计算机图形学基础

胡事民

清华大学计算机科学与技术系

计算机动画

- 动画 (Animation) 与计算机动画
  - 动画是一门通过在连续多格的胶片上拍摄一系列图像并将其连续放映来产生动态视觉的技术和艺术
  - 计算机动画采用连续播放静止图像的方法产生物体运动的效果
  - 计算机动画是图形学与艺术相结合的产物

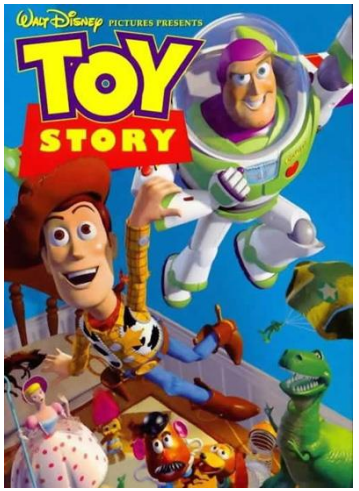
计算机动画

- 计算机动画
  - 随着计算机图形学和硬件技术的高速发展，计算机动画已渗透到人们生活的各个角落
  - 1993年，电影《侏罗纪公园》采用动画特技制作的恐龙片段获得了该年度奥斯卡最佳视觉效果奖

Edwin Catmull & Patrick Hanrahan

计算机动画

- 计算机动画
  - 1996 电影《玩具总动员》上映，该片不仅获得了破记录的票房，而且作为第一部完全用计算机动画制作的长篇动画，给电影制作开辟了一条新的道路
  - “对电影制作和计算机生成图像(CGI)等应用产生革命性的影响” – 图灵奖评语



计算机动画

- **RenderMan**
  - **1981成立的卢卡斯电影公司，于1986年被从苹果辞职的乔布斯花了1000万美元从乔治·卢卡斯手中收购，成立皮克斯动画工作室。**
  - **RenderMan使用接口规范，定义了相机、几何体、材质和灯光等，接口规范有助于3D建模、动画应用程序与生成高质量图像的渲染引擎之间的通信。**
  - **Pat Hanrahan领导了该引擎的开发，1993年获奥斯卡奖。获奖人：** Pat Hanrahan, Anthony A. Apodaca, Loren Carpenter, Rob Cook, Ed Catmull, Darwyn Peachey, and Tom Porter.

计算机动画

- 计算机动画
  - 1998年放映的电影《泰坦尼克》中，船翻沉时乘客的落水镜头有许多是采用计算机合成的，从而避免了实物拍摄中的高难度、高危险动作

计算机动画

- 计算机动画
  - 电影《阿凡达》采用全新的立体拍摄系统，仅用了6周时间，便在全球收入18亿美元
  - 时长仅160分钟，60%的画面采用计算机动画制作，由近800名特效人员，历时4年完成

计算机动画

- 计算机动画
  - 计算机动画不仅用于商业广告、影视特技，还广泛应用于教育、军事领域，例如教学演示、战斗演习、飞行模拟等
  - 简单地说，计算机动画是指用绘制程序生成一系列的景物画面，其中，当前帧画面是对前一帧画面的部分修改，所有这些帧的画面组成一部计算机动画

计算机动画的分类

- 二维动画
  - 图像变形 (Image Morphing)
  - 形状混合 (Shape Blending)
- 三维动画
  - 关键帧动画
  - 变形物体的动画 (自由体变形技术, FFD)
  - 过程动画，人体动画，智能体行为动画等…

计算机动画的分类

- 二维动画
  - 图像变形 **(Image Morphing)**
  - 形状混合 **(Shape Blending)**
- 三维动画
  - 关键帧动画
  - 变形物体的动画 **(自由体变形技术, FFD)**
  - 过程动画，人体动画，智能体行为动画等…

二维动画

- 图像变形 (Image Morphing)
  - 可以基于单张图像直接进行变形，也可以基于两张或多张图像作插值计算而获得变形过程，通过引入用户交互来指定对应的特征，可以使变形更加可控

二维动画

- 图像变形 (Image Morphing)
  - **Feature-based images metamorphosis**

    Beier T. & Neely S., SIGGRAPH 1992
  - **Animating image with drawing**

    Litwinowicz P. & Willams L., SIGGRAPH 1994

图像变形 (Image Morphing)

- 图像变形：任意给定两幅不带任何几何信息的图像 S(source) 和 D(destination)，如何将 S 很自然地变形为图像 D？
  - 图像 S 和 D 表示的可以是完全不同的两种东西
  - S 和 D 可以是灰度图像，也可以是彩色图像

S

D

图像变形 (Image Morphing)

- 关于图像变形，有以下几个核心问题：
  - 如何找到图像 S 与 D 之间各像素点之间的对应关系？
  - 对于彩色图像，如何处理颜色渐变？
  - 怎么样才能使这种变形显得更真实、更自然？

图像变形 (Image Morphing)

- 直接变形法 (cross-fading)
  - 将S和D上的像素按照其空间位置建立对应关系，在每对像素之间对其颜色信息进行插值，以得到变形的中间帧图像

$$S*(1-t) + D*t \quad t \subset [0,1]$$

  - 直接变形法所能做到的是让 S 渐渐消隐，而 D 渐渐的显现，直接变形法得到的中间帧图像常常包含 S 与 D 的交叠，有一种近视眼戴着老花镜的感觉

图像变形 (Image Morphing)

- 直接变形法 (cross-fading)
  - 直接变形法虽然简单，但却比直接切换图像在视觉效果上要平和得多

S

Cross-Fading

D

图像变形 (Image Morphing)

- 基于特征的图像变形法 (feature-based image morphing)
  - 在进行图像变形时，像素的对应关系不像直接变形法中按照空间位置直接对应那么简单
  - 而是通过分析图像的特征建立起对应关系，并在该对应关系的基础上，将几何的变形 (warping) 和颜色的交叉溶解 (cross-dissolving) 结合起来

图像变形 (Image Morphing)

- 基于特征的图像变形法 **(feature-based image morphing)**

Morphing = warping + cross-dissolving

shape (geometric)

color (photometric)

为图像变形加入了几何上的考虑

图像变形 (Image Morphing)

- 与直接变形法的比较：

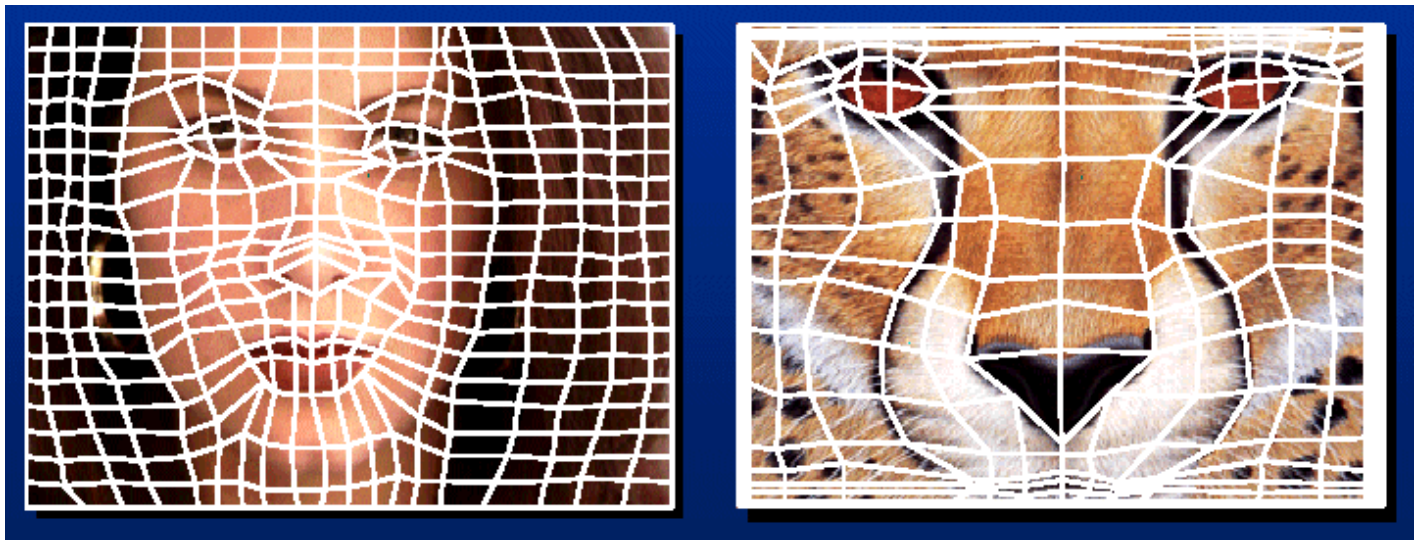
S

D

更真实、更自然！

图像变形 (Image Morphing)

- 核心问题：
  - 如何找到 S 和 D 之间各像素点的对应关系？
  - 让用户标注一组图像的基网格的控制顶点间的对应关系，然后做插值？交互量太大

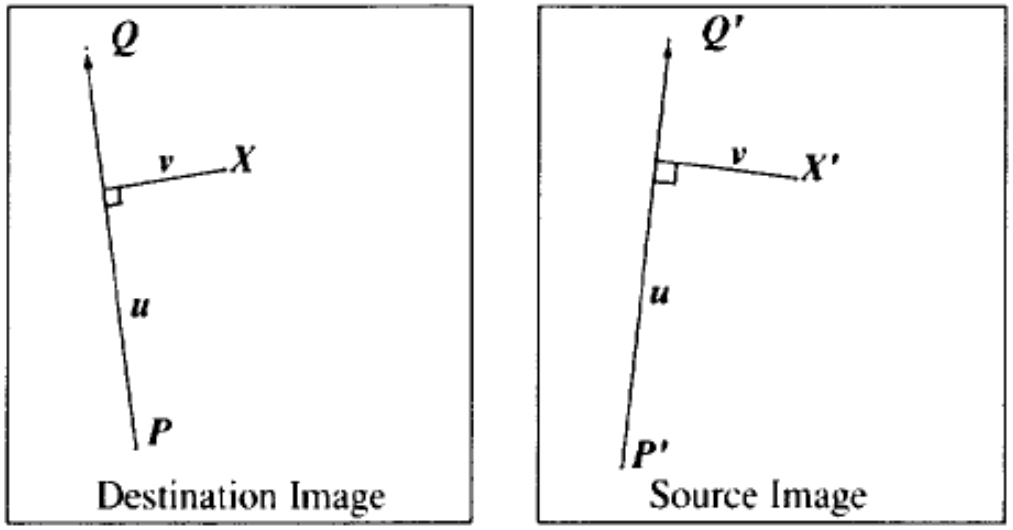


图像变形 (Image Morphing)

- **Beier 和 Neely 的解决方案：**
  - 让用户标注对应的向量 (feature vectors)，然后插值整幅图像 S 和 D 的对应关系
  - 插值过程被称为 Beier & Neely 算法

图像变形 (Image Morphing)

- **Beier & Neely 算法**
  - 首先考虑一个控制向量的情况：
  - **PQ和 P'Q' 分别是 D 和 S 上所对应的参考向量，对 D 上的一点 X，要计算 X 在 S 上的对应点 X'**



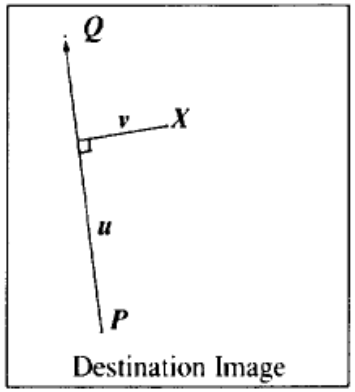
图像变形 (Image Morphing)

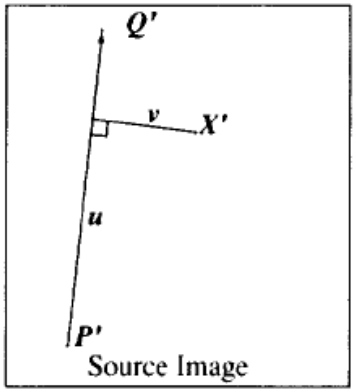
- **Beier & Neely 算法**
  - 保持 X (X’) 到参考向量 PQ (P’Q’) 的投影距离以及投影点在 PQ(P’Q’) 上的相对位置 (即到两端点的距离之比) 不变

$$u = \frac{(X-P)\cdot(Q-P)}{\|Q-P\|^2} \quad (1)$$

$$v = \frac{(X-P)\cdot Perpendicular(Q-P)}{\|Q-P\|} \quad (2)$$

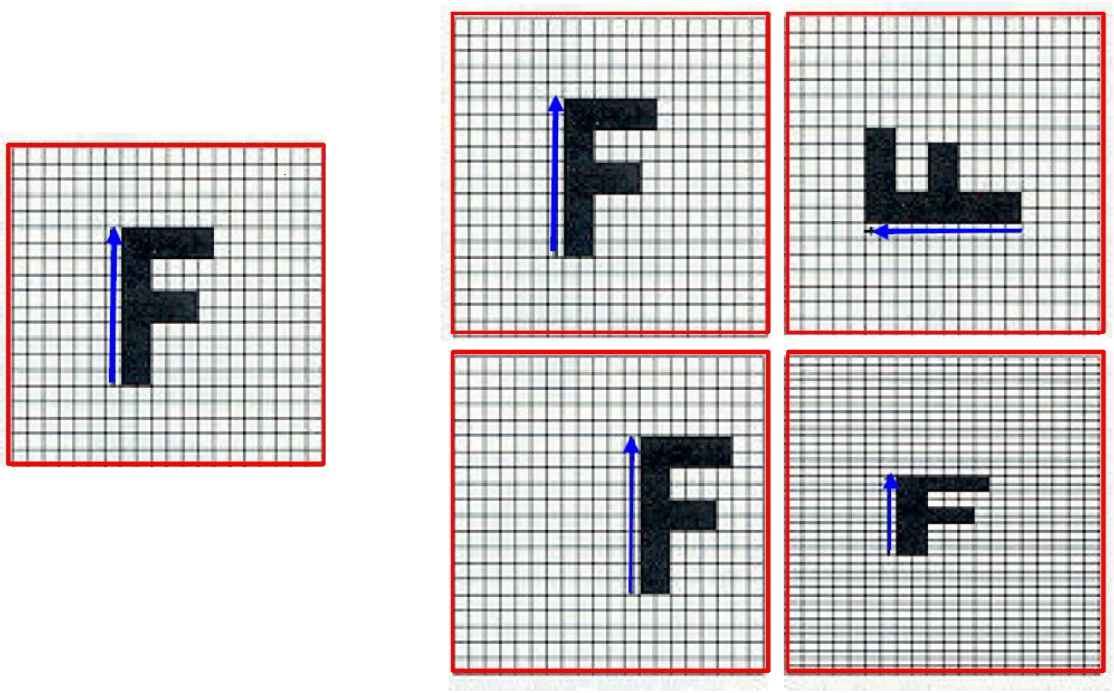
$$X' = P' + u\cdot(Q'-P') + \frac{v\cdot Perpendicular(Q'-P')}{\|Q'-P'\|} \quad (3)$$





图像变形 (Image Morphing)

- **Beier & Neely 算法**
  - 一个控制向量的例子：



图像变形 (Image Morphing)

- **Beier & Neely 算法**
  - 对于多个控制向量的情况，为了计算 X 的对应点 X’，我们针对每个单独的控制向量计算 X 的对应点：X_1’…, X_k’，最后将它们线性加权从而得到 X’：

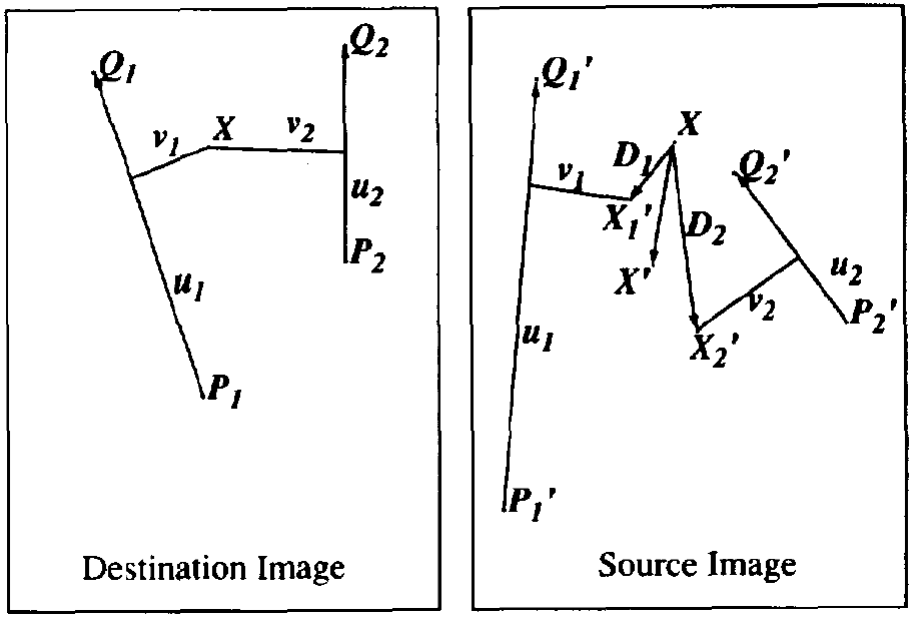
$$X' = weight_1 X_1' + \ldots + weight_k X_k'$$

图像变形 (Image Morphing)

- **Beier & Neely 算法**
  - 权重 **Weight** 的计算方法：

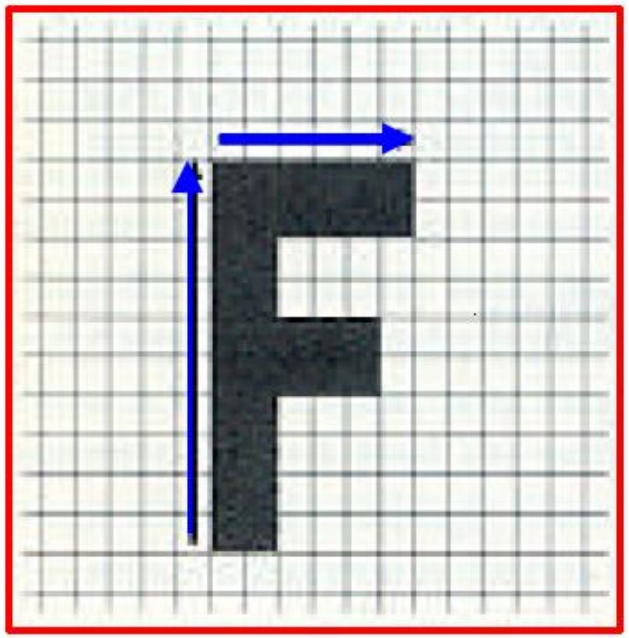
$$weight = \left(\frac{length^{p}}{(a + dist)} \right)^{b}$$

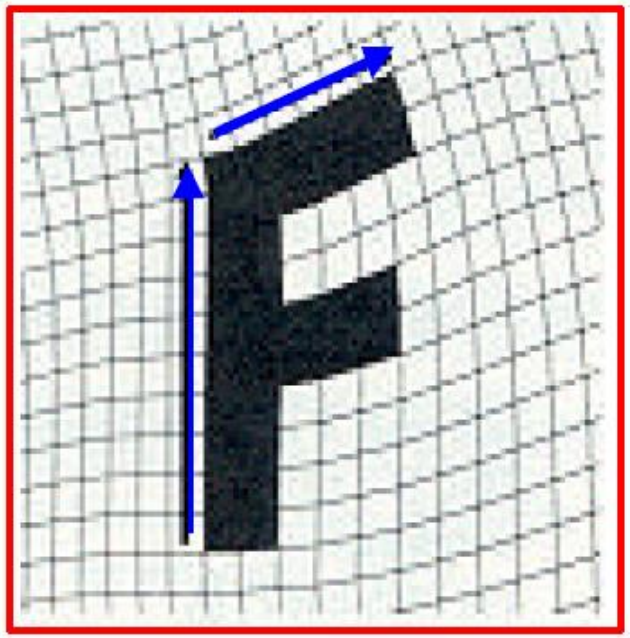
length 是 P_iQ_i 的长度
dist 是 X 到 P_iQ_i 的距离
a, b, p 均为正常数



图像变形 (Image Morphing)

- **Beier & Neely 算法**
  - 多个控制向量的例子：





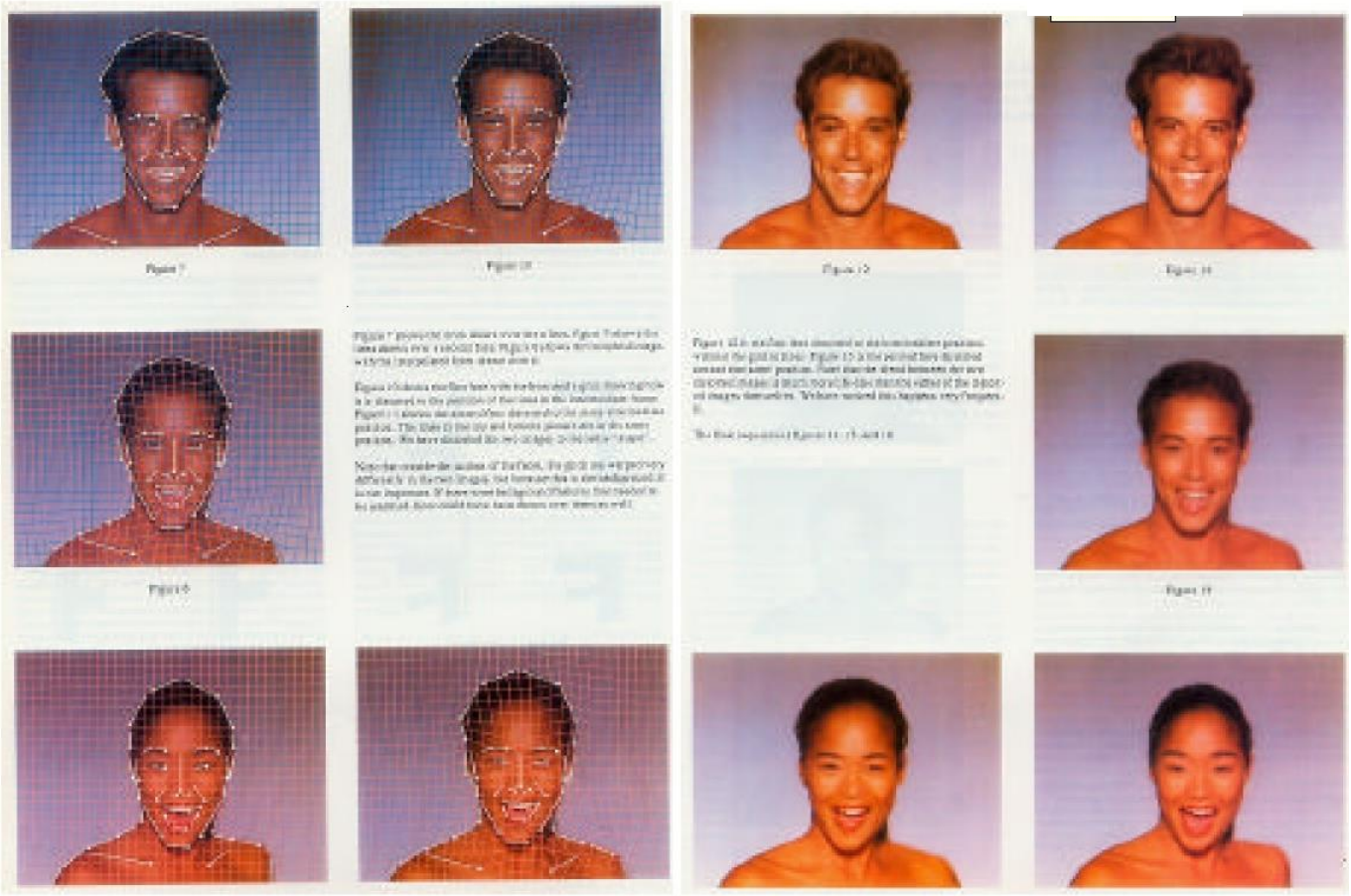
图像变形 (Image Morphing)

- 使用 **Beier & Neely** 算法进行图像变形：
  - 要计算 **S** 和 **D** 之间的任一中间帧图像 **M**
  - 首先插值出 **M** 中的所有的控制向量
  - 对 **M** 中的任一点 **X**，使用 **Beier & Neely** 算法分别计算 **X** 在 **S** 和 **D** 中的对应点：

    X_S 和 X_D
  - 将 **X** 的颜色按 X_S 和 X_D 分别在 **S** 和 **D** 中的颜色混合而成 (比例根据 **M** 的位置决定)

图像变形 (Image Morphing)

- 基于特征的图像变形示例：



多选题 1分

设置

对于有多根特征线的情况，可以采取的方法是

A 对不同特征线组先插值、插值结果再加权

B 对不同特征线组先加权求对应点，再插值

C 对不同特征线组，先求出一对最重要的，再插值

D 从多组特征线，求出一组新的特征线，按新特征线插值

提交

计算机动画的分类

- 二维动画
  - 图像变形 (Image Morphing)
  - 形状混合 (Shape Blending)
- 三维动画
  - 关键帧动画
  - 变形物体的动画 (自由体变形技术, FFD)
  - 过程动画，人体动画，智能体行为动画等…

二维动画

- 形状混合 (Shape Blending)
  - 二维图形动画，都可以简化为多边形处理，二维形状混合，实际上是在两个关键帧的多边形之间插入中间多边形
  - 二维形状混合需要解决的关键问题包括：
    - 顶点对应问题
    - 对应顶点之间的插值路径问题

二维动画

- 形状混合 (Shape Blending)
  - **A physically based approach to 2-D shape Blending**

    Sederberg T.W. & Greenberg E., SIGGRAPH 1992
  - **2-D shape Blending: An Intrinsic Solution to the vertex path problem**

    Sederberg T.W. et al., SIGGRAPH 1993
  - 计算机动画的向量混合方法

    汪国昭等，计算机学报 **1996**

形状混合 (ShapeBlending)

- 2D形状混合，关注于如何在两个多边形之间插入中间的混合多边形，使之变化尽量合理、自然
- 二维形状混合需要解决的关键问题有：
  - 顶点对应问题
  - 对应顶点之间的插值路径问题

形状混合 (ShapeBlending)

- 我们将在有了顶点对应关系的前提下，关注于对应顶点间的插值路径问题，下面介绍三个主要的算法：顶点插值法、几何内在参数法、边向量混合法
- 关于顶点对应问题可以参考：

  A physically based approach to 2–d shape blending

  Sederberg T.W. & Greenwood E., SIGGRAPH 1992

形状混合 (ShapeBlending)

- 顶点插值法
  - 给定具有相同顶点个数的两个多边形：

    $P = [P_1,\ldots, P_m]$，$Q = [Q_1,\ldots, Q_m]$

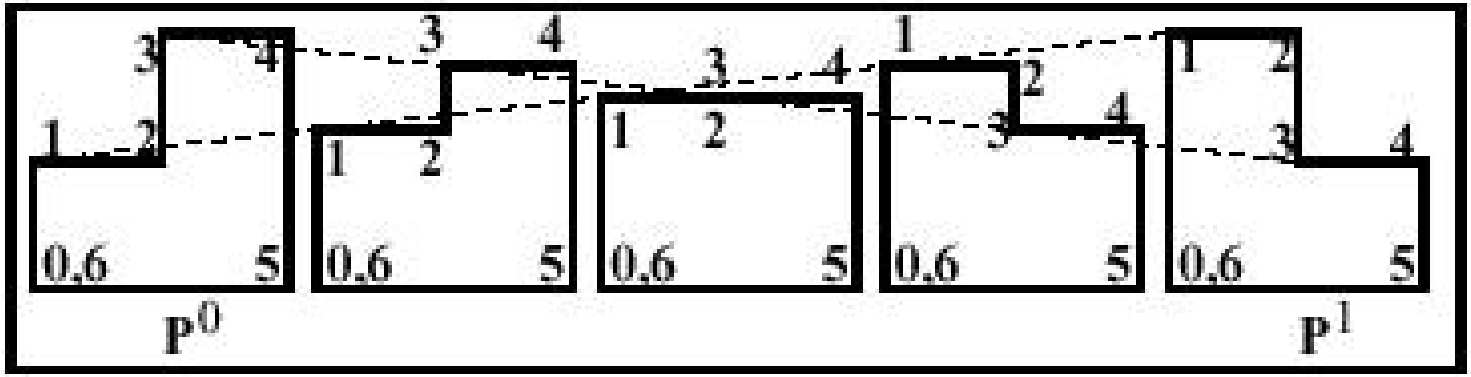
    并且顶点之间已经有了对应关系：$<P_i, Q_i>$
  - 顶点插值法使用对应点间的线性插值来实现形状混合：

    $R(t) = (1–t)P + tQ$

    $= [(1–t)P_1+tQ_1,\ldots, (1–t)P_m+tQ_m]$

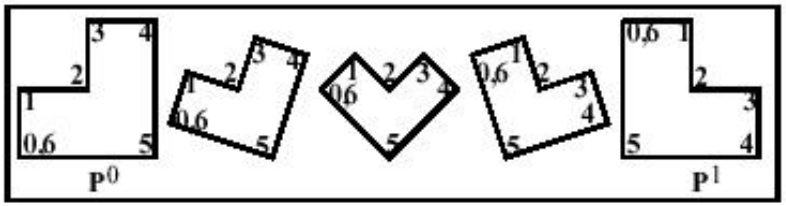
形状混合 (ShapeBlending)

- 顶点插值法
  - 顶点插值法简单快速易实现，却很容易导致不自然甚至包含边萎缩的病态变化过程

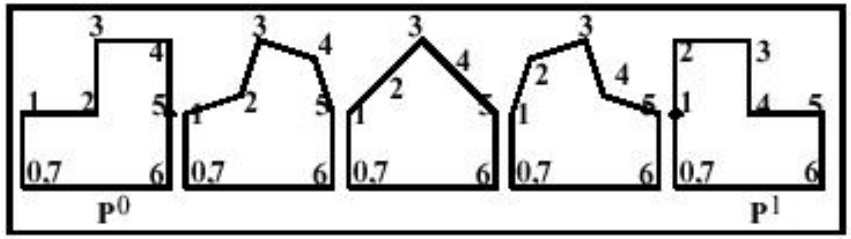


形状混合 (ShapeBlending)

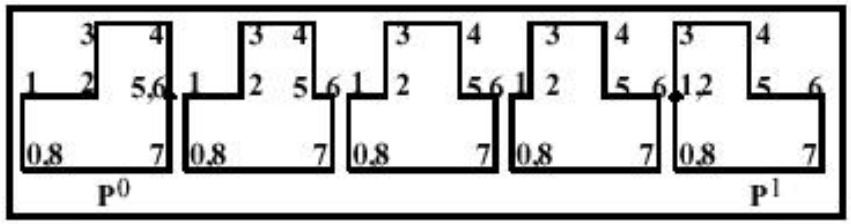
- 顶点插值法
  - 可以使用顶点重新编号和插入新顶点的策略来消除前面的退化情况，但是都无法获得最理想的混合效果



最理想的混合效果



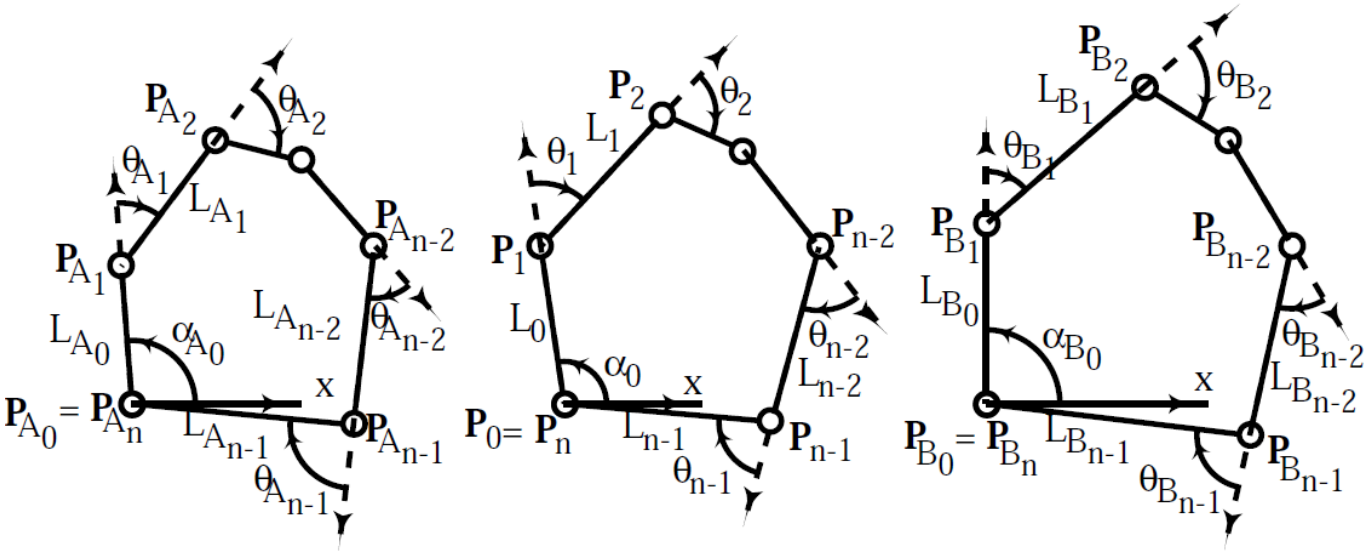
插入一个新顶点的顶点插值



插入两个新顶点的顶点插值

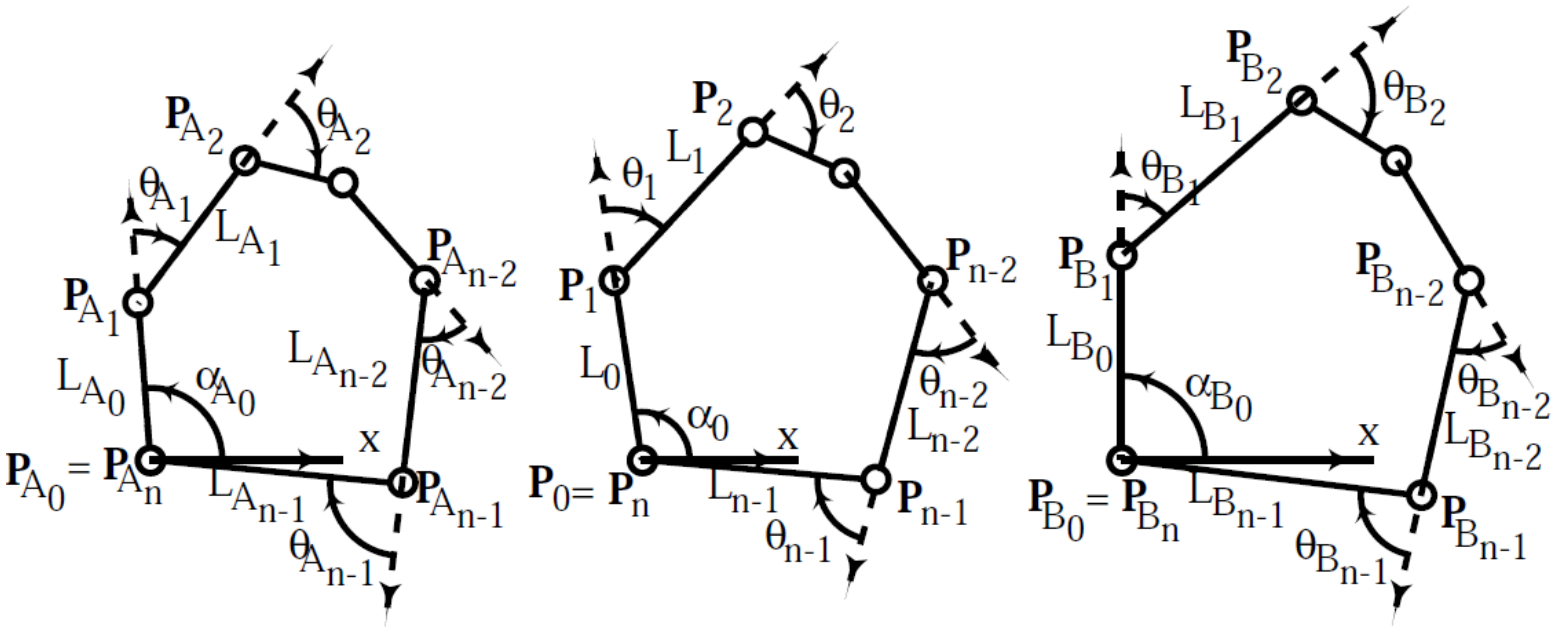
形状混合 (ShapeBlending)

- 几何内在参数法
  - 不同于顶点插值法，几何内在参数法并不对顶点的位置进行直接插值，而是先插值出多边形的边长和顶角，再计算顶点的位置



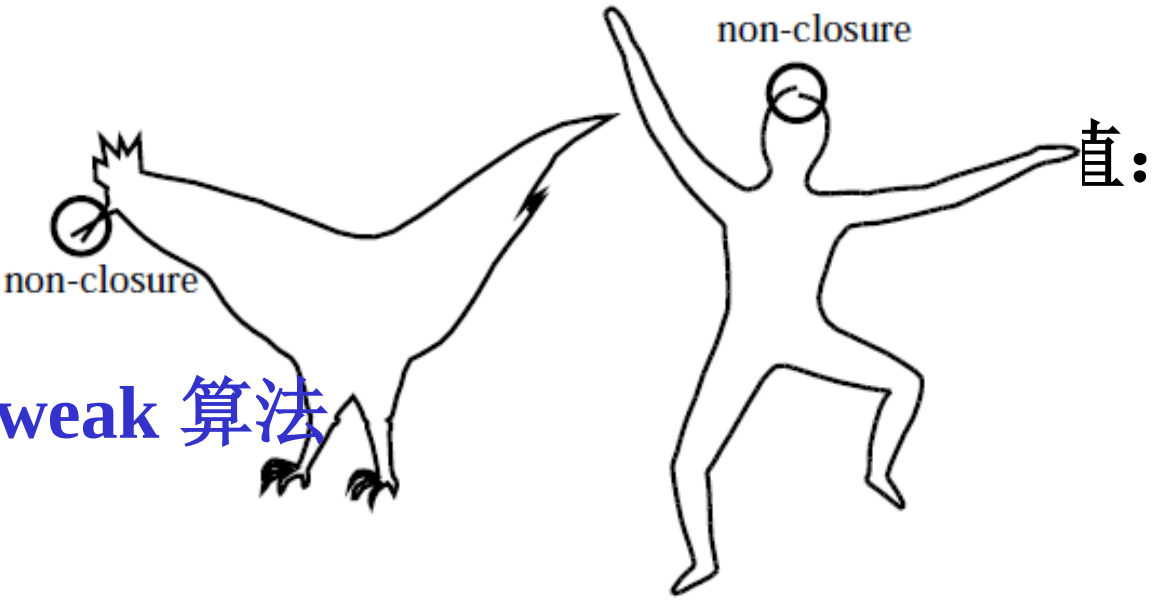
形状混合 (ShapeBlending)

- 几何内在参数法
  - $\theta_i(t) = (1-t)\theta_{Ai} + t\theta_{Bi}$
  - $L_i(t) = (1-t)L_{Ai} + tL_{Bi}$



形状混合 (ShapeBlending)

- 几何内在参数法
  - 问题：如此插值出来的边长 L_i 以及顶角 θ_i 不一定能围成一个封闭的多边形
  - S
    值：
    I



tweak 算法

形状混合 (ShapeBlending)

- 几何内在参数法
  - 为了保证多边形的封闭，对于 S_i 有两个线性约束：

$$\varphi_1(S_0, S_1, \ldots, S_m) = \sum_{i=0}^{m} [(1-t)L_{A_i} + tL_{B_i} + S_i] \cos \alpha_i = 0,$$

$$\varphi_2(S_0, S_1, \ldots, S_m) = \sum_{i=0}^{m} [(1-t)L_{A_i} + tL_{B_i} + S_i] \sin \alpha_i = 0,$$

  - α_i 是第 i 条边和 *x* 轴的夹角
  - 在这两个约束之下，我们 S_i 尽可能的小

形状混合 (ShapeBlending)

- 几何内在参数法
  - 优化如下的能量：

$$f(S_0, S_1, \ldots, S_m) = \sum_{i=0}^{m} \frac{S_i^2}{L_{AB_i}^2}$$

  - 这是一个带约束的优化问题，可以使用拉格朗日乘数法进行求解；具体地说，优化：

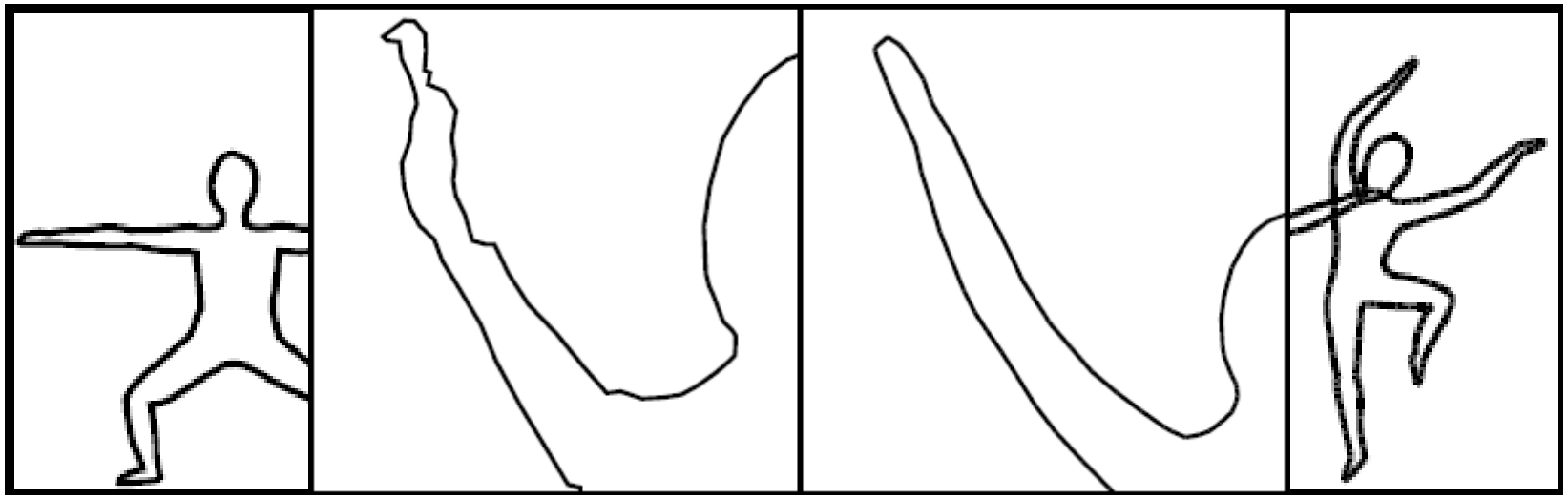
$$\Phi(\lambda_1, \lambda_2, S_0, S_1, \ldots, S_m) = f + \lambda_1 \varphi_1 + \lambda_2 \varphi_2,$$

这是一个典型的二次凸优化问题，相当于求解一个线性方程组

形状混合 (ShapeBlending)

- 几何内在参数法
  - 算法结果示例：

手臂处的局部放大图



顶点插值法　　几何内在参数法

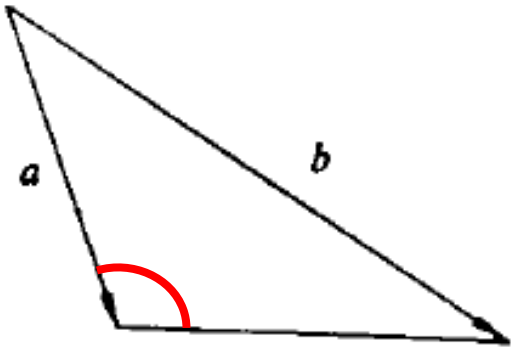
形状混合 (ShapeBlending)

- 边向量混合法
  - 顶点插值法不能保证多边形的边长在变形过程中单调，甚至有可能发生边萎缩
  - 几何内在参数法运算量大，速度慢
  - 汪国昭等人提出的边向量混合法通过对边长向量直接进行插值操作，在实时计算的同时保证的边长的单调性

形状混合 (ShapeBlending)

- 边向量混合法
  - 假设两个多边形的边向量分别为 $\{a_1,\ldots, a_m\}$ 和 $\{b_1,\ldots, b_m\}$，对边向量进行插值
  - 对于对应的向量 a_i 和 b_i，如果由 a_i 和 b_i 构成的三角形的两个底角中有一个为非锐角，那么直接进行线性插值：

$$c_i(t) = (1–t)a_i + tb_i$$

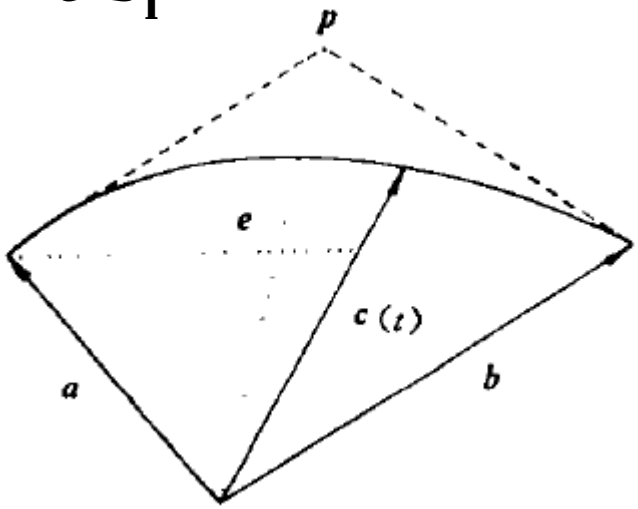


形状混合 (ShapeBlending)

- 边向量混合法
  - 如果由 a_i 和 b_i 构成的三角形的两个底角均为锐角，那么使用二次 Bezier 曲线插值：

$$c_i(t) = (1-t)^2 a_i + 2(1-t)t p_i + t^2 b_i$$

其中 $p_i = (a_i+b_i)*(1+\lambda_i)/2$

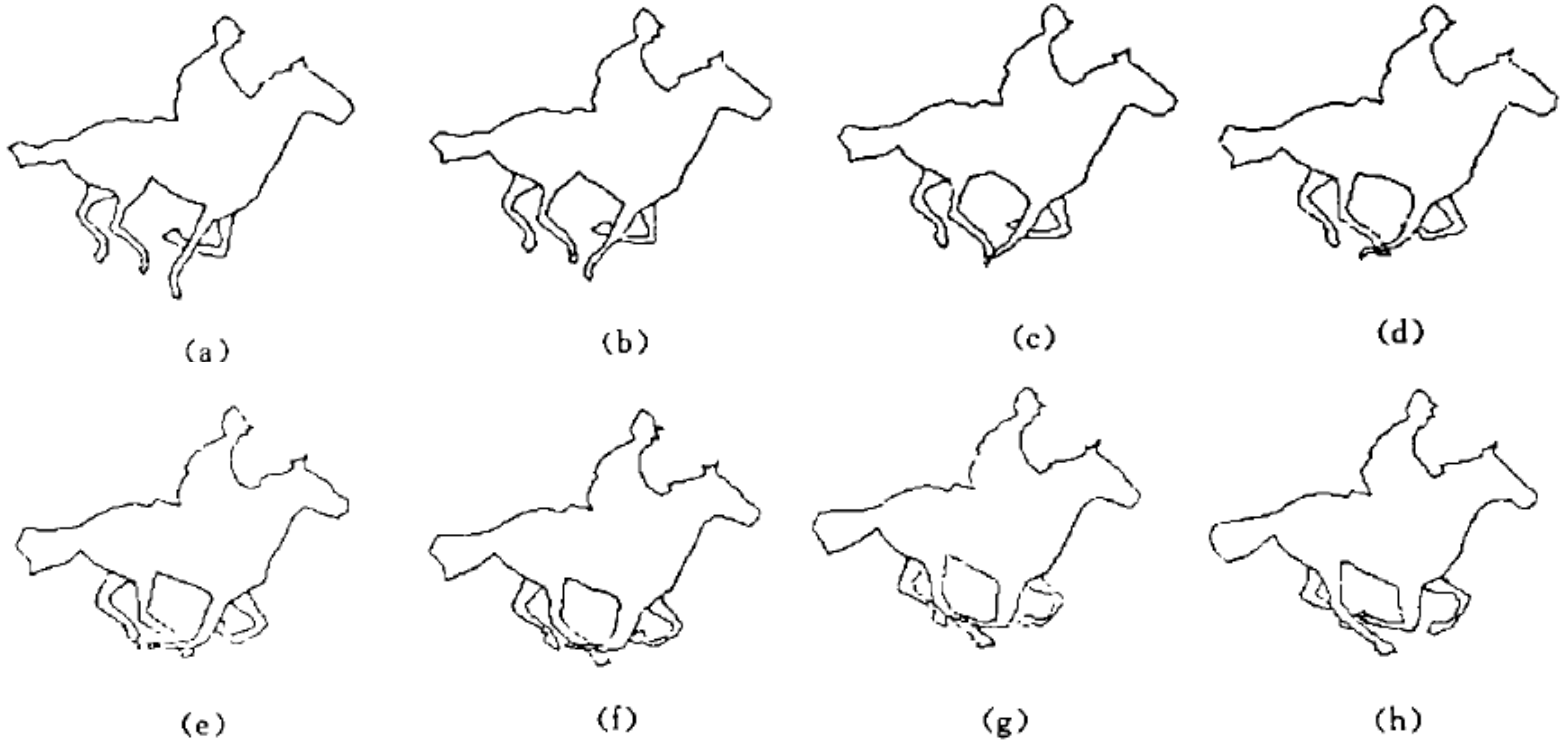


形状混合 (ShapeBlending)

- 边向量混合法
  - 一个重要的问题是如何保证多边形封闭性，即 $c_1(t) + \ldots + c_m(t) = 0$
  - 汪国昭等人在文章中通过对第一类情况 (即由 a_i 和 b_i 构成的三角形的两个底角中有一个为非锐角时) 的插值方程进行松弛，指出在一定的条件下，可以在满足多边形封闭性的同时依旧保持边长的单调性

形状混合 (ShapeBlending)

- 边向量混合法
  - 结果示例：



计算机动画的分类

- 二维动画
  - 图像变形 (Image Morphing)
  - 形状混合 (Shape Blending)
- 三维动画
  - 关键帧动画
  - 变形物体的动画 (自由体变形技术, FFD)
  - 过程动画，人体动画，智能体行为动画等…

三维动画

- 关键帧动画
  - 关键帧动画的概念，源于传统卡通片制作。在早期 **Disney** 的动画片制作室，由熟练的动画师设计卡通片中的关键画面，即关键帧，再由一般的动画师设计中间帧
  - 在计算机动画中，中间帧由计算机生成，插值计算代替了设计中间帧的动画师，而影响画面图像的参数都可以成为关键帧的参数，如位置、旋转角、纹理等参数

三维动画

- 关键帧动画
  - 关键帧技术是计算机动画中最基本，且运用最广泛的方法之一
  - 关键帧动画技术需要解决的问题是：给定一物体运动的轨迹，求物体在某一帧的位置
  - 运动轨迹可以用参数样条表示，如果直接对参数空间进行等间隔采样，势必引起运动的不均匀，因此需要对样条进行弧长参数化

三维动画

- 变形物体的动画
  - 刚体的动画缺乏生气，因此以物体的形变来渲染出夸张的效果是传统动画的特点
  - 在计算机动画中，物体变形的方法主要有：
    - **Barr 的变形变换**

      Global and local deformation of solid primitives

      Barr A.H., SIGGRAPH 1984

三维动画

- 轴变形方法

  Axial deformation: an intuitive technique

  Lazarus F. et al., CAD 1994

- 面变形方法

  一种应用参数曲面的动态自由变形方法

  冯结青，马利庄，彭群生，计算机学报 **1997**

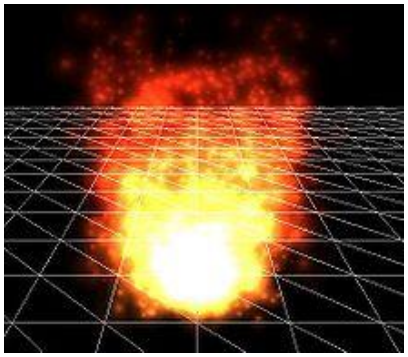
- 自由体变形方法 **(FFD)**

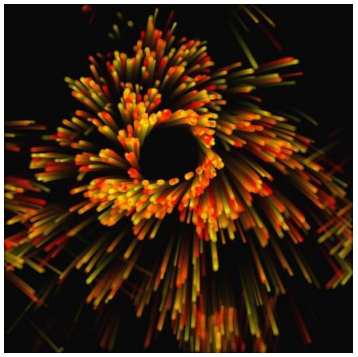
  Free-form deformation of solid geometric models

  Sederberg T.W. & Parry R., SIGGRAPH 1986

三维动画

- 过程动画
  - 过程动画是用一个过程去控制物体的动画
  - 过程动画也涉及物体的变形，但与柔性体的变形不一样，物体的变形基于一定的数学模型，或物理规律
  - 粒子系统是过程动画的典型例子





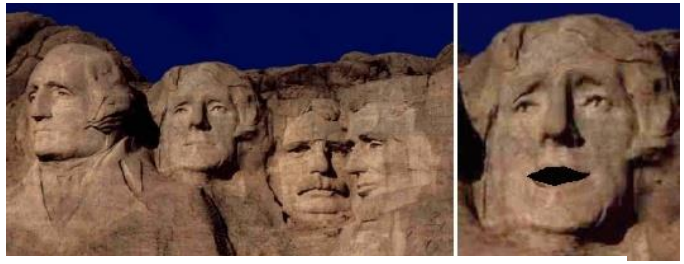
三维动画

- 关节动画与人体动画
  - 人体与动物的动画，至今仍未很好解决
    - 人体有 200 个以上的自由度和复杂的运动模式
    - 人的肢体形状不规则
    - 人的肌肉会随人体的运动而变形
    - 人的个性和表情也千变万化
  - 关节动画与人体动画通常涉及运动学与动力学的知识

三维动画

- 关节动画与人体动画
  - **Voice Puppetry (声音木偶)**

    Brand M., SIGGRAPH 1999



Speech and Facial Animation

A Deep Learning Approach for Generalized Speech Animation
Sarah Taylor (University of East Anglia), Taehwan Kim, Yisong Yue (California Institute o
Krahe, Anastasio Garcia Rodriguez (Disney Research Pittsburgh), Jessica Hodgins (Carnegie
(Disney Research Pittsburgh)

Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion
Tero Karras, Timo Aila, Samuli Laine (NVIDIA Research), Antti Herva (Remedy Entertainment
and Aalto University)

Synthesizing Obama: Learning Lip Sync from Audio
Supasorn Suwajanakorn, Steve Seitz, Ira Kemelmacher-Shlizerman (University of Washington)

VoCo: Text-based Insertion and Replacement in Audio Narration
Zeyu Jin (Princeton University), Gautham Mysore, Stephen DiVerdi, Jingwan Lu (Adobe Resea
University)

三维动画

- 智能生物的行为动画
  - 结合人工智能的一种动画形式

    涂晓媛博士的“人工鱼”，被学术界称之为“晓媛的鱼”

    Artificial fishes: Physics, locomotion, perception, behavior.

    X Tu, D Terzopoulos., SIGGRAPH 1994,

    1166 citation

中文的专著《人工鱼》出版, 介绍该工作

三维动画

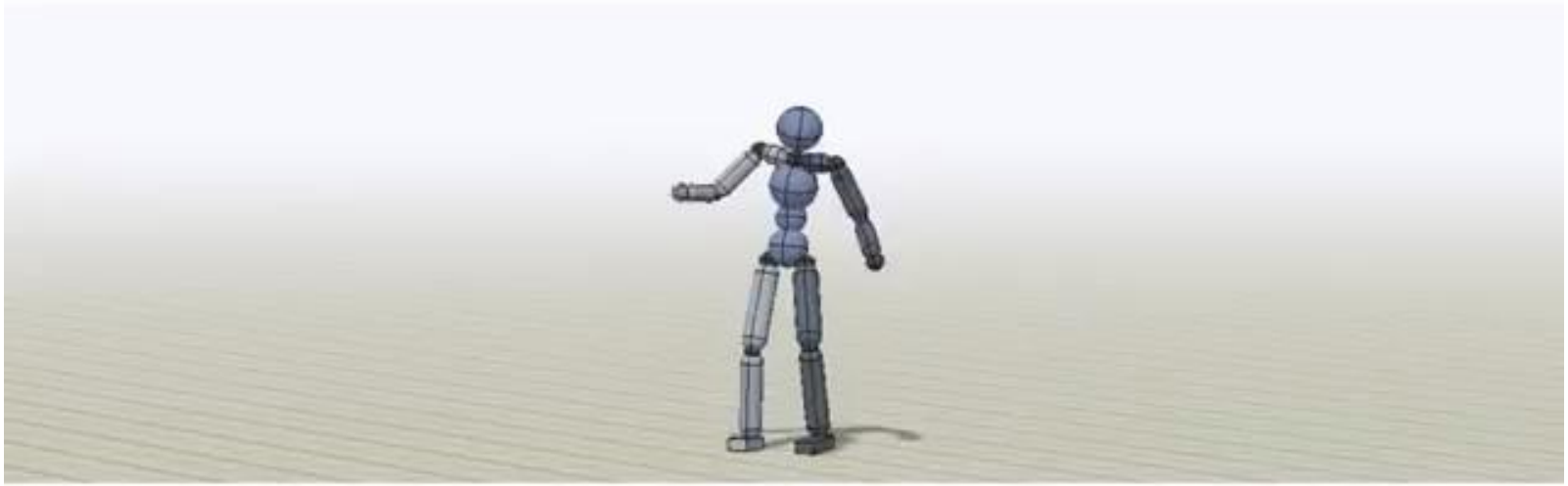
- 智能生物的行为动画
  - 晓媛的鱼

三维动画

- 智能生物的行为动画
  - 随着近年来人工智能领域的发展，许多工作脱颖而出
  - 其中，**UC Berkeley在SIGGRAPH 2018上发表的DeepMimic工作，使用深度增强学习模仿人体自然姿态动画，取得了较大的影响力**

三维动画

DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills



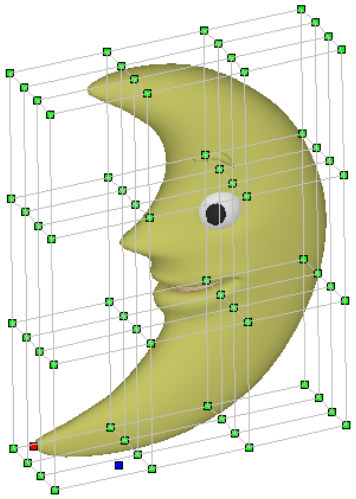
Xue Bin Peng[1], Pieter Abbeel[1], Sergey Levine[1], Michiel van de Panne[2]

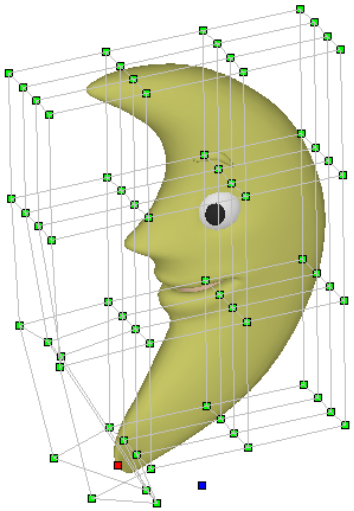
[1] University of California Berkeley

[2] University of British Columbia

自由体变形技术 (FFD)

- **FFD 方法最早由 Sederberg 于 1986 年提出**
  FFD 的思想是：将变形物体嵌入一个简单而柔韧的实体中，这个包含嵌入物的实体的变形，使嵌入物发生相应的变形





自由体变形技术 (FFD)

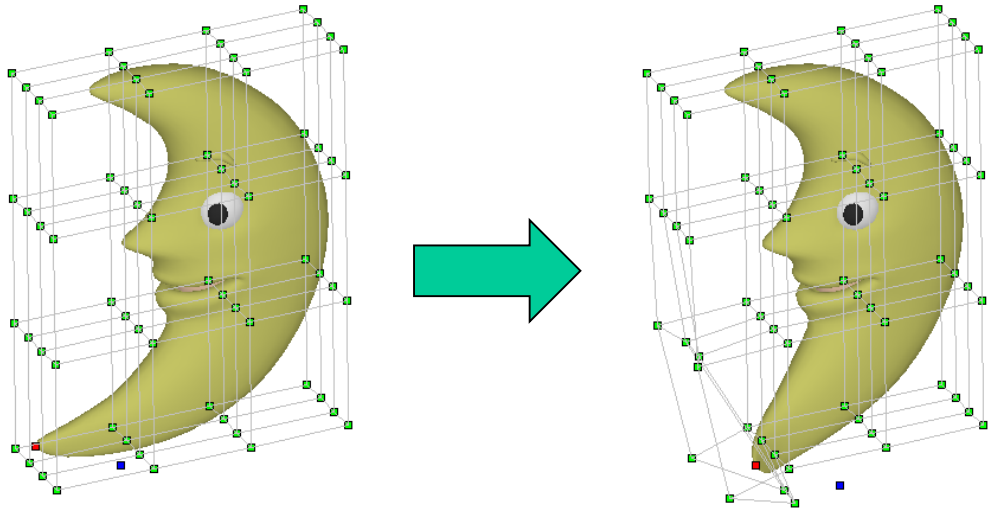
- **FFD 方法主要由以下四个步骤组成：**
  - 构造一个足够大的三参数的自由体，如张量积 **Bezier** 体
  - 将欲变形的物体“嵌入”到自由体中，变形物上各点都可以对张量积 **Bezier** 体反求参数
  - 张量积 **Bezier** 体通过改变控制顶点作变形
  - 对变形物上各点，由参数按新控制顶点计算点变形后的新位置，由此得到新的变形体

TW Sederberg, SR Parry, Free-form deformation of solid geometric models，ACM Siggraph，1986 （**3855 cites**）

自由体变形技术 (FFD)

- 基于NURBS基函数的FFD方法：

$$V(u,v,w)=\frac{\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}V_{i,j,k}W_{i,j,k}B_{i,p}(u)B_{j,q}(v)B_{k,r}(w)}{\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}W_{i,j,k}B_{i,p}(u)B_{j,q}(v)B_{k,r}(w)},\qquad 0\le u,v,w\le 1 \qquad (1)$$



修改控制点 $V_{i,j,k}$

自由体变形技术 (FFD)

- 我们下面介绍 **FFD** 的直接操作算法，即 **DFFD (Direct Manipulation of FFD)**
- 在 **DFFD** 中，用户可以直接选定物体上的一些点，并给出其想要变换到的目标位置，然后算法可以自动地计算出控制点需要进行的变换

- **Direct manipulation of free-form deformations**

  Hsu W.M., Hughes J.F., and Kaufman A., SIGGRAPH 1992 （811 cites）

自由体变形技术 (FFD)

- **DFFD 的优点：**
  - **有了 DFFD, 使用 FFD 来表示复杂的物体变形变得更加直观**
  - **用户不再需要理解复杂的Bezier 张量积或者是 NURBS 理论，而只需对一些关键点指定好目标点 (Targets)，所有的理论和运算都被 DFFD 隐藏在算法后面，以获得更好的用户体验**

自由体变形技术 (FFD)

- **DFFD 的挑战：**
  - **如何才能快速地计算控制点的目标位置，以支持实时/交互的动画变形控制**
  - **我们希望找到一个显式的解决方案 (explicit solution)**
    - SM Hu, H Zhang, CL Tai, JG Sun, Direct manipulation of FFD: efficient explicit solutions and decomposible multiple point constraints, The Visual Computer 17 (6), 370-379. **(110 cites)**

自由体变形技术 (FFD)

- 为了找到显式解，我们先考察只包含一个点约束条件 (point constraint) 的 DFFD：
- 对于基于 NURBS 的 FFD，我们有：

$$\boldsymbol{Q}(u,v,w)=\sum_{i,j,k=0}^{l,m,n}\boldsymbol{P}_{i,j,k}R_{i,j,k}(u,v,w),\quad 0\le u,v,w\le 1$$

$$R_{i,j,k}(u,v,w)=\frac{W_{i,j,k}B_{i,p}(u)B_{j,q}(v)B_{k,r}(w)}{\sum_{i,j,k=0}^{l,m,n}W_{i,j,k}B_{i,p}(u)B_{j,q}(v)B_{k,r}(w)}$$

自由体变形技术 (FFD)

- 假设希望一个点 S 变形后成为 T，那么我们实际上相当于需要计算控制顶点的位移 $\delta_{i,j,k}$，使得：

$$T = \sum_{i,j,k=0}^{l,m,n} (P_{i,j,k} + \delta_{i,j,k}) R_{i,j,k}(u_s, v_s, w_s)$$

$$= S + \sum_{i,j,k=0}^{l,m,n} \delta_{i,j,k} R_{i,j,k}(u_s, v_s, w_s)$$

- 如何计算 $\delta_{i,j,k}$ ？
- 使用带限制条件的优化算法来求解

自由体变形技术 (FFD)

- 希望优化的能量函数是：

$$\sum_{i,j,k=0}^{l,m,n} \left\| \delta_{i,j,k} \right\|^2$$

- 而限制条件是：

$$T = S + \sum_{i,j,k=0}^{l,m,n} \delta_{i,j,k} R_{i,j,k}(u_s, v_s, w_s)$$

自由体变形技术 (FFD)

- 使用拉格朗日乘数法，优化：

$$\boldsymbol{L}=\sum_{i,j,k=0}^{l,m,n}\left\|\boldsymbol{\delta}_{i,j,k}\right\|^2+\boldsymbol{\lambda}(\boldsymbol{T}-\boldsymbol{S}-\sum_{i,j,k=0}^{l,m,n}\boldsymbol{\delta}_{i,j,k}R_{i,j,k}(u_s,v_s,w_s))$$

- 这是一个典型的二次凸优化问题，相当于求解线性方程组：

$$\begin{cases}\boldsymbol{T}=\boldsymbol{S}+\sum_{i,j,k=0}^{l,m,n}\boldsymbol{\delta}_{i,j,k}R_{i,j,k}(u_s,v_s,w_s)\\ \boldsymbol{\delta}_{i,j,k}=\dfrac{\boldsymbol{\lambda}}{2}R_{i,j,k}(u_s,v_s,w_s),\quad 0\le i\le l,\ 0\le j\le m,\ 0\le k\le n\end{cases}$$

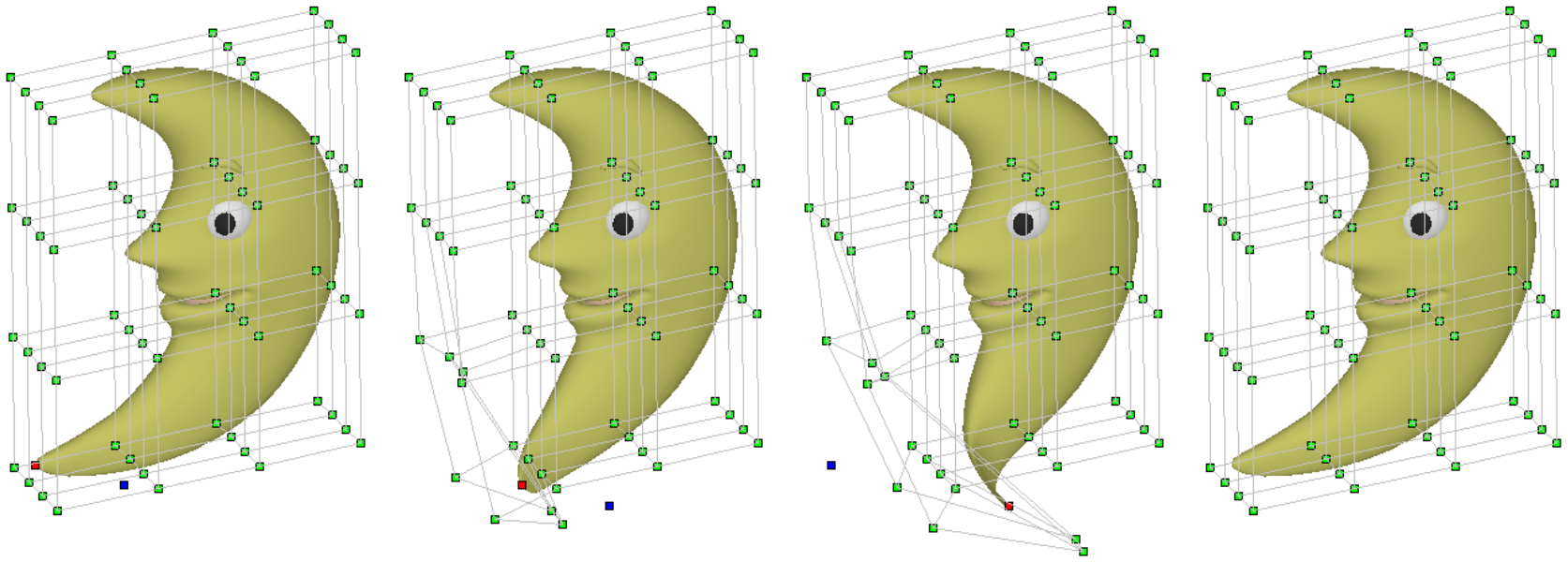
自由体变形技术 (FFD)

- 显式解为：

$$\delta_{i,j,k} = \frac{R_{i,j,k}(u_s, v_s, w_s)}{\sum_{i,j,k=0}^{l,m,n} {R_{i,j,k}}^2 (u_s, v_s, w_s)} (T - S)$$

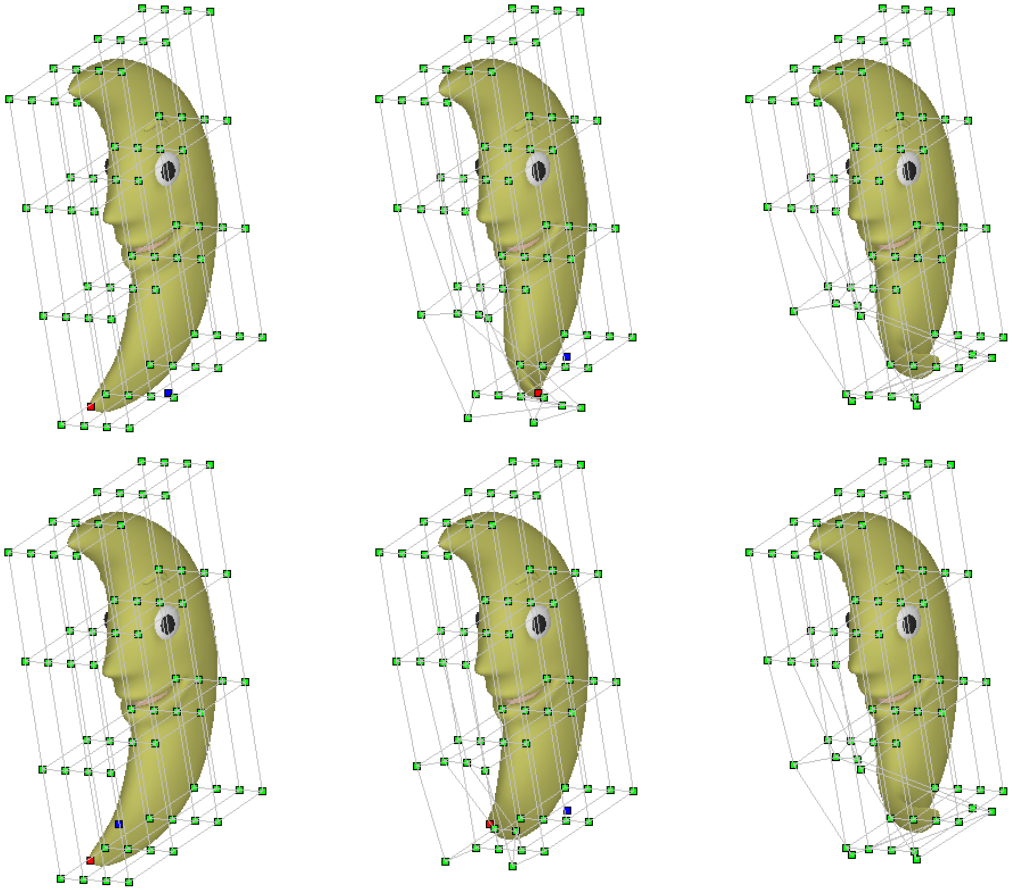
自由体变形技术 (FFD)

- 只包含一个点约束条件 (point constraint) 的 DFFD 示例：



自由体变形技术 (FFD)

- 使用 DFFD 进行连续变形时与路径无关：



自由体变形技术 (FFD)

- 现在，让我们考虑多点约束条件下 (multi-point constraints) 的 DFFD：
- 假设有 h 个点约束条件：

  希望点 S_f 变形后成为点 T_f (f=1,2…h)

  那么相当于要求：

$$T_f = S_f + \sum_{i,j,k=0}^{l,m,n} \delta_{i,j,k} R_{i,j,k}(u_f, v_f, w_f)$$

自由体变形技术 (FFD)

- 使用拉格朗日乘数法，相当于优化：

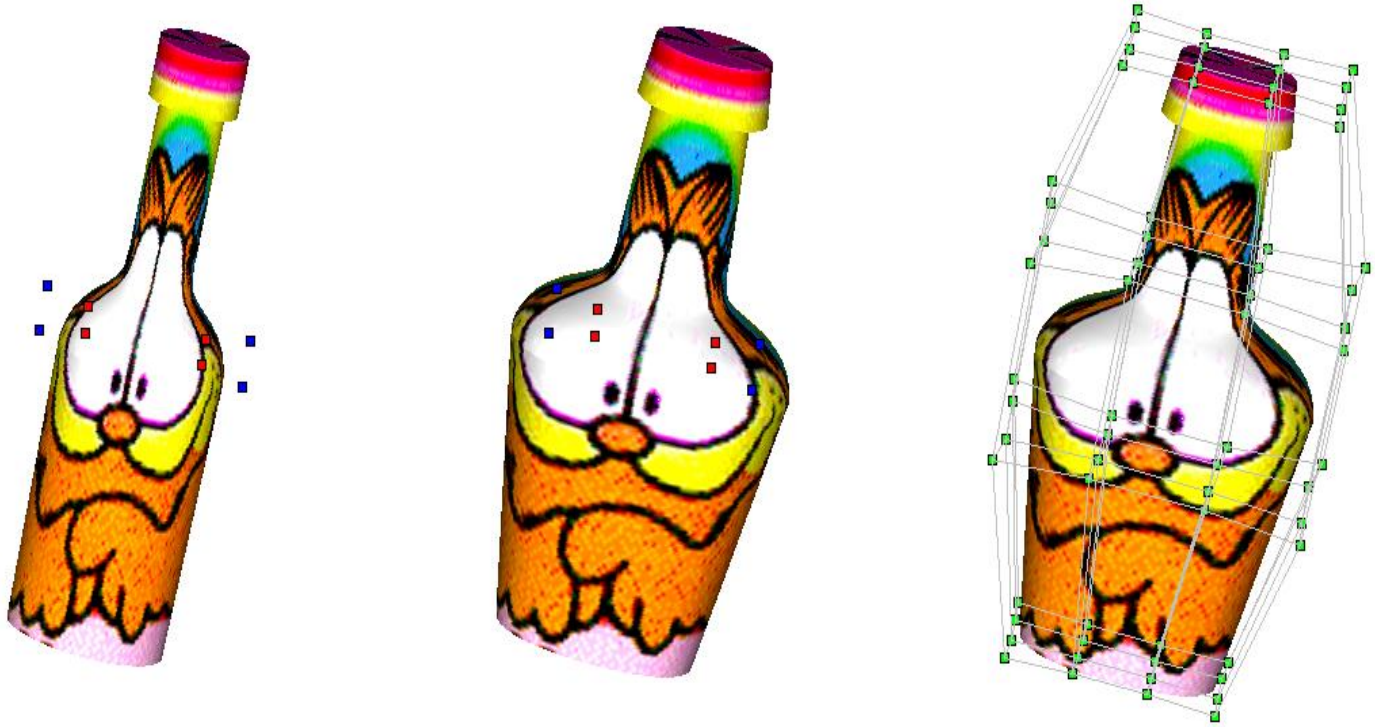
$$\boldsymbol{L}=\sum_{i,j,k=0}^{l,m,n}\left\|\boldsymbol{\delta}_{i,j,k}\right\|^2+\sum_{f=1}^{h}\boldsymbol{\lambda}_f(\boldsymbol{T}_f-\boldsymbol{S}_f-\sum_{i,j,k=0}^{l,m,n}\boldsymbol{\delta}_{i,j,k}R_{i,j,k}(u_f,v_f,w_f))$$

- 又等价于求解线性方程组：

$$\begin{cases}\boldsymbol{T}_f=\boldsymbol{S}_f+\sum\limits_{i,j,k=0}^{l,m,n}\boldsymbol{\delta}_{i,j,k}R_{i,j,k}(u_f,v_f,w_f), & f=1,\cdots,h\\ \boldsymbol{\delta}_{i,j,k}=\sum\limits_{f=1}^{h}\dfrac{\boldsymbol{\lambda}_f}{2}R_{i,j,k}(u_f,v_f,w_f), & \\ & 0\le i\le l,\ 0\le j\le m,\ 0\le k\le n\end{cases}$$

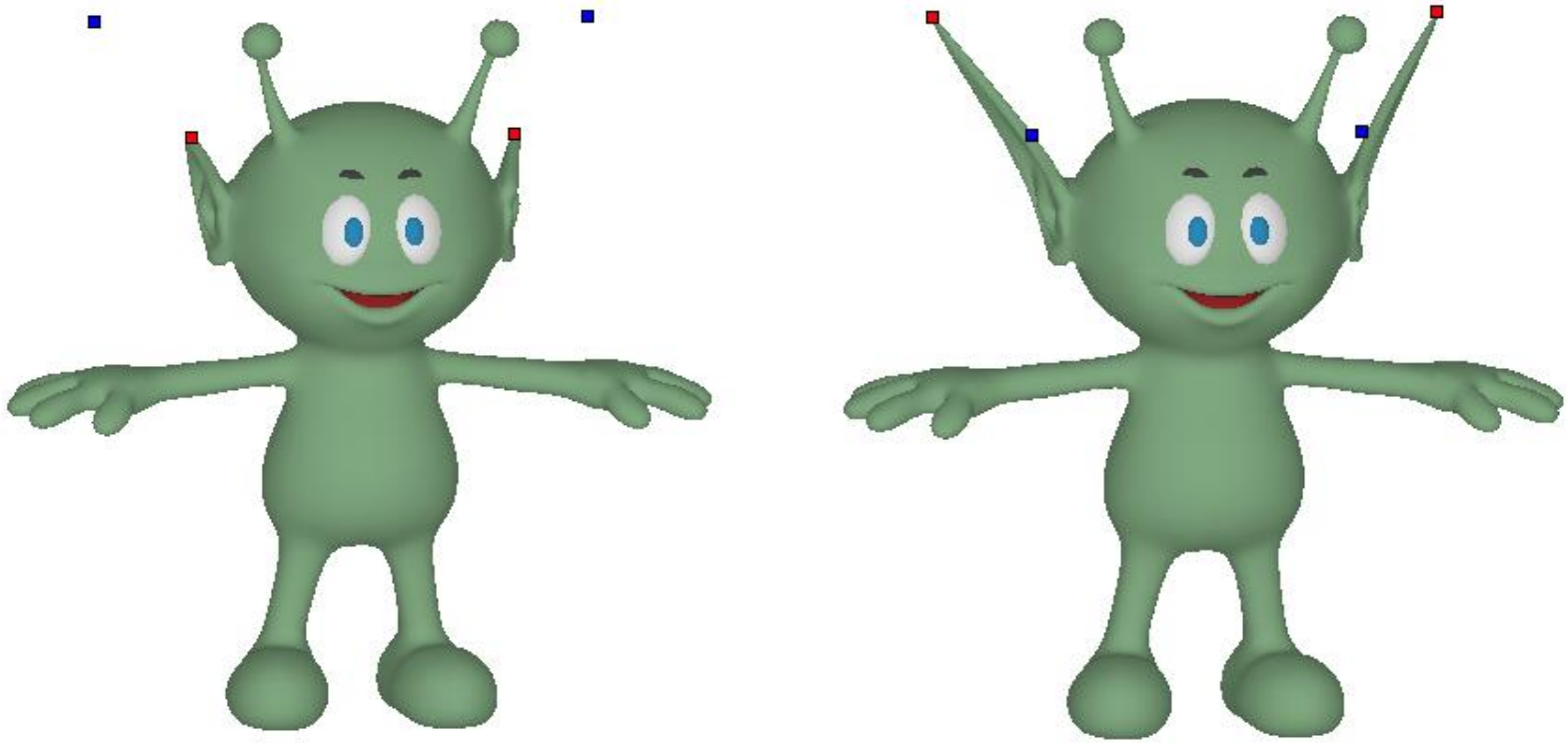
自由体变形技术 (FFD)

- 一些多点约束条件的 DFFD 的例子：



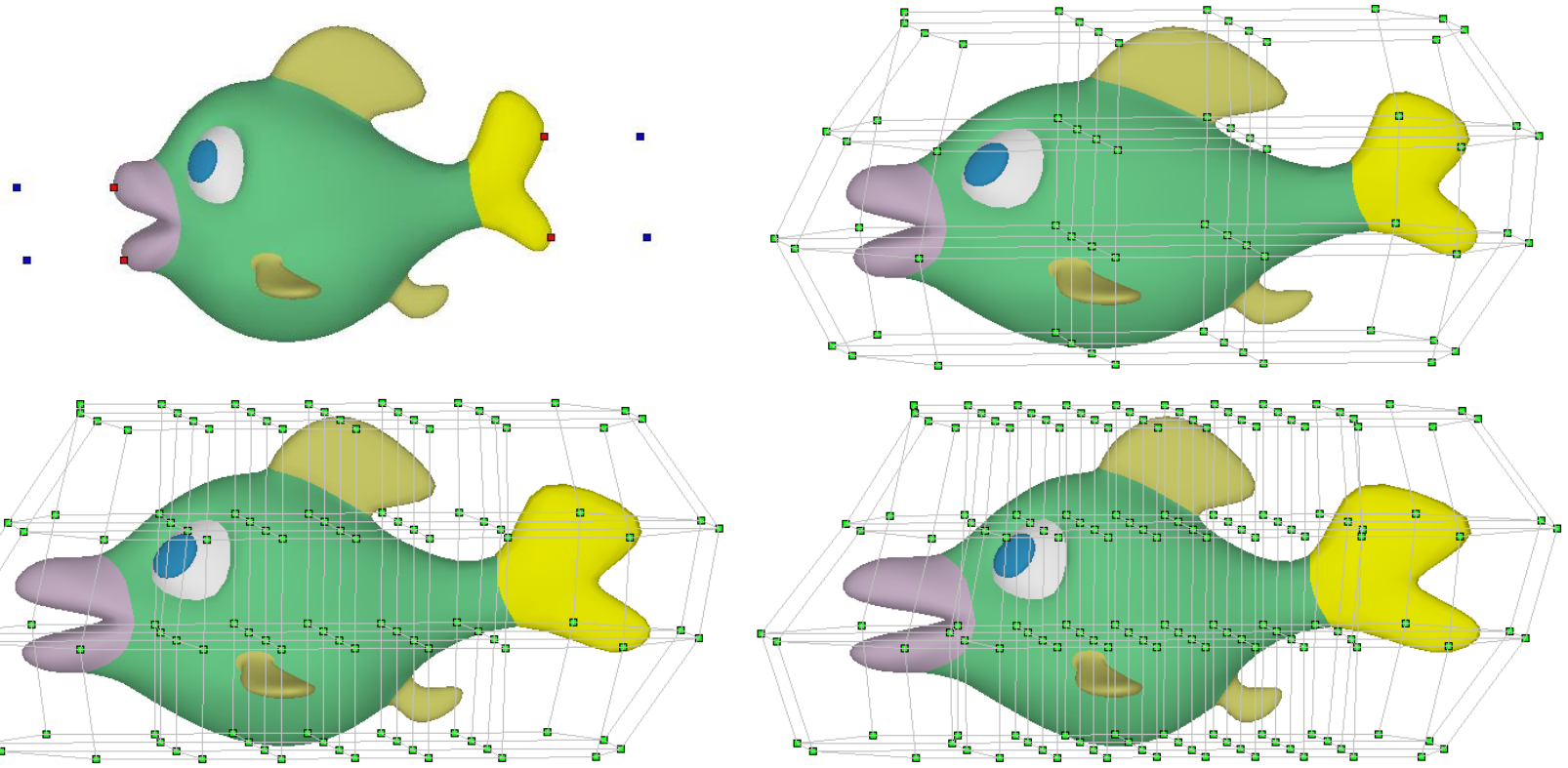
自由体变形技术 (FFD)

- 一些多点约束条件的 DFFD 的例子：



自由体变形技术 (FFD)

- 一些多点约束条件的 DFFD 的例子：



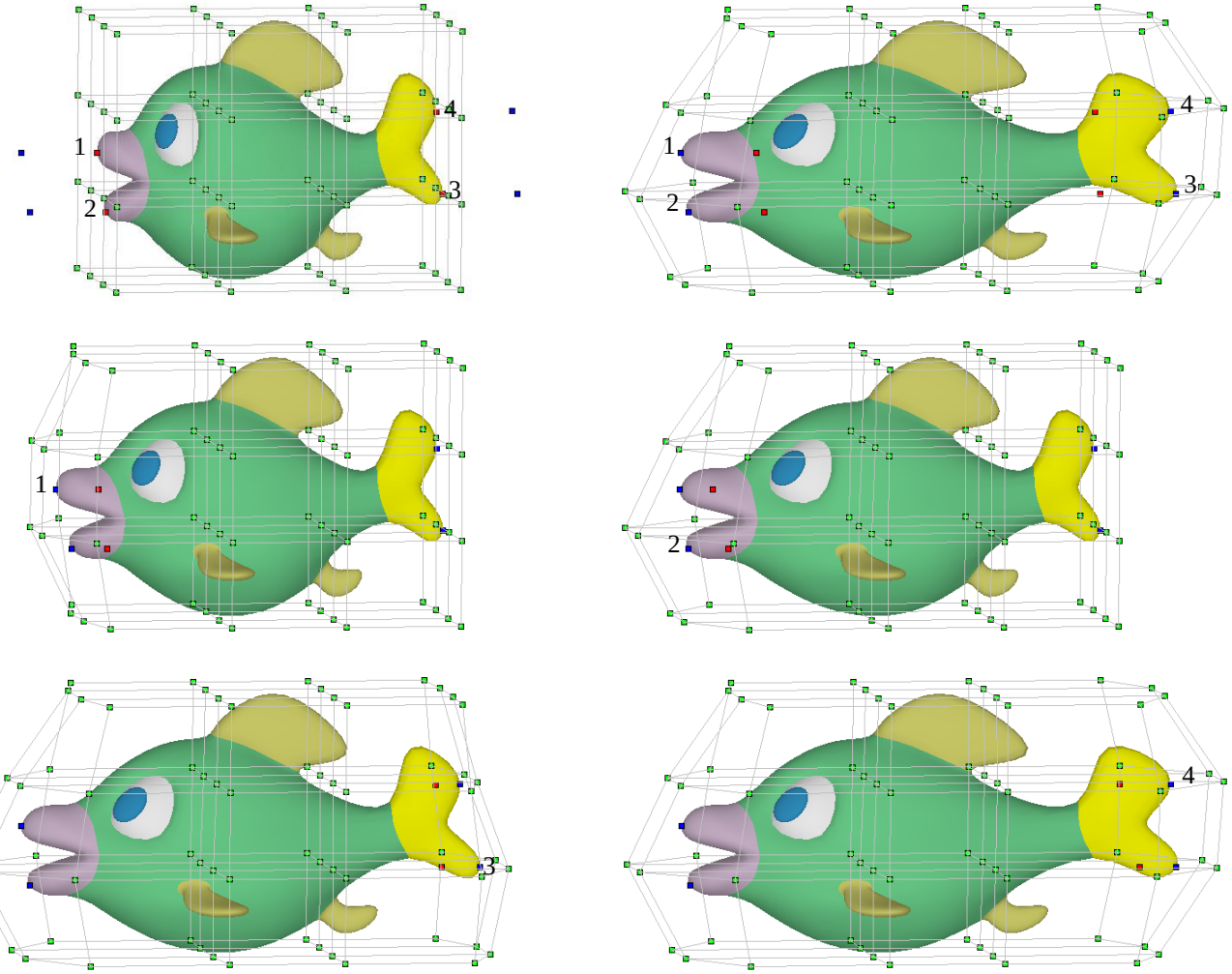
自由体变形技术 (FFD)

- 多点约束条件的分解 (Decomposability of Multiple Point Constraints)
  - 我们看到，在单点约束条件下，DFFD 存在显式解；而在多点约束条件下，却需要求解一个较大规模的复杂的线性方程组
  - 有没有办法通过对多点约束条件下的 DFFD 问题进行分解，转化成为单点约束条件下的 DFFD？

自由体变形技术 (FFD)

- 多点约束条件的分解 (Decomposability of Multiple Point Constraints)
- 定理. 一个包含 h 个点约束条件下的 DFFD 问题可以分解为 h 个单点约束条件下的 DFFD 问题依次串接而成
- 以上定理可以帮助我们做出多点约束条件下 DFFD 问题的显式解

自由体变形技术 (FFD)



自由体变形技术 (FFD)

- 基于权值修改 (Modification of Weights) 的 DFFD：
  - 不同于修改可控制顶点，我们也可以通过修改 NURBS 函数的权值来实现 DFFD：

$$\boldsymbol{T} = \frac{\sum_{i,j,k=0}^{l,m,n} \boldsymbol{P}_{i,j,k}(W_{i,j,k} + \varepsilon_{i,j,k}) B_{i,p}(u) B_{j,q}(v) B_{k,r}(w)}{\sum_{i,j,k=0}^{l,m,n} (W_{i,j,k} + \varepsilon_{i,j,k}) B_{i,p}(u) B_{j,q}(v) B_{k,r}(w)}$$

自由体变形技术 (FFD)

- 基于权值修改 (Modification of Weights) 的 DFFD：
  - 类似地，我们优化：

$$\sum_{i,j,k=0}^{l,m,n} \left\| \varepsilon_{i,j,k} \right\|^2$$

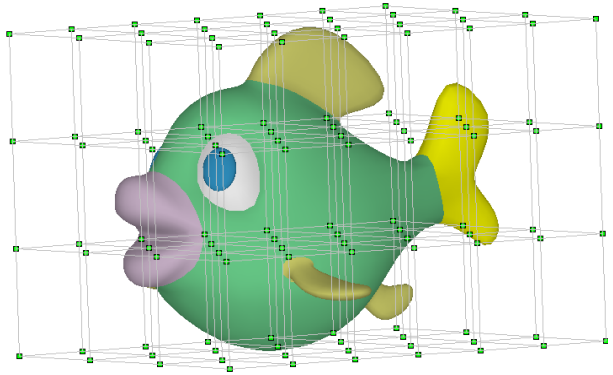
自由体变形技术 (FFD)

- 基于权值修改 (Modification of Weights) 的 DFFD：
  - 使用拉格朗日乘数法，转化为求解方程组：

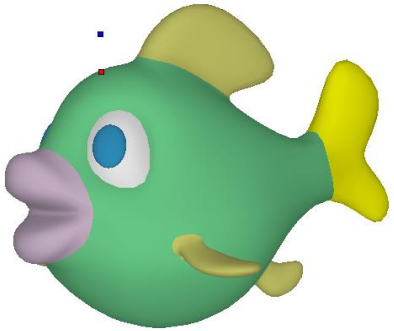
$$\begin{cases} (T-S)\sum\limits_{i,j,k=0}^{l,m,n} W_{i,j,k}B_{i,j,k}(t_s)+\sum\limits_{i,j,k=0}^{l,m,n}(T-P_{i,j,k})\varepsilon_{i,j,k}B_{i,j,k}(t_s)=0 \\ \varepsilon_{i,j,k}=-\dfrac{\lambda}{2}(T-P_{i,j,k})B_{i,j,k}(t_s),0\le i\le l,0\le j\le m,0\le k\le n \end{cases}$$

自由体变形技术 (FFD)

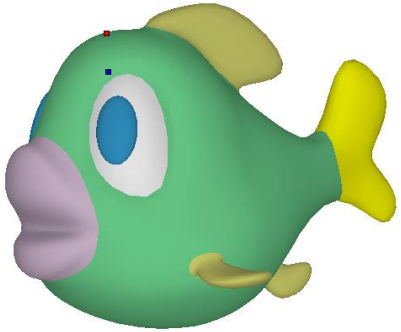
- 基于权值修改 (Modification of Weights) 的 DFFD：



1(a)



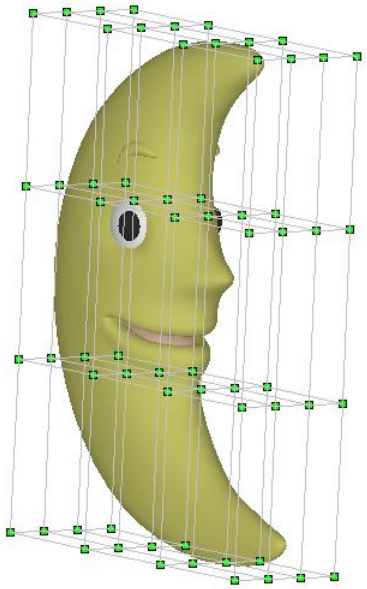
1(b)



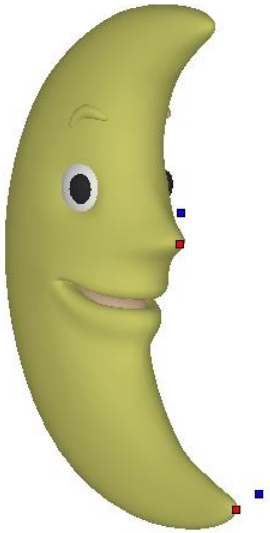
1(c)

自由体变形技术 (FFD)

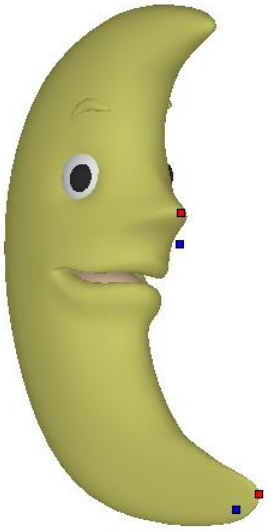
- 基于权值修改 (Modification of Weights) 的 DFFD：



4(a)



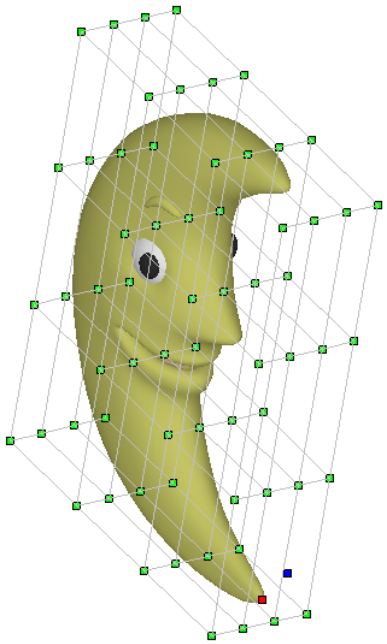
4(b)

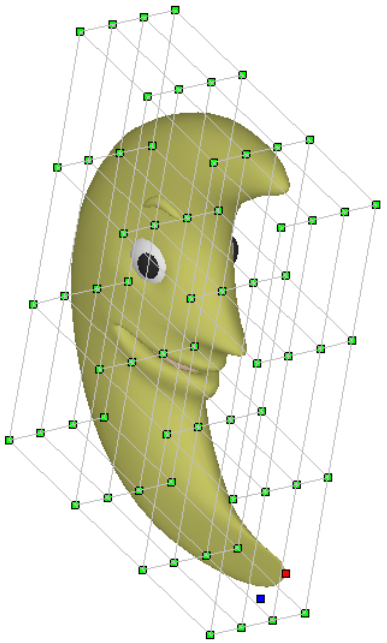


4(c)

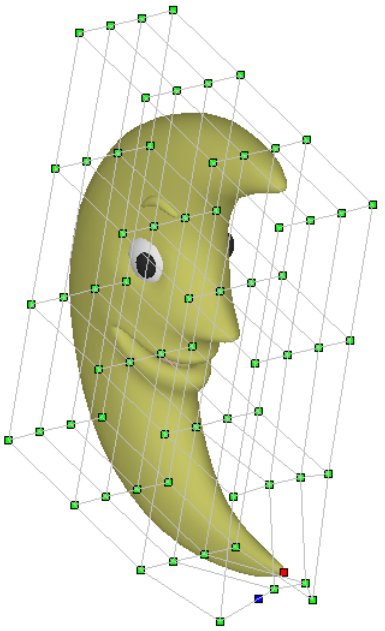
自由体变形技术 (FFD)

- 修改控制点与修改权值的两种 **DFFD** 比较：





Modification of weights



Modification of control points

今日人物： Nelson Max

- **Nelson Max**, UC Davis教授, ACM Fellow
- 论文200+篇，引用8097次
  - 1967年哈佛大学博士，毕业后在Berkeley、 Georgia等大学从事研究，并领导Topology Films Project
  - 计算机动画的早期，制作了一批动画作品，在SIGGRAPH、 Expo 85、90庐山
  - 1972年，CMU助理教授，曾在日本动画界工作多年，90年加盟UC Davis
  - 2007年获Coons奖！2008 PG 主席/2009SIG ASIA主席

谢谢！