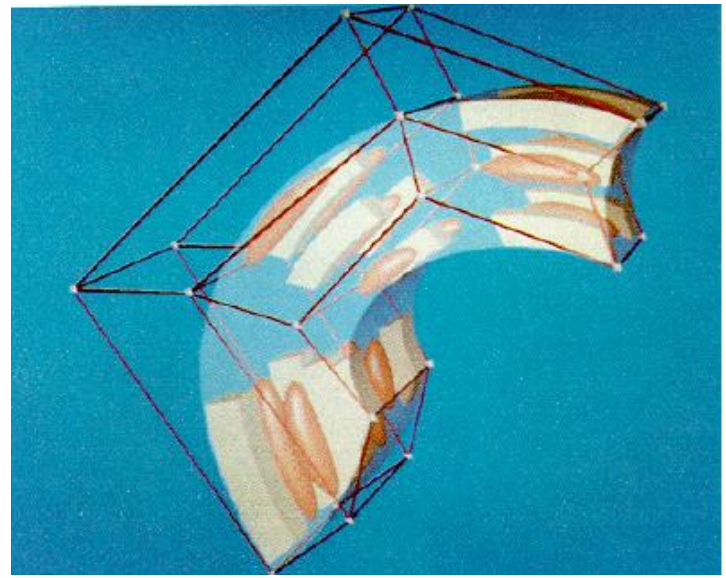
- 整体和局部变形方法
- 自由变形方法
- 轴变形方法
- 基于约束的变形

自由变形FFD

- Sederderg等在1986年提出的一种非常适合于柔性物体动画的一般性方法
- FFD方法不直接操作物体,而是将物体嵌入一个空间,当所嵌入的空间变形时,物体将随之变形







三三次Bezier超曲面

$$Q(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}P_{ijk}B_i^3(u)B_j^3(v)B_k^3(w)$$

$$(u,v,w)\in[0,1]\times[0,1]\times[0,1]$$

将一正方体转换为弯曲的物体

P_{ijk}是64个控制顶点

三三次Bezier超曲面

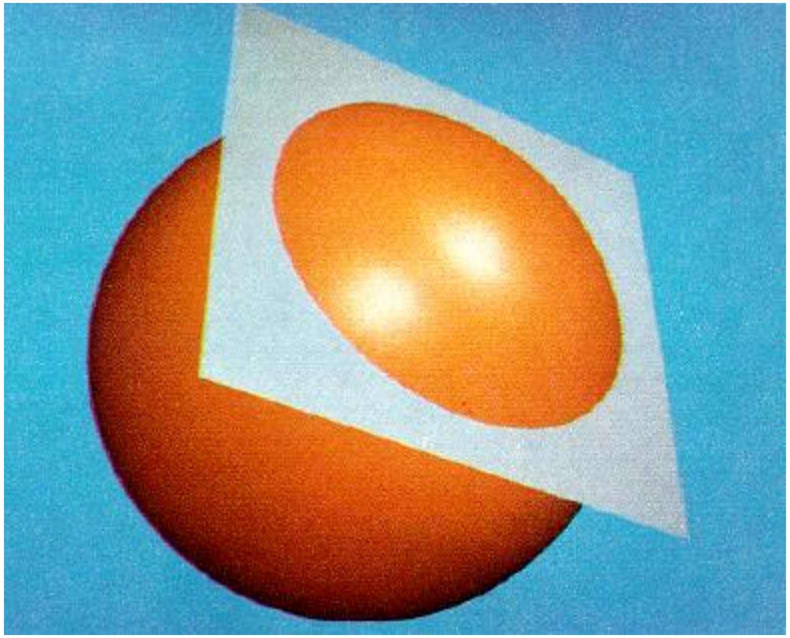
- 多个Bezier超曲面可以拼接成一个分段光滑的的Bezier体,称这种复合的Bezier体为FFD块, 令其坐标方向为(s,t,u)
- 假设采用(3l+1)*(3m+1)*(3n+1)个控制点定义一个FFD块, 其包含l*m*n个三三次Bezier超曲面

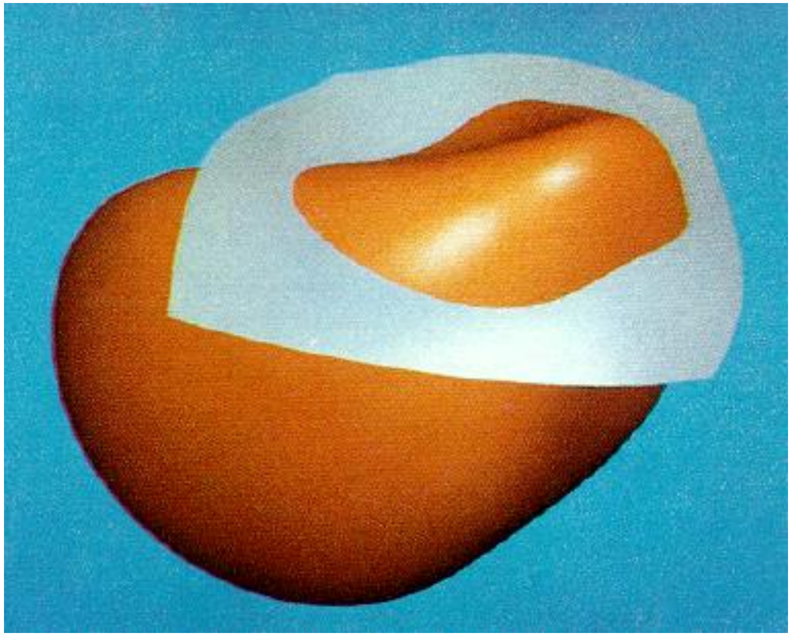
自由变形FFD

- **采用FFD块对物体变形的步骤**

1. 确定物体的顶点在网格空间中的位置
2. 根据动画设计的需要,移动控制顶点P_{ijk}
3. 确定顶点变形后的位置

自由变形FFD

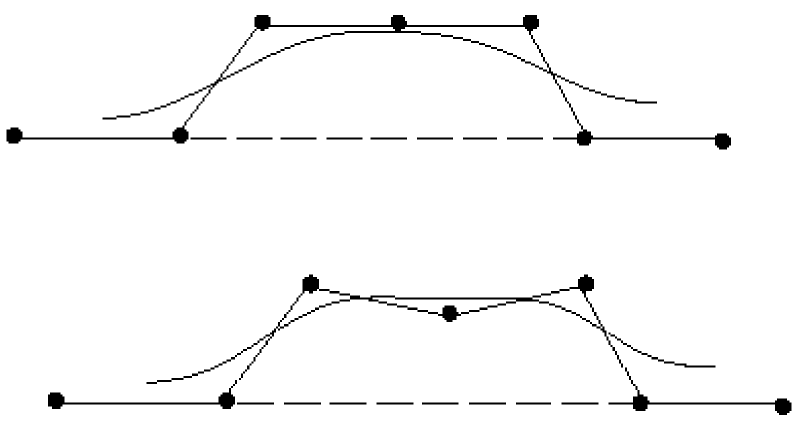





Coke

直接操纵的FFD变形方法

- 网格控制顶点的移动只提供了物体变形的某种暗示,物体的变形并不精确地跟随FFD控制顶点的移动



直接操纵的FFD变形方法

- 通过移动控制顶点的**FFD**变形方法的缺点：

1. 难以得到精确的形状
2. 难以使物体上的某些点到达指定的位置
3. 如果用户对样条不熟悉,难以了解控制顶点的移动和物体变形结果之间关系
4. 控制顶点的数目较多,移动控制顶点变得很困难

直接操纵的FFD变形方法

- Hsu在1992年提出一种直接操纵FFD变形的方法
- 主要思想
  - 用户从物体上选择一个点,并将该点移动到变形后的位置, 系统自动反求导致这种变形的FFD控制顶点的变化

单点约束问题

问题描述

- 用户将物体上的选择的一点Q移动到其目标位置Q_{new}，求解网格控制顶点的某种布局,使该点变形后的位置与目标位置重合
- 该问题是欠定的,存在多种解
- 使控制顶点在最小二乘意义下移动

单点约束问题

- **FFD变形方程**

$$Q(u,v,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}P_{ijk}B_i^3(u)B_j^3(v)B_k^3(w)$$

Q=BP

与基函数有关的行矩阵

64个控制顶点构成的列矩阵

单点约束问题

$$Q_{new}=B(P+\triangle P)$$

$$\downarrow$$

$$\triangle Q=B^*(\triangle P)$$

给定物体上点的变化$\triangle Q$, 求满足上述方程的$\triangle P$

单点约束问题

$$\triangle Q = B^*(\triangle P)$$

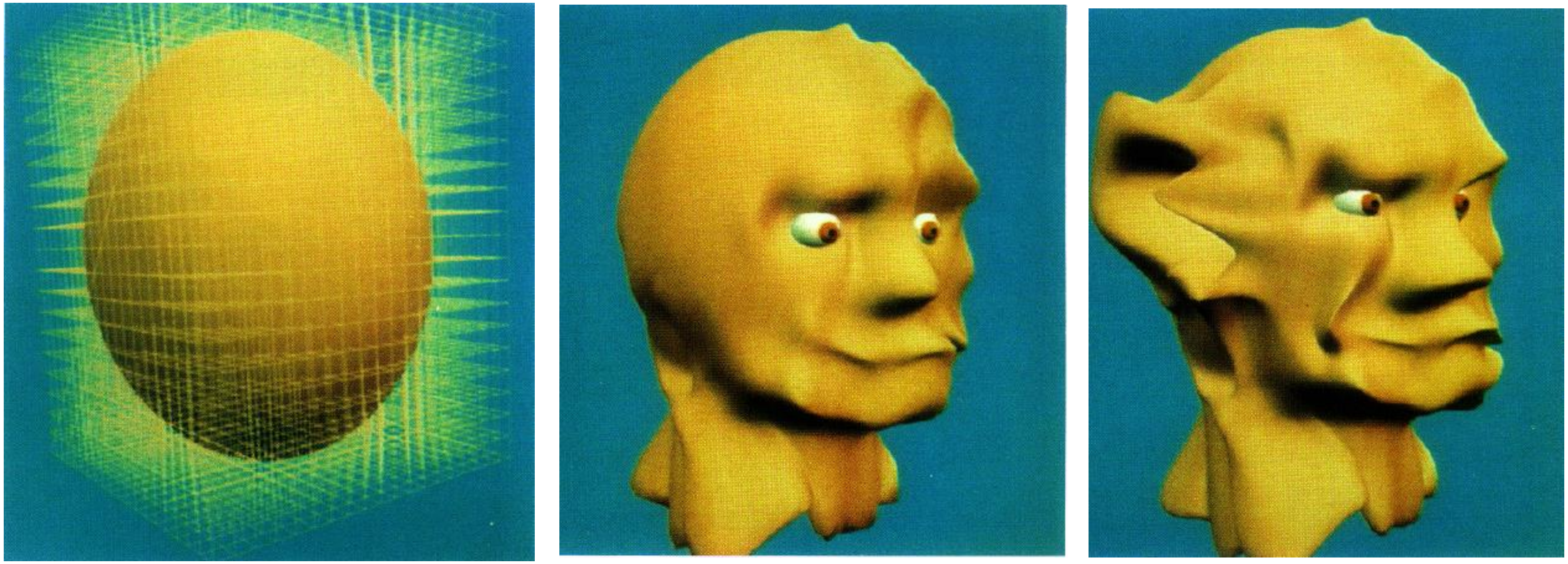
$$\downarrow$$

$$\triangle P = B^+ \triangle Q$$

最小二乘意义下的伪逆矩阵

单点约束时 $B^+ = \dfrac{1}{\|B\|^2} B^T$

直接操纵的FFD变形方法



<<Direct Manipulation of Free-Form Deformations>> SIGGRAPH92

空间变形(Deformation)

- 整体和局部变形方法
- 自由变形方法
- 轴变形方法
- 基于约束的变形

轴变形方法

- 自然界中有一类物体的运动可以看成是由某条**曲线**控制的
  - 蛇的爬行,鱼的游动,树的随风摆动
- 轴变形是一种通过**参数曲线**来控制物体自由变形的方法
  - 基于弧长的轴变形方法

轴变形方法

- 用户首先指定一条轴，即一条空间参数曲线。待变形物体上的点根据最近点规则嵌入曲线轴上相应点的局部坐标系中;
- 当曲线的形状变化时，物体的形状也会随之改变;
- 方法的优点在于可通过轴的变形来控制物体的变形, 该方法比较直观.

基于弧长的轴变形方法

- 记R(t)=(x(t),y(t),z(t))为空间参数曲线
  - K(0<K<4)次空间B样条曲线

$$T(t)=\frac{R'(t)}{\|R'(t)\|}$$

$$B(t)=\frac{R'(t)\times R''(t)}{\|R'(t)\times R''(t)\|}$$

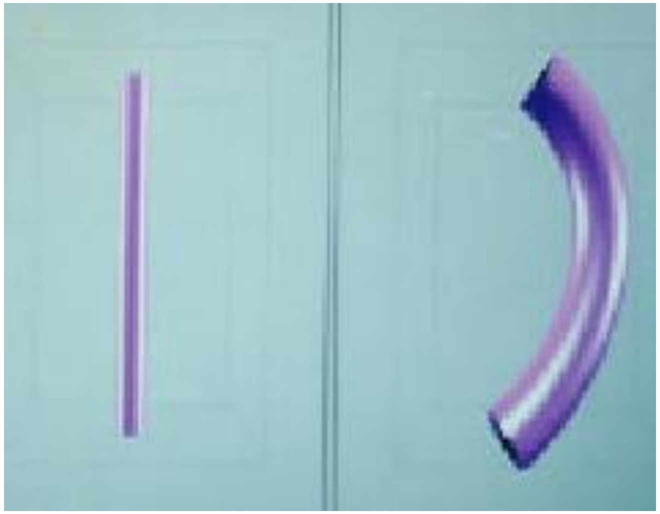
$$N(t)=B(t)\times T(t)$$

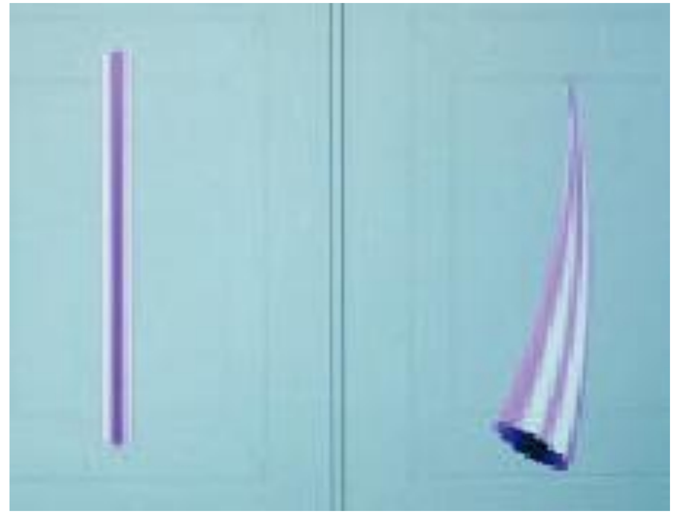
Frenet标架

T(t),N(t),B(t)分别为t点处的切向,法向和副法向,相互垂直

基于弧长的轴变形方法

1. 离散B样条曲线,建立弧长,空间位置和局部活动标架的查找表,作为物体的嵌入空间
2. 定义物体的局部坐标系 并将被变形物体变换到该局部坐标系中
3. 通过控制曲线将被变形物体嵌入到曲线的变形空间中,实现轴变形





轴变形方法作用于一个**圆柱**所产生的效果

空间变形(Deformation)

- 整体和局部变形方法
- 自由变形方法
- 轴变形方法
- 基于约束的变形

基于约束的变形

- 空间变形时,有时希望待变形物体上的一个或者一些点移动指定的偏移量,称这种变形为约束变形
- 直接操纵的**FFD**方法可以看作是约束变形的一种,但该方法通常需要求解一个大型的广义逆矩阵,计算量很大,不便于动画中的交互设计

基于约束的变形

简单点约束变形

scodef

基于广义圆球的一般约束变形

简单点约束变形模型

- 用户定义一系列约束点,约束点的偏移量和约束点的影响半径.每个约束点决定了一个以约束点为中心的的局部B样条基函数.该函数在影响半径之外为0,空间任意点变形后的位置为这些基函数的组合.

简单点约束变形模型(scodef)

- 设n为待变形空间的维数,Scodef由r个约束点定义. C_i表示约束点在未变形空间的位置, D_i为约束点在变形空间的偏移量,约束点的影响半径为R_i.
- 设Q为R^n空间的一个点. d: $R^n \text{-->} R^n$ 为变形函数, 它表示Q点的偏移量d(Q)

$$d(Q) = [M_1, M_2, \ldots, M_r] \begin{bmatrix} f_1(Q) \\ f_2(Q) \\ \cdots \\ f_r(Q) \end{bmatrix} = \sum_{i=1}^{r} M_i f_i(Q)$$

简单点约束变形模型(scodef)

M_i为一个n维列向量

f_i是一个表示第i个约束的贡献的标量函数

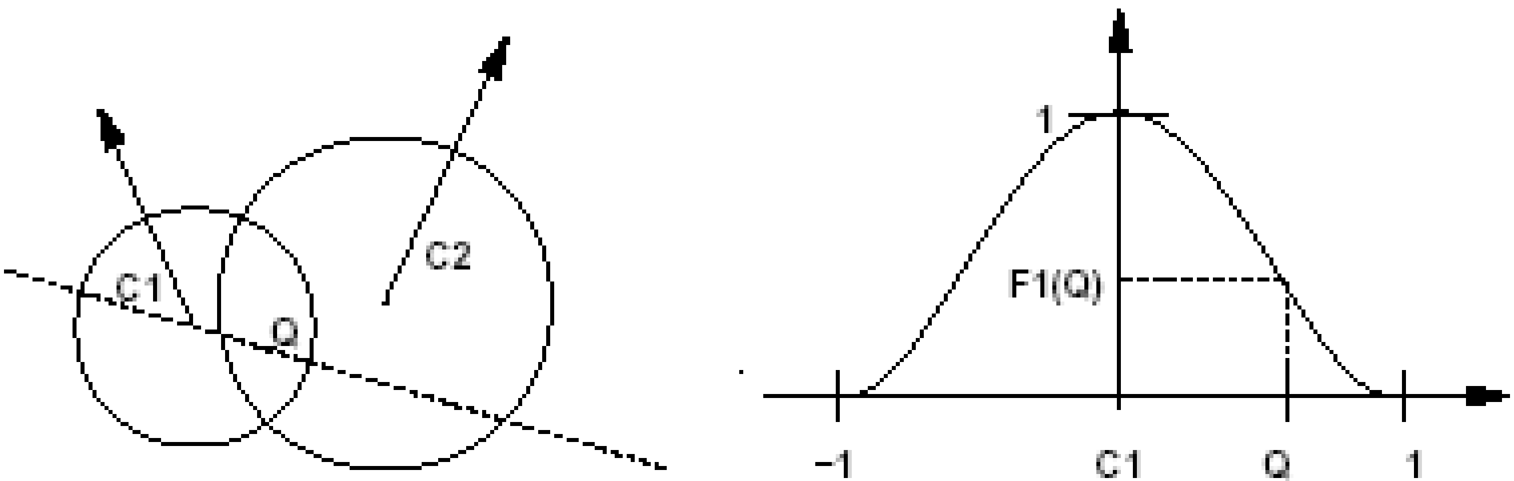
$$f_i(Q) = B_i\left(\frac{\| Q - C_i \|}{R_i}\right)$$

$B_i()$是以0为中心的B样条基函数,在中心的值为1, 影响半径外的值为0

$$f_i(Q) = B_i\left(\frac{\| Q - C_i \|}{R_i}\right)$$

$$U_i(Q) = \frac{\| Q - C_i \|}{R_i}$$

$$f_i(Q) = B_i(U_i(Q))$$



(a) (b)

对于给定约束的f值

$$d(Q)=[M_1,M_2,\dots,M_r]\begin{bmatrix} f_1(Q) \\ f_2(Q) \\ \dots \\ f_r(Q) \end{bmatrix}=\sum_{i=1}^{r} M_i f_i(Q)$$

d(Q)=M*f(Q)

通过选择适当的矩阵M，变形方程可以满足约束条件

n*r的矩阵

r维列向量

为了满足每个约束C_i,矩阵M需满足如下方程

$$D_i = d(C_i) = M * f(C_i) \quad i = 1, 2, \ldots r$$

设D_{ij}为D_i的第j个坐标,M^j为矩阵M的第j行向量

$$D_{ij} = M^j * f(C_i) = f^T(C_i) * (M^j)^T$$

将所有约束合并到一个方程, 第j个空间坐标$D_j(C)$

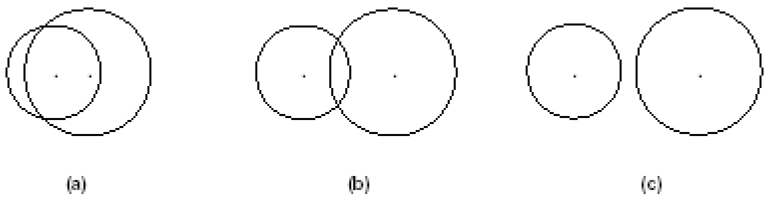
$$D_j(C)=\begin{bmatrix} D_{1j} \\ D_{2j} \\ \cdots \\ D_{rj} \end{bmatrix}=\begin{bmatrix} f^T(C_1) \\ f^T(C_2) \\ \cdots \\ f^T(C_r) \end{bmatrix}(M^j)^T = X(M^j)^T$$

X是一个与约束点有关的r*r矩阵, 不依赖于j

由上述方程的解可以计算M的第j行向量M^j

变形方程的求解

- 约束点之间的关系影响着变形方程求解的难度
- 约束点之间的关系
  - 互相影响(a)
  - 互不相交(b)
  - 完全不相交(c)



变形方程的求解

- 约束点集互不相交时,变形方程求解很简单
  - ❖矩阵M的第i列M_i等于约束点C_i的偏移量$d(C_i)$

$$D_i = M_i$$

- 对于任意点Q

$$d(Q) = \sum_{i=1}^{r} B_i\left(\frac{\| Q - C_i \|}{R_i}\right) * D_i$$

基于广义元球的一般约束变形

- 用户定义一系列由点,线,面和体组成的约束,每个约束的影响半径和偏移量 变形模型根据每个约束和其影响半径产生该约束的广义元球
- 该模型对局部空间进行变形, 与物体的表示无关,变形可以由偏移量和广义元球的参数进行细微调整
- 与其他变形模型相比,该方法不仅有效,直观,而且可以做到线,面,体约束变形和比例变换以及旋转约束变形

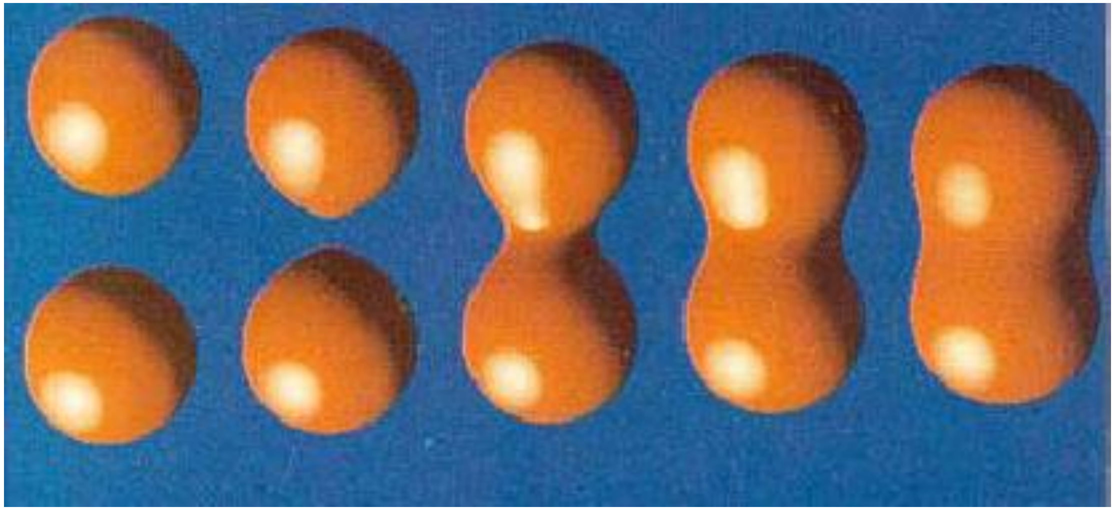
元球(Metaball)造型

- 曲面的表示方法

1. 参数曲面
   - Bezier曲面,B样条曲面,NURBS曲面
   - 参数曲面在体现人体的肌肉,器官和它们的运动方便比较困难
2. 隐式曲面
   - 元球造型是80年代出现的一种隐式曲面造型技术
   - 采用具有等势场值的点集定义曲面
   - 元球生成的曲面是一张等势面

元球(Metaball)造型

- 元球具有相互靠近到一定距离会产生变形,再进一步靠近时会融合成光滑表面的特性



- 元球数足够多时可以生成很复杂的形状

元球(Metaball)造型

- 元球造型的优点

1. 需要的数据量通常比用多边形造型少2~3个数量级
   - 用500个元球可以较好地表示一个人的造型
2. 适合于表示可变形的物体,对柔性物体的动画非常有用
3. 适合于人体,动物器官和液体的造型
4. 生成的曲面是光滑的

元球(Metaball)造型

- 元球造型中,曲面定义为一张等势面
- 需要用户指定的参数包括元球的中心,元球中心的密度,势函数,颜色…
- 对于简单的物体, 元球的造型过程与CSG造型非常相似, 用户只需根据欲造型物体的形状放置元球,并调整其大小和方向,便可以得到一张光滑的曲面

元球的势函数

- 在元球系统中,每个元球都可以有不同的势函数

$$r = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

(x_i,y_i,z_i)是第i个元球的中心

(x,y,z)是空间中的任意一点

元球的势函数

1 Blinn的幂函数

$$f_i(r) = b_i \exp(-a_i r^2)$$

2 Nishimura的分段二次多项式

$$fi(r) = \begin{cases} 1-3*(\frac{r}{R_i})^2 & 0 \le r \le \frac{R_i}{3} \\ \frac{3}{2}(1-(\frac{r}{R_i}))^2 & \frac{R_i}{3} < r \le R_i \\ 0 & r > Ri \end{cases}$$

元球的势函数

3 Murakami的四次多项式

$$fi(r)=\begin{cases}(1-(\frac{r}{R_i})^2)^2 & 0\le r\le R_i\\ 0 & r>Ri\end{cases}$$

4 Wyvill的六次多项式

$$fi(r)=\begin{cases}-\frac{4}{9}(\frac{r}{R_i})^6+\frac{17}{9}(\frac{r}{R_i})^4-\frac{22}{9}(\frac{r}{R_i})^2+1 & 0\le r\le R_i\\ 0 & r>Ri\end{cases}$$

元球的势函数

- 对于一个由n个元球构成的元球系统,其所对应的等势面为满足以下方程的点集

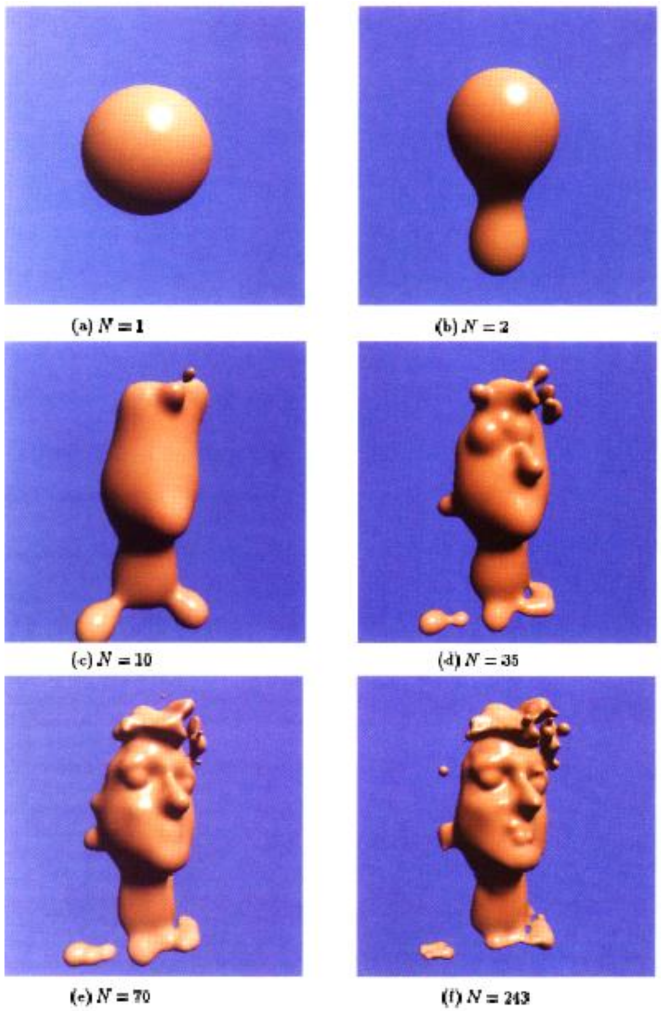
$$f(x,y,z)=\sum_{i=1}^{n} q_i * f_i - T = 0$$

方程中,q_i为第i个元球的密度值(允许为负值),T为阈值

元球(Metaball)造型

- 对于复杂的几何模型, 需要利用通过元球自动拟合物体的技术

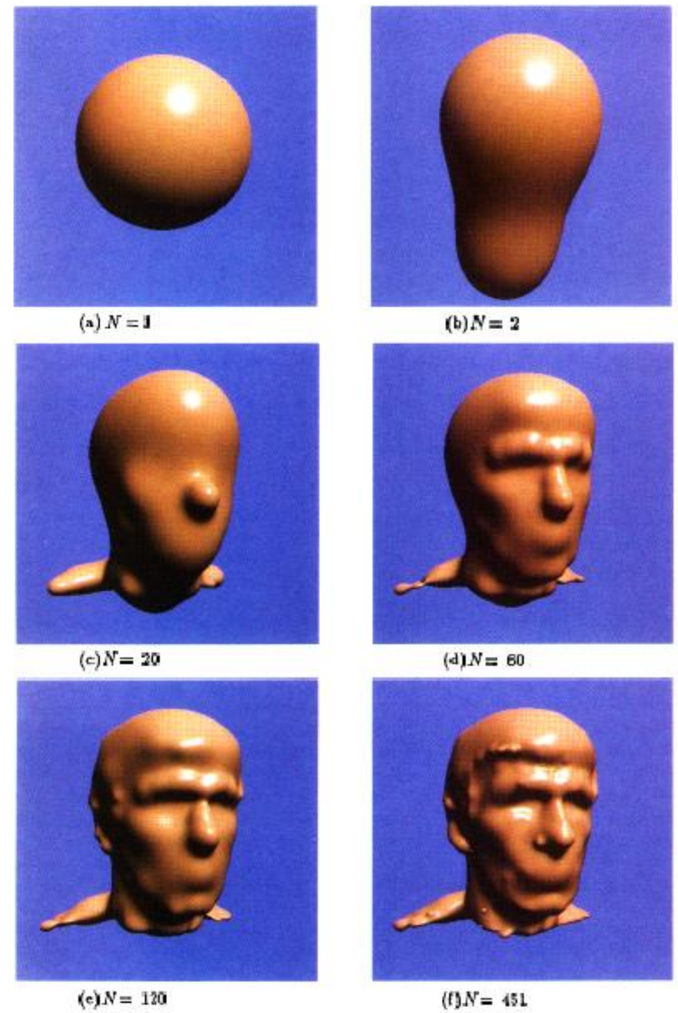
1. 给定待拟合的物体, 首先对它进行点采样. 采样点的数目应足够多,以便采样点能够基本表示物体的形状.
2. 采样后,首先用一个元球拟和采样点集, 然后依据能量最小的原则将该元球分成两个,以便增加拟合的精度.对所有元球重复这个分割过程,直至元球的描述满足拟合要求



(a) $N = 1$ (b) $N = 2$

(c) $N = 10$ (d) $N = 35$

(e) $N = 70$ (f) $N = 243$



(a) $N = 1$ (b) $N = 2$

(c) $N = 20$ (d) $N = 60$

(e) $N = 120$ (f) $N = 451$

广义元球

- 将元球造型技术中,将距离定义的参考"点"推广到"骨架"
  - 骨架可能是线,面,体
  - 设C为一骨架,P(x,y,z)为3D空间中的一点,r(P,C)为点P到骨架C上的点的最短距离

$$r(P,C) = \inf_{Q \in C} \| P - Q \|$$

势函数

- 骨架C相关的势函数F()

$$F(r(P,C),R)=f(r,R)\circ r(P,C).$$

势函数f(r,R)和元球造型中的势函数相同

R为一指定的距离，称为有效半径

广义元球

- 假设C为一个约束骨架,R为其有效半径,S为它对应的距离曲面

$$S=\left\{P(x,y,z)\in S\mid r(P,C)=R\right\}$$

称M=<S, f(r,R)>为一个定义于骨架C上的广义元球

基于广义元球的一般约束变形

当存在n个约束C_i（可以由点，线，面，体组成）时，且这些约束互不相交时，变形函数为

$$Deform(P) = P + \sum_{i=1}^{n} \Delta D_i F(r(P, C_i), R_i)$$

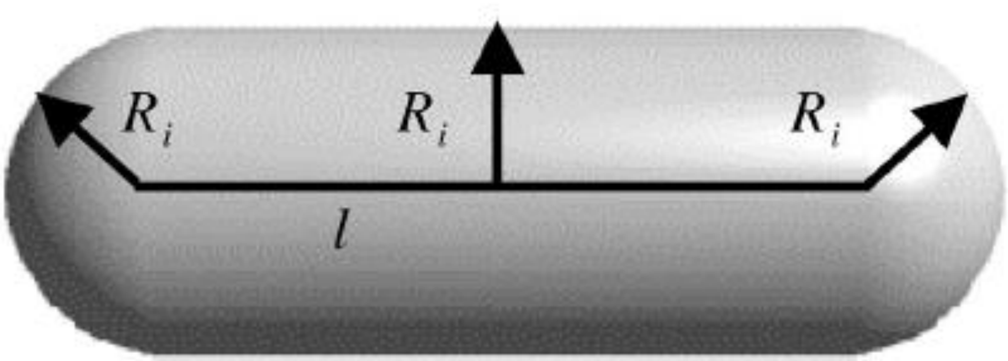
P点的偏移量为所有约束的偏移量的加权平均，权值为相应的势函数值

广义元球的计算

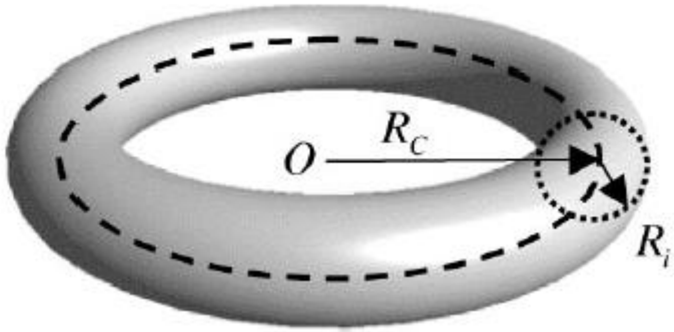
- 由变形模型可以看出, 计算变形函数的关键在于计算距离函数$r(P,C_i)$.
- 当C_i为点约束时, $r(P,C_i)$为点P与点C_i之间的距离.
- 当C_i为线约束,面约束,体约束时, 距离函数的计算稍显复杂.

线段约束对应的广义元球

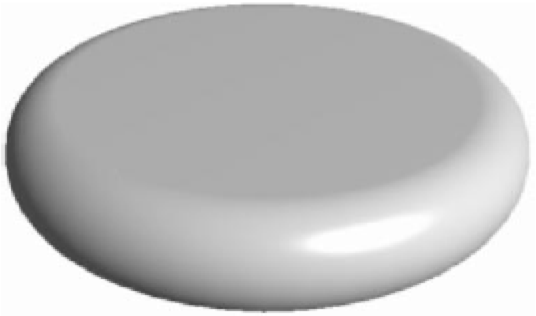
假设约束C_i为一由点P0=(x0,y0,z0)和点P1=(x1,y1,z1)决定的直线段, 长度为L.



圆线约束对应的广义元球



圆面约束对应的广义元球

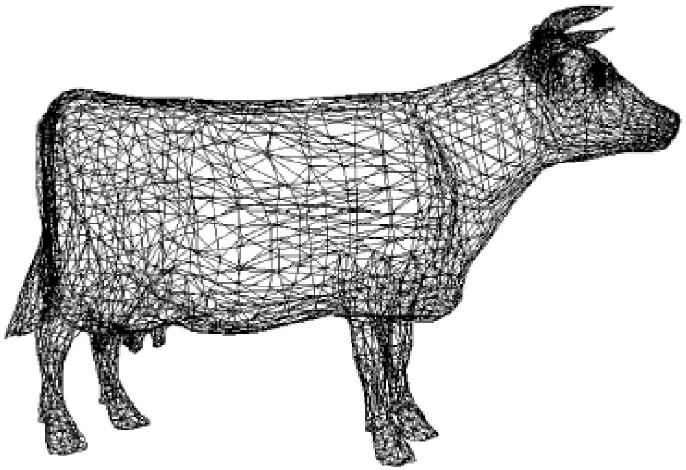


正方形约束对应的广义元球

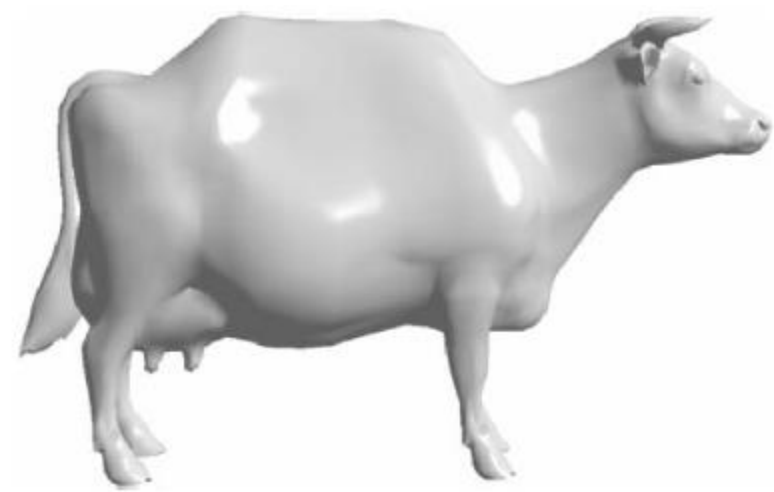
圆柱面约束对应的广义元球

假设约束C_i为一半径为R_c,高为h的圆柱面

正方体约束对应的广义元球



未变形的牛的线框图



由直线约束变形而成

(a)

变形前

(b)

两个平面约束变形后

思考题：

1. Morphing变形与空间变形(Deformation)有何区别？
2. Morphing变形有哪几种方法？
3. 空间变形有哪几种方法？

结束