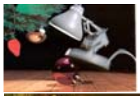
Animation



Conventional Animation

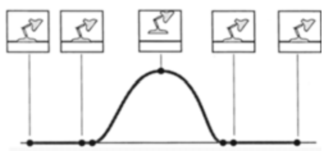
Draw each frame of the animation
- great control
- tedious

Reduce burden with cel animation
- layer
- keyframe
- inbetween
- cel panoramas (Disney's Pinocchio)
- ...

ACM © 1997 "Multiperspective panoramas for cel animation"

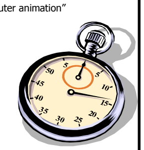
Computer-Assisted Animation

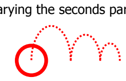
Keyframing
- automate the inbetweening
- good control
- less tedious
- creating a good animation still requires considerable skill and talent



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

Procedural animation
- describes the motion algorithmically
- express animation as a function of small number of parameteres
- Example: a clock with second, minute and hour hands
  - hands should rotate together
  - express the clock motions in terms of a "seconds" variable
  - the clock is animated by varying the seconds parameter
- Example 2: A bouncing ball
  - Abs(sin(ωt+θ_0))*e^{-kt}





Computer-Assisted Animation

Physically Based Animation
- Assign physical properties to objects (masses, forces, inertial properties)
- Simulate physics by solving equations
- Realistic but difficult to control

ACM© 1988 "Spacetime Constraints"

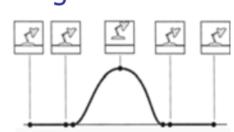
Motion Capture
- Captures style, subtle nuances and realism
- You must observe someone do something

Overview

Hermite Splines
Keyframing
Traditional Principles

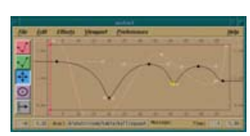
Articulated Models
Forward Kinematics
Inverse Kinematics
Optimization
Differential Constraints

Keyframing



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

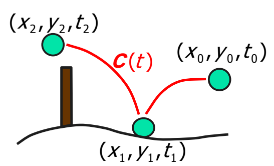
Describe motion of objects as a function of time from a set of key object positions. In short, compute the inbetween frames.



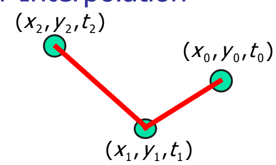
Interpolating Positions

Given positions: $(x_i, y_i, t_i),\ i = 0,\dots,n$

find curve $\boldsymbol{C}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ such that $\boldsymbol{C}(t_i) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$



Linear Interpolation



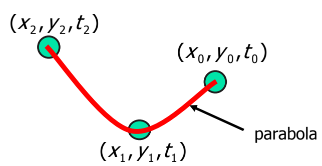
Simple problem: linear interpolation between first two points assuming t_0=0 and t_1=1: $x(t) = x_0(1-t) + x_1 t$

The x-coordinate for the complete curve in the figure:

$$x(t) = \begin{cases} \dfrac{t_1 - t}{t_1 - t_0} x_0 + \dfrac{t - t_0}{t_1 - t_0} x_1, & t \in [t_0, t_1) \\ \dfrac{t_2 - t}{t_2 - t_1} x_1 + \dfrac{t - t_1}{t_2 - t_1} x_2, & t \in [t_1, t_2] \end{cases}$$

Derivation?

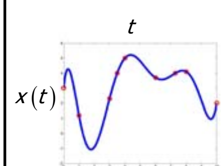
Polynomial Interpolation



An n-degree polynomial can interpolate any n+1 points. The Lagrange formula gives the n+1 coefficients of an n-degree polynomial that interpolates n+1 points. The resulting interpolating polynomials are called Lagrange polynomials. On the previous slide, we saw the Lagrange formula for n = 1.

Spline Interpolation

Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly.



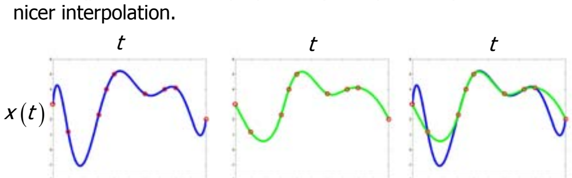
8-degree polynomial

How many n-degree polynomials interpolate n+1 points?

Spline Interpolation

Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly.

Spline (piecewise cubic polynomial) interpolation produces nicer interpolation.



8-degree polynomial spline spline vs. polynomial

Spline Interpolation

A cubic polynomial between each pair of points:

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

Four parameters (degrees of freedom) for each spline segment.

Number of parameters:

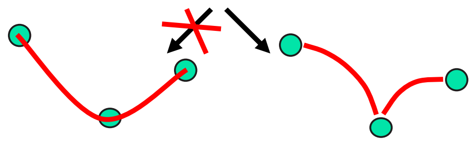
n+1 points ÿ n cubic polynomials ÿ 4n degrees of freedom

Number of constraints:

- interpolation constraints
  n+1 points ÿ 2 + 2 (n-1) = 2n interpolation constraints
  "endpoints" + "each side of an internal point"
- rest by requiring smooth velocity, acceleration, etc.

Hermite Splines

We want to support general constraints: not just smooth velocity and acceleration. For example, a bouncing ball does not always have continuous velocity:



Solution: specify position AND velocity at each point

Derivation? $c_0, c_1, c_2, c_3 = ?$ for $x_0, v_0 : t = t_0$ and $x_1, v_1 : t = t_1$

Keyframing

Given keyframes $K_i = \left(p_i^0, p_i^1, \dots, t_i\right),\ i = 0, \dots, n$

find curves $\boldsymbol{K}(t) = \begin{bmatrix} p^0(t) \\ p^1(t) \\ \vdots \end{bmatrix}$ such that $\boldsymbol{K}(t_i) = \boldsymbol{K}_i$

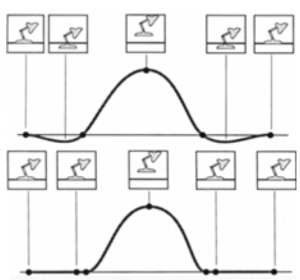
What are parameters $p_i^0, p_i^1, \dots$?

- position, orientation, size, visibility, …

Interpolate each curve separately

Interpolating Key Frames

Interpolation is not fool proof. The splines may undershoot and cause interpenetration. The animator must also keep an eye out for these types of side-effects.



Traditional Animation Principles

The in-betweening, was once a job for apprentice animators. We described the automatic interpolation techniques that accomplish these tasks automatically. However, the animator still has to draw the key frames. This is an art form and precisely why the experienced animators were spared the in-betweening work even before automatic techniques.

The classical paper on animation by John Lasseter from Pixar surveys some the standard animation techniques:

"*Principles of Traditional Animation Applied to 3D Computer Graphics,*" **SIGGRAPH'87**, pp. 35-44.

Squash and stretch

Squash: flatten an object or character by pressure or by its own power

Stretch: used to increase the sense of speed and emphasize the squash by contrast

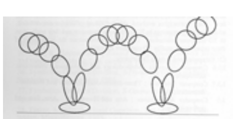


FIGURE 2. Squash & stretch in bouncing ball.

FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Timing

Timing affects weight:

- Light object move quickly
- Heavier objects move slower

Timing completely changes the interpretation of the motion. Because the timing is critical, the animators used the draw a time scale next to the keyframe to indicate how to generate the in-between frames.

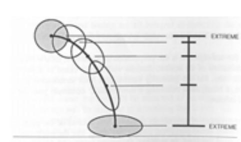


FIGURE 9. Timing chart for ball bounce.

Anticipation

An action breaks down into:

- Anticipation
- Action
- Reaction

Anatomical motivation: a muscle must extend before it can contract. Prepares audience for action so they know what to expect. Directs audience's attention. Amount of anticipation can affect perception of speed and weight.

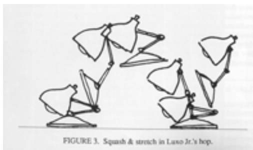


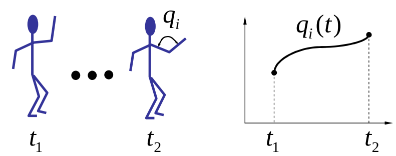
FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Articulated Models

Articulated models:

- rigid parts
- connected by joints

They can be animated by specifying the joint angles as functions of time.



Forward Kinematics

Describes the positions of the body parts as a function of the joint angles.

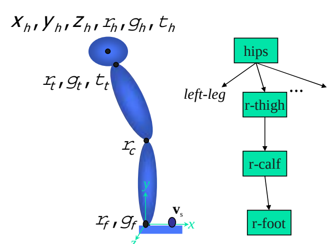
1 DOF: knee

2 DOF: wrist

3 DOF: arm

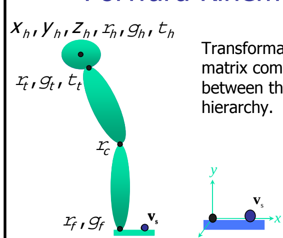
Skeleton Hierarchy

Each bone transformation described relative to the parent in the hierarchy:



Derive world coordinates v_w for an effecter with local coordinates v_s ?

Forward Kinematics



Transformation matrix for an effecter v_s is a matrix composition of all joint transformation between the effecter and the root of the hierarchy.

$$v_w = T(x_h, y_h, z_h) R(\theta_h, \phi_h, \sigma_h) TR(\theta_t, \phi_t, \sigma_t) TR(\theta_c) TR(\theta_f, \phi_f) v_s$$

$$v_w = S\left(\underbrace{x_h, y_h, z_h, \theta_h, \phi_h, \sigma_h, \theta_t, \phi_t, \sigma_t, \theta_c, \theta_f, \phi_f}_{p}\right) v_s = S(p) v_s$$

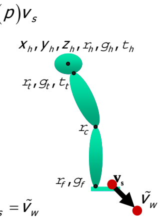
Inverse Kinematics

Forward Kinematics

- Given the skeleton parameters (position of the root and the joint angles) p and the position of the effecter in local coordinates v_s, what is the position of the sensor in the world coordinates v_w?
- Not too hard, we can solve it by evaluating $S(p)v_s$

Inverse Kinematics

- Given the position of the effecter in local coordinates v_s and the desired position $\tilde{v}_w$ in world coordinates, what are the skeleton parameters p?



- Much harder requires solving the inverse of the non-linear function: p? such that $S(p)v_s = \tilde{v}_w$
- Underdetermined problem with many solutions

Real IK Problem

Find a "natural" skeleton configuration for a given collection of pose constraints.

Definition: A scalar objective function *g(p)* measures the quality of a pose. The objective *g(p)* reaches its minimum for the most natural skeleton configurations *p.*

Definition: A vector constraint function *C(p) = 0* collects all pose constraints:

$$\underbrace{\begin{bmatrix} S_0(p)v_0 - \tilde{v}_0 \\ S_1(p)v_1 - \tilde{v}_1 \\ \vdots \end{bmatrix}}_{C(p)} = 0$$

Optimization

Compute the optimal parameters p^* that satisfy pose constraints and maximize the natural quality of skeleton configuration:

$$p^* = \underset{p}{\text{argmin}} \quad g(p)$$
$$\text{s.t.} \qquad C(p) = 0$$

Example objective functions *g(p):*

- deviation from natural pose: $g(p) = (p - \bar{p})^T M(p - \bar{p})$
- joint stiffness
- power consumption
- …

Unconstrained Optimization

Define an objective function *f(p)* that penalizes violation of pose constraints:

$$f(p) = g(p) + \sum_i w_i \left[C_i(p)\right]^2$$

$$p^* = \underset{p}{\text{argmin}} \quad f(p)$$

Necessary condition:

$$f(p^* + \Delta p) - f(p^*) \geq 0 \quad (p^* \text{ is local minimum})$$
$$\nabla f(p^*)^T \Delta p + \cdots \geq 0 \quad (\text{Taylor series})$$
$$\Downarrow$$
$$\nabla f(p^*) = 0 \quad (\Delta p \text{ is arbitrary})$$

Numerical Solution

Gradient methods

- Guess initial solution x_0
- Iterate $x_{k+1} = x_k + \alpha_k d_k, \quad \alpha_k > 0, \quad \nabla f(x_k)^T d_k < 0$
- Until $\nabla f(x_k) = 0$

The conditions $\alpha_k > 0, \quad \nabla f(x_k)^T d_k < 0$ guarantee that each new iterate is more optimal $f(x_{k+1}) < f(x_k)$. Derive?

Some choices for direction d_k:

- Steepest descent $d_k = -\nabla f(x_k) = -\frac{\partial g}{\partial p} - 2\sum_i w_i C_i \frac{\partial C_i}{\partial p}$
- Newton's method $d_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$
- Quasi-Newton methods $d_k = -D_k \nabla f(x_k)$

Gradient Computation

Requires computation of constraint derivatives:

- Compute derivatives of each transformation primitive
- Apply chain rule

Example:

$$C(p) \gg \mathbf{T}(x_h, y_h, z_h)\mathbf{R}(r_h, g_h, t_h)\,\mathbf{TR}(r_t, g_t, t_t)\,\mathbf{TR}(r_c)\,\mathbf{TR}(r_f, g_f)\,v_s \;.\; \tilde{v}_w$$

$$\frac{\bullet C}{\bullet r_c} > \mathbf{T}(x_h, y_h, z_h)\mathbf{R}(r_h, g_h, t_h)\,\mathbf{TR}(r_t, g_t, t_t)\,\mathbf{T}\frac{\bullet \mathbf{R}(r_c)}{\bullet r_c}\mathbf{TR}(r_f, g_f)\,v_s$$

Derive $\frac{\bullet \mathbf{R}(r_c)}{\bullet r_c}$ if **R** is a rotation around z-axis?

Constrained Optimization

Unconstrained formulation has drawbacks:

- Sloppy constraints
- The setting of penalty weights w_i must balance the constraints and the natural quality of the pose

Necessary condition for equality constraints:

- Lagrange multiplier theorem:

$$\nabla f(p^*) + \sum_i \lambda_i \nabla C_i(p^*) = 0$$

- $\lambda_0, \lambda_1, \ldots$ are scalars called Lagrange multipliers

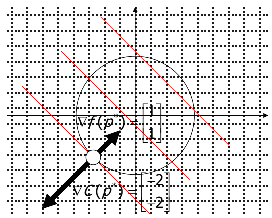
Interpretations:

- Cost gradient (direction of improving the cost) belongs to the subspace spanned by constraint gradients (normals to the constraints surface).
- Cost gradient is orthogonal to subspace of feasible variations.

Example

$$p^* = \underset{p}{\arg\min} \quad p_1 + p_2$$

$$\text{s.t.} \qquad p_1^2 + p_2^2 = 2$$



Nonlinear Programming

Use Lagrange multipliers and nonlinear programming techniques to solve the IK problem:

$$p^* = \underset{p}{\arg\min} \quad g(p)\Big\} \quad \text{nonlinear objective}$$

$$\text{s.t.} \qquad C(p) = 0\} \quad \text{nonlinear constraints}$$

In general, slow for interactive use!

Differential Constraints

Differential constraints linearize original pose constraints.

Rewrite constraints by pulling the desired effecter locations to the right hand side

$$\underbrace{\begin{bmatrix} S_0(p)v_0 \\ S_1(p)v_1 \\ \vdots \end{bmatrix}}_{C(p)} = \begin{bmatrix} \tilde{v}_0 \\ \tilde{v}_1 \\ \vdots \end{bmatrix}$$

Construct linear approximation around the current parameter p. Derive with Taylor series.

$$\frac{\partial C(p)}{\partial p} \Delta p = \Delta \tilde{v}$$

IK with Differential Constraints

Interactive Inverse Kinematics

- User interface assembles desired effecter variations $\Delta \tilde{v}$
- Solve quadratic program with Lagrange multipliers:

$$\Delta p^* = \underset{\Delta p}{\arg\min} \quad \Delta p^T M \Delta p$$

$$\text{s.t.} \qquad \frac{\partial C(p)}{\partial p} \Delta p = \Delta \tilde{v}$$

- Update current pose: $p_{k+1} = p_k + \alpha_k \Delta p^*$

Objective function is quadratic, differential constraints are linear.

Some choices for matrix *M:*

- Identity: minimizes parameter variations
- Diagonal: minimizes scaled parameter variations

Quadratic Program

Elimination procedure

Apply Lagrange multiplier theorem and convert to vector notation:

$$M \Delta p + \left(\frac{\partial C(p)}{\partial p} \right)^T \lambda = 0$$

general form in scalar notation:
$\nabla f(p^*) + \sum_i \lambda_i \nabla C_i(p^*) = 0$

Rewrite to expose Δp:

$$\Delta p = -M^{-1} \left(\frac{\partial C(p)}{\partial p} \right)^T \lambda$$

Use this expression to replace Δp in the differential constraint:

$$\frac{\partial C(p)}{\partial p} M^{-1} \left(\frac{\partial C(p)}{\partial p} \right)^T \lambda = -\Delta \tilde{v}$$

Solve for Lagrange multipliers and compute Δp^*

Kinematics vs. Dynamics

Kinematics

Describes the positions of body parts as a function of skeleton parameters.

Dynamics

Describes the positions of body parts as a function of applied forces.

Next

Dynamics

ACM© 1988 "Spacetime Constraints"