Character Animation

- 15.05.2008
- Inverse Kinematics: Inverse Jacobian + CCD
- Speaker: Fabian Lessle
What is kinematics?

- science of motion without regard to the masses or forces that bring out the motion ("how", not "why")
- Forward kinematics
- Inverse kinematics
Demo
Why IK?

- robotics (e.g. to adjust a tool to arriving parts on a conveyor belt)
- interactive computer graphics (e.g. automatic generation of key frames based on user/world input)
- assisting 3D animators (pure FK means lots of work)
How to IK?
analytical vs. numerical

- **Analytical methods:**
  - fast
  - precise
  - often impossible
  (- precise)

- **Numerical methods:**
  - not always fast
  - not always precise
  + practical for complex systems
  (+ close „solution“ better than no solution)
The Inverse Jacobian: What is a Jacobian? (I)

- Problem: How does angle configuration vector $\Theta$ correspond to end effector position vector $P$?
- Solution: $P = F(\Theta)$?

- Actual problem: How does a change in $\Theta$ correspond to a change in $P$?
- $dP = dF/d\Theta \times d\Theta$?
The Inverse Jacobian: What is a Jacobian? (II)

- $F$ is a function from $\mathbb{R}^m$ to $\mathbb{R}^n$, with $m$ being the system's degrees of freedom, $n$ being the end effector's (theoretical) DOF. E.g. in a 3D system $n$ would be 6 (3D position + 3D orientation), $m$ would be the number of joints (assuming 1 DOF per joint).

- The $nxm$ matrix of partial derivatives, $dF/d\Theta$, is called the Jacobian. The Jacobian can be thought of as a mapping of the velocities of joint angles to the end effector's velocity.
The Inverse Jacobian: What is a Jacobian? (III)

- $V = J(\Theta)\theta$

- $V ("Velocity") := P_d - P$ (Difference between desired and current end-effector position)

- $J(\Theta) :=$ Jacobian of current configuration $\Theta$

- $\Theta :=$ change in configuration (or "angle velocities")
The Inverse Jacobian: What is an example?
The Inverse Jacobian: What is an example?

- Joints: P0 (0,1)
  P1 (4,0)
  P2 (5,2)
- Rotation axis: (0,0,1) for all joints
- End effector: E (6,3), Goal: G (4,3)
- Jacobian:

\[
J = \begin{pmatrix}
(0,0,1) \times (E-P0)_x & (0,0,1) \times (E-P1)_x & (0,0,1) \times (E-P2)_x \\
(0,0,1) \times (E-P0)_y & (0,0,1) \times (E-P1)_y & (0,0,1) \times (E-P2)_y \\
(0,0,1) \times (E-P0)_z & (0,0,1) \times (E-P1)_z & (0,0,1) \times (E-P2)_z
\end{pmatrix}
\]

\[
= \begin{pmatrix}
-2 & -3 & -1 \\
6 & 2 & 1 \\
0 & 0 & 0
\end{pmatrix}
\]
The Inverse Jacobian: Why the example?

- The Example reminds us of an important fact: Most matrices are not invertible. To be invertible, the Jacobian would have to be square with a determinant $\neq 0$.
- Dimensions are (more or less) dictated by the system's number of joints.
- Values will change over time, so will the Jacobian's determinant.
- So how can we compute $\theta = J^{-1}v$?
The Inverse Jacobian: Actually **Pseudoinverse**...

- Properties of the pseudoinverse:
  (for real valued matrices)

  - $A A^+ A = A$
  - $A^+ A A^+ = A^+$
  - $(A A^+)^T = A A^+$
  - $(A^+ A)^T = A^+ A$
The Pseudoinverse Jacobian

- Instead of $J^{-1}$ compute the Moore-Penrose pseudoinverse $J^{+}$:
  1. Determine singular value decomposition $J = USV^{T}$
  2. P-invert $S$ to $S^{+}$ by taking the reciprocal of each non-zero element on the diagonal
  3. Then $J^{+} = VS^{+}U^{T}$
- Solve for $\theta$
- Increment $\theta$ by $\varepsilon \theta$
- Repeat all steps until done
Oh no, the singularity!
Problems of the pseudoinverse

- Example: No linear combination of angle velocities causes direct movement towards the goal location; only 1 DOF left
- Causes discontinuities in pseudoinverse
- Pseudoinverse tends to produce large joint velocities in vicinity of singular configurations, causing instability and wild oscillation
From Pseudoinverse to Pseudophysics

- **Virtual work:**
  \[ W = \text{force} \times \text{distance} \]
  \[ W = \text{torque} \times \text{angle} \]
- **Work must be equal:**
  \[ (1) \ F \ \Delta x = \tau \ \Delta q \]
  \[ (2) \ F^T \ \Delta x = \tau^T \ \Delta q \]
- **Forward kinematics:**
  \[ (3) \ \Delta x = J \ \Delta q \]
- **(3) into (2)**
  \[ F^T J \ \Delta q = \tau^T \ \Delta q \]
  \[ F^T J = \tau^T \]
  \[ \tau = J^T F \]
The Transposed Jacobian

- Instead of $J^+$ determine $J^T$
- Compute $\theta = K J^T v$
  where $K$ is a constant scaling matrix
- Increment $\Theta$ by $\epsilon\Theta$
- Repeat all steps until done

- $K$ is used to counter scaling problems.
  Welman suggests $K_i = 1 / (\omega_i a_i)$
  with $\omega_i$ proportional to the length of link $i$
Another method: Cyclic Coordinate Descend

- minimizing system error by adjusting each joint one at a time
- start at the last link, work backwards, repeat if necessary
Cyclic Coordinate Descend: Optimization Step (in 2D)

- EffGoal = Goal position
- EffCur = Current position of the end-effector
- $j_i$ = Current joint
- GoalV = normalize(EffGoal - $j_i$.position);
- CurV = normalize(EffCur - $j_i$.position);
- Angle = acos(CurV $\cdot$ GoalV);
- Direction = ((CurV $\times$ GoalV).z > 0 ? -1 : 1);
- $j_i$.rotation += Direction $\cdot$ Angle;
- i++;
CCD Example:
Done after 1 move

- The first example again.
  And this time it's actually being solved. :D
CCD Example:
Two iterations away
CCD Example: Barely reachable
Cyclic Coordinate Descend: Improvements

- easy to implement joint restrictions since each joint is handled as a single problem
- damping may be used to avoid very unnatural looking results
- constraints via penalty method
Jacobian vs. CCD: Comparison (I)

- **Transposed Jacobian:**
  + intuitive, smooth results
  - can be made quite stable around kinematic singularities
  - slow convergence

- **CCD:**
  - less intuitive results
  + complete immunity to singularities
  + better convergence
Jacobian vs. CCD: Comparison (II)

- Illustrating the „smooth results“ argument:
Jacobian vs. CCD: Comparison (III)
Jacobian vs. CCD: Comparison (IV)
Sources

- Welman (1993) Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation
- Wikipedia, the free encyclopedia