

计算机动画学复习

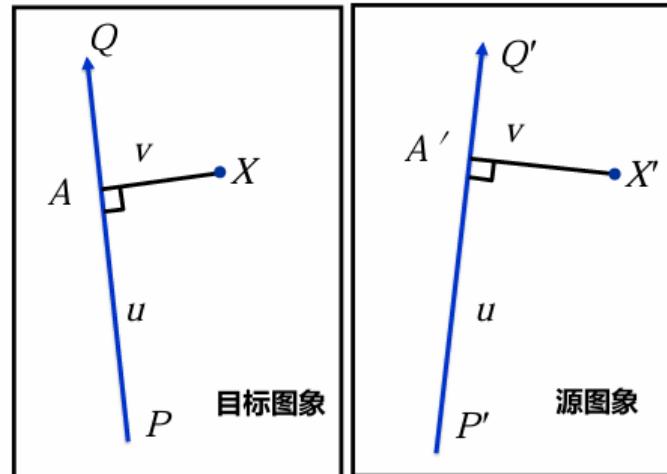
数学知识考点

基于内在形状插值的多边形渐变方法的数学原理

详见下文第四章

图像Morphing中由一对直线确定的变换公式

- 两幅图象之间的变换可以用一对直线来指定。
- 设 I_S 为源图象, I_D 为目标图象。若先在源图象中定义一条有向线段, 再在目标图象中定义一条有向线段, 那么这一对线段定义了一个从一幅图象到另一幅图象的映射, 该映射把一条直线映为另一条直线。
- 设从 $I_S \rightarrow I_D$ 的映射为 $f: X' \rightarrow X$, A 为 X 在 PQ 上的投影, u 为 PA 与 PQ 的比, v 为 XA 的长度, 如右图所示。



单对线对

24

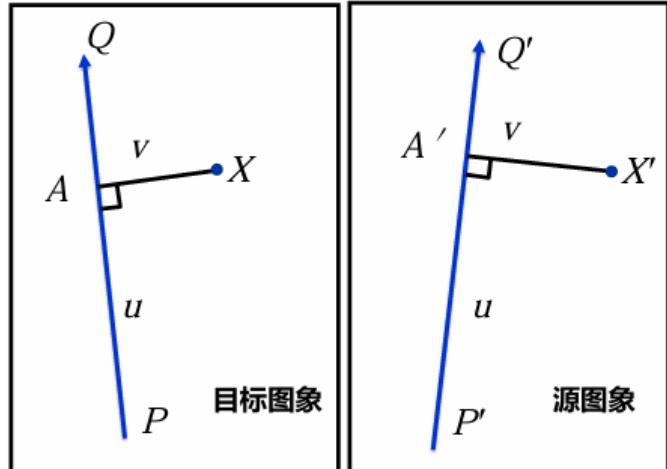
由一对直线确定的变换

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

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

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

其中函数Perpendicular()返回一长度与输入矢量相等且与输入矢量垂直的矢量。



单对线对

Velocity Verlet积分方法

- 1) 速度半步演化 $v(t+\frac{1}{2}h) = v(h) + a(t)\frac{1}{2}h$
- 2) 坐标单步演化 $x(t+h) = x(t) + v(t)h + \frac{1}{2}a(t)h^2 = x(t) + v(t+\frac{1}{2}h)h$
- 3) 通过 $x(t+h)$ 和系统中相互作用条件计算出 $a(t+h)$

- 4) 将半步演化速度同步到单步演化 $v(t+h) = v(t + \frac{1}{2}h) + a(t+h)\frac{1}{2}h = v(h) + a(t)\frac{1}{2}h + a(t+h)\frac{1}{2}h$

第一章

人的五感?

- 视觉、听觉、触觉、味觉、嗅觉

动画形成的视觉原理

- 所谓动画，就是指通过以每秒若干帧的速度顺序地播放静止图像帧以产生运动错觉的艺术。
- 动画利用了人的**视觉残留**这一特点，即上个画面的残留还未消失，下一个画面又进入视觉，这样循环往复，在人的眼中形成动态的画面。
- 动画利用了人的视觉残留特性。连续画面的基本单位为单幅静态画面，在图形学和动画中称为一帧(**Frame**)

计算机动画在影视特技起到什么作用?

- 现实世界中**无法拍摄到的**，如恐龙、变形金刚、阿凡达、
- **成本太高和太危险的**，如龙卷风、核爆炸、地震、大规模人群、

Blender

- Blender是一套跨平台的**开源动画软件**

运动是动画的要素

- 计算机动画中的运动包括
 - 景物位置、方向、大小和形状的变化
 - 虚拟摄像机的运动
 - 景物表面纹理、色彩的变化
- 运动的**表示、捕捉、合成、仿真、模拟、编辑、控制**
- 运动方程的建模 (数学、物理...)
- 运动方程的求解 (如何更快、更精准...)

动画中的“恐怖谷效应”

- 恐怖谷理论(Uncanny Valley)是一个关于人类对机器人和非人类物体感觉的假设，于1969年提出，说明了**当机器人与人类相似程度超过一定程度的时候，人类对他们的反应便会突然变得极其反感，即哪怕机器人与人类有一点点的差别都会显得非常显眼刺目**，从而整个机器人有非常僵硬恐怖的感觉，有如面对行尸走肉。

SORA的动画生成流程和原理

- Sora是一种视频生成模型
- Sora的训练集为**短视频集**，每个样本是一个短视频，同类的短视频构成一个数据流形。Sora将其**编码到隐空间进行降维**，然后在隐空间中将隐特征向量切割成补丁，加上时间顺序，构成**时空词元 (time-space token)**。这里时空的概念是比较关键的，每个词元 (Token) 在短视频的帧序列号 (时间)，在当前帧的行列序号 (空间) 都被记录在**时空词元里**

第二章

传统动画应用于三维计算机动画的12条基本原则及其原理

- Squash and Stretch
 - 在使用“挤压和伸展”原则时，通常使变形的物体保持其**体积不变**。
 - 蕴含的物理原理：影响运动的因素包括质量、外力、材料属性、表面接触的位置等。
- Anticipation
 - 动画中的动作通常包括动作的准备、实际的动作和动作的完成三部分。第一部分就叫做预期性。
- Staging
 - 布局就是以一种容易理解的方式展示动作或对象。
 - 在设置场景时，一个至关重要的因素是**要考虑到观众**。
- Straight-Ahead Action and Pose-to-Pose Action 连贯动作法与关键动作法
 - 前者根据连续的动作依序制作每一帧画面；
 - 后者是先定义关键的主要动作，而后再制作关键动作间的画面(关键帧方法)。
- Follow-Through and Overlapping Action 跟随动作与重叠动作
 - 动者恒动
- Slow In and Slow Out
- Arcs
- Secondary Action
- Timing
- Exaggeration
- Solid drawing
- Appeal
- Depth of Field (衍生原则)
- Balance&Weight (衍生原则)

第三章

旋转的5种表示方法

- Rotation Matrix(旋转矩阵)
 - **orthonormal columns/rows**
 - **bad for interpolation**
- Fixed Angle(定角)
 - **rotate about global axes**
 - **bad for interpolation, has gimbal lock**
 - Ordered triple of rotations about **fixed axes**
 - Any triple can be used that doesn't repeat an axis
- Euler Angle(欧拉角)
 - **rotate about local axes**
 - **same problem as fixed angle (also has gimbal lock)**
 - Euler变换是一种**直观**的使一个物体（或摄像机）朝向一指定方向的有效方法。其来源：瑞士数学家Leonard Euler
 - Ordered triple of rotations about local axes
 - As with fixed angles, any triple can be used that doesn't immediately repeat an axis, e.g., x-y-z, is fine, so is x-y-x. But x-x-z is not.
 - Euler angle ordering is equivalent to **reverse ordering** in fixed angles.
- Axis angle (轴线角)

- **rotate about A by q , (q, Ax,Ay,Az)**
- **good interpolation, no gimbal lock**
- **bad for compounding rotations**
- Euler's rotation theorem
- One orientation can be derived from another by a single rotation about an axis
- So, we can use an axis and a single angle to represent an orientation (with respect to the object's initial orientation)
- Interpolation can be implemented by interpolating axes of rotation and angles separately; but the transformation concatenation cannot be done easily
- Quaternion(四元数)
 - **similar to axis angle but in different form**
 - **$q=(s, x, y, z)$ or $[s, v]$, s is a scalar; v is a vector**
 - **good for interpolation and compounding rotations**
 - A point in space, v , is represented as $[0, v]$
 - To rotate a vector v using quaternion q
 - Represent the vector as $v = [0, v]$
 - Represent the rotation as a quaternion q
 - Using quaternion multiplication

$$\mathbf{v}' = \text{Rot}_q(\mathbf{v}) = q\mathbf{v}q^{-1}$$

- The proof isn't that hard
- Note that the result \mathbf{v}' always has zero scalar value

- If a is unit length, then q will be also
- Rotating some angle around an axis is the same as rotating the negative angle around the negated axis
- Addition $[[s_1, v_1] + [s_2, v_2]] = [s_1 + s_2, v_1 + v_2]$
- Multiplication $[[s_1, v_1][s_2, v_2]] = [s_1s_2 - v_1v_2, s_1v_2 + s_2v_1 + v_1 \times v_2]$
- Multiplication is associative but not commutative (乘法满足结合律, 不满足交换律) $q_1(q_2q_3) = (q_1q_2)q_3$

Gimbal Lock 万向节死锁

- Ordered triple of rotations about fixed axes
 - Any triple can be used that doesn't repeat an axis

欧拉角中的Gimbal lock

- **Gimbal lock现象**: 当一个自由度丧失时。

- 当 $p = \pi/2$ 时, 矩阵只依赖一个角($r+h$)

$$\begin{aligned} E\left(h, \frac{\pi}{2}, r\right) &= \begin{pmatrix} \cos r \cosh - \sin r \sinh & 0 & \cos r \sinh + \sin r \cosh \\ \sin r \cosh + \cos r \sinh & 0 & \sin r \sinh - \cos r \cosh \\ 0 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos(r+h) & 0 & \sin(r+h) \\ \sin(r+h) & 0 & -\cos(r+h) \\ 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

Gimbal Lock(万向节死锁)

- For an orientation $(0, 90, 0)$,
 - A slight change in the first value $(+/-\varepsilon, 90, 0)$
 - A slight change in the 3rd value $(0, 90, +/- \varepsilon)$
 - 90-degree y-axis rotation essentially makes x-axis align with z-axis → **gimbal lock**
 - From $(0, 90, 0)$, the object can no longer be rotated about x-axis by a small change since orientation actually performed is $(90, 90+\varepsilon, 90)$

旋转矩阵到四元数的转换公式

四元数到旋转矩阵的转换

- 对于单位四元数 $q = (q_w, q_x, q_y, q_z)$, 把它转化为对应的旋转矩阵, 可得到:

$$\mathbf{M}^q = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) & 0 \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) & 0 \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

旋转矩阵到四元数的转换

- 由 \mathbf{M}^q 可得到:

$$\begin{aligned} m_{21}^q - m_{12}^q &= 4q_w q_x \\ m_{02}^q - m_{20}^q &= 4q_w q_y \\ m_{10}^q - m_{01}^q &= 4q_w q_z \end{aligned}$$

故若得到 q_w , 则 q_x 、 q_y 、 q_z 便可得。因为:

$$\begin{aligned} \text{tr}(\mathbf{M}^q) &= 4 - 2s(q_x^2 + q_y^2 + q_z^2) \\ &= 4 \left(1 - \frac{q_x^2 + q_y^2 + q_z^2}{q_x^2 + q_y^2 + q_z^2 + q_w^2} \right) \\ &= \frac{4q_w^2}{q_x^2 + q_y^2 + q_z^2 + q_w^2} = \frac{4q_w^2}{n(\mathbf{q})} \end{aligned}$$

- 故单位四元数为

$$q_w = \frac{1}{2} \sqrt{\text{tr}(\mathbf{M}^q)},$$

$$q_x = \frac{m_{21}^q - m_{12}^q}{4q_w},$$

$$q_y = \frac{m_{02}^q - m_{20}^q}{4q_w},$$

$$q_z = \frac{m_{10}^q - m_{01}^q}{4q_w}.$$

Spherical Linear Interpolation (Slerp)公式的证明

Spherical Linear Interpolation (Slerp)

Let $q = \alpha q_1 + \beta q_2$

We can solve for given:

$$\|q\| = 1,$$

$$q_1 \cdot q_2 = \theta,$$

$$q_1 \cdot q = u\theta$$

to give

$$\text{slerp}(q_1, q_2, u) = \frac{\sin((1-u)\theta)}{\sin \theta} q_1 + \frac{\sin(u\theta)}{\sin \theta} q_2$$

第四章

2D Shape Blending涉及哪两个子问题?

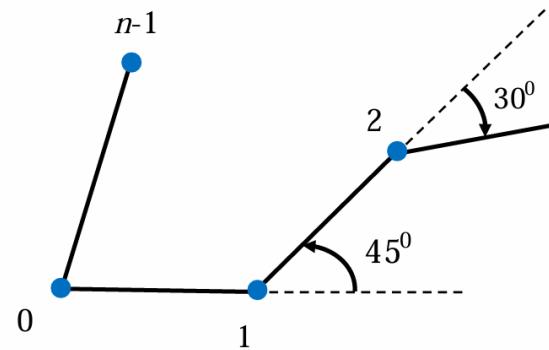
- 该问题实际上分为两个子问题: (1). 顶点的对应关系问题; (2). 顶点的插值问题。
- 在计算机动画中, “Shape blending” (形状混合) 是指在两个或多个不同的几何形状之间进行插值以产生平滑的过渡。

基于内在形状插值的多边形渐变方法的基本思想。Edge Tweaking的思想。

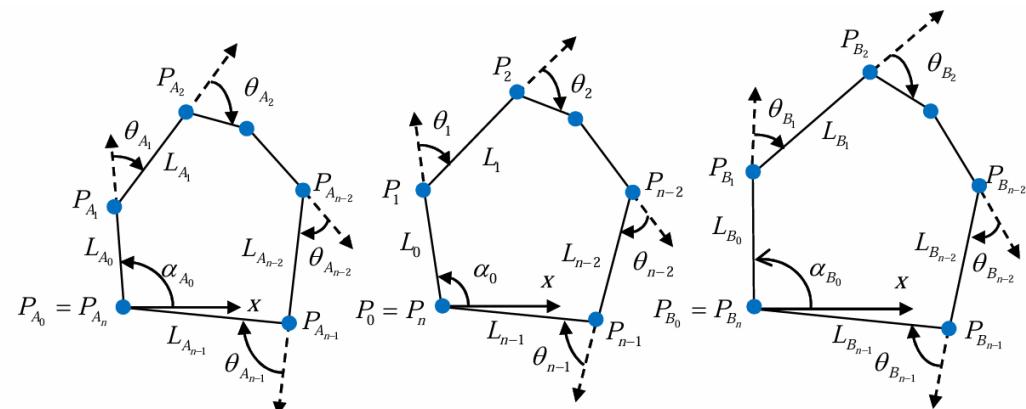
- 线性插值会带来收缩(shrinkage)和纽结现象(Kink),
- Intrinsic Shape Interpolation的数学原理: 乌龟几何 + Lagrange乘数法优化

乌龟几何

- 在笛卡尔坐标系中, 多边形是通过顶点的坐标显式给出的。
- 但多边形也可以通过乌龟几何来定义, 即通过顶点处的边长和有向角来定义。例如, 以某一点为起点, 向东往前走10米, 往左拐 45° , 继续往前走6米, 往右拐 30° , 向前走5米, ..., 最后得到一多边形。
- 因而一个自然的想法是, 能否对关键帧多边形的边长和顶点角进行插值来产生比线性插值更好的效果?
- 回答是肯定的, 因为这种插值具有更好的几何意义。



多边形的内在定义



- 设 $m=n-1$, 逆时针方向的角为正的, P_0 为形状的平移锚点(anchor point), 我们的目标是计算中间多边形的顶点 $P_i (i=1, 2, \dots, m)$, $0 \leq t \leq 1$ 。
- 通过计算多边形 P_A 和 P_B 的边长和有向顶点角, 得到多边形的内在定义 $\{\alpha_0, L_0, (\theta_i, L_i)_{i=1}^m\}$:

$$\begin{aligned} \alpha_{A_0} &= \theta(x, P_{A_0}, P_{A_1}) & \theta_{A_i} &= \theta(P_{A_{i-1}}, P_{A_i}, P_{A_{i+1}}) & L_{A_i} &= |P_{A_{i+1}} - P_{A_i}| & \text{其中: } \theta(P_{A_{i-1}}, P_{A_i}, P_{A_{i+1}}) \\ \alpha_{B_0} &= \theta(x, P_{B_0}, P_{B_1}) & \theta_{B_i} &= \theta(P_{B_{i-1}}, P_{B_i}, P_{B_{i+1}}) & L_{B_i} &= |P_{B_{i+1}} - P_{B_i}| & \text{表示边 } P_{A_{i-1}}P_{A_i} \text{ 和边 } \\ & & & & & & P_{A_i}P_{A_{i+1}} \text{ 之间的有向夹角} \end{aligned}$$

中间多边形生成

- 一个解决方法为保持插值的顶点角不变，适当调整插值得到的边长(Edge Tweaking)：

$$L_i = (1-t)L_{A_i} + tL_{B_i} + S_i, \quad (i=0,1,2,\dots,m)$$

- 如果关键多边形的某一条对应的边具有相同的长度 L ，那么我们可认为中间多边形的边长也为 L ，因此可认为 $|S_i| \propto |L_{A_i} - L_{B_i}|$ 。为了防止除以零，我们定义：

$$L_{AB_i} = \max \left\{ |L_{A_i} - L_{B_i}|, L_{tol} \right\}, \quad (i=0,1,2,\dots,m)$$

其中： $L_{tol} = 0.0001 \times (\max_{i \in [0,m]} |L_{A_i} - L_{B_i}|)$

中间多边形生成

- 我们的目标是为了求得 S_0, S_1, \dots, S_m ，使目标函数：

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

最小，并且 S_0, S_1, \dots, S_m 应满足约束条件(强迫多边形封闭)：

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

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

其中 α_i 是由矢量 $P_i P_{i+1}$ 和x轴构成的有向角： $\alpha_i = \alpha_{i-1} + \theta_i$, ($i=1, 2, \dots, m$)

中间多边形生成

- 这是个具有约束条件的极值问题，可用Lagrange乘数法来求解。引进Lagrange函数：

$$\Phi(\lambda_1, \lambda_2, S_1, S_2, \dots, S_m) = f + \lambda_1 \varphi_1 + \lambda_2 \varphi_2$$

其中 λ_1, λ_2 为Lagrange乘数。对 Φ 求偏导得：

$$\begin{aligned} \frac{\partial \Phi}{\partial S_i} &= \frac{2S_i}{L_{AB_i}^2} + \lambda_1 \cos \alpha_i + \lambda_2 \sin \alpha_i = 0 \quad (i=0,1,\dots,m) \\ \sum_{i=0}^m [(1-t)L_{A_i} + tL_{B_i} + S_i] \cos \alpha_i &= 0 \\ \sum_{i=0}^m [(1-t)L_{A_i} + tL_{B_i} + S_i] \sin \alpha_i &= 0 \end{aligned}$$

アラシナルエクス

- 如果 $EG - F^2 \neq 0$ ，则

$$\begin{aligned} \lambda_1 &= \begin{vmatrix} U & F \\ V & G \end{vmatrix} / \begin{vmatrix} E & F \\ F & G \end{vmatrix} \\ \lambda_2 &= \begin{vmatrix} E & U \\ F & V \end{vmatrix} / \begin{vmatrix} E & F \\ F & G \end{vmatrix} \end{aligned}$$

$$S_i = -\frac{1}{2} L_{AB_i}^2 (\lambda_1 \cos \alpha_i + \lambda_2 \sin \alpha_i), \quad (i=0,1,2,\dots,m)$$

因而可得到中间多边形顶点 (x_i, y_i) 的坐标：

$$\begin{cases} x_i = x_{i-1} + L_{i-1} \cos \alpha_{i-1} \\ y_i = y_{i-1} + L_{i-1} \sin \alpha_{i-1} \end{cases}$$

20

- 总结：

输入：两个关键帧多边形的顶点

- 计算两个关键帧多边形的内在表示；
- 解线性方程组，计算 λ_1, λ_2 ；
- 计算 S_i ；
- 中间多边形顶点的坐标；

输出：中间帧多边形的顶点

优点：

- 简单；

- 能在一定程度上避免形状插值中出现的收缩和扭结现象，这在二维角色动画中尤其有用。

缺点：

- 顶点对应关系需要用户交互给出；
- 结果依赖于顶点对应关系的好坏；
- 当源和目标多边形中包含短边时，由于计算短边的方向不稳定，中间帧多边形有可能产生较严重的畸变。

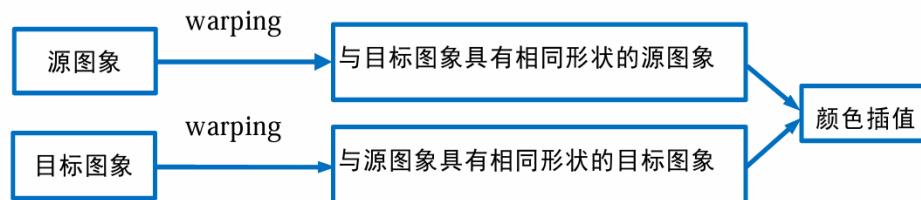
图像morphing的原理

图像morphing的过程

- 设 P_0 为源图象上的点， P_1 为目标图象上的点，则源图象和目标图象的warping函数 W_0 和 W_1 分别定义为：

$$\begin{aligned} W_0(P_0, t) &= (1-t)P_0 + tC_0(P_0) & t \in [0,1] \\ W_1(P_1, t) &= (1-t)C_1(P_1) + tP_1 \end{aligned}$$

- 当两幅图象变形对齐后，我们可采用简单的颜色插值得到中间帧图象。



9

基于网格的图象morphing 原理

基于网格的图象morphing方法

- 中间帧图象可以下面的四个步骤来生成：

- 线性插值网格 M_s 和 M_d ，得到网格 M 。
- 应用由网格 M_s 和 M_d 定义的变换，使源图象 I_s 扭曲变形到 I_0 。

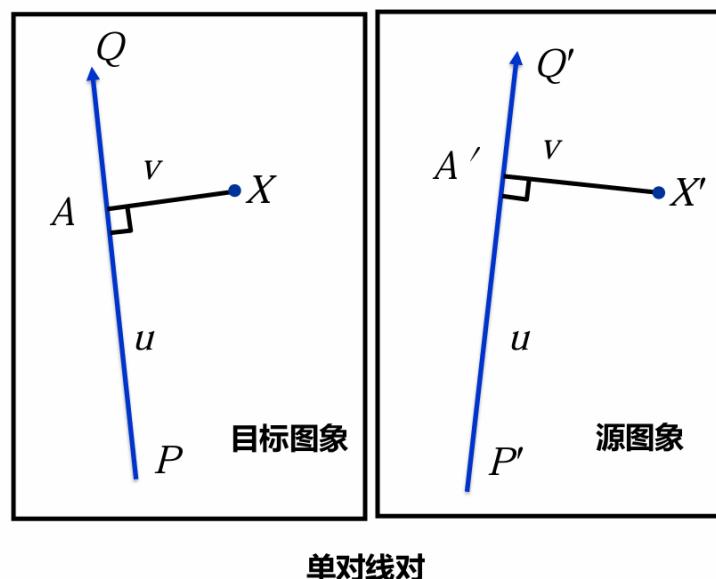
假设 M_s 对应的曲面为 $P^s(u, v) = \sum \sum P_{ij}^s B_i(u) B_j(v)$ ， M_d 对应的曲面为 $P(u, v) = \sum \sum P_{ij} B_i(u) B_j(v)$ ，则图象 I_0 的计算过程为：对于 I_0 上的每一像素 \tilde{P} ，由 $\sum \sum P_{ij} B_i(u) B_j(v) = \tilde{P}$ 计算出满足该公式的参数 (\tilde{u}, \tilde{v}) ，则 I_0 上 \tilde{P} 点的颜色值为图象 I_s 上点 $P^s(\tilde{u}, \tilde{v})$ 的颜色值(或双线性插值)。

- 应用由网格 M_d 和 M 定义的变换，使目标图象 I_d 变形到 I_1 。
- 对图象 I_0 和 I_1 进行线性插值，得到中间帧图象。

基于线对的图象morphing方法的原理

由一对直线确定的变换

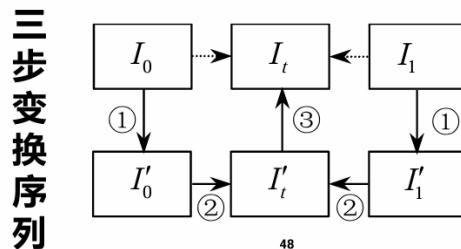
- 两幅图象之间的变换可以用一对直线来指定。
- 设 I_S 为源图象, I_D 为目标图象。若先在源图象中定义一条有向直线段, 再在目标图象中定义一条有向线段, 那么这一对直线段定义了一个从一幅图象到另一幅图象的映射, 该映射把一条直线映为另一条直线。
- 设从 $I_S \rightarrow I_D$ 的映射为 $f: X' \rightarrow X$, A 为 X 在 PQ 上的投影, u 为 PA 与 PQ 的比, v 为 XA 的长度, 如右图所示。



视域morphing的三个步骤

视域morphing的三个步骤

- 给定中间帧的投影矩阵 $\Pi_t = [\mathbf{H}_t | -\mathbf{H}_t \mathbf{C}_t]$, 对应于投影矩阵 Π_t 的投影图象 I_t 可用下面的三步算法来生成:
 - 前置变形(Prewarp)。对图象 I_0 应用投影变换 \mathbf{H}_0^{-1} , 对图象 I_1 应用投影变换 \mathbf{H}_1^{-1} , 生成前置变形后的图象 I'_0 和 I'_1 。
 - Morphing。采用平行摄像机的方法, 线性插值图象 I'_0 和 I'_1 相应点的位置和颜色, 得到图象 I'_t 。
 - 后置变形(Postwarp)。对图象 I'_t 应用变换 \mathbf{H}_t , 生成图象 I_t 。

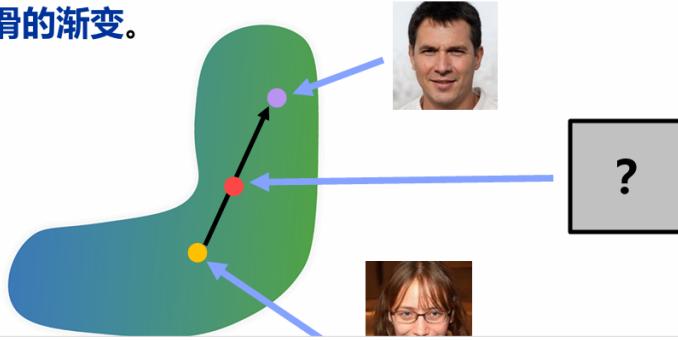


48

基于StyleGAN的肖像Morphing的原理

StyleGAN中的图像Morphing

- StyleGAN倾向于学习一个**平滑的**隐空间，即**隐空间的接近区域中采样的隐码能生成相似的图像**。这表明，在隐空间中的连续移动会产生一条**平滑变化的图像路径**，并且每个图像都接近目标域（真实的人脸）。
- 因此，对StyleGAN的两个隐码进行插值，可以在两个人脸图像之间产生非常**自然平滑的渐变**。



StyleGan 中的Coarse styles、Middle styles、Fine styles分别对应人脸的那些语义特征？

StyleGAN生成器中的风格控制

- 而在**StyleGAN**的图像合成网络中，输入不同**Synthesis Layer**的style控制不同粒度的语义特征，即不同粒度的风格。
- Coarse styles**: 控制人脸姿态、人脸几何形状等。**大约对应网络的1-4层**。
- Middle styles**: 发型、部分人脸形状等。**大约对应网络的5-8层**。
- Fine styles**: 肤色、发色、光影、背景等。**大约对应网络剩下的层数**。
- 因此，通过**混合不同粒度**的style，能产生不同层级的**风格混合**效果。

第五章

二、三维形状渐变(morphing)各有什么优缺点？

- 二维：
- 优点：是一种达到特殊视觉效果的有效方法；可以让人产生神奇的三维形状改变的错觉，可以避免复杂的三维造型过程。缺点：可能生成一些意料之外的图像（线对特征法）；缺乏三维几何信息，无法像其它三维物体一样进行几何变换、改变材质属性、改变光照、阴影等，摄像机的动画会受到很大的限制。
- 三维：

- 优点：能生成更逼真和生动的特技效果，且得到的中间帧是物体的模型而不是图像，三维morphing得到的中间帧是物体的模型而不是图象，所以三维morphing的结果与视点和光照参数无关，并能够生成精确的光照和阴影效果，运用范围更广。缺点：物体之间的对应关系很难建立，而且该方法对物体的几何表示也相当苛刻

三维morphing技术

- 虽然二维morphing技术在影视特技、广告等行业中取得了广泛的应用，但二维图象对应的物体存在**没有三维几何信息**这一很严重的局限性，故它不能象其他三维物体一样进行几何变换，从而使摄像机的动画受到了很大的限制。
- 尽管三维morphing比二维图象morphing要复杂得多，但由于能生成更逼真和生动的特技效果，所以它还是吸引了许多研究者。
- 与二维图象morphing相比，三维morphing得到的中间帧是**物体的模型**而不是图象，所以三维morphing的结果与视点和光照参数无关，并能够生成精确的光照和阴影效果。在三维morphing中，一旦得到中间帧物体序列，就可以用不同的摄像机角度和光照条件来对它们进行重新绘制，也可以把它们与其它的三维场景相结合进行绘制。

6

基于星形物体的多面体morphing方法

- 合并一对三维多面体物体模型的拓扑结构，使得它们具有相同的顶点/边/面网络结构，然后对相应的顶点进行插值。
- 通过合并拓扑结构将一个物体的形状变换为另外一个物体的形状通常可分为以下两步：
 - 建立源物体表面上的点和目标物体上的点对应关系（对应问题）
 - 插值对应的点（插值问题）
- 对于亏格为零的两个多面体，它们都同构于球。把它们都投影到单位球面上，然后将投影在单位球面上的两个拓扑结构合并在一起构成一个新的拓扑结构，再将新的拓扑结构映射回原来的两个多面体，即建立好了相应的点对关系。
- 插值相应的点点通常采用线性插值或者Hermite插值

基于体表示的三维morphing方法

- 思想：给定源物体S和目标物体T，首先根据指定的对应特征生成一个空间变换，该变换使给定的两个体扭曲变形（warp）成S'和T'，达到几何对齐的目的；然后对得到的两个扭曲变形体S'和T'进行混合（对两个体素的颜色和不透明度进行交溶处理，然后对混合后的体素进行绘制）。
- 混合方法：对两个体素的颜色和不透明度进行交溶处理，然后对混合后的体素进行绘制。
- 优点：与物体的几何和拓扑结构无关；提供了一种统一的处理方法，具有一般性（几何表示的物体都可以转化成体表示）。

- 缺点：走样现象严重，精度没有基于几何表示的好；几何模型转换为体素表示的计算时间较费。

几何对齐

- 在基于体表示的三维morphing中，首先要使给定的两个体扭曲变形，达到几何对齐。
- 为了实现几何对齐，动画师在源体和目标体中指定特征元素对(e_s, e_t)，特征元素包括点、线、长方形和长方体。
- 为了便于在三维空间指定特征，Lerios把特征的位置和朝向编码为一个包含四个向量的特征局部坐标系，这四个向量包括一个位置向量 c 和三个定义坐标轴方向的正交单位向量 x, y, z 。
- 另外，特征元素还包括比例缩放因子属性 s_x, s_y, s_z ，它们定义了特征沿每个轴的伸展程度。
- 在morphing过程中，这些特征彼此变换到对方。这些变换把S中的特征平移、旋转、和比例缩放后，分别与T中对应特征的位置、朝向和大小相匹配。

29

基于变分隐式曲面的三维Morphing

- 建立源物体和目标物体所对应的隐式曲面: $f(x,y,z)=0$;
 $g(x,y,z)=0$;
- 2). 插值这两个函数:
 $(1-t)^* f(x,y,z) + t^* g(x,y,z) = 0; 0 <= t <= 1$;

优点：即使对于不同拓扑的三维物体，该方法仍能生成平滑自然的中间帧结果。

变分插值(Variational Interpolation)

- 采用散乱数据插值(scattered data interpolation)来解决形状的变换问题。
- 散乱数据插值：给定一系列数据点，构建一个光滑函数，通过给定的这些点。
- 2D散乱数据插值问题：给定 k 个约束点 $\{c_1, c_2, \dots, c_k\}$ ，它们为平面上的散乱点，并给定在这些点的标量值 $\{h_1, h_2, \dots, h_k\}$ ，构建一个光滑曲面函数 $f(\mathbf{x})$ ，使得曲面在给定约束点的高度值为给定的值，即 $f(c_i) = h_i, i=1,2,3,\dots,k$ 。
- 上述问题可通过变分法进行求解。通过构建度量插值函数质量的能量函数，在插值给定点的同时，使得能量函数极小。在二维情形，当采用如下能量函数 E 时： $E = \int_{\Omega} f_{xx}^2(\mathbf{x}) + 2f_{xy}^2(\mathbf{x}) + f_{yy}^2(\mathbf{x})$ ，薄板插值即为其变分解。

粒子系统的基本原理、粒子系统可以生成哪些自然现象？

- 粒子系统即模拟不规则自然景物生成和动画的系统，使造型和动画巧妙地连成一体。景物被定义为由成千上万个不规则的，随机分布的粒子所组成，而每个粒子均有一定的生命周期，它们不断改变形状，不断运动。

粒子系统基于的假设：

- 粒子一般不与其它粒子碰撞
- 除非处于聚集状态，粒子不向其它粒子投影
- 粒子只向其它环境投影
- 粒子不反射光
- 粒子通常有有限的生命周期

粒子系统的基本思想：

- 将许多简单形状的微小粒子作为基本元素聚集起来形成一个不规则模糊物体。每个粒子均经历出生，成长，衰老和死亡的过程，与粒子有关的每一参数均将受到一个随机过程的控制。

生成一帧的基本步骤：

- 生成新的粒子并加入系统中；
- 赋予每一新粒子以一定的属性；
- 删除那些已经超过其生命周期的粒子；
- 根据粒子的动态属性对粒子进行移动和变换；
- 绘制并显示由有生命的粒子组成的图形。

粒子的生成：

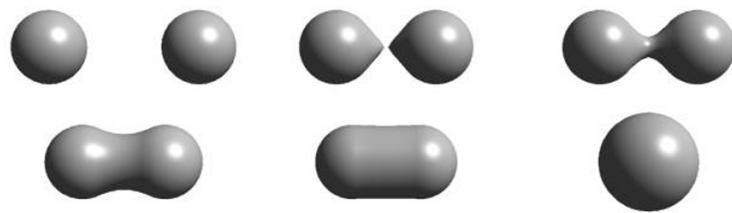
- 对于每一帧，根据一个控制的随机过程生成粒子：
 - 用户可以控制每帧的平均粒子数和其概率分布；
 - 粒子数可以是时间的函数。

粒子的属性：

- 位置，速度，大小，周期，质量，力加速器，生命周期，绘制属性.....

元球的融合原理

粒子绘制——把粒子作为元球



- 元球相互靠近到一定距离产生变形，再进一步靠近时则融合成光滑表面。
- 以两个元球为例，元球靠近时的变形过程如上图所示。最初是两个独立的球，彼此接近时，相对的面开始隆起变形，接近到一定程度就会象水滴（水银）一样融合成一个面。然后变成花生形状、胶囊形状、最后变成一个球。
- 上述过程实际上提供了一种模拟两滴水（水银）融合的动画过程，而用参数曲面和多边形网格造型方法来模拟此类动画过程是很困难的。
- 隐式曲面很适合表示可变形和可变拓扑的物体，因而对动画非常有用(如morphing)。

Xiaogang Jin

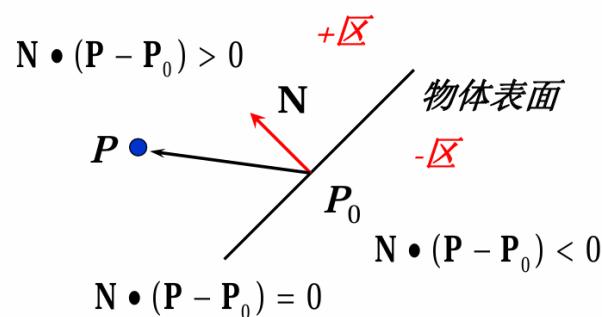
36

10/5/2024

粒子的碰撞检测和碰撞响应

粒子的碰撞检测

- 判断一个粒子是否与物体发生碰撞



- 当且仅当 $N \cdot (P - P_0) < 0$ 时，粒子与平面发生碰撞

Xiaogang Jin

42

10/5/2024

碰撞响应(Collision Response)

- 当一个粒子已经发生碰撞时，该做什么？
- 正确的做法是：把模拟**退到接触点**(the point of contact))
- 只需修改粒子的**位置**和**速度**



为什么需要群组动画？

- 得到大场面的震撼效果
- 减少动画师的工作量
- 节约成本

Flocking(群体) System的包含哪三种层次的行为？

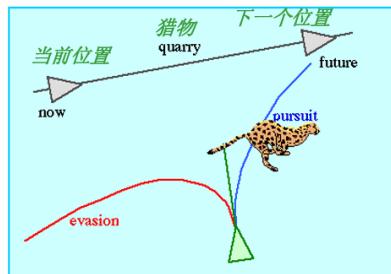
- Action Selection: 基于人工智能的 行为选择
- Steering: 抽象化的导航行为
- Locomotion: 实体化的个体行为

Boids模型的三条原则，每条原则的含义

- 类似鸟集群运动模型（空中）
- Boid是一个模拟的类似于鸟一样的物体。粒子系统中的个体没有智能。而群组中个体是智能体(Agent)。
- 优先级递减的群体模拟三大原则：
 - 碰撞避免原则(Collision Avoidance): 避免与相邻的群体成员相碰
 - 速度匹配原则(Velocity Matching): 尽量保持与周围邻居群体成员的速度匹配
 - 群体合群原则(Flock Centering): 群体成员尽量靠近。Boid局部感知的群体中心，实际上是相邻群体成员的中心，使得boid飞向与相邻boids的中心

Reynolds导航方法中的追逐和躲避模型、障碍避免模型、路径跟随、流场跟随行为模型、领导模型

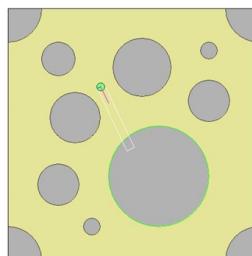
追逐和躲避(Pursuit and Evasion)



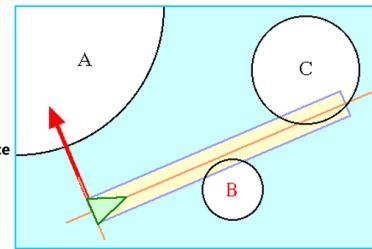
- 追逐(Pursuit)与寻找(Seek)非常类似，其区别在于目标为一移动的角色(猎物)。
 - 假设猎物在预测区间T内不会转向。
 - 猎物在将来的位置可通过把它的当前速度乘以T，并把该偏移量与当前位置相加来得到。
- <http://www.red3d.com/cwr/steer/PursueEvade.html>

障碍避免(Obstacle Avoidance)

Obstacle Avoidance steering behavior



anticipatory collision avoidance



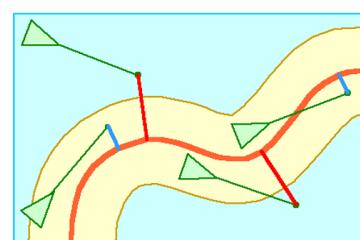
- 该行为的目标是：在该角色前面，保证有一个假想圆柱的自由空间（实际上是预期性碰撞避免）。
- <http://www.red3d.com/cwr/steer/Obstacle.html>

- 障碍避免返回的值：
 - 返回避免最有威胁障碍物的导航值；
 - 如果没有碰撞是迫在眉睫的，返回一个特殊值（空值，或零向量），表示在这一时刻不需要纠正量。

路径跟随(Path following)

- <http://www.red3d.com/cwr/steer/PathFollow.html>

- 沿着一条路径移动角色，并同时保持在脊柱线的指定半径内。
- 投影距离：**小于路径半径时，不需要进行导航校正。
- 否则，把预测的位置投影到路径上，把该点作为目标点，并进行寻找(Seeking)行为。



投影距离(Projection distance)：
从预测位置到最近路径点的距离

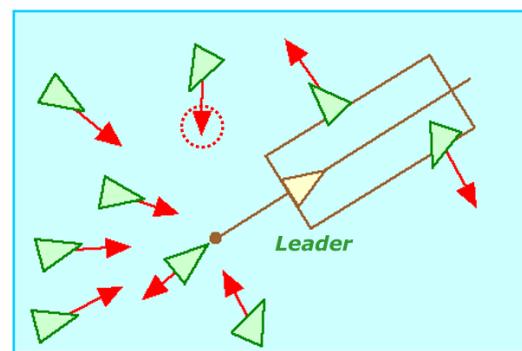
66 / 78

流场跟随行为(Flow Field Following)

- <http://www.red3d.com/cwr/steer/FlowFollow.html>
- 假设运动区域已经有一个速度流场
- 估算角色将来的位置并计算在这点的流场
- 得到的速度(F)即为我们期望的速度，导航方向(S)为期望速度和当前速度的差

跟随领导(Leader Following)

- 如果一个跟随成员发现自己处于领导前面的一个矩形区域，它会横向远离领导者的路径；
- 否则，到达(arrival)目标为领导后面的一个偏移点；
- 跟随成员采用分离行为来避免相互拥挤；



<http://www.red3d.com/cwr/steer/LeaderFollow.html>

第九章

Helbing基于社会力模型的群体行为模拟方法的基本原理

- 社会力模型以牛顿动力学为基础，由各个力的表达式来体现行人不同的动机和影响。在社会力模型中，由于对影响个体的因素考虑得比较全面，对个体行为的建模比较合理，该模型可以逼真地模拟人群的疏散过程。
- 个体的实际行为受**主观意识、其他个体及障碍物**三方面因素的影响，均可等效为力在个体上的作用

羊群效应的原理

- 当处于恐慌状态时，人的行为动作会变得笨拙和危险
- 恐慌的人群呈现羊群效应、个体行为或两者的混合；

羊群效应

- 恐慌的人群呈现羊群效应、个体行为或两者的混合；

- 羊群效应用参数 p_i 来模拟；

$$\mathbf{e}_i^0(t) = \text{Norm} \left[(1 - p_i) \mathbf{e}_i + p_i \langle \mathbf{e}_j^0(t) \rangle_i \right]$$

↓ ↓
个体行为的方向 个体邻居行为的平均

相互速度障碍物(RVO)原理

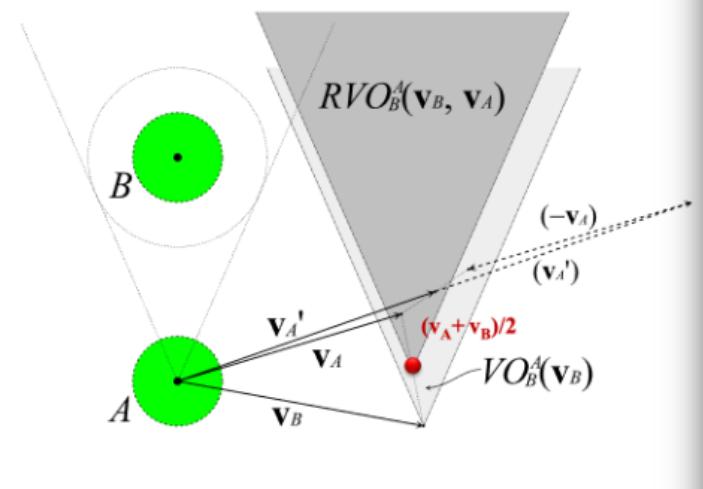
- 假设个体以匀速前进。每个个体在保持与周围个体相对运动的同时，在速度域中计算出可能导致碰撞的速度集合，并对自身速度进行必要的调整。在调整过程中，碰撞避免的任务同时分配给相关的个体，使它们相互协调完成碰撞避免任务。

相互速度障碍物

- B 对 A 的相互速度障碍物包含 A 的所有以下速度的集合：该集合的成员为 A 的当前速度 \mathbf{v}_A 和 B 的速度障碍物内的一个速度的平均：

$$RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) = \left\{ \mathbf{v}_A' \mid 2\mathbf{v}_A' - \mathbf{v}_A \in VO_B^A(\mathbf{v}_B) \right\}$$

- 几何上，可解释为把 B 的速度障碍物 $VO_B^A(\mathbf{v}_B)$ 平移到顶点 $(\mathbf{v}_A + \mathbf{v}_B)/2$ ；



连续人群(Continuum Crowds)的原理

- Treuille等人提出了基于连续体动力学的实时人群模型。在该模型中，**动态势能场**同时将全局导航与移动障碍物（例如其他人）整合在一起，**无需明确碰撞避免**即可有效地解决大规模人群的运动。仿真可以以**交互速率**运行，在各种条件下平稳流动，并自然展现了在真实人群中观察到的紧急现象
- 最优路径计算

- 密度和速度计算
- 控制方程(The Governing Equations)
 - 最大速度场(Maximum Speed Field)
 - 不舒服场(Discomfort Field)
 - 单位代价场(Unit Cost Field)
- 离散网格结构 (Discretized grid structure)
 - 密度转换 (Density conversion)
 - 单位代价计算 (Unit cost computation)
 - 动态势能场构建(Dynamic Potential Field Construction)

正向运动学、逆向运动学原理(IK), 逆向雅克比方法求解IK的原理

- 正向运动学(FK):
 - 动画师通过**直接指定关节处的关节运动参数**来控制物体的运动;
 - 从关节空间映射到笛卡尔空间;
 - 计算整棵树 (深度优先遍历)。
 - 优点: 精确、确定性、实时。运算简单, 计算速度较快。
 - 缺点: 难以控制末端执行器的位置, 不能直接控制末端执行器在工作空间中的位置和方向。
- 逆向运动学(IK):
 - 动画师**指定目标位置**, 系统求解满足要求的关节角;
 - 从笛卡尔空间映射到关节空间;
 - 给定初始姿态向量和目标姿态向量, **计算关节向量的值**, 使得物体满足所需的姿势;
 - 一旦得到关节向量值后, 可以对角色的初始姿态和最终姿态的关节向量值进行插值, 从而得到角色动画。
 - 优点: 直观的末端执行器位置控制, 允许机器人在空间中执行特定的任务, 适用于处理复杂的环境。
 - 缺点: 逆向运动学问题通常较复杂, 需要数值方法或迭代算法来解决。可能存在多个解, 或者无解情况, 需要进行解决方案选择。另外还可能需要约束。
- 逆向雅克比方法求解IK原理:
 - 通过把雅克比矩阵求逆, 把该问题在当前位置局部线性化; 把笛卡尔空间的速度映射到关节空间的速度; 给定初始姿势和所需要的姿势, 迭代变化关节角, 使得末端影响器朝目标位置和方向前进。

逆向雅克比方法

$$X(t) = f(\theta(t)) \quad X \in R^n \text{ (通常 } n = 6) \\ \theta \in R^m (m = \text{自由度})$$

- 雅克比矩阵为 $n \times m$ 的矩阵，它把 θ 的微分 ($d\theta$) 与 X 的微分相关联 (dX)

$$\frac{dX}{dt} = J(\theta) \frac{d\theta}{dt} \quad \text{其中 } J_{ij} = \frac{\partial f_i}{\partial \theta_j}$$

- 所以雅克比矩阵把 **关节空间的速度** 映射到 **笛卡尔空间的速度**

$$V = J(\theta) \dot{\theta}$$

逆向雅克比方法

- 逆向雅克比问题为： $\theta = f^{-1}(X)$
 - f 是一个高度非线性函数
- 通过把雅克比矩阵求逆，把该问题在当前位置局部线性化

$$\dot{\theta} = J^{-1} V$$

- 通过一系列增量步骤，迭代到所需要的位置

逆向雅克比方法

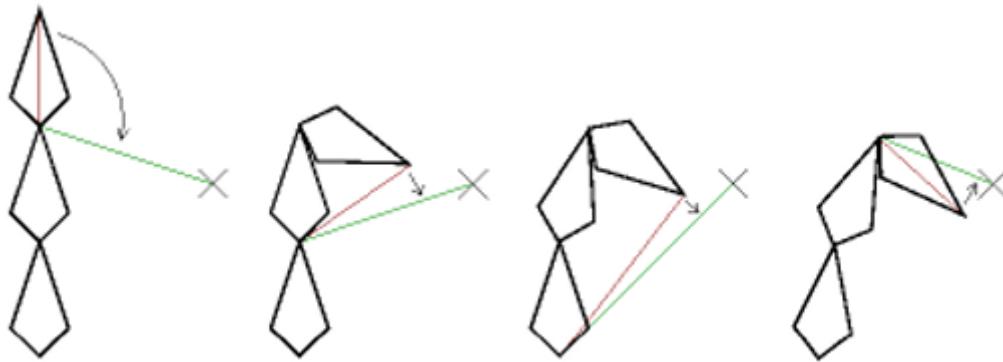
- 给定初始姿势和所需要的姿势，**迭代**变化关节角，使得末端影响器朝目标位置和方向前进
 - 对于中间帧，插值得到所需的姿态向量
 - 对于每一步 k ，通过上述公式得到关节的角速度 $\dot{\theta}$ ，然后执行

$$\theta_{k+1} = \theta_k + \Delta t \dot{\theta}$$

循环坐标下降法(CCD)原理。

- 循环坐标下降法(CCD)原理：对所有受IK影响的骨骼，按从最远侧子骨骼到父骨骼的顺序执行迭代操作：旋转当前骨骼，使当前骨骼位置到目标骨骼的连线指向IK目标位置。

- 由于所有骨骼是从一个特定状态出发开始IK计算，所得到的结果也会比较稳定。通常5~10次迭代之后就能得到很好的结果。



FK vS. IK优缺点

- 见上文

关机动画中的关节空间 vs. 笛卡尔空间数 数值求解|包括哪些方法?

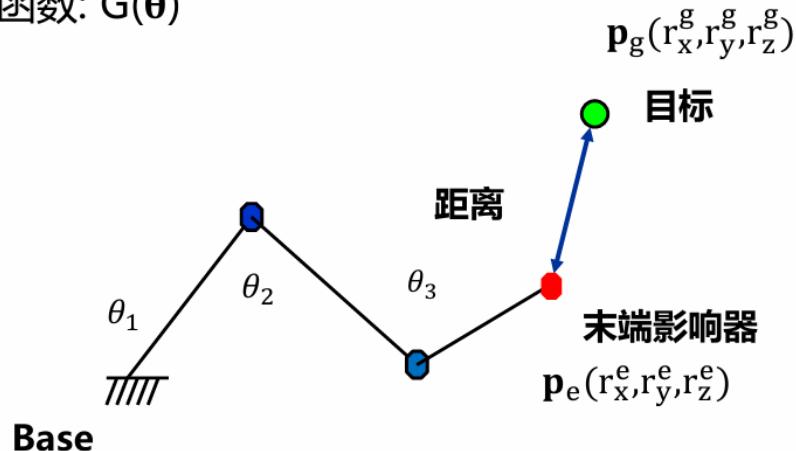
- 逆向雅克比方法 (Inverse-Jacobian Method):
 - 使用数值迭代，不断微调关节角度，以使末端执行器达到目标位置和方向。
 - 适用于各种机器人结构，但可能需要较多计算资源。
- 循环坐标下降法 (CCD):
 - 逐一处理每个关节，每次微调一个关节的角度，以逼近目标位置和方向。
 - 通常用于串联机器人，但结果可能受处理顺序影响。
- 基于优化的方法 (Optimization-based Method):
 - 视问题为数学优化，通过最小化误差函数找到最佳关节角度。
 - 适用于各种机器人结构和复杂任务，但可能需要更多计算资源。
- 基于样例的方法 (Example-based Method):
 - 利用机器学习和训练数据，建立从末端执行器位置到关节角度的映射。
 - 适应多变任务，但需要大量训练数据和计算资源

基于优化的|计算方法和目标函数

- 把IK转化为一非线性优化问题
 - 例如 $\text{minimize } x^2(y + 1) + \sin(x + y); \text{ subject to } x \geq 0, y \geq 0$
- 目标函数(Objective function)
- 约束(Constraint)
- 迭代算法(Iterative algorithm)

目标函数(Objective Function)

- 末端影响器到目标位置/方向的“距离”
- 关节角的函数: $G(\theta)$



15

目标函数

位置目标函数

$$\|p_g - p_e\|^2$$

方向目标函数(方向由一对正交向量来定义)

$$\|r_x^g - r_x^e\|^2 + \|r_y^g - r_y^e\|^2$$

总目标函数为位置/方向目标的加权和:

$$G(\theta) = \omega \|p_g - p_e\|^2 + (1 - \omega)(\|r_x^g - r_x^e\|^2 + \|r_y^g - r_y^e\|^2)$$

16

非线性优化

- 约束的非线性优化问题：

$$\begin{cases} \text{minimize} & G(\boldsymbol{\theta}) \\ \text{subject to} & \begin{cases} \mathbf{a}^T \boldsymbol{\theta} = \mathbf{b}_1 & \text{limb coordination} \\ \mathbf{a}^T \boldsymbol{\theta} \leq \mathbf{b}_2 & \text{joint limits} \end{cases} \end{cases}$$

- 求解

- 标准的数值方法
- MATLAB或其它优化软件包
 - 求得的通常是局部极小
 - 依赖于初始条件

17

基于深度学习的姿态恢复的优缺点

基于深度学习的姿态恢复

优 点

- **较高的精度：**深度学习模型能够从大量数据中学习复杂的人体姿态和运动模式，提供较高精度的姿态恢复。
- **自动化和实时处理：**可以自动识别和跟踪人体关节，支持实时姿态分析，非常适合需要快速反馈的应用场景。
- **适应性强：**模型能够适应不同的环境和个体差异，处理多样化的姿态数据。
- **多应用场景：**从运动科学到动画制作，再到医疗康复，基于深度学习的姿态恢复技术在多个领域都有广泛应用。

缺 点

- **数据依赖性：**模型的性能极大地依赖于训练数据的质量和多样性。不足或偏差的数据会导致模型表现不佳。
- **计算资源消耗：**深度学习模型通常需要大量的计算资源，这在有限资源的环境下可能是一个限制。
- **可解释性差：**相比传统算法，深度学习模型的决策过程更加复杂和不透明，这在某些应用场景（如医疗诊断）中可能成为问题。
- **过度拟合的风险：**模型可能过度拟合特定的训练数据，导致其在面对新的、未见过的姿态数据时表现不佳。

基于运动捕获的关节动画制作的优缺点？

- 优点：只要能被捕获，可以记录人体运动的所有细节，运动真实
- 缺点：不容易进行编辑和控制，较昂贵

骨架与角色模型的绑定原理，顶点混合(VertexBlending)的原理。顶点混合的数学表示。

- 骨架与角色模型的绑定原理
 - 角色的表面（外皮）必须随着骨架的运动而运动（变形）；
 - 在骨架绑定中，皮肤的运动定义为对应控制骨架的函数；

- 很多骨架绑定系统采用一个称为中性姿势或静止姿势的几何信息；
- 自动绑定：首先在未知的三维模型中嵌入骨架，然后计算骨骼对表面网格上每个顶点的影响权值，并将表面皮肤依附在骨骼上。
- 顶点混合(Vertex Blending)的原理
 - 采用一个物体模拟手臂，前臂和后臂仍单独设置动画，但关节处用一柔性的“skin”来连接。
 - 柔性部分的一部分顶点由前臂的矩阵来变换，另一部分由后臂的矩阵来变换。即：一个三角形的顶点可以由不同的矩阵来变换，而不是一个矩阵。
 - 进一步推广：为一个顶点可以由几个不同的矩阵进行加权变换
 - 实施方法：通过在物体上放置关节骨架，每个骨架按用户给定的权因子影响顶点来实现。

顶点混合的数学表示

$$\mathbf{u}(t) = \sum_{i=0}^{n-1} \omega_i \mathbf{B}_i(t) \mathbf{M}_i^{-1} \mathbf{P}, \quad \text{其中 } \sum_{i=0}^{n-1} \omega_i = 1, \quad \omega_i \geq 0$$

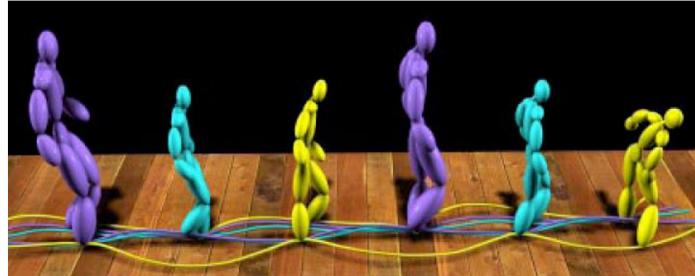
- \mathbf{P} 为变换前的顶点， $\mathbf{u}(t)$ 为变换后的顶点， n 为影响 \mathbf{P} 的关节数目
- \mathbf{M}_i ：把第*i*个关节骨架的**局部坐标系**变换到世界坐标系
- $\mathbf{B}_i(t)$ ：第*i*个关节随时间变化的世界变换，通常是一系列矩阵的连乘
- w_i : 第*i*个关节骨架作用于 \mathbf{P} 的权因子

32

运动数据的重定向

- 把运动映射到不匹配的虚拟角色上，并满足一些重要的约束条件
- 通常具有相同的结构 (with the same structure)

运动重定向



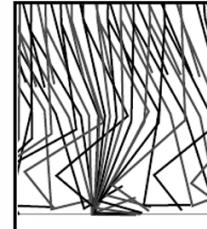
Michael Gleicher,
“Retargetting motion to
new characters”,
SIGGRAPH '98

- 把捕获的运动数据映射到不匹配的虚拟角色上，并满足一些重要的约束
- 约束（运动所应满足的性质）：
 - 避免脚穿透地板，避免自身穿透，避免走路时的打滑；
- 生成一个与原始运动尽可能接近的新的运动，并强制满足约束条件；
- 新的运动可看成是一个时空(space-time)、非线性约束的优化问题。

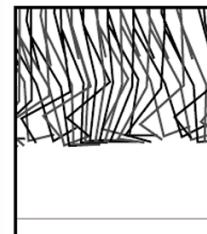
运动重定向

- $m(t) = m_o(t) + d(t)$
 - $m_o(t)$ ：原始运动； $m(t)$ ：重定向的运动
 - $d(t)$ ：两个运动之间的距离
- 如果只是简单重用原始的运动，则目标角色与其它物体之间的相互作用会失败。
- 把运动重定向问题看成一个带约束的优化问题：
 - 找到运动 $m(t)$ ，满足函数 $f(m(t)) \diamond c$
 - $f(\cdot)$ ：约束方程， c ：常数
 - $\diamond \in \{\leq, \geq, =\}$
 - 最小化目标函数： $g(m) = \int_t (m(t) - m_o(t))^2 = \int_t d(t)^2$

原始运动



Naïve重定向运动



运动图基本原理

- 将一组预先获取的运动片段通过特定的相似点连接成一个图结构，来生成新的连续动作。每个运动片段作为图的节点，节点之间的连接通过运动的相似性决定，连接处的平滑过渡依赖于帧的姿态和速度匹配。一旦构建了运动图，系统可以从中选取不同的路径，从而生成各种连续、自然的动画，而不需要重新录制新的动作。

基于Blend Shapes表情动画的原理

- 对人脸表情进行三维扫描，构建Blend Shapes模型（把每个扫描模型映射到手工建立的主网格上，找到主网格每个顶点和扫描表情某个点的对应关系，处理模型的缺陷和不一致的覆盖，对所有顶点建立稠密对应关系），根据插值这些Blend Shapes来得到任意的表情。

Face IK原理

• 动画师更愿意控制脸部上的一些点（类似于IK），而不是权因子

$$S_i = C_1 S_{i,1} + C_2 S_{i,2} + \dots + C_F S_{i,F}, i = 1, \dots, n; \quad \sum_{f=1}^F C_f = 1$$

• 我们希望约束 L 个点（由动画师控制），使得顶点 P_l 的位置为 \mathbf{P}_l

$$\sum_{f=1}^F C_f S_{l,f} = \mathbf{P}_l, l = 1, \dots, L, \text{ 其中 } \sum_{f=1}^F C_f = 1$$

• 采用最小二乘法求解系数 C_f

Deformation与Morphing的区别

- 变形Deformation是指将几何对象的形状作某种扭曲，形变，使它形成动画师所需要的形状。在这种变化中，**几何对象的拓扑关系保持不变**。
- 与Morphing不同，空间变形更具有某种随意性，所以空间变形也长称为自由变形。

与物体表示无关的变形的原理

- 与物体表示有关的变形。是指针对物体的某种具体表示形式，如多边形网格、细分曲面、参数曲面等的变形方式。
- 与物体表示无关的变形。既可作用于多边形表示的物体，又可作用于参数曲面表示的物体。
 - 用少量的点去有效控制更多的点。在传统CSG造型方法的基础上，进行非线性整体变形（变换是物体顶点位置的函数）和局部变形（改变物体的切向量空间，积分得到物体变形后的整体位置）。

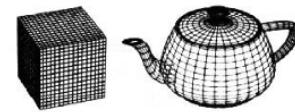
- 在几何造型中，切向量和法向量是两个非常重要的向量，因为切向量决定了物体的局部几何信息，法向量决定了物体的方向和光照信息

Tapering、Twisting和Bending变形的原理

整体和局部变形方法

- Barr推广了传统的运算操作，他提出把整体和局部变形(Global and local deformation)作为新的算子。
- 他提出的算子有：
 - Twisting(使成螺旋形)
 - Bending(弯曲)
 - Tapering(渐细)
- 这些算子的优点在于：① 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。② 变形后物体的法向量可用原物体的法向量和变换矩阵解析求得。

• original



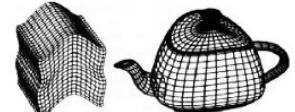
• tapering



• twisting

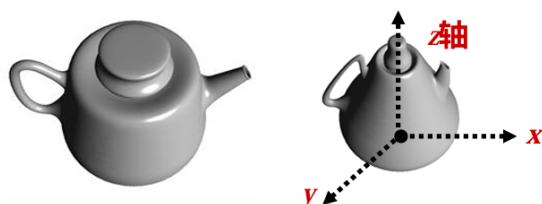


• bending



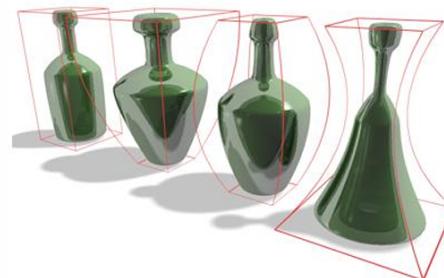
8

变形例子——Tapering



- 沿z轴的渐细变形Tapering为：

$$\mathbf{F}: \begin{cases} x' = rx \\ y' = ry, \quad r = f(z) \\ z' = z \end{cases}$$



- 当 $f'(z)>0$ 时，变形物体的大小沿z轴逐渐增大；
- 当 $f'(z)<0$ 时，变形物体的大小沿z轴逐渐变小。

[3DS MAX Taper Modifier](#)

变形例子——Twisting(沿轴螺旋形变形)

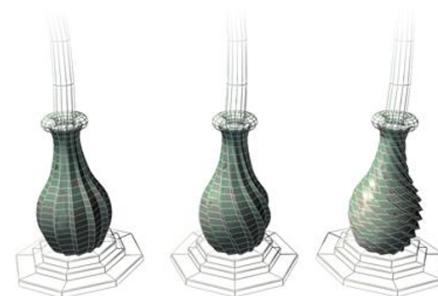
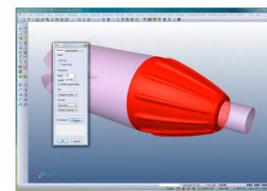
- 该变换旋转其中两个坐标分量而不改变第三个坐标分量（像扭麻花）：

$$\mathbf{F}: \begin{cases} x' = x c_\theta - y s_\theta \\ y' = x s_\theta + y c_\theta \\ z' = z \end{cases}$$



其中 $\theta = f(z), c_\theta = \cos \theta, s_\theta = \sin \theta$ 。切向量
变换矩阵为：

$$\mathbf{J} = \begin{pmatrix} c_\theta & -s_\theta & -x s_\theta f'(z) - y c_\theta f'(z) \\ s_\theta & c_\theta & x c_\theta f'(z) - y s_\theta f'(z) \\ 0 & 0 & 1 \end{pmatrix}$$



3DS MAX Twist Modifier

19

变形例子——Bending(弯曲变形)

- 沿y轴的弯曲变形(Bending)。
- 设变形的区域为 $y_{\min} \leq y \leq y_{\max}$, 中心为 y_0 ,
弯曲的曲率半径为 $1/k$,

$$\theta = k(\hat{y} - y_0),$$

$$\text{其中: } \hat{y} = \begin{cases} y_{\min}, & y \leq y_{\min} \\ y, & y_{\min} < y < y_{\max} \\ y_{\max}, & y \geq y_{\max} \end{cases}$$

- 即弯曲角 θ 在变形区域外为常数, 在中间区域为线性变化, 变形中中心线长度保持不变



3DS MAX Bend Modifier

21

FFD的变形原理

- 1986年, Sederberg等提出了一种非常适合于柔性物体动画的更为一般的方法, 该方法不直接操作物体, 而是将物体嵌入一空间, 当所嵌的空间变形时, 物体也随之变形。
- 这类方法的本质: **物体参数化!**
 - 1.确定物体的顶点 (或控制顶点) 在lattice空间的位置(参数化)
 - 2.变形FFD块。根据动画设计的需要, 移动控制顶点Pijk

- 3.确定顶点变形后的位置

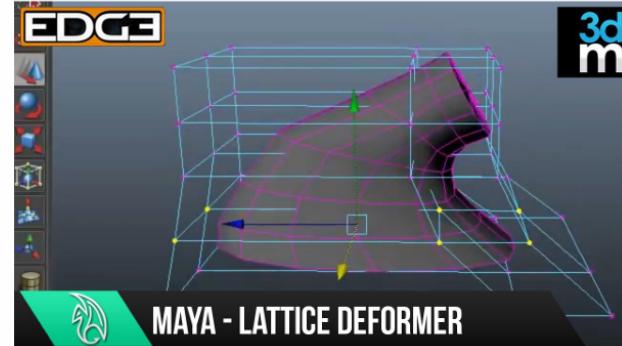
FFD总结

- **优点:**

- 与物体的几何表示无关；
- 直观；
- 高效；

- **缺点:**

- Lattice为平行六面体形状；
- 难以控制复杂的变形；
- **细节很难被保持；**
- 不遵循物理规律；



扩展的FFD方法EFFD的原理。

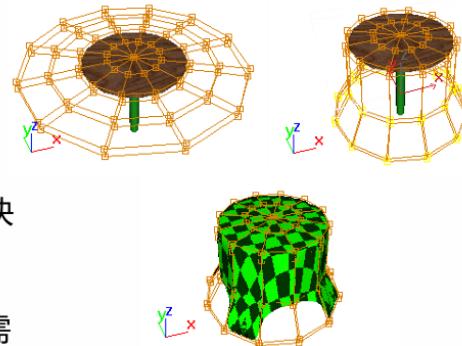
- FFD是一个非常直观的变形工具，但它只适合于平行六面体的lattice形状。
- 1990年，Coquillart提出了一种称为扩展的FFD方法，该方法**允许非平行六面体的lattice形状，从而能实现更任意的变形**
- EFFD型lattice**允许FFD型lattice作为它结构的一部分，许多个FFD型lattice可合并构成EFFD的lattice。通过把多个基本EFFD块融合，得到更复杂的复合EFFD块。使用迭代求解采用EFFD块对物体进行变形。**
 - 通过把多个基本EFFD块融合，我们可以得到更复杂的复合EFFD块。在合并超曲面块的控制顶点时，必须注意合并后的块之间的切向连续性问题

扩展的FFD方法EFFD

- EFFD块构造好以后，我们可以采用与FFD方法类似的方法来使物体变形。
- 在FFD方法中，由于FFD块的三条轴与景物空间的坐标轴重合，景物空间的点在lattice空间的局部坐标很容易求得（**线性关系！**）。
- 但在EFFD变形中，两个空间之间不再有这种简单的对应关系（**非线性关系！**），**通常需要迭代求解**。采用EFFD块对物体变形的步骤如下：

EFFD方法的优缺点

- EFFD方法的优点：**
 - 是允许更加复杂的变形空间。
- EFFD方法的缺点：**
 - 在移动内部控制顶点时必须保持块与块之间的**连续性**；
 - 计算景物点在lattice空间的局部坐标需**要数值求解方法**，导致**计算速度变慢**。

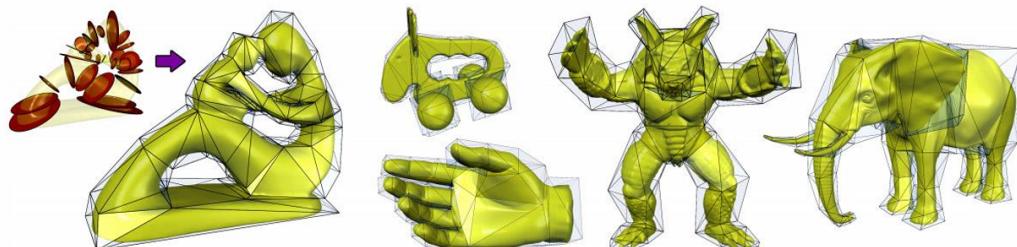


3DS MAX中对应的类似功能：
FFD(Cyl) Space Warp

53

基于Cage的变形原理

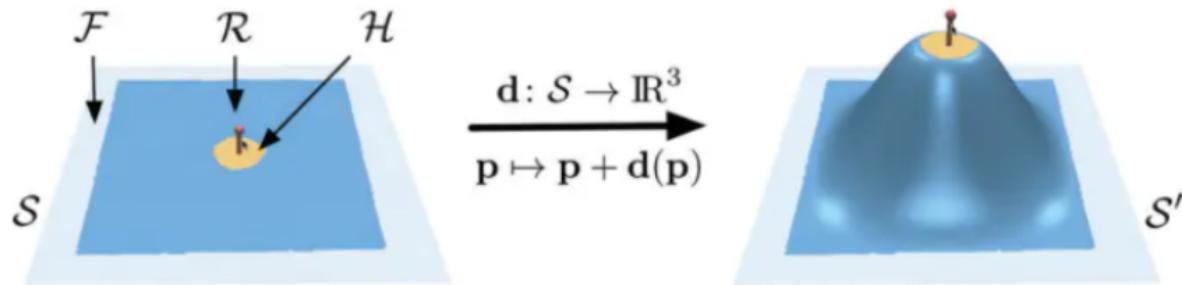
更一般：基于Cage的变形



- Cage：**一个包裹高分辨率模型的**低分辨率控制网格**。
- 基于cage的变形方法具有**直观性、简单性和高效性**，在最近十多年得到了越来越多的关注。
- 变形原理：**(1). 构建模型的Cage；(2). 计算模型的Cage坐标；(3). 编辑Cage模型；(4). 把Cage的变形通过预先计算的Cage坐标光滑传播到其包裹的模型。

Laplacian微分坐标及其变形原理

原理：形变前的点的拉普拉斯坐标与形变后的点的拉普拉斯坐标尽可能相等



一图以蔽之，左图S是初始网格，右图S'是变形后的网格。

F, 淡蓝色区域，是S的子区域，固定初始位置不变，不参与变形。

R, 深蓝色与黄色区域，ROI区域，是位置改变的区域，其中（差集R-H）区域参与变形计算

H, 黄色区域，olga Sokraine称其为Handle Vertices，形象理解为用手拽着移动的区域。

由上两个例子可以看出，网格变形的一个需求就是要保证局部的细节特征尽量不变，而常规的笛卡尔坐标只保存了顶点的空间信息，这时引入**拉普拉斯坐标 (δ-Coordinate)** 的概念。我们将每个顶点的笛卡尔坐标转换为拉普拉斯坐标，再来思索问题。

$$\delta_i = (\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)}) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v}_j ,$$

拉普拉斯坐标，其数学意义为一个顶点的笛卡尔坐标和这个点所有邻接点的笛卡尔坐标的加权平均，物理意义为拉普拉斯坐标能够记录模型的局部细节特征。

当网格形变后，我们希望网格的局部信息不要被破坏，换成坐标的说法就是，网格上顶点的笛卡尔坐标变了，但是我们希望网格上所有点的拉普拉斯坐标没变，也就是点和点的相对位置关系没变。

总结

- 与物体表示无关的变形方法
 - 整体和局部变形方法
 - FFD方法及其变种
 - 基于Cage的变形方法
- 基于拉普拉斯微分坐标的Mesh变形方法

布料动画和仿真中的核心问题有哪4个

- 建立布料的物理(力学)模型 (表达能力)
 - 需要考虑布料的拉伸、剪切、弯曲等物理力;
 - 布料不是各项同性的连续体;
- 布料物理参数的确定 (缝线、纽扣、装饰物等都会影响布料的物理属性)
- 物理模型的求解 (偏微分方程数值解)
 - 速度
 - 稳定性
- 碰撞检测和响应
 - 衣服与人体的碰撞、衣服与衣服的碰撞(自碰撞)
 - 多层衣服
 - 胳膊窝等特殊位置的处理
 - 纽扣、装饰物与衣服的碰撞

布料的物理机械性能主要有哪 4 种？布料的弹簧-质点模型主要有哪 3 种弹簧，并图示。

- 物理机械性能：
 - 拉伸力(In-plane stretch)
 - 压缩力(In-plane compression)
 - 剪切力(In-plane shear(trellising))
 - 弯曲力(Out-of-plane bending)
- 弹簧-质点模型的弹簧：

- 结构弹簧(Stretch spring): 结构弹簧连接上下左右的相邻的质点，用于结构力(拉力或者压力)的计算；
- 剪切弹簧(Shear spring): 剪切弹簧连接左上左下右上右下对角相邻的质点，用于剪力的计算；
- 弯曲弹簧(Bend spring): 弯曲弹簧连接上下左右隔一个质点的质点，用于弯矩的计算。

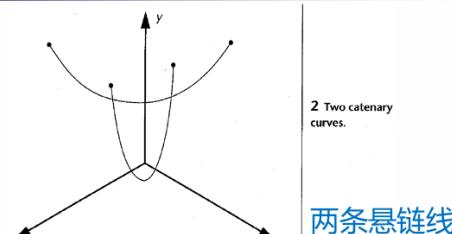
Weil的悬链线表示悬垂布料方法原理

- Weil的悬链线表示悬垂布料

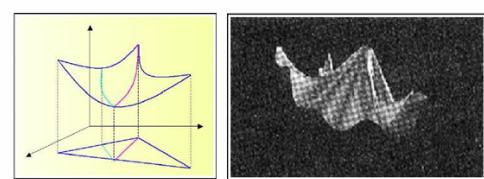
- J. Weil, *The Synthesis of Cloth Objects*, Computer Graphics (Proc. SIGGRAPH 86), 20 (4), pp. 49-54, 1986.

- 悬链线的定义为(双曲余弦函数):

- 把布料悬挂在一些约束点上，基于悬链线计算出布料自由悬挂时的形状。
- Weil方法把一个三角形细分为两个三角形，并沿着新的边生成一条新的悬链线，如下图左所示，最后生成的具有真实感的悬挂的布料如下图右所示。
- 该方法不考虑布料的质量、弹性系数等物理因素，不能生成布料的动态效果，只适用于表示悬垂的布料。



两条悬链线



用悬链线细分三角形 生成的布料模型

Provot的衣服模型的原理

- Provot提出了一种可以看作是从Breen的粒子系统简化而来的弹簧-质点模型，并在此基础上发展了一套非常简单有效的算法。
- Provot采用了简单高效的**显式Euler法**求解微分方程组，得益于物理模型的简单方便，Provot的方法在效率上明显比基于Terzopoulos模型的方法要来的高，而且显示效果也不错。

显式欧拉法

- 显式欧拉法公式：

$$\begin{cases} \mathbf{a}^{t+h} = \mathbf{M}^{-1}\mathbf{f}^t \\ \mathbf{v}^{t+h} = \mathbf{v}^t + h\mathbf{a}^{t+h} \\ \mathbf{x}^{t+h} = \mathbf{x}^t + h\mathbf{v}^{t+h} \end{cases} \longrightarrow \begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^t \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix}$$

- 其中 \mathbf{M} 是质点的质量， \mathbf{x} 、 \mathbf{v} 、 \mathbf{a} 、 \mathbf{f} 分别表示粒子的位置、速度、加速度和所受合力， h 是时间步长。上标 t 和 $t+h$ 是时间，分别表示当前时间和下一步的时间。
- 优点：
 - 单步计算快。
 - 并行性好。在一个计算步中，每一个质点的状态都是独立计算。

Provot的衣服模型

- 但由于采用了显式积分方法，算法的**稳定性**成为一个明显的问题。
- h 存在一个最大的临界值 H ，一旦超过 H ，方程就会病态而失稳。而这个 H 跟弹簧的刚度有关(弹簧的刚度 K 是载荷增量 dF 与变形增量 $d\lambda$ 之比，即产生单位变形所需的载荷，弹簧的刚度计算公式为 $K=dF/d\lambda$)：

$$H \approx \pi \sqrt{\frac{M}{K}}$$

- 为了保证算法稳定，必须采用**很小的时间步长**或者减小方程组的刚度，也即是弹簧的刚度。
- 取小的计算步长意味着计算次数增加，取小的弹簧刚度则会形成一种 Provot 称之为**超弹性的问题**，即布料产生了现实中不可能发生的伸长，使得算法的真实感明显下降。

布料动画中大步长隐式方法的原理

- 布料模拟的积分计算部分的主要开销在由于稳定性限制计算步长只能取很小值，因而被迫在两帧画面之间计算多次；在布料模拟的积分计算部分设法取较大的时间步长可以有效的提高算法效率。使用隐式方法来积分，在不降低刚度的条件下可以取得较大的时间步长。
- 从数值方法的理论可知，隐式方法具有比显式方法高得多的稳定性。因此，Baraff and Witkin 使用了隐式方法来积分，在不降低刚度的条件下可以取得较大的时间步长。

大步长隐式方法

$$\begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^t \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix} \xrightarrow{\text{显式欧拉法}} \begin{pmatrix} \mathbf{v}^{t+h} \\ \mathbf{x}^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^t + h\mathbf{M}^{-1}\mathbf{f}^{t+h} \\ \mathbf{x}^t + h\mathbf{v}^{t+h} \end{pmatrix} \xrightarrow{\text{隐式欧拉法}}$$

$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n]^T, \quad \mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n]^T, \quad \mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$

$$\mathbf{M}_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{M}_n \end{bmatrix}$$

这里 $\mathbf{f}_i, \mathbf{v}_i, \mathbf{x}_i, m_i$ 分别表示质点 i 的力、速度、位置和质量, n 为质点的数目。

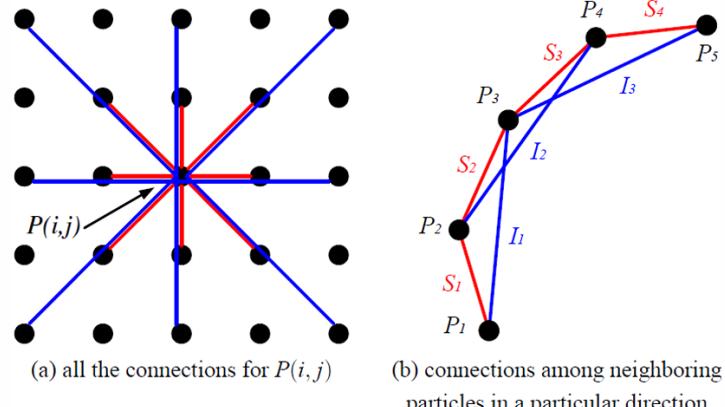
稳定但反应灵敏的布料的原理

- 这个模型考虑到布料局部在受到压缩作用时**并不会出现很大的压缩抗力**, 而是会很快发生屈曲, 因此物理模型中**将压缩和弯曲行为合二为一**进行建模, 因此模型中将不会出现传统弹簧—质点系统中所具有的压缩弹簧

稳定但反应灵敏的布料

Stable but Responsive Cloth

- 采用2种连接弹簧
 - 红色: 类型 1
 - 蓝色: 类型 2
- 红色连接弹簧:** 用于抗拉伸和剪切
 - $P(i, j)$ 连接 $P(i \pm 1, j), P(i, j \pm 1)$ 和 $P(i \pm 1, j \pm 1)$; 这种连接称为**顺序连接**。
- 蓝色连接弹簧:** 用于抗弯曲和抗压缩
 - $P(i, j)$ 连接 $P(i \pm 2, j), P(i, j \pm 2)$ 和 $P(i \pm 2, j \pm 2)$ 。这种连接称为**隔行连接**。
- 右图图示了在一个特定方向的顺序连接和隔行连接。 S_1-S_4 是顺序连接; I_1-I_3 是隔行连接。



相互作用粒子的连接关系

弹簧质点法优缺点:

- 优点: 直接利用布料的Mesh表示, 简单高效, 碰撞容易处理, 可扩展性好;
- 缺点: 精度有限, 仿真结果依赖于所使用的网格表示。

各种方法的比较

	动画形式	优点	缺点
几何方法	几何曲线	速度快	真实感差
物理方法(显式)	动力学	真实感好， 单步计算快	稳定性差， 总体效率不高
物理方法(隐式)	动力学	稳定性好	需要求解线性方程， 单步计算复杂
近似隐式法	近似动力学	速度快， 稳定性好	真实感差
混合方法	动力学+几何	速度快	真实感较差