Physics-based Animation

lionel.reveret@inria.fr

2013-14

Basic Concepts

- Particles dynamics
- Differential Equations Solver
- Constraints
- Rigid body dynamics
- Collisions

$\Rightarrow$ see Pixar Courses (Baraff & Witkin)

http://www.pixar.com/companyinfo/research/pbm2001

Constraints

- Constraints forces (general case)

$\ddot{q}=W(Q+Q_C)$

$C(q)=0 \quad \Rightarrow \dot{C} = J\dot{q} = 0 \quad \Rightarrow \quad \ddot{C}=J\ddot{q} + \dot{J}\dot{q}=0$

$\Rightarrow JW(Q+Q_C) + \dot{J}\dot{q} = 0$

$\Rightarrow JWQ_C = -\dot{J}\dot{q} - JWQ$

Q_C doesn't work (no energy introduced) : $Q_C{}^t\,\dot{q}=0$

$\Rightarrow Q_C = J^t\,\lambda$

$\Rightarrow JWJ^t\,\lambda = -\dot{J}\dot{q} - JWQ \;\; -(k_sC + k_d\dot{C})$

Solving for λ provides Qc, new forces to integrate

Constraints

- Alternative formulation

$\ddot{C}=0 \Rightarrow J\ddot{q} = - \dot{J}\dot{q} + \text{(damping)}$

$M\ddot{q} = Q + J^t \lambda$

$$\begin{pmatrix} M & -J^t \\ J & 0 \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} Q \\ b \end{pmatrix}$$

Sparse matrix

Reduced coordinates

- Use a more appropriate set of explicit parameters instead of implicit constraints

$q = q(u) \quad \Rightarrow \quad \dot{q} = J\dot{u} \Rightarrow \ddot{q} = \dot{J}\dot{u} + J\ddot{u}$

$M\ddot{q} = Q + Q_c \quad$ and $Q_c.\dot{q} = 0$ (no work)

applying J^t makes Q_c disappears:

$(J^tMJ)\,\ddot{u} + (J^tMJ)\,\dot{u} - J^t\,Q = 0$

Lagrangian Mechanics

- Reduced coordinates (vs Maximal Coordinates + constraints)

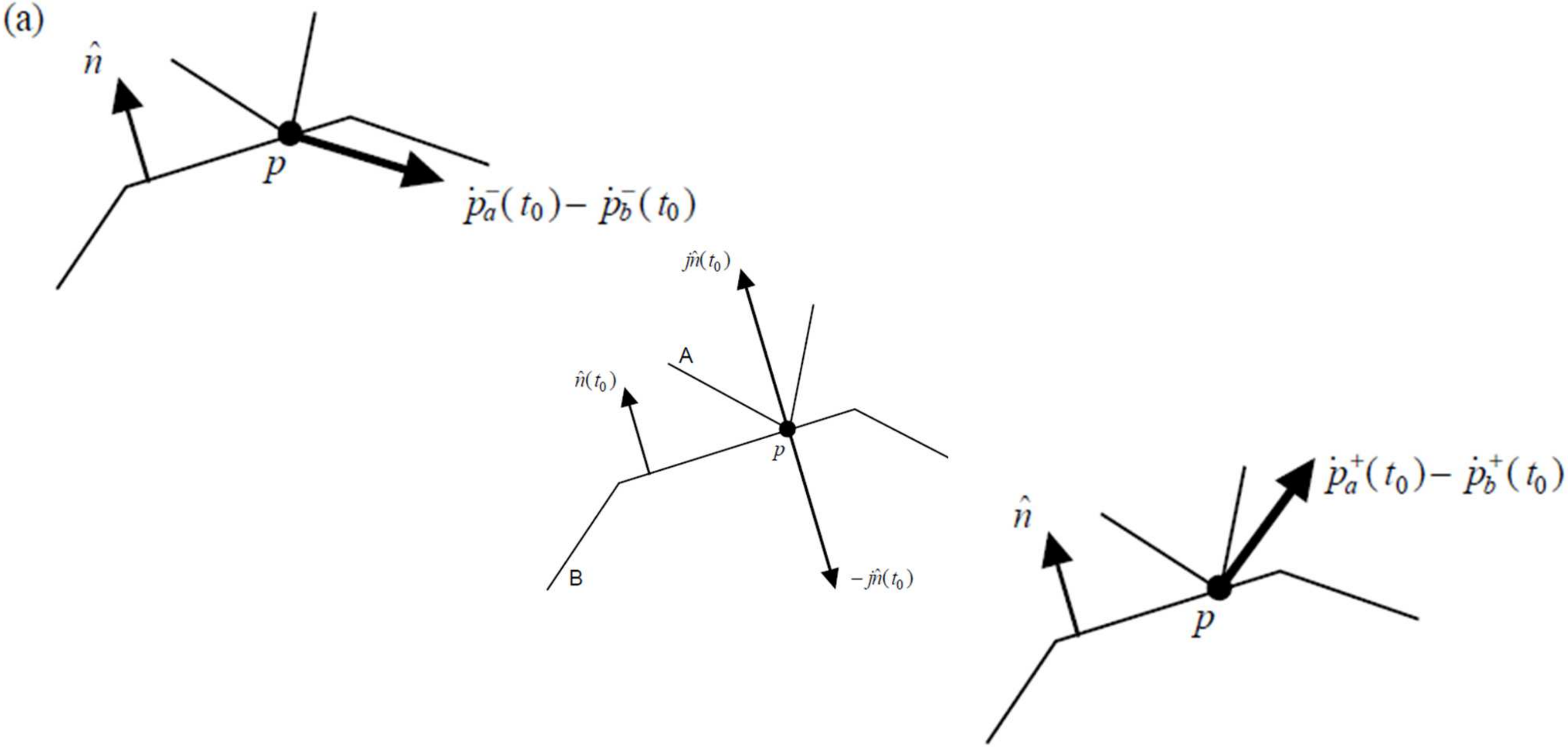
$$\mathcal{L} = T - V.$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathcal{L}}{\partial q_j} = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial V}{\partial q_j} = 0.$$

T : kinetic energy
V : potential from conservative forces
qj : any motion parameter (position, angle, etc),
Note: one equation for each parameter

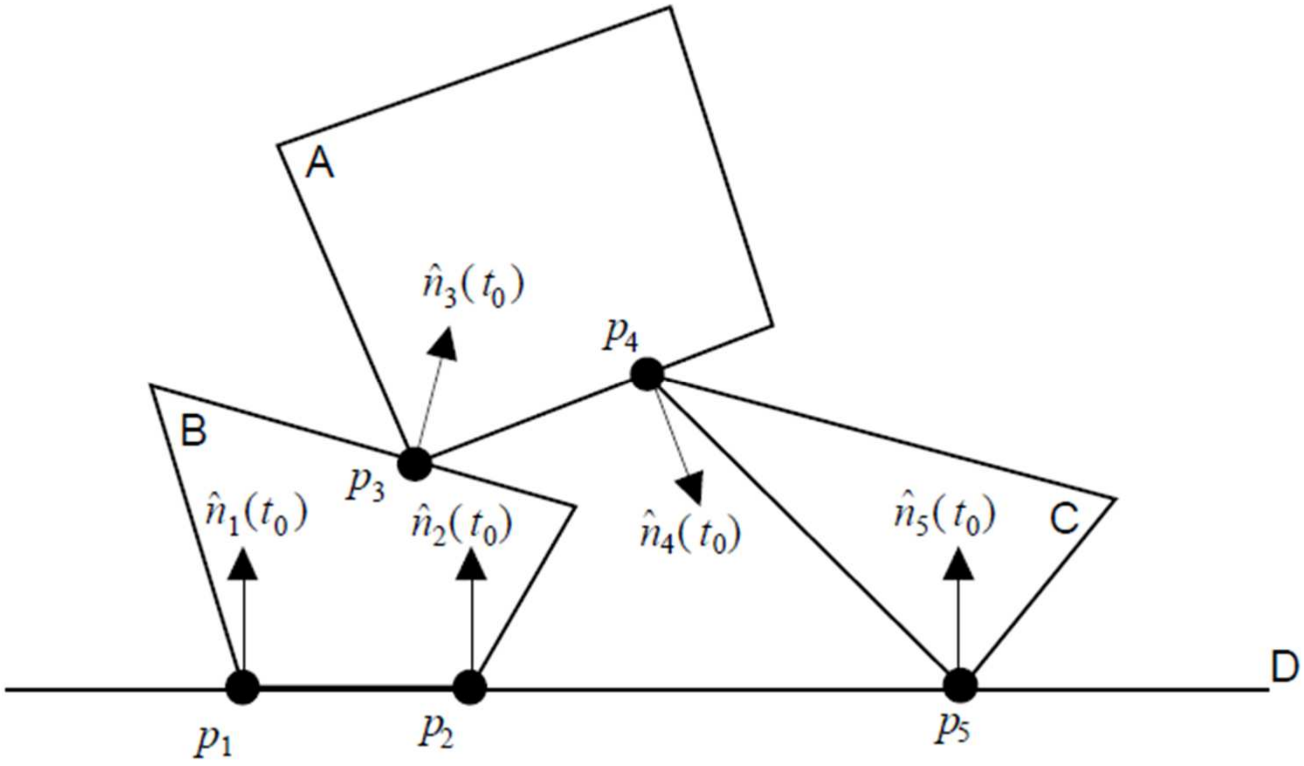
For a rigid body under gravity field only:
$T = ½\, v^t\, M\, v + ½\, \omega^t\, I\, \omega$ and $V = mgh$

Collision contacts



$v^{+}_{rel} = - \varepsilon\, v^{-}_{rel}$ => solves for j => update velocities, restart solver

Resting contact



Goal : solve for f_i

Resting contacts

- Three conditions to satisfy

1. No inter-penetration between objects

   - Forces are strong enough to push objects apart

2. Forces must not stick objects together

   - No extra forces if objects separates (no « glue »)

3. Forces must be 0 if contact breaks

Resting contact

- Relative distance

  $d_i(t) = n_i(t) \cdot [\, p_a(t) - p_b(t) \,]$

  $\ddot{d}_i(t_c) = n_i(t_c) \cdot [\, \ddot{p}_a(t_c) - \ddot{p}_b(t_c) \,] + 2\, \dot{n}_i(t_c) \cdot [\, \dot{p}_a(t_c) - \dot{p}_b(t_c) \,]$

- Condition 1 is $\ddot{d}_i(t_c) \geq 0$

  It can be shown that $\ddot{d}_i(t_c) = A_i\, f + b_i \geq 0$
- Condition 2 is $f_i \geq 0$
- Condition3 is $f_i \cdot \ddot{d}_i\, (t_c) = 0$

$\Rightarrow$solved with a Quadratic Programming solver (QP)

Character animation

- Skeleton
  - Set of rigid bodies => limbs
  - Connected together => joints

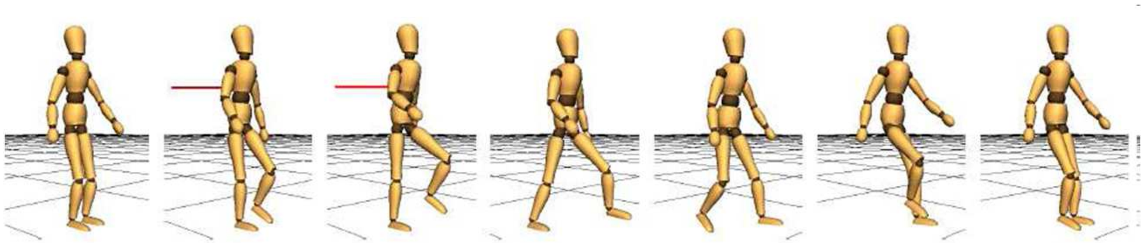
- Two approaches
  - Maximal coordinates (Newton)
    - ☺ • limbs : independent, 6D inertial frame space coordinates
    - ☹ • joints : set of constraints
  - Reduced coordinates (Lagrange)
    - ☺ • limbs : local joint orientations
    - ☺ • joints : no need for constraints
    - ☹ • complex derivatives

Dynamics controllers

- Compute external torques

  Proportional Derivative (PD Controller) + FSM

$$\tau = k_p (\theta_d - \theta) - k_v \dot{\theta}$$



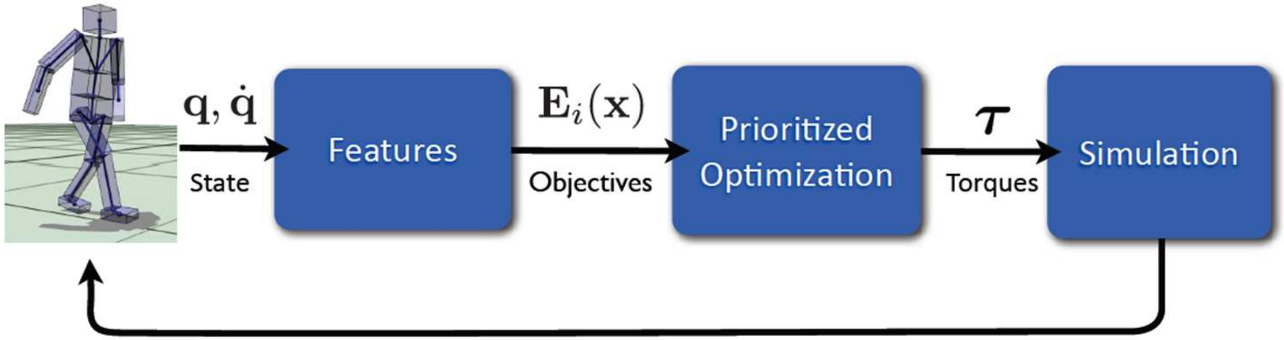
available code

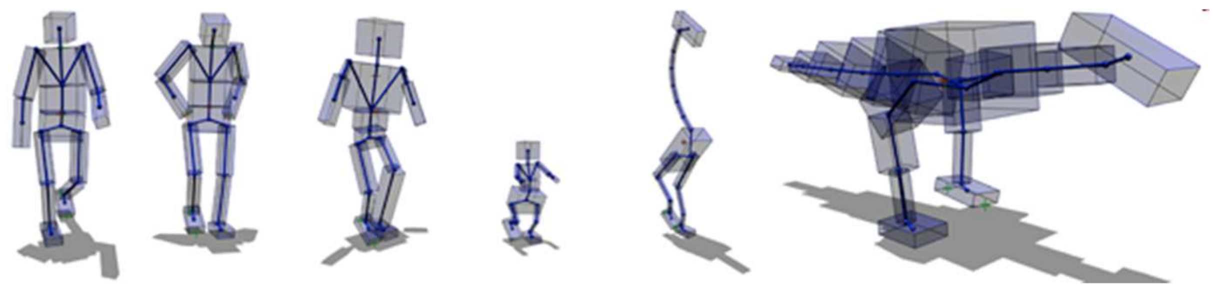
SIMBICON: Simple Biped Locomotion Control, KangKang Yin Kevin Loken, Michiel van de Panne , SIGGRAPH 2007.

http://www.cs.ubc.ca/~van/papers/Simbicon.htm

Dynamics controllers

- Compute external torques through optimization





Martin de Lasa, Igor Mordatch, Aaron Hertzmann, Feature-Based Locomotion Controllers, SIGGRAPH 2010

Fluids

- Navier-Stokes formula

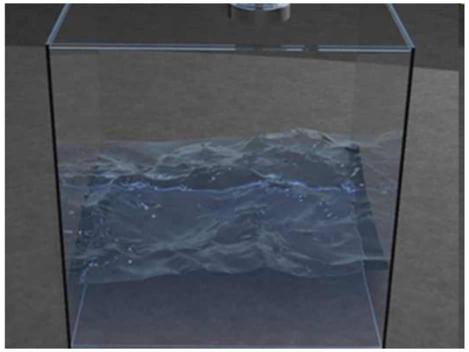
$$
\begin{aligned}
\nabla \cdot \mathbf{u} &= 0 \\
\frac{\partial \mathbf{u}}{\partial t} &= -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f},
\end{aligned}
$$

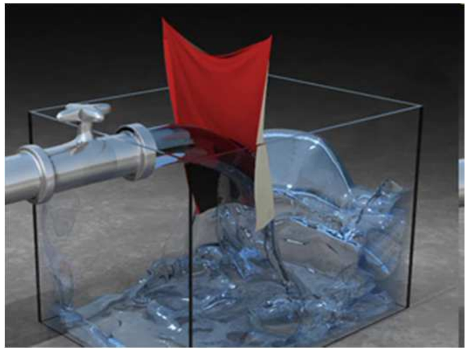
u : velocity field of the fluid (grid)
p : pressure field
ν : viscosity
f : external forces

Mass and Momentum are conserved

Fluids

- From fine grain modeling…





http://physbam.stanford.edu/~fedkiw/

- … to a more phenomenological approach

http://www-evasion.imag.fr/Membres/Fabrice.Neyret/

Fluids

- Available code
  www.dgp.toronto.edu/~stam/reality/Research/pub.html

« Stable Fluids », Jos Stam, SIGGRAPH'99

« Real-Time Fluid Dynamics for Games », Jos Stam, Proceedings of the Game Developer Conference, March 2003

Physics-based Animation

- How to start ?
  - Write your own engine (physics + solver)
    - especially for fluids or large collision problems

  - Start from an existing engine
    - especially for character animation and work on controller, but you may want to tune your own solver
    - SDK
      - PhysX (NVidia), www.nividia.com
      - Havok (Intel), www.havok.com
    - Open source
      - ODE (Open Dynamics Engine), www.ode.org
      - Bullet, bulletphysics.org
    - A comparative paper:
      - LocoTest: Deploying and Evaluating Physics-based Locomotion on Multiple Simulation Platforms, Stevie Giovanni and KangKang Yin. Lecture Notes in Computer Science, volume 7060 (Proc. Motion in Games 2011), pp. 227–241, Springer-Verlag Berlin Heidelberg.