Iterative Dynamics with Temporal Coherence

Erin Catto
ecatto@crystald.com

Crystal Dynamics
Menlo Park, California

Game Developers Conference, 2005

Introduction

The Problem

Many rigid body physics algorithms are slow, use too much memory, are difficult to implement, or have other nasty limitations.

The Idea

- Use an approximate contact model that is easy to solve.
- Use a sloppy but fast constraint solver.
- Clean up the solution over several frames.

The Toolkit

- Contact point calculator.
- Rigid bodies, constraints, and Jacobians.
- Gauss-Seidel constraint solver and simple integrator.
- Contact cache.

High Level Algorithm

Time Stepping

1. Generate contact points.
2. Initialize contact forces λ using a contact cache (generated in the previous step).
3. Compute the Jacobian J for non-penetration and friction constraints.
4. Form an equation for λ.
5. Use a Gauss-Seidel solver to refine λ.
6. Compute new velocities v and ω using λ.
7. Compute new positions x and q from v and ω.
8. Store λ in the contact cache.
9. Go to step 1.

Constraints

Pairwise position constraint

$$C(x_i, q_i, x_j, q_j) = 0$$

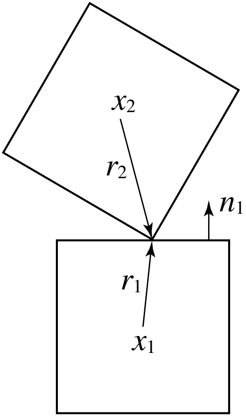
Velocity constraint and Jacobian

$$\frac{dC}{dt} = JV$$

The Recipe

1. Determine each constraint equation as a function of body positions and rotations.
2. Differentiate the constraint equation with respect to time.
3. Identify the coefficient matrix of V. This matrix is J.

Contact Constraint



Normal constraint

$$C_n = (x_2 + r_2 - x_1 - r_1) \cdot n_1$$

$$\frac{dC_n}{dt} = (v_2 + \omega_2 \times r_2 - v_1 - \omega_1 \times r_1) \cdot n_1$$

Friction constraint

$$\frac{dC_{u1}}{dt} = (v_2 + \omega_2 \times r_2 - v_1 - \omega_1 \times r_1) \cdot u_1$$

$$\frac{dC_{u2}}{dt} = (v_2 + \omega_2 \times r_2 - v_1 - \omega_1 \times r_1) \cdot u_2$$

Equations of Motion

Kinematics

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dq}{dt} &= \frac{1}{2} q * \omega \end{aligned}$$

Newton's Law for a system of constrained rigid bodies

$$M \frac{dV}{dt} = J^T \lambda + F_{ext}$$

$$JV = \zeta$$

Time Stepping

Approximate acceleration

$$\frac{dV}{dt} \approx \frac{V^2 - V^1}{\Delta t}$$

Eliminate V^2

$$JB\lambda = \eta$$

where $B = M^{-1}J^T$ and

$$\eta = \frac{1}{\Delta t}\zeta - J(\frac{1}{\Delta t}V^1 + M^{-1}F_{ext})$$

Constraint Forces

Contact and friction are simulated by bounding λ.

Normal force:

$$0 \leq \lambda_n < \infty$$

Approximate friction model:

$$-\mu m_c g \leq \lambda_{u1} \leq \mu m_c g$$

$$-\mu m_c g \leq \lambda_{u2} \leq \mu m_c g$$

In general:

$$\lambda_i^- \leq \lambda_i \leq \lambda_i^+$$

Gauss-Seidel

- An iterative method for solving linear equations.
- The basic algorithm: approximately solve $Ax = b$ given x^0.
- Iterate for a fixed number of steps.

$x = x^0$
for *iter* $= 1$ to iteration limit **do**
  for $i = 1$ to n **do**
    $\Delta x_i = \left[b_i - \sum_{j=1}^{n} A_{ij} x_j \right] / A_{ii}$
    $x_i = x_i + \Delta x_i$
  end for
end for

Projected Gauss-Seidel

- Solve $JB\lambda = \eta$ given λ^0.
- Clamp λ_i to its bounds.
- Use sparsity to avoid forming the s-by-s matrix JB.
- See the paper for details.

The Jacobian J is sparse because constraints are pairwise.

$$J_{sp} = \begin{pmatrix} J_{11} & J_{12} \\ \vdots & \vdots \\ J_{s1} & J_{s2} \end{pmatrix}$$

$$J_{map} = \begin{pmatrix} b_{11} & b_{12} \\ \vdots & \vdots \\ b_{s1} & b_{s2} \end{pmatrix}$$

Temporal Coherence

Why?

- When things move fast, sloppiness is okay.
- When things settle down, jiggle looks bad.
- Gauss-Seidel is iterative. It needs a good starting guess to be accurate.

Issues

- Contact points appear and disappear.
- How can contact points persist?
- There is too much stuff to keep track of (n^2 pairs).

Contact Caching

The Idea

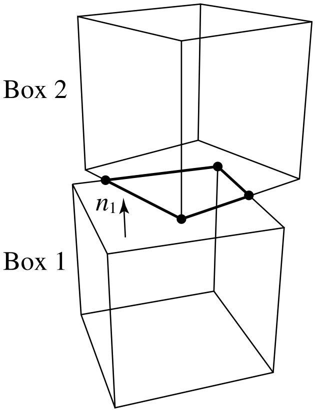
- Build a contact cache at the end of each time step.
- Rediscover all the contact points at the beginning of the next time step.
- Try to match the the new points to the cached points.
- If there is a cache hit then $\lambda = \lambda_{cache}$, else $\lambda = 0$.

Matching Points

1. Compare global coordinates of the points
2. Compare local coordinates of the points.
3. Compare contact identifiers.

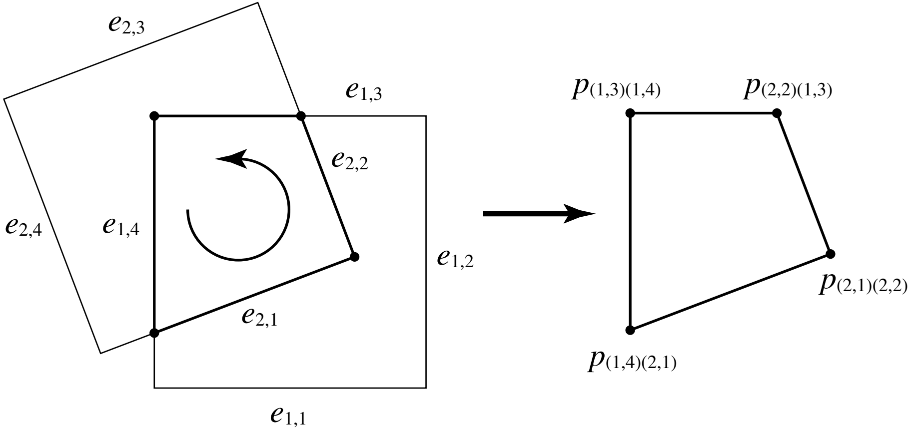
Contact Identifiers

Typical contact configuration:

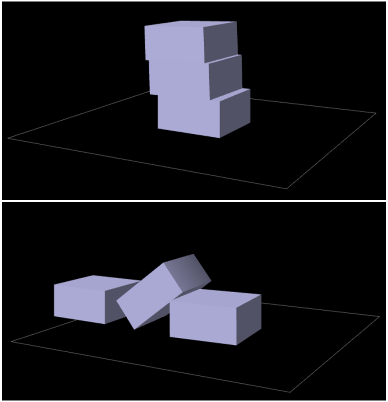


Edge Labels

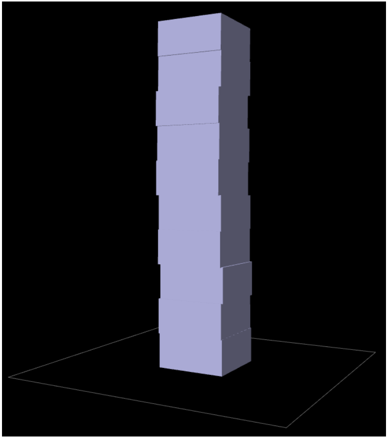
Clipping leads to contact identifiers.



Box Stacking



Without Caching



With Caching