Meshless Approximation Methods and Applications in Physics Based Modeling and Animation

Bart Adams  Martin Wicke
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares

- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- Conclusions
Part I: Meshless Approximation Methods
Meshless Approximations

Approximate a function from discrete samples

1D

2D, 3D
Meshless Approximation Methods

Smoothed Particle Hydrodynamics (SPH)
- simple, efficient, no consistency guarantee
- popular in CG for fluid simulation

Meshfree Moving Least Squares (MLS)
- a little more involved, consistency guarantees
- popular in CG for elasto-plastic solid simulation
Meshless Approximation Methods

Fluid simulation using SPH

Elastic solid simulation using MLS
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares

- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- Conclusions
Smoothed Particle Hydrodynamics
Smoothed Particle Hydrodynamics (SPH)

Integral representation of a scalar function $f$

$$f(x) = \int f(y) \delta(x - y) dy$$

Dirac delta function

$$\delta(x - y) = \begin{cases} \infty & x = y \\ 0 & x \neq y \end{cases}$$
Smoothed Particle Hydrodynamics (SPH)

Replace Dirac by a smooth function \( w \)

\[
f(x) \approx \int f(y)w(\|x - y\|/h)\,dy
\]

Desirable properties of \( w \)

1. compactness: \( w(\|x - y\|/h) = 0 \) when \( \|x - y\|/h > 1 \)
2. delta function property: \( \lim_{h \to 0} w(\|x - y\|/h) = \delta(x - y) \)
3. unity condition (set \( f \) to 1): \( \int w(\|x - y\|/h)\,dy = 1 \)
4. smoothness
Smoothed Particle Hydrodynamics (SPH)

Example: designing a smoothing kernel in 2D

For simplicity set $h = 1, \|x - y\| = r$

We pick $w(r) = \begin{cases} A(1 - r)^3 & r < 1 \\ 0 & r \geq 1 \end{cases}$

Satisfy the unity constraint

$$\int_0^{2\pi} \int_0^1 A(1 - r)^3 r dr d\theta = 1 \iff A = \frac{10}{\pi}$$
Smoothed Particle Hydrodynamics (SPH)

Particle approximation by discretization

\[ f(x) \approx \int f(y) w\left(\frac{\|x - y\|}{h}\right) dy \]

\[ \Downarrow \]

\[ f(x) \approx \sum_{i=1}^{N} f_i w\left(\frac{\|x - x_i\|}{h_i}\right) V_i \]

\[ \Uparrow \]

\[ f(x) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f_i w\left(\frac{\|x - x_i\|}{h_i}\right) \]
Smoothed Particle Hydrodynamics (SPH)

Example: density evaluation

\[ f(x) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f_i w(\|x - x_i\|/h_i) \]

\[ \rho(x) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} \rho_i w(\|x - x_i\|/h_i) \]

\[ \rho(x) \approx \sum_{i=1}^{N} m_i w(\|x - x_i\|/h_i) \]
Smoothed Particle Hydrodynamics (SPH)

\[ f(x) = \sum_{i=1}^{N} \Phi_i(x) f_i \]

\[ \Phi_i(x) = \frac{m_i}{\rho_i} w(\|x - x_i\|/h_i) \]
Smoothed Particle Hydrodynamics (SPH)

Derivatives

\[ f(x) \approx \int f(y)w(\|x - y\|/h)dy \]

\[ \nabla_x f(x) \approx \nabla_x \int f(y)w(\|x - y\|/h)dy \]

\[ \nabla, \int \text{ linear, product rule} \]

\[ \nabla_x f(x) \approx \int \nabla_x f(x)w(\|x - y\|/h)dy + \int f(y)\nabla_x w(\|x - y\|/h)dy \]

\[ \nabla_x f(y) = 0 \]

\[ \nabla_x f(x) \approx \int f(y)\nabla_x w(\|x - y\|/h)dy \]
Smoothed Particle Hydrodynamics (SPH)

Particle approximation for the derivative

\[ \nabla f(x) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f_i \nabla w(\|x - x_i\|/h_i) \]

Some properties:
- simple averaging of function values
- only need to be able to differentiate \( w \)
- gradient of constant function not necessarily 0
  - will fix this later
Smoothed Particle Hydrodynamics (SPH)

Example: gradient of our smoothing kernel

We have \( w(r) = \begin{cases} \frac{10}{\pi}(1 - r)^3 & r < 1 \\ 0 & r \geq 1 \end{cases} \)

with \( r = \|x - y\|, h = 1 \)

Gradient using product rule:

\[
\nabla_x w = \frac{\partial w}{\partial r} \cdot \nabla_x r = -\frac{30}{\pi}(1 - r)^2 \cdot \frac{x - y}{\|x - y\|}
\]
Smoothed Particle Hydrodynamics (SPH)

Alternative derivative formulation

Old gradient formula: \( \nabla f(x_i) \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \nabla w(\|x_i - x_j\|/h_j) \) \hspace{1cm} (1)

Product rule: \( \nabla (\rho f) = f \nabla \rho + \rho \nabla f \Rightarrow \nabla f = \frac{1}{\rho} \nabla (\rho f) - \frac{1}{\rho} f \nabla \rho \) \hspace{1cm} (2)

Use (1) in (2):

\[
\nabla f(x_i) \approx \frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j f_j \nabla w(\|x_i - x_j\|/h_j) - \frac{1}{\rho_i} f_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j \nabla w(\|x_i - x_j\|/h_j)
\]

\[
\nabla f(x_i) \approx \frac{1}{\rho_i} \sum_{j=1}^{N} m_j (f_j - f_i) \nabla w(\|x_i - x_j\|/h_j)
\]

Gradient of constant function now always 0.
Similarly, starting from

\[
\nabla \left( \frac{f}{\rho} \right) = -\frac{f}{\rho^2} \nabla \rho + \frac{1}{\rho} \nabla f \quad \Leftrightarrow \quad \nabla f = \rho \left( \nabla \left( \frac{f}{\rho} \right) + \frac{f}{\rho^2} \nabla \rho \right)
\]

\[
\nabla (\frac{f}{\rho}) = -\frac{f}{\rho^2} \nabla \rho + \frac{1}{\rho} \nabla f \quad \Leftrightarrow \quad \nabla f = \rho \left( \nabla \left( \frac{f}{\rho} \right) + \frac{f}{\rho^2} \nabla \rho \right)
\]

\[
\nabla f(x_i) \approx \rho_i \sum_{j=1}^{N} m_j \left( \frac{f_j}{\rho_i^2} + \frac{f_i}{\rho_j^2} \right) \nabla w(||x_i - x_j||/h_j)
\]

This gradient is symmetric: \( \nabla f(x_i) = \sum_j g_{ij} \) : \( g_{ij} = -g_{ji} \)
Smoothened Particle Hydrodynamics (SPH)

- Other differential operators
  - Divergence
    \[ \nabla \cdot f(x_i) \approx \sum_j \frac{m_j}{\rho_j} (f_j - f_i) \cdot \nabla w(\|x_i - x_j\|/h) \]
  - Laplacian
    \[ \Delta f(x_i) \approx \sum_j \frac{m_j}{\rho_j} (f_j - f_i) \Delta w(\|x_i - x_j\|/h) \]
Smoothed Particle Hydrodynamics (SPH)

Problem: Operator inconsistency

- Theorems derives in continuous setting don’t hold

\[ \Delta f \neq \nabla \cdot \nabla f \]

Solution: Derive operators for specific guarantees
Smoothed Particle Hydrodynamics (SPH)

Problem: particle inconsistency

- constant consistency in continuous setting
  \[ \int w(||x - y||/h)dy = 1 \]

- does not necessarily give constant consistency in discrete setting (irregular sampling, boundaries)
  \[ \sum_{i=1}^{N} \frac{m_i}{\rho_i} w(||x - x_i||/h_i) = 1 \]

Solution: see MLS approximation
Smoothed Particle Hydrodynamics (SPH)

Problem: particle deficiencies near boundaries

- integral/summation truncated by the boundary
- example: wrong density estimation

\[ \rho(x) \approx \sum_{i=1}^{N} m_i w(\|x - x_i\|/h_i) \]

Solution: ghost particles
A scalar function $f$ satisfies

$$f(x) = \int f(y) \delta(x - y) dy$$

Replace Dirac by a smooth function $w$

$$f(x) \approx \int f(y) w(\|x - y\|/h) dy$$

Discretize

$$f(x) \approx \sum_{i=1}^{N} V_i f_i w(\|x - x_i\|/h_i)$$
SPH Summary (2)

Function evaluation:

\[ f(x_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j w(\|x_i - x_j\|/h_j) \]

Gradient evaluation:

\[ \nabla f(x_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \nabla w(\|x_i - x_j\|/h_j) \]

\[ \nabla f(x_i) = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j (f_j - f_i) \nabla w(\|x_i - x_j\|/h_j) \]

\[ \nabla f(x_i) = \rho_i \sum_{j=1}^{N} m_j \left( \frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2} \right) \nabla w(\|x_i - x_j\|/h_j) \]
SPH Summary (3)

Further literature

- Smoothed Particle Hydrodynamics, Monaghan, 1992
- Smoothed Particles: A new paradigm for animating highly deformable bodies, Desbrun & Cani, 1996
- Smoothed Particle Hydrodynamics, A Meshfree Particle Method, Liu & Liu, 2003
- Particle-Based Fluid Simulation for Interactive Applications, Müller et al., 2003
- Smoothed Particle Hydrodynamics, Monaghan, 2005
- Adaptively Sampled Particle Fluids, Adams et al., 2007
- Fluid Simulation, Chapter 7.3 in Point Based Graphics, Wicke et al., 2007
- Many more
Solve the Navier-Stokes momentum equation

\[ \rho \left( \frac{Dv}{Dt} \right) = -\nabla P + \mu \nabla^2 v + g \]

\[ \text{Lagrangian derivative} \quad \text{pressure force} \quad \text{viscosity force} \quad \text{gravity} \]
Particle Fluid Simulation

Discretized and solved at particles using SPH

\[ \rho_i \left( \frac{Dv_i}{Dt} \right) = -\nabla P_i + \mu \nabla^2 v_i + g \]

- **density estimation**
  \[ \rho_i = \rho(x_i) = \sum_{j=1}^{N} V_j \rho_j w(\|x_i - x_j\|/h_j) = \sum_{j=1}^{N} m_j w(\|x_i - x_j\|/h_j) \]

- **pressure force**
  \[ -\nabla P_i = - \sum_{j=1}^{N} V_j P_j \nabla w(\|x_i - x_j\|/h_j) \]

- **viscosity force**
  \[ \mu \nabla^2 v_i = \mu \sum_{j=1}^{N} V_j v_j \nabla^2 w(\|x_i - x_j\|/h_j) \]
Preview: Particle Fluid Simulation
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares

- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- Conclusions
Moving Least Squares
Meshless Approximations

Same problem statement:
Approximate a function from discrete samples

1D

2D, 3D
Moving Least Squares (MLS)

Moving least squares approach

Locally fit a polynomial \( f(x) = p^T(x)a \)

\[
a = [a \ b \ c \ d]^T \quad p(x) = [1 \ x \ y \ z]^T
\]

By minimizing \( E = \sum_{i=1}^{N} w(||x - x_i||/h_i) \left( p^T(x_i)a - f_i \right)^2 \)
Moving Least Squares (MLS)

\[ E = \sum_{i=1}^{N} w(\|x - x_i\|/h_i) \left( p^T(x_i) a - f_i \right)^2 \]

\[
\begin{align*}
\frac{\partial E}{\partial a} &= 2 \sum_{i=1}^{N} w(\|x - x_i\|/h_i) \left( p^T(x_i) a - f_i \right) = 0 \\
\frac{\partial E}{\partial b} &= 2 \sum_{i=1}^{N} w(\|x - x_i\|/h_i) \left( p^T(x_i) a - f_i \right) x_i = 0 \\
\frac{\partial E}{\partial c} &= 2 \sum_{i=1}^{N} w(\|x - x_i\|/h_i) \left( p^T(x_i) a - f_i \right) y_i = 0 \\
\frac{\partial E}{\partial d} &= 2 \sum_{i=1}^{N} w(\|x - x_i\|/h_i) \left( p^T(x_i) a - f_i \right) z_i = 0 \\
\end{align*}
\]

Solution: \( a = M(x)^{-1} \sum_{i=1}^{N} w(\|x - x_i\|/h_i) p(x_i) f_i \)

with \( M(x) = \sum_{i=1}^{N} w(\|x - x_i\|/h_i) p(x_i) p^T(x_i) \)

Approximation: \( f(x) = p^T(x) a = p^T(x) M(x)^{-1} \sum_{i=1}^{N} w(\|x - x_i\|/h_i) p(x_i) f_i \)
Moving Least Squares (MLS)

Approximation: \( f(x) = p^T(x)a = p^T(x)M(x)^{-1} \sum_{i=1}^{N} w(||x-x_i||/h_i)p(x_i)f_i \)

\[
f(x) = \sum_{i=1}^{N} \Phi_i(x)f_i
\]

with shape functions \( \Phi_i(x) \)

\[
\Phi_i(x) = w(||x-x_i||/h_i)p(x)^TM(x)^{-1}p(x_i)
\]

\( \rightarrow \) by construction they are consistent up to the order of the basis

\( \rightarrow \) by construction they build a partition of unity
Moving Least Squares (MLS)

\[ f(x) = \sum_{i=1}^{N} \Phi_i(x) f_i \]

\[ \Phi_i(x) = w(\|x - x_i\|/h_i)p^T(x)M(x)^{-1}p(x_i) \]
Moving Least Squares (MLS)

Derivatives

\[
\frac{\partial f(x)}{\partial x(k)} = \sum_{i=1}^{N} \frac{\partial \Phi_i(x)}{\partial x(k)} f_i
\]

\[
\frac{\partial \Phi_i(x)}{\partial x(k)} = \frac{\partial w(||x-x_i||/h_i)}{\partial x(k)} p^T(x) M(x)^{-1} p(x_i)
\]

\[
+ w(||x-x_i||/h_i) p^T(x) \frac{\partial M(x)^{-1}}{\partial x(k)} p(x_i)
\]

\[
+ w(||x-x_i||/h_i) \frac{\partial p^T(x)}{\partial x(k)} M(x)^{-1} p(x_i)
\]

\[
\frac{\partial (M^{-1})}{\partial x(k)} = -M^{-1} \left( \frac{\partial M}{\partial x(k)} \right) M^{-1}
\]
Moving Least Squares (MLS)

Consistency

▪ have to prove: \( p(x) = \sum_{i=1}^{N} \Phi_i(x)p(x_i) \)

▪ or: \( p^T(x) = \sum_{i=1}^{N} \Phi_i(x)p^T(x_i) \)

\[ \downarrow \quad \Phi_i(x) = w(\|x - x_i\|/h_i)p^T(x)M(x)^{-1}p(x_i) \]

\[ p^T(x) = \sum_{i=1}^{N} w(\|x - x_i\|/h_i)p^T(x)M(x)^{-1}p(x_i)p^T(x_i) \]

\[ p^T(x) = p^T(x)M(x)^{-1} \sum_{i=1}^{N} w(\|x - x_i\|/h_i)p(x_i)p^T(x_i) \]

\[ \downarrow \quad M(x) = \sum_{i=1}^{N} w(\|x - x_i\|/h_i)p(x_i)p^T(x_i) \]

\[ p^T(x) = p^T(x)M(x)^{-1}M(x) = p^T(x) \]
Moving Least Squares (MLS)

Problem: moment matrix can become singular

- Example:
  - particles in a plane $z = 0$ in 3D
  - Linear basis $p(x) = [1 \ x]^T = [1 \ x \ y \ z]^T$

$$M(x) = \sum_{i=1}^{N} w(||x - x_i||/h_i) p(x_i) p^T(x_i)$$

$$M(x) = \sum_{i=1}^{N} w(||x - x_i||/h_i) \begin{pmatrix}
1 & x_i & y_i & z_i \\
 x_i & x_i^2 & x_iy_i & x_iz_i \\
y_i & x_iy_i & y_i^2 & y_iz_i \\
z_i & x_iz_i & y_iz_i & z_i^2 \\
\end{pmatrix}$$

$$M(x) = \sum_{i=1}^{N} w(||x - x_i||/h_i) \begin{pmatrix}
1 & x_i & y_i & 0 \\
x_i & x_i^2 & x_iy_i & 0 \\
y_i & x_iy_i & y_i^2 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}$$
Moving Least Squares (MLS)

Stable computation of shape functions

\[ \Phi_i(x) = w(||x - x_i||/h_i)p(x)^T M(x)^{-1} p(x_i) \]
\[ M(x) = \sum_i w(||x - x_i||/h) p(x_i) p^T(x_i) \]

\[ \downarrow \text{ translate basis by } -x \]
\[ \downarrow \text{ scale by } 1/h \]

\[ \Phi_i(x) = w(||x - x_i||/h_i)p(0)^T M(x)^{-1} p\left(\frac{x_i-x}{h}\right) \]
\[ M(x) = \sum_i w(||x - x_i||/h) p\left(\frac{x_i-x}{h}\right) p^T\left(\frac{x_i-x}{h}\right) \]

It can be shown that this moment matrix has a lower condition number.
MLS Summary

\[ f(x) = \sum_{i=1}^{N} \Phi_i(x) f_i \]

\[ \Phi_i(x) = w(\|x - x_i\|/h_i)p(0)^T M(x)^{-1} p(\frac{x_i - x}{h}) \]

\[ M(x) = \sum_i w(\|x - x_i\|/h)p(\frac{x_i - x}{h})p^T(\frac{x_i - x}{h}) \]
Literature

- Moving Least Square Reproducing Kernel Methods (I) Methododology and Convergence, Liu et al., 1997
- Classification and Overview of Meshfree Methods, Fries & Matthies, 2004
- Point Based Animation of Elastic, Plastic and Melting Objects, Müller et al., 2004
- Meshless Animation of Fracturing Solids, Pauly et al., 2005
- Meshless Modeling of Deformable Shapes and their Motion, Adams et al., 2008
Preview: Elastic Solid Simulation

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]
Preview: Elastic Solid Simulation

\[ f_{ext} \]
\[ \downarrow \]
\[ f_t \]
\[ \downarrow \]
\[ u_{t+\Delta t} \]
\[ \downarrow \]
\[ \nabla u_{t+\Delta t} \]
\[ \downarrow \]
\[ U_{t+\Delta t} \]
\[ \downarrow \]
\[ \sigma_{t+\Delta t} \]
\[ \downarrow \]
\[ \epsilon_{t+\Delta t} \]

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]
Preview: Elastic Solid Simulation
Part I: Conclusion
SPH – MLS Comparison

\[ f(x) = \sum_{i=1}^{N} \Phi_i(x) f_i \]

**SPH**

\[ \Phi_i(x) = V_i w(\|x - x_i\|/h_i) \]

local  
fast  
simple weighting  
not consistent

**MLS**

\[ \Phi_i(x) = w(\|x - x_i\|/h_i)p(x)^T M(x)^{-1} p(x_i) \]

\[ M(x) = \sum_i w(\|x - x_i\|/h_i)p(x_i)p(x_i)^T \]

local  
slower  
matrix inversion (can fail)  
consistent up to chosen order
Lagrangian vs Eulerian Kernels

Lagrangian kernels neighbors remain constant

Eulerian kernels neighbors change

[Fries & Matthies 2004]
Lagrangian vs Eulerian Kernels

Lagrangian kernels are OK for elastic solid simulations, but not for fluid simulations

[Fries & Matthies 2004]
Moving Least Squares Particle Hydrodynamics (MLSPH)

Use idea of variable rank MLS

\[ \Phi_i(x) = V_i w(\|x - x_i\|/h_i) \quad \text{(SPH)} \]

\[ \Downarrow \]

\[ \Phi_i(x) = w(\|x - x_i\|/h_i)p(x)^T M(x)^{-1} p(x_i) \quad \text{(MLS)} \]

- start for each particle with basis of highest rank
- if inversion fails, lower rank

Consequence: shape functions are not smooth
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares

- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- Conclusions
Application 1:

Particle Fluid Simulation
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares

- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- Conclusions
Fluid Simulation
Eulerian vs. Lagrangian

- Eulerian Simulation
  - Discretization of space
  - Simulation mesh required
  - Better guarantees / operator consistency
  - Conservation of mass problematic
  - Arbitrary boundary conditions hard
Eulerian vs. Lagrangian

- Lagrangian Simulation
  - Discretization of the material
  - Meshless simulation
  - No guarantees on consistency
  - Mass preserved automatically (particles)
  - Arbitrary boundary conditions easy (per particle)
Navier-Stokes Equations

- Momentum equation:
  \[
  \frac{\partial v}{\partial t} + v \cdot \nabla v = \frac{1}{\rho} \left( -\nabla p + \mu \Delta v + f_{\text{ext}} \right)
  \]

- Continuity equation:
  \[
  \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0
  \]
Continuum Equation

- Continuum equation automatically fulfilled
  - Particles carry mass
  - No particles added/deleted $\Rightarrow$ No mass loss/gain

- Compressible Flow
  - Often, incompressible flow is a better approximation
  - Divergence-free flow (later)
Momentum Equation

\[ \frac{\partial v}{\partial t} + v \cdot \nabla v = \frac{1}{\rho} \left( -\nabla p + \mu \Delta v + f_{\text{ext}} \right) \]

- Left-hand side is *material derivative*
  - "How does the velocity of this piece of fluid change?"
- Useful in Lagrangian setting

\[ \frac{Dv}{Dt} = \frac{1}{\rho} \left( -\nabla p + \mu \Delta v + f_{\text{ext}} \right) \]
Momentum Equation

\[
\frac{Dv}{Dt} = \frac{1}{\rho} (-\nabla p + \mu \Delta v + f_{\text{ext}})
\]

\[a = 1/m \cdot F\]

- Instance of Newton’s Law
- Right-hand side consists of
  - Pressure forces
  - Viscosity forces
  - External forces
Density Estimate

- SPH has concept of density built in

\[ \rho_i = \sum_j w_{ij} m_j \]

- Particles carry mass
- Density computed from particle density
Pressure

- Pressure acts to equalize density differences

\[ p = K \left( \left( \frac{\rho}{\rho_0} \right)^\gamma - 1 \right) \]

- CFD: \( \gamma = 7 \), computer graphics: \( \gamma = 1 \)
- large \( K \) and \( \gamma \) require small time steps
Pressure Forces

\[
\frac{Dv}{Dt} = \frac{1}{\rho} \left( -\nabla p + \mu \Delta v + f_{\text{ext}} \right)
\]

- Discretize \( a_p = \frac{-\nabla p}{\rho} \)
- Use symmetric SPH gradient approximation
  \[
  a_{p,i} = \frac{\nabla p(x_i)}{\rho_i} \approx \sum_j m_j \left( \frac{p_j}{\rho^2_j} + \frac{p_i}{\rho^2_i} \right) \nabla w_{ij}
  \]
- Preserves linear and angular momentum
Pressure Forces

- Symmetric pairwise forces: all forces cancel out
  - Preserves linear momentum
- Pairwise forces act along $x_i - x_j$
  - Preserves angular momentum
Viscosity

\[
\frac{Dv}{Dt} = \frac{1}{\rho} \left( -\nabla p + \mu \Delta v + f_{\text{ext}} \right)
\]

- Discretize using SPH Laplace approximation

\[
a_{v,i} = \frac{\mu \Delta v(x_i)}{\rho} \approx \mu \sum_j \frac{m_j}{\rho_i \rho_j} (v_j - v_i) \Delta w_{ij}
\]

- Momentum-preserving
- Very unstable
XSPH (artificial viscosity)

- Viscosity an artifact, not simulation goal
- Viscosity needed for stability
- Smoothes velocity field
- Artificial viscosity: stable smoothing

\[ \tilde{v}_i = (1 - \xi)v_i + \xi \sum_j w_{ij} v_j \]
Integration

- Update velocities
  \[ v_i \leftarrow v_i + \Delta t \left( a_{p,i} + \frac{f(x_i)}{\rho_i} \right) \]
- Artificial Viscosity
  \[ v_i \leftarrow (1 - \xi)v_i + \xi \sum_j w_{ij}v_j \]
- Update Positions
  \[ x_i \leftarrow x_i + \Delta t v_i \]
Boundary Conditions

- Apply to individual particles
  - Reflect off boundaries

- 2-way coupling
  - Apply inverse impulse to object
Surface Effects

- Density estimate breaks down at boundaries
- Leads to higher particle density
Surface Extraction

- Extract iso-surface of density field
- Marching cubes
<table>
<thead>
<tr>
<th>Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Sampling</td>
</tr>
<tr>
<td>Incompressible flow</td>
</tr>
<tr>
<td>Multiphase flow</td>
</tr>
<tr>
<td>Interaction with deformables</td>
</tr>
<tr>
<td>Interaction with porous materials</td>
</tr>
</tbody>
</table>
Tutorial Overview

- **Meshless Methods**
  - smoothed particle hydrodynamics
  - moving least squares

- **Applications**
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- **Conclusions**
Application 2: Elastic Solid Simulation
Goal

Simulate elastically deformable objects
Goal

Simulate elastically deformable objects

efficient and stable algorithms

~

different materials
elastic, plastic, fracturing

~

highly detailed surfaces
Elasticity Model

What are the strains and stresses for a deformed elastic material?
Elasticity Model

Displacement field

\[ u(x) = (u, v, w)^T : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]

\[ u(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R} \]

\[ v(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R} \]

\[ w(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R} \]
Elasticity Model

Gradient of displacement field

\[ u(x) = (u, v, w)^T : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \]

\[ \nabla \begin{bmatrix} u, x & u, y & u, z \\ v, x & v, y & v, z \\ w, x & w, y & w, z \end{bmatrix} \]
Elasticity Model

Green-Saint-Venant non-linear strain tensor

\[ \epsilon = \frac{1}{2}(\nabla u + \nabla u^T + \nabla u^T \nabla u) \]

symmetric 3x3 matrix
Elasticity Model

Stress from Hooke’s law

\[ \sigma = E \epsilon \]

symmetric 3x3 matrix
Elasticity Model

For isotropic materials

\[
\begin{bmatrix}
\sigma_{xx} \\ \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{xz}
\end{bmatrix} = \frac{E}{(1 + \nu)(1 - 2\nu)} \begin{bmatrix}
1 - \nu & \nu & \nu & 0 & 0 & 0 \\
\nu & 1 - \nu & \nu & 0 & 0 & 0 \\
\nu & \nu & 1 - \nu & 0 & 0 & 0 \\
0 & 0 & 0 & 1 - 2\nu & 0 & 0 \\
0 & 0 & 0 & 0 & 1 - 2\nu & 0 \\
0 & 0 & 0 & 0 & 0 & 1 - 2\nu
\end{bmatrix} \begin{bmatrix}
\epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{xy} \\ \epsilon_{yz} \\ \epsilon_{xz}
\end{bmatrix}
\]

Young’s modulus \( E \)

Poisson’s ratio \( \nu \)
Elasticity Model

Strain energy density

\[ U = \frac{1}{2} \epsilon \cdot \sigma \]

Elastic force

\[ f_{\text{elastic}} = -\nabla_u U \]
Elasticity Model

Volume conservation force

\[ f^{\text{vol.}} = -\frac{k_v}{2} \nabla u (|I + \nabla u(x)| - 1)^2 \]

prevents undesirable shape inversions
Elasticity Model

Final PDE

\[
\rho \frac{\partial^2 x'}{\partial t^2} = \rho \frac{\partial^2 u}{\partial t^2} = f^{\text{elastic}} + f^{\text{volume}} + f^{\text{body}}
\]

\[f^{\text{elastic}} = -\frac{1}{2} \nabla u \epsilon \cdot \sigma\]

\[f^{\text{volume}} = -\frac{k_v}{2} \nabla u (|I + \nabla u(x)| - 1)^2\]

\[f^{\text{body}} = \rho g\]
Particle Discretization

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x) u_i \]
Simulation Loop

\[ f_{ext} \]
\[ \downarrow \]
\[ f_t \]
\[ \downarrow \]
\[ u_{t+\Delta t} \]
\[ \downarrow \]
\[ \nabla u_{t+\Delta t} \]
\[ \Downarrow \]
\[ U_{t+\Delta t} \]
\[ \uparrow \]
\[ \sigma_{t+\Delta t} \]
\[ \uparrow \]
\[ \epsilon_{t+\Delta t} \]

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]
Surface Animation

Two alternatives

- Using MLS approximation of displacement field
- Using local first-order approximation of displacement field
Surface Animation – Alternative 1

Simply use MLS approximation of deformation field

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]

Can use whatever representation: triangle meshes, point clouds, ...
Surface Animation – Alternative 1

Vertex position update

\[ x' = x + u(x) \]

Approximate normal update

- first-order Taylor for displacement field at normal tip

\[ u(x + n) \approx u(x) + \nabla u(x)^T n \]

- tip is transformed to

\[
\begin{align*}
(x + n)' &= x + n + u(x) + \nabla u(x)n \\
x' + n' &= x' + n + \nabla u(x)n \\
n' &= n + \nabla u(x)n
\end{align*}
\]
Surface Animation – Alternative 1

Easy GPU Implementation

\[ x' = x + u(x) = x + \sum_{i=1}^{N} \Phi_i(x)u_i \]

- Scalars remain constant

\[ n' = n + \nabla u(x)^T n = n + \sum_{i=1}^{N} (\nabla \Phi_i^T(x)n)u_i \]

→ only have to send particle deformations to the GPU
Surface Animation – Alternative 2

Use weighted first-order Taylor approximation for displacement field at vertex

\[ \tilde{u}(x) = \sum_{j} \bar{\omega}_{ij} \left( u_j + \nabla u(x_j)^T (x - x_j) \right) \]

Updated vertex position

\[ x' = x + \tilde{u}(x) \]

→ avoid storing per-vertex shape functions
→ at the cost of more computations
Plasticity

Include plasticity effects
Plasticity

Store some amount of the strain and subtract it from the actual strain in the elastic force computations

\[
\varepsilon_i^{\text{elastic}} = \varepsilon_i - \varepsilon_i^{\text{plastic}}
\]

strain state variable

\[
f_i^{\text{elastic}} = -\frac{1}{2} \nabla u_i \varepsilon_i^{\text{elastic}} \cdot \sigma_i
\]
Plasticity

Strain state variables updated by absorbing some of the elastic strain

Absorb some of the elastic strain:

\[
\text{if } \|\epsilon_i^{\text{elastic}}\| > c_{\text{yield}} \text{ then } \epsilon_i^{\text{plastic}} \leftarrow \epsilon_i^{\text{plastic}} + c_{\text{creep}} \cdot \epsilon_i^{\text{elastic}}
\]

Limit amount of plastic strain:

\[
\text{if } \|\epsilon_i^{\text{plastic}}\| > c_{\text{max}} \text{ then } \epsilon_i^{\text{plastic}} \leftarrow \epsilon_i^{\text{plastic}} \cdot \left(\frac{c_{\text{max}}}{\|\epsilon_i^{\text{plastic}}\|}\right)
\]
Plasticity

Update the reference shape and store the plastic strain state variables

\[ \varepsilon_{i}^{\text{plastic}} \leftarrow \varepsilon_{i}^{\text{plastic}} - \varepsilon_{i}, \]

\[ x_{i} \leftarrow x_{i} + u_{i}, \]

\[ u_{i} \leftarrow 0. \]
Ductile Fracture

Initial statistics:
2.2k nodes
134k surfels

Final statistics:
3.3k nodes
144k surfels

Simulation time:
23 sec/frame
Modeling Discontinuities

Only *visible* nodes should interact

`crack`
Modeling Discontinuities

Only \textit{visible} nodes should interact

- collect nearest neighbors
Modeling Discontinuities

Only *visible* nodes should interact

- collect nearest neighbors
- perform visibility test
Modeling Discontinuities

Only *visible* nodes should interact

- collect nearest neighbors
- perform visibility test
Modeling Discontinuities

Only **visible** nodes should interact

Discontinuity along the crack surfaces
Modeling Discontinuities

Only *visible* nodes should interact

Discontinuity along the crack surfaces

But also within the domain → undesirable!
Modeling Discontinuities

Visibility Criterion

Weight function

Shape function
Modeling Discontinuities

Solution: transparency method

- nodes in vicinity of crack partially interact
- by modifying the weight function

\[ \omega'_{ij} = \omega(\|x_j - x_i\|/h_i + (d_s/\kappa h)^2) \]

\[ \rightarrow \text{crack becomes transparent near the crack tip} \]

---

1 Organ et al.: Continuous Meshless Approximations for Nonconvex Bodies by Diffraction and Transparency, Comp. Mechanics, 1996
Modeling Discontinuities

Visibility Criterion

Weight function

Shape function

Transparency Method
Re-sampling

- Add simulation nodes when number of neighbors too small

- **Local** re-sampling of the domain of a node
  - distribute mass
  - adapt support radius
  - interpolate attributes

- Shape functions adapt automatically!
Re-sampling: Example
Brittle Fracture

Initial statistics:
4.3k nodes
249k surfels

Final statistics:
6.5k nodes
310k surfels

Simulation time:
22 sec/frame
Summary

\[ f_{ext} \]
\[ \downarrow \]
\[ f_t \]
\[ \downarrow \]
\[ u_{t+\Delta t} \]
\[ \downarrow \]
\[ \nabla u_{t+\Delta t} \]
\[ \downarrow \]
\[ \nabla u_{t+\Delta t} \rightarrow U_{t+\Delta t} \]
\[ \downarrow \]
\[ \sigma_{t+\Delta t} \]
\[ \uparrow \]
\[ \epsilon_{t+\Delta t} \]
\[ \uparrow \]
\[ \nabla u_{t+\Delta t} \rightarrow \epsilon_{t+\Delta t} \]

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]
Summary

Efficient algorithms
- for elasticity: shape functions precomputed
- for fracturing: local cutting of interactions

No bookkeeping for consistent mesh
- simple re-sampling
- shape functions adapt automatically

High-quality surfaces
- representation decoupled from volume discretization
- deformation on the GPU
Limitations

Problem with moment matrix inversions
- cannot handle shells (2D layers of particles)
- cannot handle strings (1D layer of particles)

Plasticity simulation rather expensive
- recomputing neighbors
- re-evaluating shape functions

Fracturing in many small pieces expensive
- excessive re-sampling
Tutorial Overview

- **Meshless Methods**
  - smoothed particle hydrodynamics
  - moving least squares

- **Applications**
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling

- **Conclusions**
Application 3: Shape & Motion Modeling
Shape Deformations
Shape Deformations: Objective

Find a realistic shape deformation given the user’s input constraints.
Shape Deformations
Shape Deformations
Shape Deformations

\[ f(x) = x + u(x) \]
Use meshless shape functions to define a continuous deformation field.

\[
\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) \mathbf{u}_i
\]

\[
f(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x})
\]
Deformation Field Representation

\[ u(x) = \sum_{i=1}^{N} \Phi_i(x)u_i \]

Precompute for every node and neighbor

\[ \Phi_i(x) = w(||x - x_i||/h_i)p(x)^T M(x)^{-1} p(x_i) \]

Complete linear basis in 3D

\[ p(x) = [1 \ x \ y \ z]^T \]
Deformation Field Optimization

We are optimizing the displacement field

\[ f(x) = x + u(x) \]

\[ = x + \sum_{i=1}^{N} \Phi_i(x) u_i \]

nodal deformations
unknowns to solve for
Deformation Field Optimization

The displacement field should satisfy the input constraints.

Position constraint

\[ \| f(x) - y \|^2 \rightarrow \min \]

\[ \| x + u(x) - y \|^2 \rightarrow \min \]

\[ \| x + \sum_i \Phi_i(x)u_i - y \|^2 \rightarrow \min \]

→ quadratic in the unknowns
Deformation Field Optimization

The displacement should be realistic.

Locally rigid (minimal strain)
\[ (\nabla f^T(x) \nabla f(x) - I)^2 \rightarrow \min \]
\[ (\nabla u(x) + \nabla u^T(x) + \nabla u^T(x) \nabla u(x))^2 \rightarrow \min \]

Volume preserving
\[ (|\nabla f(x)| - 1)^2 \rightarrow \min \]
\[ (|I + \nabla u(x)| - 1)^2 \rightarrow \min \]
→ degree 6 in the unknowns
→ non-linear problem
Deformation Field Optimization

The total energy to minimize

\[
E = \sum_{\text{constraints}} \| \mathbf{x} + \sum_i \Phi_i(x) \mathbf{u}_i - \mathbf{y} \|^2 \\
+ \sum_{\text{nodes}} \| \nabla f^T(x_i) \nabla f(x_i) - \mathbf{I} \|^2_F \\
+ \sum_{\text{nodes}} (|\nabla f(x_i)| - 1)^2
\]

Solve using LBFGS

- unknowns: nodal displacements \( \mathbf{u}_i = [u_i \ v_i \ w_i]^T \)
- need to compute derivatives \( \frac{\partial E}{\partial u_i}, \frac{\partial E}{\partial v_i}, \frac{\partial E}{\partial w_i} \) with respect to unknowns
Nodal Sampling & Coupling

Keep number of unknowns as low as possible.
Nodal Sampling & Coupling

Ensure proper coupling by using material distance in weight functions.
Nodal Sampling & Coupling

Set of candidate points:
vertices and interior set of dense grid points
Nodal Sampling & Coupling

Grid-based fast marching to compute material distances.
Nodal Sampling & Coupling

Resulting sampling is roughly uniform over the material. Resulting coupling respects the topology of the shape.
Surface Deformation

Use Alternative 1 of the surface animation algorithms discussed before

\[ \mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) u_i \]

Vertex positions and normals updated on the GPU
Shape Deformations

100k vertices, 60 nodes → 55 fps
Shape Deformations

500k vertices, 60 nodes $\rightarrow$ 10 fps
Deformable Motions
Deformable Motions: Objective

Find a smooth path for a deformable object from given key frame poses.
Deformation Field Representation

\[ u(x, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t) \Phi_i(x) u_{i,tj} \]

- \( \Phi_j(t) \): shape functions in time
- \( \Phi_i(x) \): shape functions in space
Deformation Field Representation

Frames: discrete samples in time

Solve only at discrete frames: nodal displacements $u_{i,t_j}$

Use meshless approximation to define continuous displacement field

$$u(x,t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t)\Phi_i(x)u_{i,t_j}$$
Deformation Field Representation

\[ u(x, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t) \Phi_i(x) u_{i,t_j} \]

Precompute for each frame for every neighboring frame

\[ \Phi_j(t) = w(||t - t_j||/r_j) p(t)^T M(t)^{-1} p(t_j) \]

Complete quadratic basis in 1D

\[ p(t) = [1 \ t \ t^2]^T \]
We want a realistic motion interpolating the keyframes.

handle constraints
keyframe 1
keyframe 2
keyframe 3

rigidity constraints
volume preservation constraints
acceleration constraints
obstacle avoidance constraints
Deformation Field Optimization

We want a smooth motion.

Acceleration constraint

\[ \| \frac{\partial^2 u}{\partial t^2}(x, t) \| ^2 \rightarrow \min \]

\[ u(x, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \phi_j(t) \phi_i(x) u_{i,t_j} \]

\[ \| \sum_{j=1}^{T} \sum_{i=1}^{N} \frac{\partial^2 \phi_j(t)}{\partial t^2} \phi_i(x) u_{i,t_j} \| ^2 \rightarrow \min \]

for all nodes in all frames
Deformation Field Optimization

We want a collision free motion.

Obstacle avoidance constraint

\[ d^2(f(x, t), t) \rightarrow \min \]
\[ f(x, t) = x + u(x, t) \]

\[ d^2(x + u(x, t), t) \rightarrow \min \]

for all nodes in all frames

\[ d(x, t) > 0 \]
\[ d(x, t) = 0 \]
Deformable Motions

59 nodes
500k vertices
2 keyframes

solve time: 10 seconds, 25 frames
Adaptive Temporal Sampling

Number of unknowns to solve for: $3NT$
→ keep as low as possible!

Constraints only imposed at frames
→ what at interpolated frames?

Adaptive temporal sampling algorithm
Adaptive Temporal Sampling

Solve only at the key frames.
Adaptive Temporal Sampling

Evaluate over whole time interval.
Adaptive Temporal Sampling

Introduce new frame where energy highest and solve.
Adaptive Temporal Sampling

Evaluate over whole time interval.
Adaptive Temporal Sampling

Iterate until motion is satisfactory.
Deformable Motions

- 66 nodes
- 166k vertices
- 7 keyframes

Interaction rate: 60 fps, modeling time: 2.5 min, solve time: 16 seconds, 28 frames
Summary

Realistic shape and motion modeling

- constraints from physical principles

Interactive and high quality

- MLS particle approximation
- low number of particles
- shape functions adapt to sampling and object’s shape
- decoupled surface representation
- adaptive temporal sampling

Rotations are however not interpolated exactly