T6: Position-Based Simulation Methods in Computer Graphics

Jan Bender   Miles Macklin   Matthias Müller
Jan Bender

• Organizer

• Professor at the Visual Computing Institute at Aachen University

• Research topics
  – Rigid bodies, deformable solids, fluids
  – Collision detection, fracture, real-time visualization
  – Position based methods

• Maintains open source PBD code base
  – github.com/InteractiveComputerGraphics/PositionBasedDynamics
Miles Macklin

• Principal engineer at NVIDIA
• Inventor and author of FLEX
  – Unified, particle based, position based solver, GPU accelerated
  – UE4 integration
  – developer.nvidia.com/flex
• Research
  – Position based fluids
  – Inventor of XPBD, making PBD truly physical with a simple trick!
Matthias Müller

• Leader of physics research group at NVIDIA
• Co-initiator of PBD (with Thomas Jakobsen)
• Co-founder of NovodeX which became physics group at NVIDIA
• Research
  – Co-rotational FEM, SPH
  – Position based methods: cloth, soft bodies, shape matching, oriented particles, air meshes
• www.matthiasmueller.info
Tutorial Outline

• Matthias
  – Motivation, Basic Idea
  – The solver
  – Constraint examples for solids
  – Solver accelerations

• Miles
  – Fluids
  – XPBD
  – Continuous materials
  – Rigid bodies
Motivation
Physical Simulations

- Well studied problem in the computational sciences (since 1940s)
- Complement / replace real experiments
- Extreme conditions, spatial scale, time scale
- **Accuracy most important factor**
- Low accuracy – useless result
Computer Graphics

• Early 1980s

• Adopted methods: FEM, SPH, grid based fluids, ..

• Applications
  – Special effects in movies and commercials
  – Computer games
  – VR

• Requirements
  – Speed, stability, controllability
  – Only visual plausibility

• New methods needed: e.g. PBD
Funhouse
Traditional Methods

• Typically force based

• Explicit integration
  – Simple and fast
  – Only conditionally stable (bad for real time apps)

• Implicit integration
  – Expensive (multiple linearizations and solves per time step)
  – Numerical damping
Basic Idea
Force Based Update

- Reaction lag
- Small spring stiffness $\rightarrow$ squashy system
- Large spring stiffness $\rightarrow$ stiff system, overshooting

Diagram:
- Penetration causes forces
- Forces change velocities
- Velocities change positions
Position Based Update

- Controlled position change
- Only as much as needed → no overshooting
- Velocity update needed to get 2\textsuperscript{nd} order system!
Position Based Integration

\[
\text{init } x_0, v_0 \\
\text{loop} \\
\quad v_n \leftarrow v_n + \Delta t \cdot f_{ext}(x_n) \quad \text{velocity update} \\
\quad p \leftarrow x_n + \Delta t \cdot v_n \quad \text{prediction} \\
\quad x_{n+1} \leftarrow \text{modify } p \quad \text{position correction} \\
\quad u \leftarrow (x_{n+1} - x_n)/\Delta t \quad \text{velocity update} \\
\quad v_{n+1} \leftarrow \text{modify } u \quad \text{velocity correction} \\
\text{end loop}
\]
Position Correction

• Example: Particle on circle
Velocity Correction

• External forces: $v_{n+1} = u + \Delta t \frac{g}{m}$
• Internal damping
• Friction
• Restitution
Distance Constraint

\[ \Delta x_1 = -\frac{w_1}{w_1 + w_2} (|x_1 - x_2| - l_0) \frac{x_1 - x_2}{|x_1 - x_2|} \]

\[ \Delta x_2 = +\frac{w_2}{w_1 + w_2} (|x_1 - x_2| - l_0) \frac{x_1 - x_2}{|x_1 - x_2|} \]

- Conservation of momentum
- Stiffness: scale corrections by \( k \in [0,1] \)
  - Easy to tune
  - Effect dependent on time step size and iteration count
  - Fixed! See XPBD

\[ w_i = \frac{1}{m_i} \]
General Internal Constraint

- Define constraint via scalar function:
  \[ C_{\text{stretch}}(x_1, x_2) = |x_1 - x_2| - l_0 \]
  \[ C_{\text{volume}}(x_1, x_2, x_3, x_4) = [(x_2 - x_1) \times (x_3 - x_1)] \cdot (x_4 - x_1) - 6v_0 \]

- Find configuration for which \( C = 0 \)
- Search along \( \nabla C \)
Constraint Projection

\[ C(x + \Delta x) = 0 \]

- Linearization (equal for distance constraint)
  \[ C(x + \Delta x) \approx C(x) + \nabla C(x)^T \Delta x = 0 \]

- Correction vectors

\[
\Delta x = \lambda \nabla C(x) \\
\lambda = -\frac{C(x)}{\nabla C(x)^T \nabla C(x)}
\]

\[
\Delta x = \lambda M^{-1} \nabla C(x) \\
\lambda = -\frac{C(x)}{\nabla C(x)^T M^{-1} \nabla C(x)}
\]

\[ M = \text{diag}(m_1, m_2, \ldots, m_n) \]
The Solver
Constraint Solver

• Gauss-Seidel
  – Iterate through all constraints and apply projection
  – Perform multiple iterations
  – Simple to implement

• Modified Jacobi
  – Process all constraints in parallel
  – Accumulate corrections
  – After each iteration, average corrections [Bridson et al., 2002]

• Both known for slow convergence
Global Solver

- Constraint vector

\[
C(x) = \begin{bmatrix}
C_1(x) \\
\vdots \\
C_M(x)
\end{bmatrix} \quad \nabla C(x) = \begin{bmatrix}
\nabla C_1(x)^T \\
\vdots \\
\nabla C_M(x)^T
\end{bmatrix} \quad \lambda = \begin{bmatrix}
\lambda_1 \\
\vdots \\
\lambda_M
\end{bmatrix}
\]

\[
\Delta x = M^{-1} \nabla C(x) \lambda
\]

\[
\lambda = -\frac{C(x)}{\nabla C(x)^T M^{-1} \nabla C(x)}
\]

\[
\Delta x = M^{-1} \nabla C(x)^T \lambda
\]

\[
[\nabla C(x) M^{-1} \nabla C(x)^T] \lambda = -C(x)
\]

[Goldenthal et al., 2007]
Global vs. Gauss-Seidel

- Gradients fixed
- Linear solution ≠ true solution
- Multiple Newton steps necessary
- Current gradients at each constraint projection
- Solver converges to the true solution
Other Speedup Tricks

• Use as smoother in a multi-grid method
• Long range distance constraints (LRA)
• Hierarchy of meshes
• Shape matching

→ more details later
Powerful Gauss-Seidel

• Can handle inequality constraints trivially (LCPs, QPs)!
  – Fluids: separating boundary conditions [Chentanez et al., 2012]
  – Rigid bodies: LCP solver [Tonge et al., 2012]
  – Deformable objects: Long range attachments [Kim et al., 2012]

• Works on non-linear problem directly
• Handles under and over-constrained problems
• GS + PBD: garbage in, simulation out (almost 😊)
• Fine grained interleaved solver trivial
• Easy to implement and parallelize
Constraint Examples
\[ C_{\text{bending}}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos\left( \frac{\mathbf{n}_1 \cdot \mathbf{n}_2}{\|\mathbf{n}_1\| \|\mathbf{n}_2\|} \right) \times (\mathbf{x}_4 - \mathbf{x}_1) - \varphi_0 \]

- More expensive than constraint \( C_{\text{stretch}}(\mathbf{x}_3, \mathbf{x}_4) \)
- But: Orthogonal to stretching
Stretching – Bending Independence

bending resistance

stretching resistance
Triangle Collision

\[ C_{coll}(x_1, x_2, x_3, x_4) = (q - x_1) \cdot \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|} - h \]
Cloth Example

King of Wushu
\[ C_{\text{air}}(x_1, x_2, x_3, x_4) = \det [x_2 - x_1, x_3 - x_1, x_4 - x_1] - 6V_0 \]
Soft Body Example
Global Volume - Balloons

\[ C_{balloon}(x_1, \ldots, x_N) = \]

\[
\frac{1}{6} \left( \sum_{i=1}^{n_{\text{triangles}}} \left( x_{t_1}^i \times x_{t_2}^i \right) \cdot x_{t_3}^i \right) - k_{\text{pressure}} V_0
\]
Air Meshes

• Triangulate air
• Prevent volume from inverting

• Add one **unilateral** constraint per cell:

\[
C_{air}(x_1, x_2, x_3) = |(x_2 - x_1) \times (x_3 - x_1)| \geq 0
\]
Locking

- Elements can invert without collisions
- Solution: Mesh optimization (edge flips)
2D Boxes
Boxes Recovery
3D Air Meshes

• Per tetra unilateral constraint:

\[ C_{air}(x_1, x_2, x_3) = \det[x_2 - x_1, x_3 - x_1, x_4 - x_1] \geq 0 \]

• Mesh optimization more expensive!
3D Air Meshes

• Two cases that work well without optimization

• Multi-layered clothing  
  • Tissue collision

• No large relative translations / rotations
Multi-Layered Clothing
Untangling
High Resolution Air Mesh
Tissue Collision Handling
Position Based Fluids

- Particle based
- Pair-wise lower distance constraints → granular behavior
- Move particles in local neighborhood such that density = rest density
- Density constraint

\[ C(x_1, \ldots, x_n) = \rho_{SPH}(x_1, \ldots, x_n) - \rho_0 \]

[Macklin et al. 2013]
Position Based Fluids
Shape Matching

• Optimally match rest with deformed shape
• Only allow translation and rotation

\[
\Delta x_i
\]

• Global correction, no propagation needed
• No mesh needed!
2d Demo
Optimal Translation

- Given rest positions \( \bar{X}_i \), current positions \( x_i \) and masses \( m_i \)

- Compute

\[
\begin{align*}
\bar{c} &= \frac{1}{M} \sum_i m_i \bar{x}_i \\
M &= \sum_i m_i \\
c &= \frac{1}{M} \sum_i m_i x_i
\end{align*}
\]

\[ t = c - \bar{c} \]
Optimal Transformation

• The optimal linear transformation is:

\[
A = \left( \sum_i m_i \mathbf{r}_i \tilde{\mathbf{r}}_i^T \right) \left( \sum_i m_i \tilde{\mathbf{r}}_i \tilde{\mathbf{r}}_i^T \right)^{-1} \\
= A_r A_s
\]

\[
\tilde{\mathbf{r}}_i = \bar{x}_i - \bar{c} \\
\mathbf{r}_i = x_i - c
\]
Optimal Rotation

\[
A = \left( \sum_i m_i \mathbf{r}_i \mathbf{r}_i^T \right) \left( \sum_i m_i \mathbf{\tilde{r}}_i \mathbf{\tilde{r}}_i^T \right)^{-1} = A_r A_s
\]

- \( A_s \) is symmetric \( \rightarrow \) contains no rotation
- Extract rotational part of \( A_r \)
- Polar decomposition

\( \mathbf{c} \)
Region Based Shape Matching

- Shape matching allows only small deviations from the rest shape.
- Performing shape matching on several overlapping regions.
- Each particle is part of multiple regions.
Fast Summation

- Compute prefix sum
On Irregular Mesh
Oriented Particles

• For co-linear, co-planar or isolated particles optimal transformation is not unique → Numerical instabilities

• Add orientation information to particles!
Oriented Particles

• Orientation information can be used
  – to stabilize simulation
  – to position anisotropic collision shapes
  – for robust skinning of visual mesh
Generalized Shape Matching

- Optimal translation is still \( t = \bar{c} - c \)
- Small modification in the calculation of \( A_r \)

\[
A_r = \left( \sum_i m_i r_i \bar{r}_i^T + A_i \right)
\]

where \( A_i^{\text{sphere}} = \frac{1}{5} mr^2 R \) and \( R \) the particle's rotation matrix
Oriented Particles Demo
Large Elasto-Plastic Deformation

• Handle splits, merges, large deformations
• Use explicit surface mesh to define object
  – Explicit surface tracking for merges and splits
  – Move with particles using linear blend skinning
• Dynamically add and remove particles
  – Remove particles outside surface, resample under-sampled regions
• Dynamically update clusters
  – Control cluster sizes
Doug Simulation
Solver Accelerations
Hierarchical PBD

- Create hierarchical mesh

- Next coarser mesh:
  - Subset of vertices
  - Each fine vertex is connected to at least $k$ coarse vertices
Hierarchical Constraints

• Constraints on coarse meshes

• Unilateral, upper bounds!
Hierarchical Solver

• Solve coarse → fine
• Interpolate displacements from next coarser level
Hierarchical PBD
Wrinkle Meshes

- 4 constraint types, geometric projection
Wrinkle Meshes

Simulated base mesh
2K triangles
Long Range Attachments (LRA)

- Very often cloth is attached (curtain, flags, clothing)
- Upper distance constraint to closest attachment point
- Only radial stretch resistance
Long Range Attachments (LRA)

[Kim et al., 2012], 90k particles
Follow The Leader (FTL)

- From top to bottom
- Only move lower particle
- All constraints satisfied!
Follow The Leader (FTL)

• Momentum not conserved!
Dynamic Follow The Leader (DFTL)

- Update positions one-sided
- Update velocities symmetrically