Spacetime Optimization for Predictive Simulation of Motion in OpenSim

Karianne Bergen

Institute for Computational and Mathematical Engineering, Stanford University

November 21, 2013
OpenSim

“OpenSim is a powerful and freely available tool for modeling and simulation of movement” developed by the National Center for Simulation in Rehabilitation Research (NCSRR) under Principal Investigator Scott Delp.

“Musculoskeletal modeling and dynamic simulation have recently emerged as powerful tools to uncover the biomechanical causes of movement abnormalities and to design improved treatments.”

\(^1\)http://opensim.stanford.edu/
Predictive Simulation of Motion

- Predictive simulations using OpenSim are currently under development.

- **Goal**: to generate realistic motion trajectories for a human/character that achieve a particular task using numerical optimization and biomechanics modeling tools in OpenSim.

- “Realistic” motion in this context means that it must obey the laws of physics and be consistent with our understanding of biomechanics.
**Predictive Simulation of Motion**

- **Motivation:**
  this could provide a tool to simulate motions for new/modified tasks without new data.

- Possible applications include
  - Loaded activity
  - Performance-enhancing or assistive devices
  - Medical interventions (e.g. physical therapy, surgery)
Forward Dynamics Approach for Predictive Simulation

- Parameterize motion using a set of initial conditions and motion controllers (design variables).

- Objective function evaluation requires forward dynamics simulation (forward integration) to compute motion trajectories and torques.
  - Slow - computation for a single simulation is approximately real-time.
  - No gradients available and noise in objective from numerical integration.

- Gradient-free optimization (Covariance Matrix Adaptation)
Figure: Related work\(^2\) on simulation of human motion.

Inverse Dynamics Approach

- **Idea:** reformulate optimization problem to avoid using forward dynamics.
  - Inverse dynamics computes net forces and torques on a model at a specified time given its motion and the applied forces.
  - Inverse dynamics faster to compute.

- Formulation of problem that uses inverse dynamics must include a direct representation of the motion trajectory and applied forces.
  - *New approach:* model problem of motion simulation as a trajectory optimization problem using *spacetime method.*
Spacetime Method for Optimization

- Introduced by Witkin and Kass (1988) \(^3\)

- Basic idea: solve for motion trajectory and unknown forces over the entire time interval, rather than relying on forward dynamics to generate motion.

- Formulated as an optimization problem, this takes the form:

\[
\text{minimize } \left( \text{trajectory } q(t), \text{forces } F(t) \right) \quad \text{performance metric}
\]

subject to

\[
\text{physical laws, pose/task constraints}
\]

Simple Example: Spacetime Particle

- Consider a particle of mass $m$ with a jetpack that generates a time-varying force $f(t)$.

Problem: find the trajectory, from initial position $x(0) = a$ to target position $x(T) = b$, that minimizes fuel consumption.

- Unknowns: trajectory $x(t)$, and jet force $f(t)$, $0 \leq t \leq T$.
- Physical law: $m\ddot{x}(t) - f(t) - mg = 0$, $\forall t$.
- Total fuel consumption: $F = \int_{t=0}^{T} |f(t)|^2 dt$

---

4 from Witkin and Kass (1988)
Simple Example: spacetime particle (2)

- Spacetime optimization formulation:

\[
\begin{align*}
\text{minimize} & \quad \Delta t \sum_{i} |f_i|^2 \\
\text{subject to} & \quad m \left( \frac{\dot{x}_i - \dot{x}_{i-1}}{\Delta t} \right) - f_{i-1} - mg = 0, \quad i = 1, \ldots, n \\
& \quad \dot{x}_i = \frac{x_{i+1} - x_i}{\Delta t}, \quad i = 1, \ldots, n \\
& \quad \dot{x}_0 = 0 \\
& \quad x_0 = a, \quad x_n = b
\end{align*}
\]

- This a convex optimization problem, and therefore the global minimum can be computed efficiently (e.g. using cvx).
Simple Example: spacetime particle (3)

(A) original formulation;
obj value: 54.78

(B) rate limit
\[ |f_i| \leq f_{max}; \]
obj value: 67.84

(C) intermediate target point;
obj value: 64.78
**Jumping Luxo Lamp Problem**

- *Task*: use spacetime method to generate a physically realistic trajectory for a jumping Luxo lamp in OpenSim.

- Test problem, used in Witkin and Kass (1988) paper, is restricted to 2 dimensions and uses simple character with only 6 degrees of freedom.
Jumping Luxo Lamp Model

- generalized coordinates $q(t)$
  - lamp base and joint coordinates are denoted $q_b(t)$ and $q_j(t)$.

  $$ q(t) = \begin{bmatrix} q_b(t) \\ q_j(t) \end{bmatrix}^T $$

  $$ = \begin{bmatrix} x(t), y(t), \theta_z(t) \\ \theta_1(t), \theta_2(t), \theta_3(t) \end{bmatrix}^T $$

- contact forces $F(t; \mathcal{T})$, which are non-zero for $t \in \mathcal{T}$.
- residual forces $\tau(t)$
  - generalized forces not accounted for by inertial and applied forces.

  $$ \tau(t, q, \dot{q}, \ddot{q}) = M(q)\ddot{q} + C(q, \dot{q}) - (G(q) + F(t; \mathcal{T})) $$

where $M$ is the mass matrix, $C$ are the inertial forces (coriolis and gyroscopic forces), $G$ is the gravitational force.
Jumping Luxo Lamp Formulation

- Spacetime optimization formulation:

  \[
  \text{minimize} \quad \int_{t=0}^{T} f(q(t), \tau(t); \rho) \, dt \\
  \left(= \int_{t=0}^{T} \|\tau_j(t) - k (q_j(t) - q_j^{\text{rest}}(t))\|_2^2 dt \right)
  \]

  subject to \( \tau_b(t) = 0, \quad 0 \leq t \leq T \)

  \( q_b(t) = p(t), \quad t \in \mathcal{T} \)

  for performance metric \( f(\cdot) \), with lamp base positions \( p(t) \), time interval(s) \( \mathcal{T} \), and lamp model parameters \( \rho \) given.

- This problem is \textit{non-convex} and more difficult to solve than the spacetime particle problem.
Challenges: Model formulation

- Parameterization of the trajectory
  - Must represent continuous functions $q(t)$ (also $\dot{q}(t)$ and $\ddot{q}(t)$) with a finite set of parameters.
  - Initially used piece-wise linear approximation of $q(t)$ and approximate derivatives using finite differences.
  - Currently using cubic Bezier splines: require fewer control points to represent smooth motion, but are not well-suited for representing discontinuities (e.g. contact).

- Modeling contact forces
  - Currently solving for the magnitude/direction of a single contact force centered at lamp base, but the position and times of contact must be specified.
  - Contact forces are still represented by a piece-wise linear function due to sharp discontinuity.
Challenges: Model formulation (2)

- Constraints and objective should be realistic (i.e. represent physical laws and constraints).
  - Physical laws are modeled as (nonlinear) constraints - must be satisfied with equality.
  - Modeling of hard versus soft constraints: e.g. enforcing target joint angles less critical than avoiding ground penetration.
  - Balancing multiple objectives: e.g. limit work/effort expenditure, achieve desired task, and avoid falling over/tripping.
**Challenges: Computing the Solution**

- The problem is non-convex, and finding the global minimum of a non-convex problem is hard!

- Choice of optimization algorithm.
  - Available in OpenSim: L-BFGS, L-BFGS-B, IPOPT, and CFSQP
  - Mostly using L-BFGS with quadratic penalty terms to handle constraints - other algorithms may be better at handling constraints.
  - Also used IPOPT, but so far this seems prohibitively slow (computing numerical Jacobian by finite differences).

- Choosing an initial point for the algorithm.
  - Choice of initial point may determine which local minima the algorithm identifies as the solution.
  - Difficult to find a feasible initial point.
**Challenges specific to implementation in OpenSim**

- No analytic gradients.
  - Gradients/Jacobians computed by finite differences, Hessian approximation generated by BFGS update.
  - Finite differences are less accurate than analytic gradients.
  - Finite difference computation slows down the algorithms significantly (although this is parallelizable).
  - Adding gradient calculation / automatic differentiation requires significant changes to software.

- Limitations of optimization tools
  - Solvers provide very limited diagnostic information to user.
  - Reading and modifying these codes can be challenging.
Results: Jumping Luxo Lamp

Click here to open demo in YouTube.
Results: Jumping Luxo Lamp
Results: Convergence

- Slow convergence: for 200-300 variables can take $\approx 2.0-5.5 \times 10^4$ iterations to converge to a local minima.

- In most cases, algorithm terminates due to failure in line-search before converging to a local minima.
REFERENCES


