CS 775: Advanced Computer Graphics

Lecture 9: Kinematics
Character Animation

- Traditional
  - Cell Animation, hand drawn, 2D
  - Lead Animator for keyframes

http://animation.about.com/od/flashanimationtutorials/ss/flash31detanim2.htm
Character Animation

- Traditional, hand drawn animation
  - Lead Animator for keyframes and many secondary animators for the in-betweens

http://animation.about.com/od/flashanimationtutorials/ss/flash31detanim2.htm
Character Animation

- Computer assisted keyframing
  - Keyframes created/posed by hand
  - In-betweens interpolated by the semi-automatic techniques

Luxo Jr, PIXAR, 1986

How is this done?
Character Animation

- Interpolating Position and Orientation parameters and animating (rigid) transformations.
- Linear interpolation, spline interpolation, quaternions.

Luxo Jr, PIXAR, 1986

How is this done?
Character Animation

- Creating the keyframe pose for the lamp
  - Modify the joint angles at each internal joint
  - Modify global position and orientation
Character Animation

- **Kinematics**
  - Study of motion of objects by studying the change in their orientation and position. The cause of the motion, i.e., the forces are *not* studied.

- **Dynamics**
  - Study of motion of objects in relation to the forces and torques that cause the motion.
Character Animation

- **Kinematics**
  - Forward and Inverse Kinematics – used for posing characters and interpolation in keyframed animation.
  - Faster to compute, easier to control

- **Dynamics**
  - Physically-based animation – used for animations involving simulations of real world physics, for e.g., collisions.
  - Harder to compute and control, more realistic (if modelled correctly),
  - Can automatically adjust to changes in the environment.
Character Animation

- Terminology

- Root Joint
- Internal Joints
- Links or Bones
- End-effector

Kinematic Chain
Character Animation

- Terminology

\[ \theta_0, \phi_0, \delta_0, t_x, t_y, t_z \]

Joint DoF (parameters)

Root DoF
Character Animation

- Forward Kinematics
  - Given the joint parameters, find the position of the end effector.
  - Position of the root is given as a global transformation.
  - Joint parameters are given as relative rotations (local transformations).
  - We already know how to solve this! (from CS475/CS675)
Character Animation

- Inverse Kinematics
  - Given the desired end effector position, compute the joint parameters.
  - More interesting and easier to animate actions like – reaching, walking, grabbing.
  - Much harder to solve.
  - Why?
  - Demo
Character Animation

- Inverse Kinematics

Given the link lengths $l_1$ and $l_2$ the desired position of the end-effector, $P(x, y)$ find the joint parameter $\theta_1$ and $\theta_2$ that will make the end-effector reach the goal.
Character Animation

- Inverse Kinematics

\[ x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \]

\[ y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \]
Character Animation

• Inverse Kinematics

$P(x, y)$

IK a hard problem because it is an ill-posed problem.
Character Animation

- Posing the Kinematics problem
  
  - Forward
    \[ X = f(\Theta) \]
  
  - Inverse
    \[ \Theta = f^{-1}(X) \]

\( X \) is the vector of end-effector position and orientation. \( \Theta \) is the vector of joint parameters. \( f() \) is the mapping between the two.
Character Animation

- Assumptions
  - The mapping $f()$ is usually non-linear in the general case. However, we linearize the problem by localizing around the current position and use the Jacobian, i.e.,

  $$d\mathbf{X} = J(\Theta) \, d\Theta \quad \Rightarrow \quad d\Theta = J^{-1}(\Theta) \, d\mathbf{X}$$

- Note that the Jacobian is always a function of the current set of joint parameters – so it has to recalculated every time they change!
Character Animation

• Assumptions
  – We will attempt to solve the Inverse Kinematics problems under the following assumptions:
    • All joints are hinge joints.
    • All links are rigid and do not change in length.
  – Note that these assumptions are not there because the method we will use to solve the Jacobian require these assumptions but instead to simplify our discussion a bit.
Character Animation

- Construct the Jacobian
- Invert the Jacobian
- Iterate
- Detect singularities and avoid ill-conditioning
Character Animation

• Steps:
  - Construct the Jacobian
  - Invert the Jacobian
  - Iterate
  - Detect singularities and avoid ill-conditioning
Character Animation

- Forming a kinematic chain (Hierarchical modelling revisit)

- Given a kinematic chain with links (local coordinate frames) numbered from \{1\} to \{j-1\}.

- The position and orientation of the root is known in the global coordinate frame, \{0\}.

- Each has its local x-axis oriented along the length of the link and the local z-axis as the axis of rotation about the joint.
Character Animation

- Forming a kinematic chain

- Then the $j^{th}$ link can be added as follows:
  - Translating along the local x-axis by the length of link $\{j-1\}$, $l_{j-1}$
Character Animation

- Forming a kinematic chain

Then the $j^{th}$ link can be added as follows:

- Translating along the local x-axis by the length of link $\{j-1\}$, $l_{j-1}$
- Rotating at the new joint origin so that the local x-axis lies along the length of the new link.
Character Animation

- Forming a kinematic chain

- So if this the transformation between the \{ j \} and \{ j-1 \} frame is given by \( j^{-1} M_j \)
  - Any point \( p_j \) in the frame \{ j \} can be moved to a corresponding point \( p_{j-1} \) in frame \{ j-1 \} as \( p_{j-1} = j^{-1} M_j p_j \)
  - Therefore, the coordinates of the point in the global frame is given by
    \[
p_0 = M_0 \cdot M_1 \cdot \ldots \cdot j^{-2} M_{j-1} \cdot j^{-1} M_j p_j
    \]
Character Animation

- Moving to global coordinates

The transformation from any local frame \( \{ j \} \) to the global frame is thus, given by

\[
T_j = {^0}M_1 \ldots {^{j-2}}M_{j-1} \cdot {^{j-1}}M_j
\]

\[
T_j = \begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_1 \\
  r_{21} & r_{22} & r_{23} & t_2 \\
  r_{31} & r_{32} & r_{33} & t_3 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]
Character Animation

- Constructing the Jacobian

\[
d X = J(\Theta) \, d \Theta
\]

\[
\dot{X} = J(\Theta) \, \dot{\Theta}
\]

- The Jacobian relates the end-effector velocities to velocities of the joint parameters.

\[
\begin{bmatrix}
0 V_{nx} \\
0 V_{ny} \\
0 V_{nz} \\
0 \Omega_{nx} \\
0 \Omega_{ny} \\
0 \Omega_{nz}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial f_1}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_1}{\partial \theta_{n-1}} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\partial f_i}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_i}{\partial \theta_{j}} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\partial f_6}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_6}{\partial \theta_{n-1}}
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\vdots \\
\vdots \\
\dot{\theta}_{n-2} \\
\dot{\theta}_{n-1}
\end{bmatrix}
\]
Character Animation

- Constructing the Jacobian

\[ d \mathbf{X} = J (\Theta) \, d \Theta \]

\[ \dot{\mathbf{X}} = J (\Theta) \, \dot{\Theta} \]

\[ \text{Known} \rightarrow \text{Unknown} \]

The Jacobian relates the end-effector velocities to velocities of the joint parameters.

\[ \begin{bmatrix} V_{nx} \\ V_{ny} \\ V_{nz} \\ \Omega_{nx} \\ \Omega_{ny} \\ \Omega_{nz} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_1}{\partial \theta_{n-1}} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \vdots \\ \frac{\partial f_i}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_i}{\partial \theta_{j}} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \vdots \\ \frac{\partial f_6}{\partial \theta_1} & \cdots & \cdots & \cdots & \frac{\partial f_6}{\partial \theta_{n-1}} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_{n-2} \\ \dot{\theta}_{n-1} \end{bmatrix} \]

Simplifying Notation
Character Animation

- The functions $f_1, f_2, f_3$ relate the joint parameter *angular* velocities to the linear velocity of the end-effector.

- The functions $f_4, f_5, f_6$ relate the joint parameter *angular* velocities to the angular velocity of the end-effector.

- The velocities of the end-effector are known in the global coordinate system. So the Jacobian has to be formed in the global coordinate system.
Character Animation

- Constructing the Jacobian

  - The $j^{th}$ column of the Jacobian relates the $j^{th}$ joint parameter *angular* velocity to the velocity of the end-effector.

$$ J(\Theta) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \cdots & \frac{\partial f_1}{\partial \theta_{n-1}} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_i}{\partial \theta_1} & \cdots & \cdots & \frac{\partial f_i}{\partial \theta_j} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_6}{\partial \theta_1} & \cdots & \cdots & \frac{\partial f_6}{\partial \theta_{n-1}} \end{bmatrix} $$
Character Animation

• Constructing the Jacobian

  For the angular velocity, \( \Omega_j = z_j \dot{\theta}_j \)

  – The contribution of this angular velocity to the angular velocity of the end-effector (in global coordinates) is given by \( ^0\Omega_j = T_j z_j \dot{\theta}_j = u_j \dot{\theta}_j \)

  – So

\[
\frac{\partial f_4}{\partial \theta_j} = (T_j z_j)_x = u_{jx} \quad \frac{\partial f_5}{\partial \theta_j} = (T_j z_j)_y = u_{jy} \quad \frac{\partial f_6}{\partial \theta_j} = (T_j z_j)_z = u_{jz}
\]
Character Animation

- Constructing the Jacobian
  - For the angular velocity, \( \Omega_j = z_j \dot{\theta}_j \)
    - The contribution of this angular velocity to the linear velocity of the end-effector (in global coordinates) is given by
      \[
      ^0V_j = ((T_j z_j) \times (P_n - P_j)) \dot{\theta}_j = v_j \dot{\theta}_j
      \]
    - So
      \[
      \frac{\partial f_1}{\partial \theta_j} = v_{jx} \quad \frac{\partial f_2}{\partial \theta_j} = v_{jy} \quad \frac{\partial f_3}{\partial \theta_j} = v_{jz}
      \]
      where
      \[
      P_j = T_j O_j \\
      P_n = T_n O_n
      \]
Character Animation

- Construct the Jacobian
- Invert the Jacobian
- Iterate
- Detect singularities and avoid ill-conditioning
Character Animation

- Inverting the Jacobian
  
  \[ d \Theta = J^{-1}(\Theta) d \, X \]
  \[ \dot{\Theta} = J^{-1}(\Theta) \dot{X} \]

- Note that the Jacobian is a *fat* matrix, i.e., it has more columns than rows.

- So the system is underconstrained – there is more than one possible solution.

- We choose the simplest solution, i.e., one of minimum norm:

  Find the unique \( \dot{\Theta} \) for which we have:

  \[ \min ||\dot{\Theta}||^2 \quad \text{s.t.} \quad \dot{X} = J(\Theta) \dot{\Theta} \]
Character Animation

- Linear Least Squares (Digression)
  - For the system
    \[ AX = B \]
  - If A has full row rank then the unique X for which we get
    \[ \min_{X} \| X \|^2 \quad \text{s.t.} \quad AX = B \]
    is given by
    \[ X = A^T (AA^T)^{-1} B = A^+ B \]
  - To see how this is so, solve the Lagrangian,
    \[ L = \| X \|^2 + \lambda^T (AX - B) \]
Character Animation

- Linear Least Squares (Digression)
  
  For the system
  
  \[ AX = B \]

  - If \( A \) has full row rank then the unique \( X \) for which we get
  
  \[ \min \| X \|^2 \quad \text{s.t.} \quad AX = B \]

  is given by

  \[ X = A^T (AA^T)^{-1} B = A^+ B \]

  - To see how this is so, solve the Lagrangian,

  \[ L = \| X \|^2 + \lambda^T (AX - B) \]
Character Animation

- **Singular Value Decomposition (Digression)**

  Any rectangular, real, $m \times n$ matrix $A$ can be decomposed as:
  
  $$A = U \Sigma V^T$$

  where $U$ is a $m \times m$ orthogonal matrix
  
  $\Sigma$ is a $m \times n$ diagonal matrix
  
  $V$ is a $n \times n$ orthogonal matrix

  - The pseudoinverse can be computed from the SVD as

  $$A^+ = V \Sigma^+ U^T$$

  where $\Sigma^+$ is formed by transposing $\Sigma$ and taking a reciprocal of every non-zero singular value
Character Animation

- Inverting the Jacobian

  - So we solve

  \[ d \Theta = J^{-1}(\Theta) \, d \, X \]

  - By computing the pseudoinverse of the Jacobian using the SVD

  \[ J^+ = V \sum_{i=1}^{r} \sigma_i^{-1} v_i u_i^T \]

  where \( r = m \) if the Jacobian has full row rank
Character Animation

- Construct the Jacobian
- Invert the Jacobian
- Iterate
- Detect singularities and avoid ill-conditioning
Character Animation

- Starting the IK solver and iterating

  - Initialize the linear velocity components of the end-effector to $dX_{1,3} = G - E$, compute the Jacobian, invert, update the joints parameters and iterate.

  - If $dX$ is too large then due to local nature of the solution, tracking errors will occur, given by $\|J(\Theta)\,d\Theta - dX\|$

  - So sometimes we go from $E$ to $G$ in by taking smaller steps in the $G - E$ direction.
Character Animation

- Starting the IK solver and iterating

- Iterate only a fixed number of times and check whether $\|G - E\|$ has fallen below some tolerance.

- Also check after a few iterations whether the arm has become fully extended, is parallel to $G - E$ and the goal is still out of reach.

- What about the angular velocity components of the end-effector?
Character Animation

- Construct the Jacobian
- Invert the Jacobian
- Iterate
- Detect singularities and avoid ill-conditioning
Character Animation

• Detect singularities and avoid ill-conditioning
  – The Jacobian becomes singular when it loses rank.
  – This is detectable using the condition number of the Jacobian matrix.

\[ C = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \]
Character Animation

- Detect singularities and avoid ill-conditioning
  - The solution also becomes ill-conditioned near a singularity, resulting in very high joint space velocities.

$\|d\theta_1\| \gg \|dX\|$
Character Animation

- Detect singularities and avoid ill-conditioning
  - To prevent ill-conditioning we damp the joint space velocities by searching for a solution that minimizes the sum:
    \[
    \| J (\Theta) d \Theta - dX \|^2 + \lambda^2 \| d \Theta \|^2
    \]
  - The solution to this is given by
    \[
    d \Theta = J^T (J J^T + \lambda^2 I)^{-1} dX
    \]
    \[
    = V \Sigma^T (\Sigma \Sigma^T + \lambda^2 I)^{-1} U^T dX
    \]
    \[
    = (\sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T) dX
    \]
- This is known as the damped least squares solution.