Kinematics

Thomas Funkhouser
Princeton University
COS 426, Spring 2004

Overview

- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics

Example: 2-Link Structure

- Two links connected by rotational joints

Forward Kinematics

- Animator specifies joint angles: \( \Theta_1 \) and \( \Theta_2 \)
- Computer finds positions of end-effector: \( X \)

\[
X = (l_1 \cos \Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin \Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))
\]

Forward Kinematics

- Joint motions can be specified by initial conditions and velocities

\[
\begin{align*}
\Theta_1(0) &= 60^\circ \\
\Theta_2(0) &= 250^\circ \\
\frac{d\Theta_1}{dt} &= 1.2 \\
\frac{d\Theta_2}{dt} &= -0.1
\end{align*}
\]

Forward Kinematics

- Joint motions can be specified by spline curves
**Example: 2-Link Structure**

- What if animator knows position of “end-effector”

```
\( \text{End-Effector} \)
```

\( X = (x, y) \)

\( \Theta_1, \Theta_2 \)

**Inverse Kinematics**

- Animator specifies end-effector positions: \( X \)
- Computer finds joint angles: \( \Theta_1 \) and \( \Theta_2 \):

\[
\Theta_2 = \cos^{-1}\left( \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \right)
\]

\[
\Theta_1 = \frac{-(l_2 \sin(\Theta_2)x + (l_1 + l_2 \cos(\Theta_2))y)}{(l_2 \sin(\Theta_2)y + (l_1 + l_2 \cos(\Theta_2))x)}
\]

**Inverse Kinematics**

- End-effector positions can be specified by spline curves

**Inverse Kinematics**

- Problem for more complex structures
  - System of equations is usually under-defined
  - Multiple solutions

Three unknowns: \( \Theta_1, \Theta_2, \Theta_3 \)

Two equations: \( x, y \)

**Inverse Kinematics**

- Solution for more complex structures:
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization

**Inverse Kinematics**

- “Ballboy”

Fujito, Milliron, Ngan, & Sanocki
Princeton University
**Summary of Kinematics**

- Forward kinematics
  - Specify conditions (joint angles)
  - Compute positions of end-effectors
- Inverse kinematics
  - "Goal-directed" motion
  - Specify goal positions of end effectors
  - Compute conditions required to achieve goals

Inverse kinematics provides easier specification for many animation tasks, but it is computationally more difficult.

**Overview**

- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics

**Dynamics**

- Simulation of physics insures realism of motion

**Spacetime Constraints**

- Computer finds the “best” physical motion satisfying constraints
- Example: particle with jet propulsion
  - \( x(t) \) is position of particle at time \( t \)
  - \( f(t) \) is force of jet propulsion at time \( t \)
  - Particle’s equation of motion is:
    \[
    m \ddot{x} - f - mg = 0
    \]
  - Suppose we want to move from \( a \) to \( b \) within \( t_0 \) to \( t_1 \) with minimum jet fuel:
    Minimize \( \int_{t_0}^{t_1} [f(t)]^2 dt \) subject to \( x(t_0) = a \) and \( x(t_1) = b \)

**Spacetime Constraints**

- Animator specifies constraints:
  - What the character’s physical structure is
    - e.g., articulated figure
  - What the character has to do
    - e.g., jump from here to there within time \( t \)
  - What other physical structures are present
    - e.g., floor to push off and land
  - How the motion should be performed
    - e.g., minimize energy

**Spacetime Constraints**

- Discretize time steps:
  \[
  x'_i = \frac{x_{i+1} - x_i}{h} \quad x''_i = \frac{x'_{i+1} - 2x'_i + x'_{i-1}}{h^2} \\
  m \left( x''_i = \frac{x_{i+1} - 2x'_i + x_{i-1}}{h^2} \right) - f_i - mg = 0
  \]
  Minimize \( h \sum_i [f_i]^2 \) subject to \( x_i = a \) and \( x_i = b \)
**Spacetime Constraints**

- Solve with iterative optimization methods

- Advantages:
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints

- Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

- Advantages:
  - Easy to vary motions due to new parameters and/or new constraints

- Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

- Adapting motion:

  - Original Jump
  - Heavier Base

- Adapting motion:

  - Hurdle

- Editing motion:

  - Ski Jump

- Editing motion:

  - Li et al. '99

- Easy to vary motions due to new parameters

- Adapting motion:

  - Free animator from having to specify details of physically realistic motion with spline curves

- Specifying constraints and objective functions

- Avoiding local minima during optimization
Spacetime Constraints

• Morphing motion:

Gliczer '98

Dynamics

• Other physical simulations:
  - Rigid bodies
  - Soft bodies
  - Cloth
  - Liquids
  - Gases
  - etc.

Hot Gases
(Foster & Metaxas '97)

Cloth
(Barsouf & Wikin '96)

Summary

• Kinematics
  - Forward kinematics
    » Animator specifies joints (hard)
    » Compute end-effectors (easy - assn 4!)
  - Inverse kinematics
    » Animator specifies end-effectors (easier)
    » Solve for joints (harder)

• Dynamics
  - Space-time constraints
    » Animator specifies structures & constraints (easiest)
    » Solve for motion (hardest)
  - Also other physical simulations

• Advantages:
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints

• Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization