Meshless Approximation Methods and Applications in Physics Based Modeling and Animation

Bart Adams

Martin Wicke

STANFORD UNIVERSITY

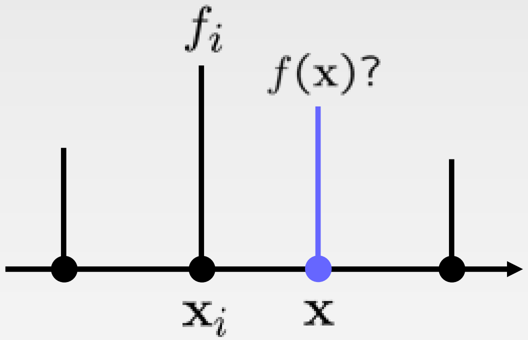
Tutorial Overview

- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares
- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling
- Conclusions

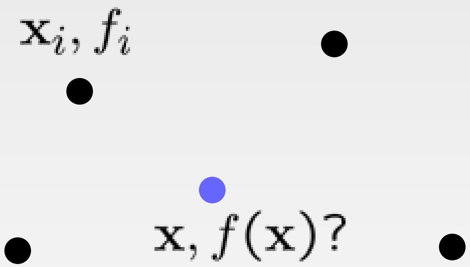
Part I: Meshless Approximation Methods

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

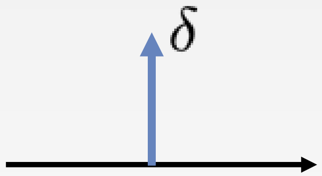
Smoothed Particle Hydrodynamics (SPH)

Integral representation of a scalar function f

$$f(\mathbf{x}) = \int f(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})d\mathbf{y}$$

Dirac delta function

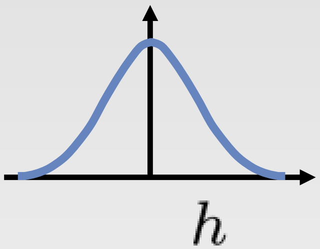
$$\delta(\mathbf{x}-\mathbf{y}) = \begin{cases} \infty & \mathbf{x}=\mathbf{y} \\ 0 & \mathbf{x}\neq\mathbf{y} \end{cases}$$



Smoothed Particle Hydrodynamics (SPH)

Replace Dirac by a smooth function w

$$f(\mathbf{x}) \approx \int f(\mathbf{y}) w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$



Desirable properties of w

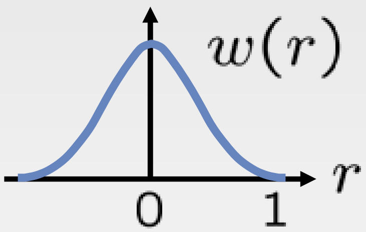
1. compactness: $w(\|\mathbf{x}-\mathbf{y}\|/h) = 0$ when $\|\mathbf{x}-\mathbf{y}\|/h > 1$
2. delta function property: $\lim_{h \to 0} w(\|\mathbf{x}-\mathbf{y}\|/h) = \delta(\mathbf{x}-\mathbf{y})$
3. unity condition (set f to 1): $\int w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y} = 1$
4. smoothness

Smoothed Particle Hydrodynamics (SPH)

Example: designing a smoothing kernel in 2D

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

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

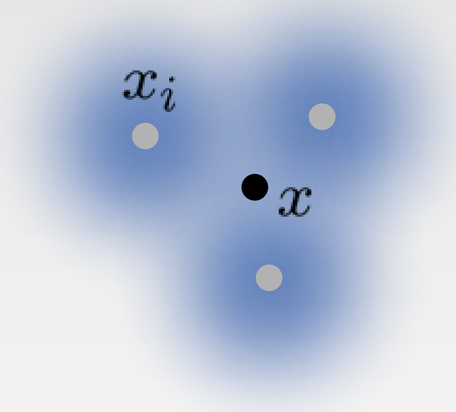
$$f(\mathbf{x}) \approx \int f(\mathbf{y}) w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$

$$\Downarrow$$

$$f(\mathbf{x}) \approx \sum_{i=1}^{N} f_i w(\|\mathbf{x}-\mathbf{x}_i\|/h_i) V_i$$

$$\Updownarrow$$

$$\boxed{f(\mathbf{x}) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f_i w(\|\mathbf{x}-\mathbf{x}_i\|/h_i)}$$



Smoothed Particle Hydrodynamics (SPH)

Example: density evaluation

$$f(\mathbf{x}) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} f_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)$$

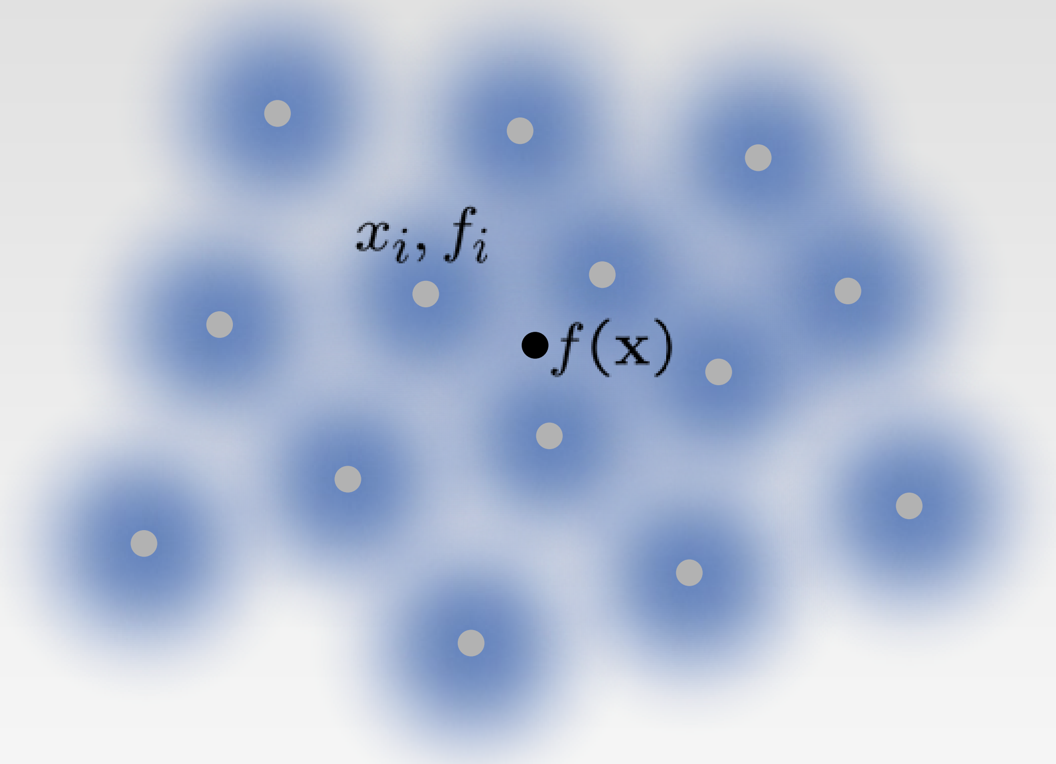
$$\Downarrow$$

$$\rho(\mathbf{x}) \approx \sum_{i=1}^{N} \frac{m_i}{\rho_i} \rho_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)$$

$$\Updownarrow$$

$$\rho(\mathbf{x}) \approx \sum_{i=1}^{N} m_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)$$

Smoothed Particle Hydrodynamics (SPH)



$$f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i$$

$$\Phi_i(\mathbf{x}) = \frac{m_i}{\rho_i} w(\|\mathbf{x} - \mathbf{x}_i\| / h_i)$$

Smoothed Particle Hydrodynamics (SPH)

Derivatives

$$f(\mathbf{x}) \approx \int f(\mathbf{y}) w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$

$$\Downarrow \text{ replace } f \text{ by } \nabla f$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \approx \nabla_{\mathbf{x}} \int f(\mathbf{y}) w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$

$$\Downarrow \quad \nabla, \int \text{ linear, product rule}$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \approx \int \nabla_{\mathbf{x}} f(\mathbf{x}) w(\|\mathbf{x}-\mathbf{y}\|/h)] d\mathbf{y} + \int f(\mathbf{y}) \nabla_{\mathbf{x}} w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$

$$\Downarrow \quad \nabla_{\mathbf{x}} f(\mathbf{y}) = 0$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) \approx \int f(\mathbf{y}) \nabla_{\mathbf{x}} w(\|\mathbf{x}-\mathbf{y}\|/h) d\mathbf{y}$$

Smoothed Particle Hydrodynamics (SPH)

Particle approximation for the derivative

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

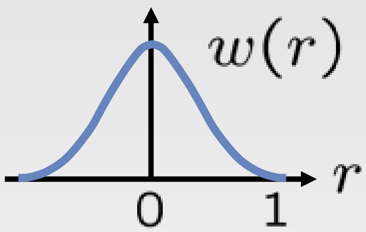
Some properties:

- 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 $r = \|\mathbf{x} - \mathbf{y}\|, h = 1$

Gradient using product rule:

$$\nabla_{\mathbf{x}} w = \frac{\partial w}{\partial r} \cdot \nabla_{\mathbf{x}} r = -\frac{30}{\pi}(1-r)^2 \cdot \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|}$$

Smoothed Particle Hydrodynamics (SPH)

Alternative derivative formulation

Old gradient formula: $\nabla f(\mathbf{x}_i) \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$ (1)

Product rule: $\nabla(\rho f) = f\nabla\rho + \rho\nabla f \Leftrightarrow \nabla f = \frac{1}{\rho}\nabla(\rho f) - \frac{1}{\rho} f \nabla\rho$ (2)

Use (1) in (2):

$$\nabla f(\mathbf{x}_i) \approx \frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j f_j \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j) - \frac{1}{\rho_i} f_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

$$\nabla f(\mathbf{x}_i) \approx \frac{1}{\rho_i} \sum_{j=1}^{N} m_j (f_j - f_i) \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

Gradient of constant function now always 0.

Smoothed Particle Hydrodynamics (SPH)

Similarly, starting from

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

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

$$\nabla f(\mathbf{x}_i) \approx \rho_i \sum_{j=1}^{N} m_j(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2})\nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

This gradient is *symmetric*: $\nabla f(\mathbf{x}_i) = \sum_j g_{ij} \quad : \quad g_{ij} = -g_{ji}$

Smoothed Particle Hydrodynamics (SPH)

- Other differential operators
  - Divergence

$$\nabla \cdot \mathbf{f}(\mathbf{x}_i) \approx \sum_j \frac{m_j}{\rho_j}(\mathbf{f}_j - \mathbf{f}_i) \cdot \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h)$$

  - Laplacian

$$\Delta f(x_i) \approx \sum_j \frac{m_j}{\rho_j}(f_j - f_i)\Delta w(\|\mathbf{x}_i - \mathbf{x}_j\|/h)$$

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

$$\int w(\|\mathbf{x} - \mathbf{y}\|/h)d\mathbf{y} = 1$$

- does not necessarily give constant consistency in discrete setting (irregular sampling, boundaries)

$$\sum_{i=1}^{N} \frac{m_i}{\rho_i} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) = 1$$

Solution: see MLS approximation

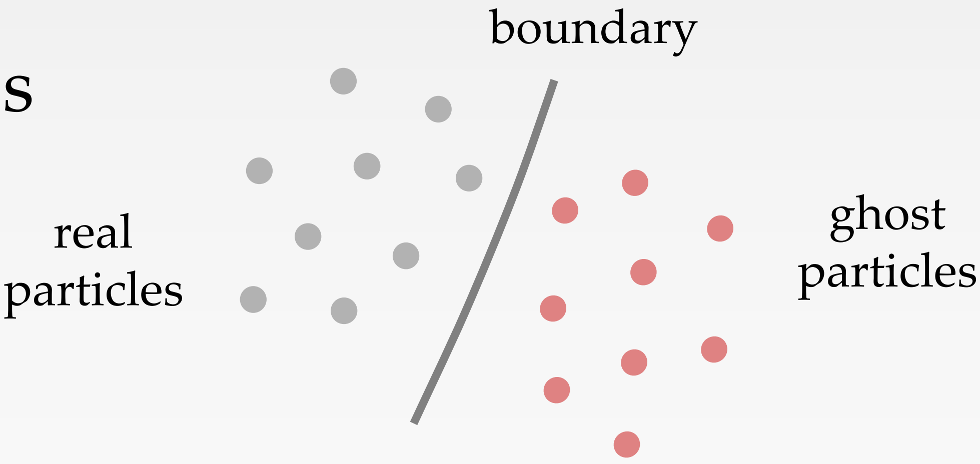
Smoothed Particle Hydrodynamics (SPH)

Problem: particle deficiencies near boundaries

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

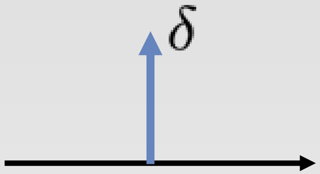
$$\rho(\mathbf{x}) \approx \sum_{i=1}^{N} m_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)$$

Solution: ghost particles



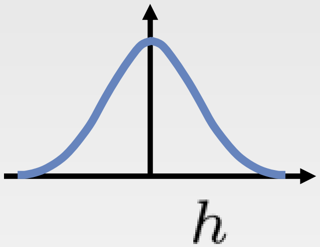
SPH Summary (1)

A scalar function f satisfies



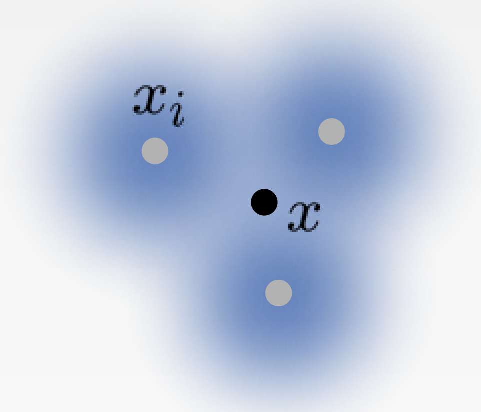
$$f(\mathbf{x}) = \int f(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})d\mathbf{y}$$

Replace Dirac by a smooth function w



$$f(\mathbf{x}) \approx \int f(\mathbf{y})w(\|\mathbf{x}-\mathbf{y}\|/h)d\mathbf{y}$$

Discretize

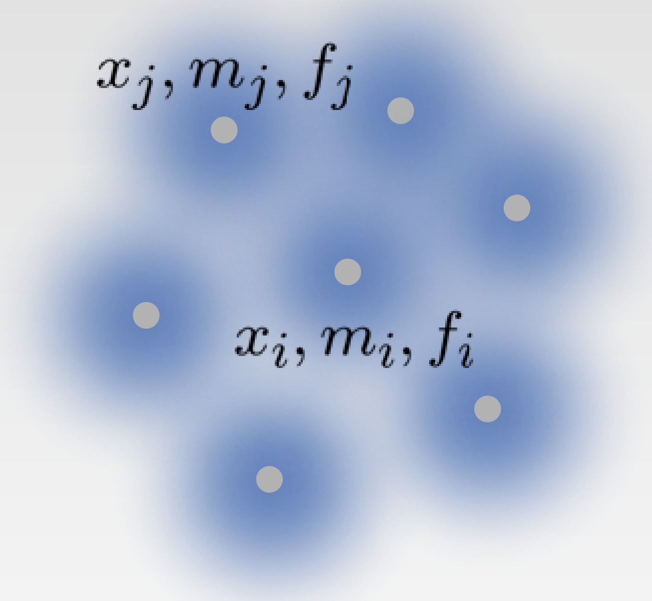


$$f(\mathbf{x}) \approx \sum_{i=1}^{N} V_i f_i w(\|\mathbf{x}-\mathbf{x}_i\|/h_i)$$

SPH Summary (2)

Function evaluation:

$$f(\mathbf{x}_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$



Gradient evaluation:

$$\nabla f(\mathbf{x}_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

$$\nabla f(\mathbf{x}_i) = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j (f_j - f_i) \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

$$\nabla f(\mathbf{x}_i) = \rho_i \sum_{j=1}^{N} m_j (\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2}) \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

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

Preview: Particle Fluid Simulation

Solve the Navier-Stokes momentum equation

$$\underbrace{\rho\left(\frac{D\mathbf{v}}{Dt}\right)}_{\text{Lagrangian derivative}} = \underbrace{-\nabla P}_{\text{pressure force}} + \underbrace{\mu\nabla^2\mathbf{v}}_{\text{viscosity force}} + \underbrace{\mathbf{g}}_{\text{gravity}}$$

Preview: Particle Fluid Simulation

Discretized and solved at particles using SPH

$$\rho_i \left(\frac{D\mathbf{v}_i}{Dt} \right) = -\nabla P_i + \mu \nabla^2 \mathbf{v}_i + \mathbf{g}$$

- density estimation

$$\rho_i = \rho(\mathbf{x}_i) = \sum_{j=1}^{N} V_j \rho_j w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j) = \sum_{j=1}^{N} m_j w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

- pressure force

$$-\nabla P_i = -\sum_{j=1}^{N} V_j P_j \nabla w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

- viscosity force

$$\mu \nabla^2 \mathbf{v}_i = \mu \sum_{j=1}^{N} V_j \mathbf{v}_j \nabla^2 w(\|\mathbf{x}_i - \mathbf{x}_j\|/h_j)$$

Preview: Particle Fluid Simulation

Tutorial Overview

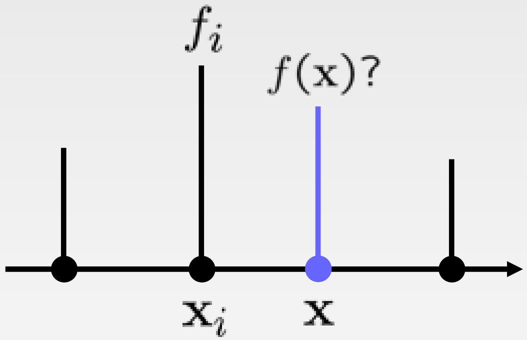
- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares
- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling
- Conclusions

Moving Least Squares

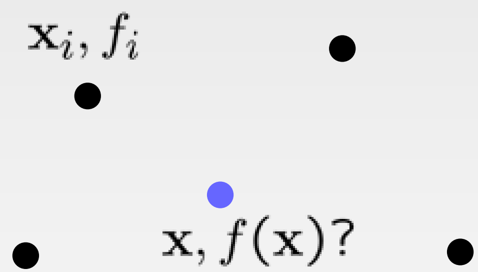
Meshless Approximations

Same problem statement:

Approximate a function from discrete samples



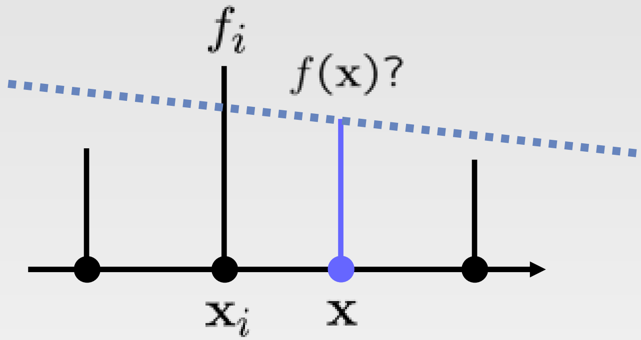
1D



2D, 3D

Moving Least Squares (MLS)

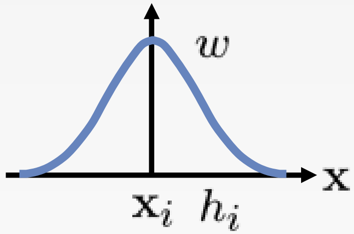
Moving least squares approach



Locally fit a polynomial $f(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}$

$$\mathbf{a} = [a\ b\ c\ d]^T \qquad \mathbf{p}(\mathbf{x}) = [1\ x\ y\ z]^T$$

By minimizing $E = \sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right)^2$



Moving Least Squares (MLS)

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

$$\begin{cases} \dfrac{\partial E}{\partial a} = 2\sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right) 1 = 0 \\ \dfrac{\partial E}{\partial b} = 2\sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right) x_i = 0 \\ \dfrac{\partial E}{\partial c} = 2\sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right) y_i = 0 \\ \dfrac{\partial E}{\partial d} = 2\sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right) z_i = 0 \end{cases}$$

$$\sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x}_i) \left(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a} - f_i\right) = 0$$

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

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

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

Moving Least Squares (MLS)

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

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i}$$

with shape functions $\Phi_i(\mathbf{x})$

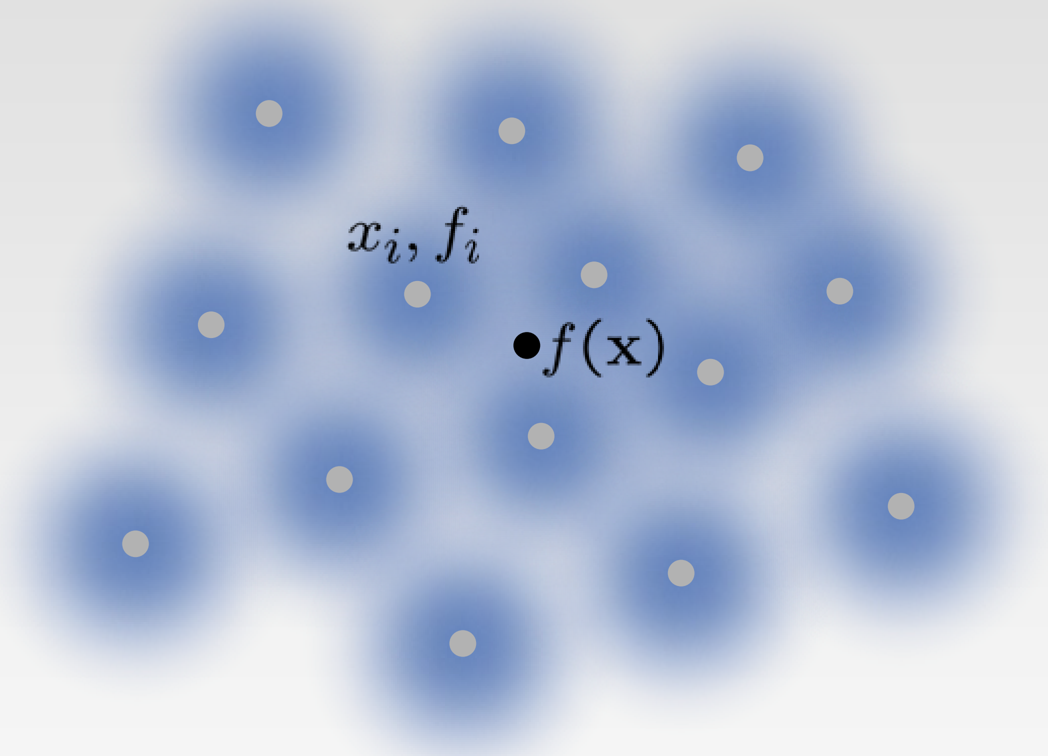
$$\Phi_i(\mathbf{x}) = \underbrace{w}_{\text{weight function}}(\|\mathbf{x}-\mathbf{x}_i\|/h_i)\underbrace{\mathbf{p}(\mathbf{x})^T}_{\text{complete polynomial basis}}\underbrace{\mathbf{M}(\mathbf{x})^{-1}}_{\text{moment matrix}}\mathbf{p}(\mathbf{x}_i)$$

moment matrix
$\mathbf{M}(\mathbf{x}) = \sum_i w(\|\mathbf{x}-\mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)^T$

$\rightarrow$ 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)



$$f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i$$

$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}^T(\mathbf{x})\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i)$$

Moving Least Squares (MLS)

Derivatives

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{(k)}} = \sum_{i=1}^{N} \frac{\partial \Phi_i(\mathbf{x})}{\partial \mathbf{x}_{(k)}} f_i$$

$$\begin{aligned}\frac{\partial \Phi_i(\mathbf{x})}{\partial \mathbf{x}_{(k)}} = & \frac{\partial w(\|\mathbf{x}-\mathbf{x}_i\|/h_i)}{\partial \mathbf{x}_{(k)}} \mathbf{p}^T(\mathbf{x}) \mathbf{M}(\mathbf{x})^{-1} \mathbf{p}(\mathbf{x}_i) \\ & + w(\|\mathbf{x}-\mathbf{x}_i\|/h_i) \mathbf{p}^T(\mathbf{x}) \frac{\partial \mathbf{M}(\mathbf{x})^{-1}}{\partial \mathbf{x}_{(k)}} \mathbf{p}(\mathbf{x}_i) \\ & + w(\|\mathbf{x}-\mathbf{x}_i\|/h_i) \frac{\partial \mathbf{p}^T(\mathbf{x})}{\partial \mathbf{x}_{(k)}} \mathbf{M}(\mathbf{x})^{-1} \mathbf{p}(\mathbf{x}_i)\end{aligned}$$

$$\frac{\partial (\mathbf{M}^{-1})}{\partial \mathbf{x}_{(k)}} = -\mathbf{M}^{-1} \left(\frac{\partial \mathbf{M}}{\partial \mathbf{x}_{(k)}}\right) \mathbf{M}^{-1}$$

Moving Least Squares (MLS)

Consistency

- have to prove: $\mathbf{p}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{p}(\mathbf{x}_i)$
- or: $\mathbf{p}^T(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{p}^T(\mathbf{x}_i)$

$$\Downarrow \; \Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}^T(\mathbf{x})\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i)$$

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

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

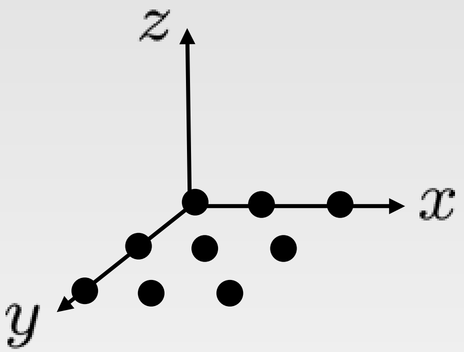
$$\Downarrow \; \mathbf{M}(\mathbf{x}) = \sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^T(\mathbf{x}_i)$$

$$\mathbf{p}^T(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{M}(\mathbf{x})^{-1}\mathbf{M}(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})$$

Moving Least Squares (MLS)

Problem: moment matrix can become singular

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



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

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

$$\mathbf{M}(\mathbf{x}) = \sum_{i=1}^{N} w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \begin{pmatrix} 1 & x_i & y_i & 0 \\ x_i & x_i^2 & x_i y_i & 0 \\ y_i & x_i y_i & y_i^2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Moving Least Squares (MLS)

Stable computation of shape functions

$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x})^T\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i)$$

$$\mathbf{M}(\mathbf{x}) = \textstyle\sum_i w(\|\mathbf{x} - \mathbf{x}_i\|/h)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^T(\mathbf{x}_i)$$

$\Downarrow$ translate basis by $-\mathbf{x}$
scale by $1/h$

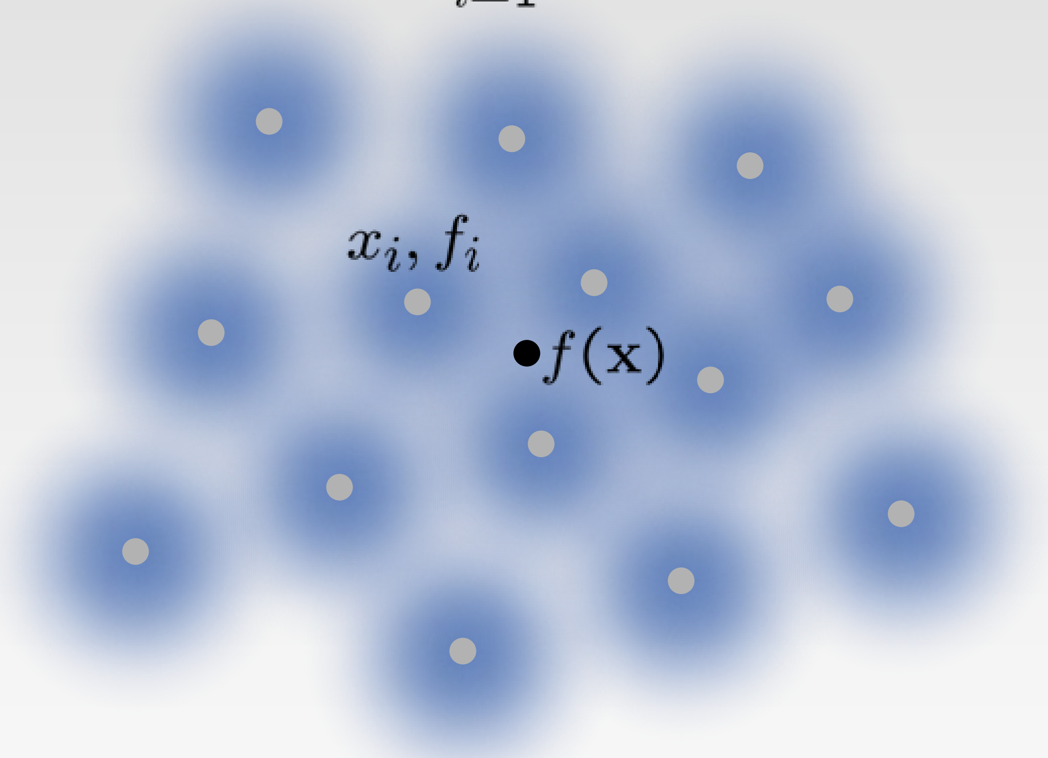
$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{0})^T\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\tfrac{\mathbf{x}_i-\mathbf{x}}{h})$$

$$\mathbf{M}(\mathbf{x}) = \textstyle\sum_i w(\|\mathbf{x} - \mathbf{x}_i\|/h)\mathbf{p}(\tfrac{\mathbf{x}_i-\mathbf{x}}{h})\mathbf{p}^T(\tfrac{\mathbf{x}_i-\mathbf{x}}{h})$$

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

MLS Summary

$$f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i$$



$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{0})^T\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\tfrac{\mathbf{x}_i - \mathbf{x}}{h})$$

$$\mathbf{M}(\mathbf{x}) = \textstyle\sum_i w(\|\mathbf{x} - \mathbf{x}_i\|/h)\mathbf{p}(\tfrac{\mathbf{x}_i - \mathbf{x}}{h})\mathbf{p}^T(\tfrac{\mathbf{x}_i - \mathbf{x}}{h})$$

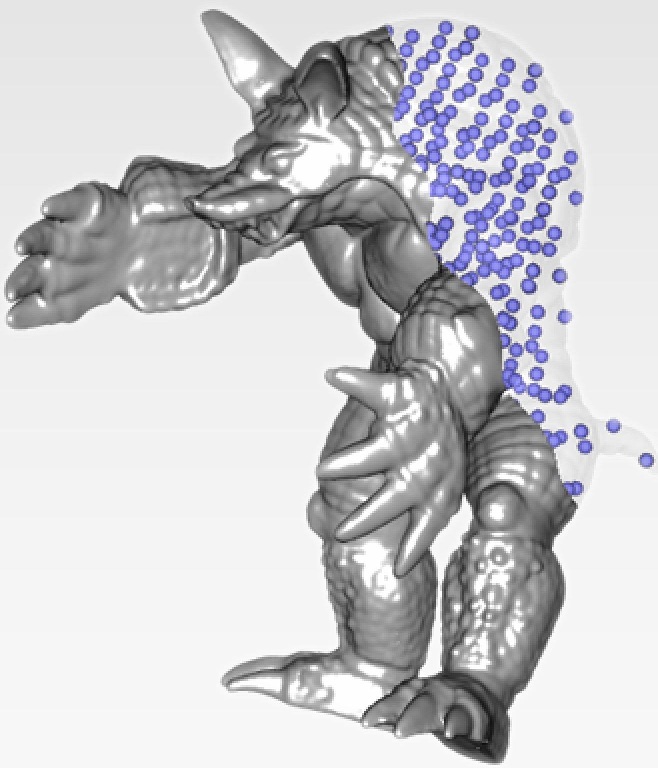
MLS Summary (2)

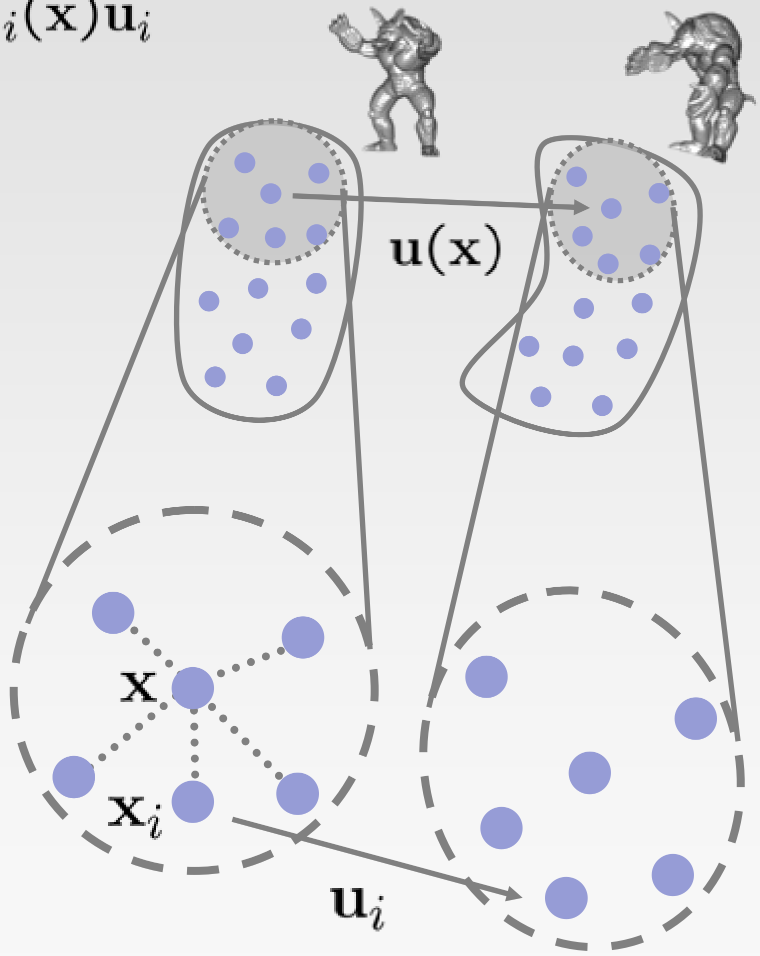
Literature

- Moving Least Square Reproducing Kernel Methods (I) Methododology and Convergence, Liu et al., 1997
- Moving-Least-Squares-Particle Hydrodynamics –I. Consistency and Stability, Dilts, 1999
- 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

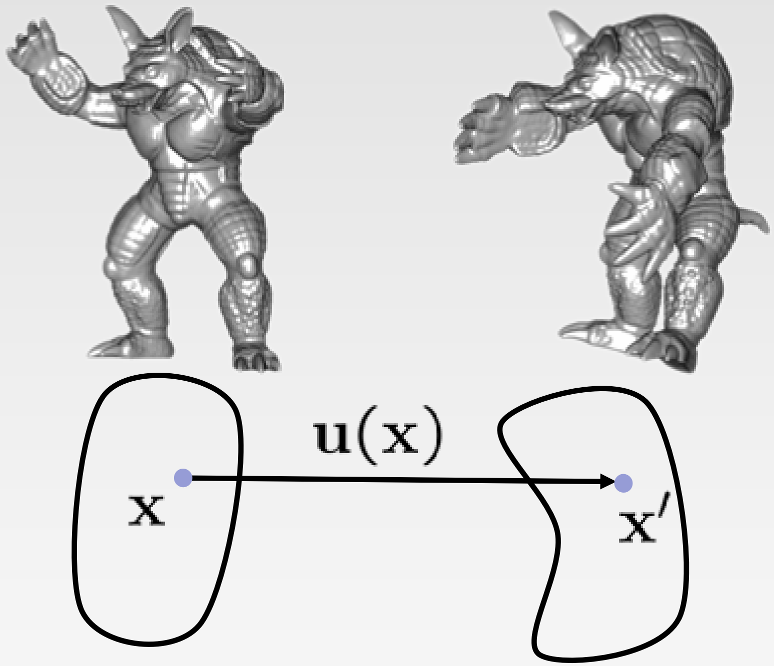
$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_i$$





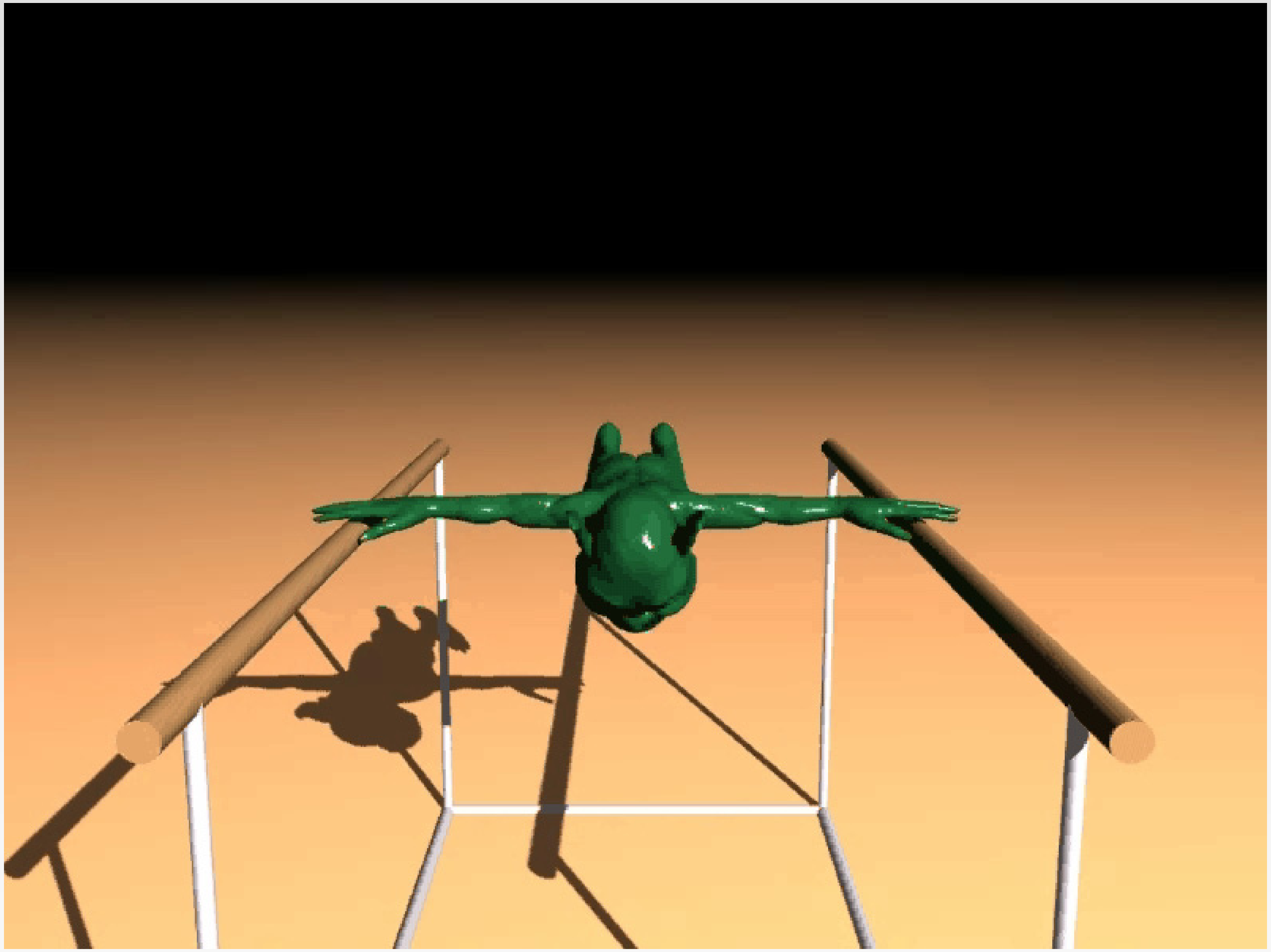
Preview: Elastic Solid Simulation

$$\begin{tikzcd}
\mathbf{f}_{ext} \arrow[d] & \\
\mathbf{f}_t \arrow[d] & U_{t+\Delta t} \arrow[l] \\
\mathbf{u}_{t+\Delta t} \arrow[d] & \sigma_{t+\Delta t} \arrow[u] \\
\nabla \mathbf{u}_{t+\Delta t} \arrow[r] & \epsilon_{t+\Delta t} \arrow[u]
\end{tikzcd}$$



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

Preview: Elastic Solid Simulation



Part I: Conclusion

SPH – MLS Comparison

$$f(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x}) f_i$$

SPH

$$\Phi_i(\mathbf{x}) = V_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)$$

local
fast
simple weighting
not consistent

MLS

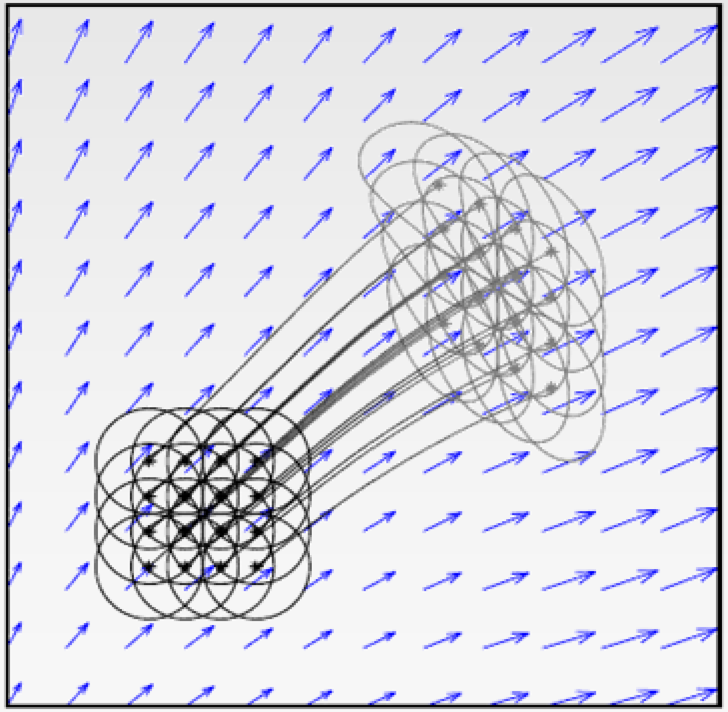
$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x})^T\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i)$$

$$\mathbf{M}(\mathbf{x}) = \sum_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)^T$$

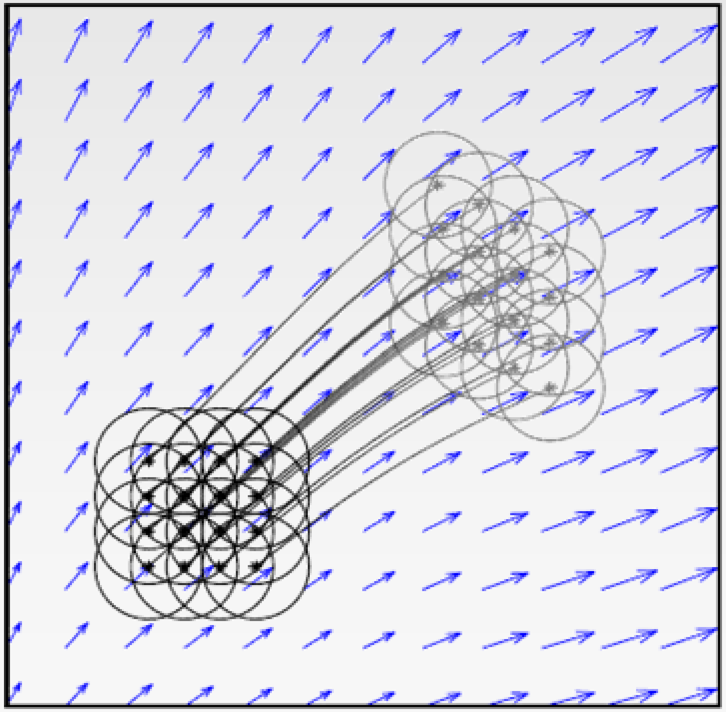
local
slower
matrix inversion (can fail)
consistent up to chosen order

Lagrangian vs Eulerian Kernels

Lagrangian kernels
neighbors remain constant



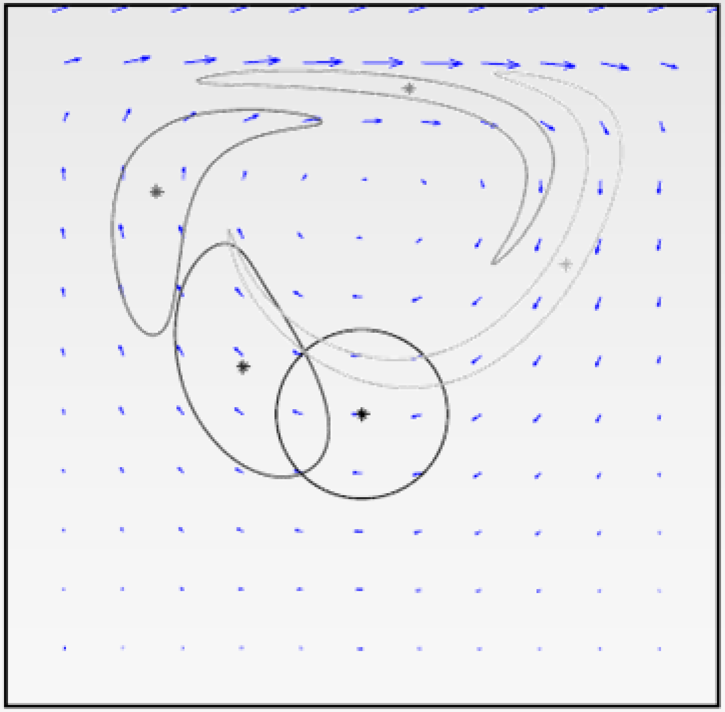
Eulerian kernels
neighbors change



[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

$$\Phi_i(\mathbf{x}) = V_i w(\|\mathbf{x} - \mathbf{x}_i\|/h_i) \qquad \text{(SPH)}$$

$$\Downarrow$$

$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x})^T\mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i) \qquad \text{(MLS)}$$

- 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

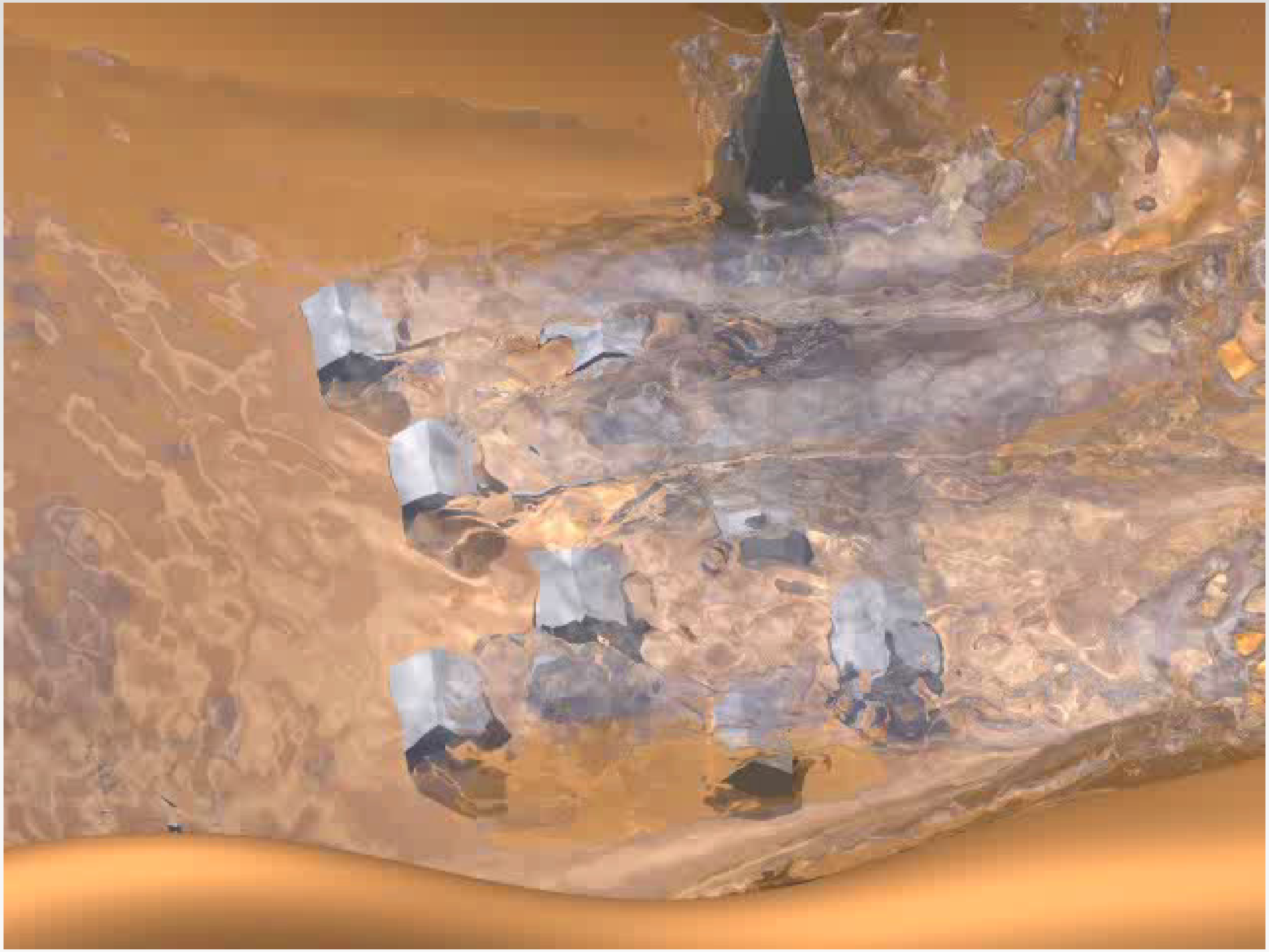
Application 1:

Particle Fluid Simulation

Tutorial Overview

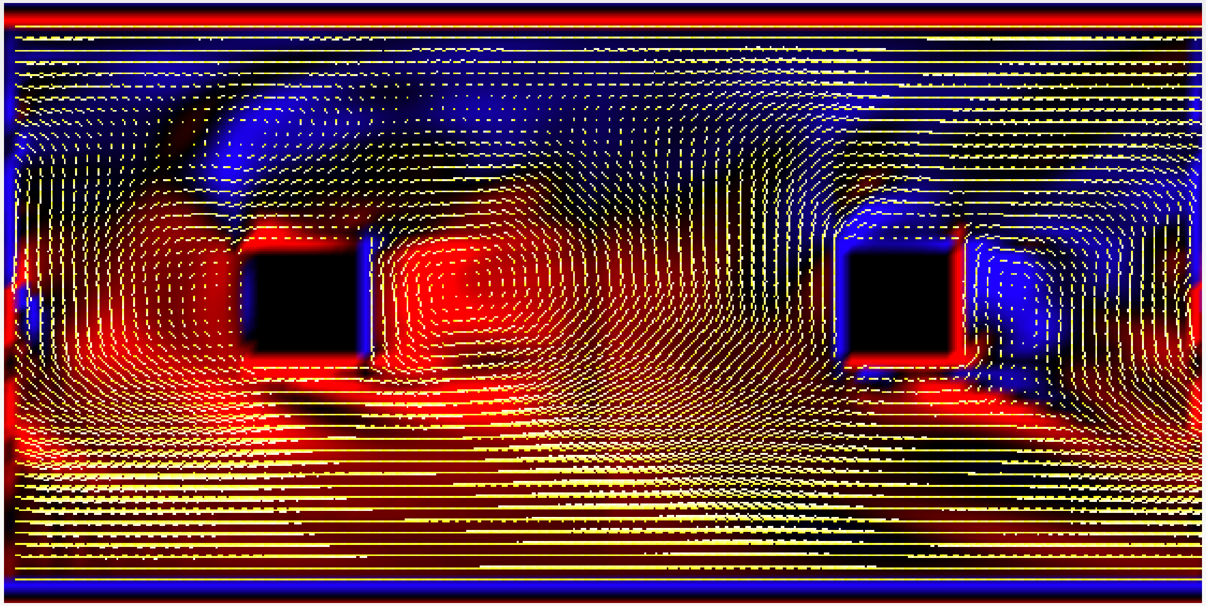
- Meshless Methods
  - smoothed particle hydrodynamics
  - moving least squares
- Applications
  - particle fluid simulation
  - elastic solid simulation
  - shape & motion modeling
- Conclusions

Fluid Simulation



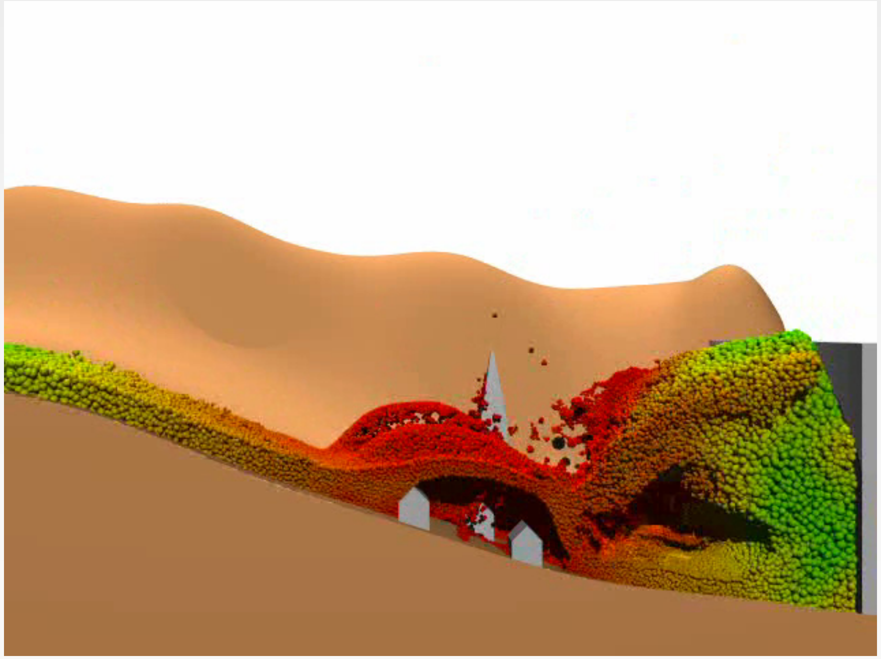
Eulerian vs. Lagrangian

- 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:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = \frac{1}{\rho}\left(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\mathsf{ext}}\right)$$

- Continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

Continuum Equation

- Continuum equation automatically fulfilled
  - Particles carry mass
  - No particles added/deleted → No mass loss/gain
- Compressible Flow
  - Often, incompressible flow is a better approximation
  - Divergence-free flow (later)

Momentum Equation

$$\boxed{\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}} = \frac{1}{\rho}\left(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\mathsf{ext}}\right)$$

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

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}\left(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\mathsf{ext}}\right)$$

Momentum Equation

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\text{ext}})$$

$$\mathbf{a} = 1/m \quad \cdot \quad \mathbf{F}$$

- 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 γ require small time steps

Pressure Forces

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\text{ext}})$$

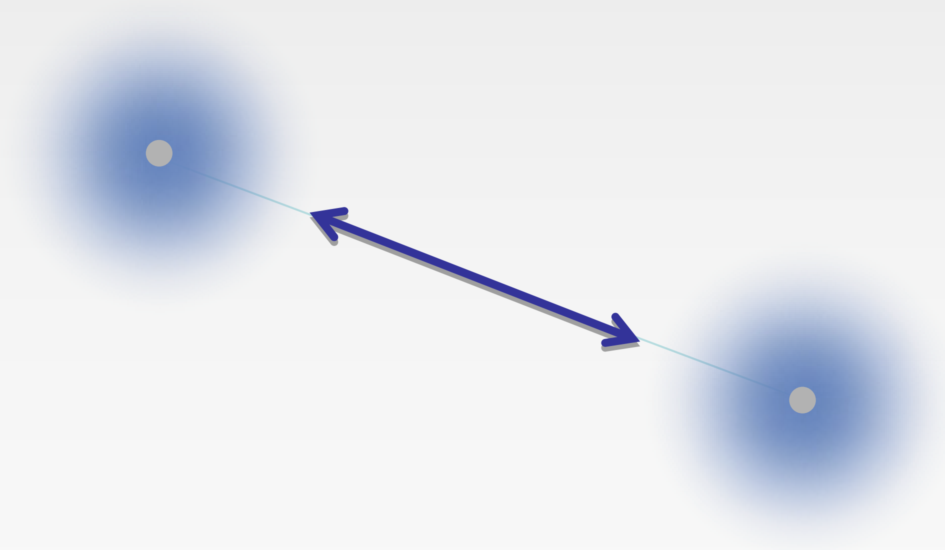
- Discretize $\mathbf{a}_p = \dfrac{-\nabla p}{\rho}$
- Use symmetric SPH gradient approximation

$$\mathbf{a}_{p,i} = \frac{\nabla p(\mathbf{x}_i)}{\rho_i} \approx \sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \nabla w_{ij}$$

- Preserves linear and angular momentum

Pressure Forces

- Symmetric pairwise forces: all forces cancel out
  - Preserves linear momentum
- Pairwise forces act along $\mathbf{x}_i - \mathbf{x}_j$
  - Preserves angular momentum



Viscosity

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}\left(-\nabla p + \mu \Delta \mathbf{v} + \mathbf{f}_{\mathsf{ext}}\right)$$

- Discretize using SPH Laplace approximation

$$\mathbf{a}_{v,i} = \frac{\mu \Delta \mathbf{v}(x_i)}{\rho} \approx \mu \sum_j \frac{m_j}{\rho_i \rho_j}(\mathbf{v}_j - \mathbf{v}_i) \Delta w_{ij}$$

- Momentum-preserving
- Very unstable

XSPH (artificial viscosity)

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

$$\tilde{\mathbf{v}}_i = (1-\xi)\mathbf{v}_i + \xi \sum_j w_{ij}\mathbf{v}_j$$

Integration

- Update velocities

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \left(\mathbf{a}_{p,i} + \frac{\mathbf{f}(\mathbf{x}_i)}{\rho_i}\right)$$

- Artificial Viscosity

$$\mathbf{v}_i \leftarrow (1-\xi)\mathbf{v}_i + \xi \sum_j w_{ij}\mathbf{v}_j$$

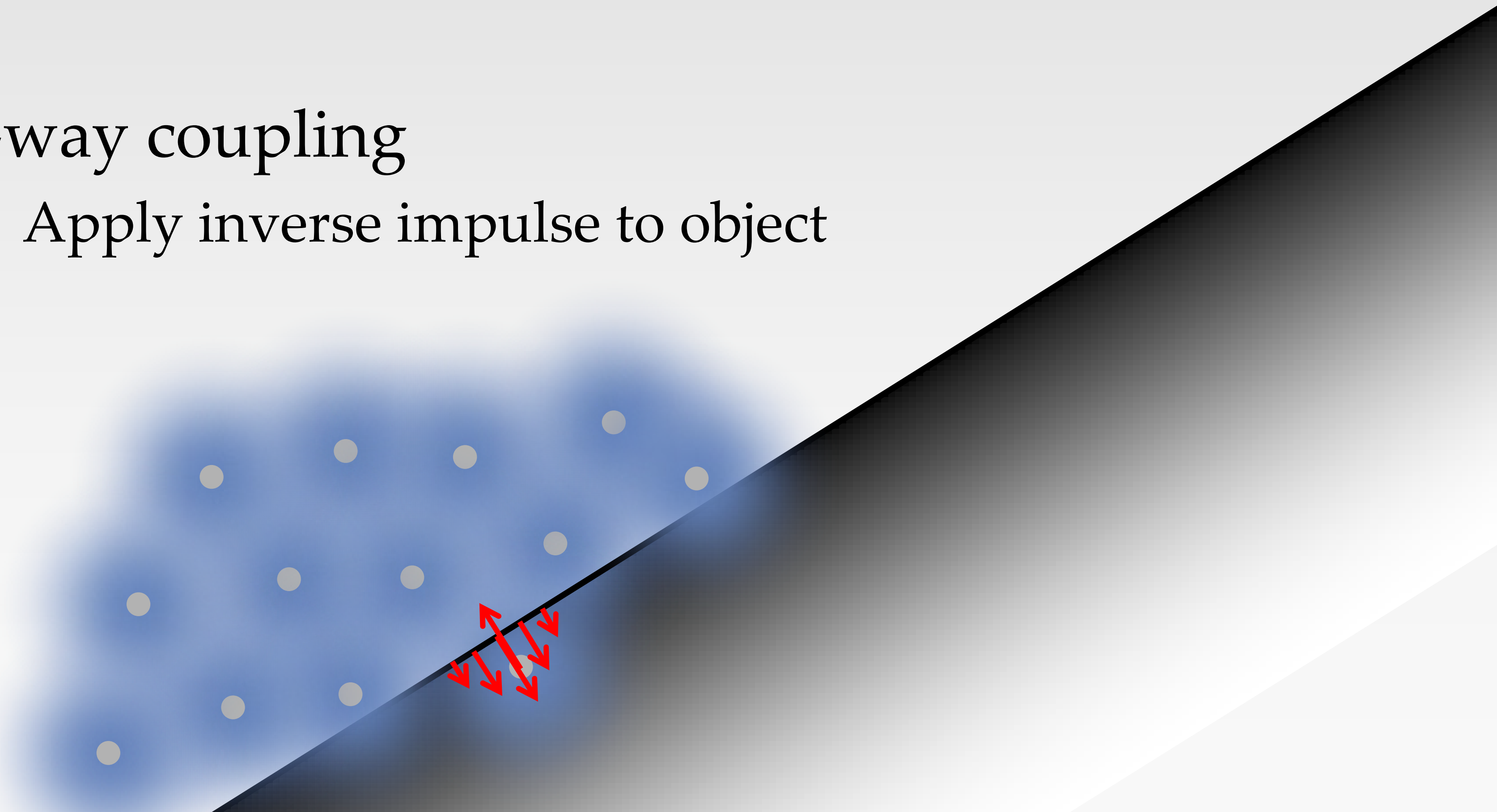
- Update Positions

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$$

Boundary Conditions

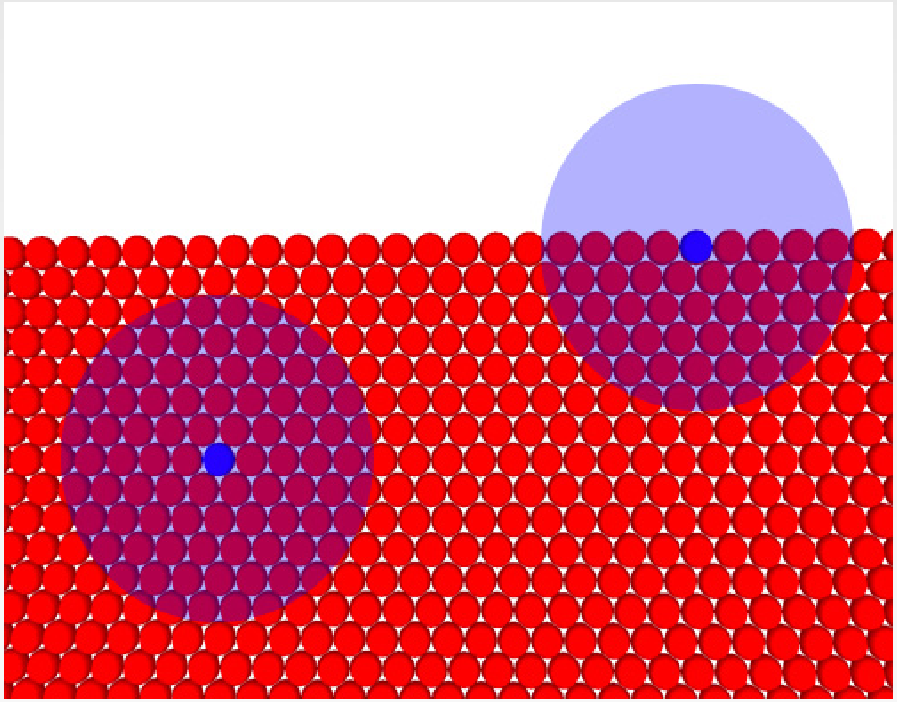
- Apply to individual particles
  - Reflect off boundaries

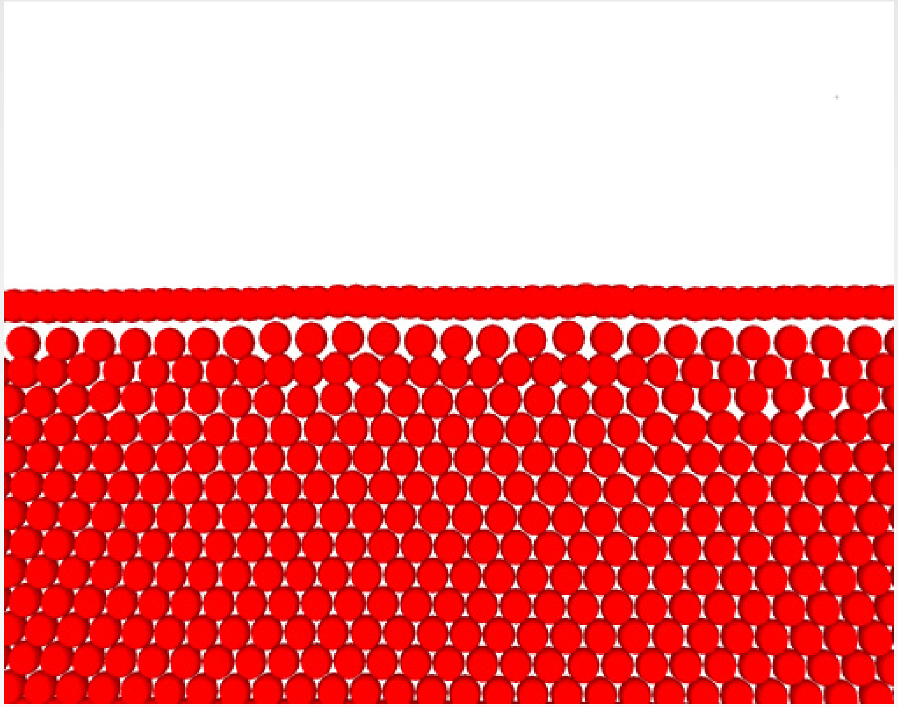
- 2-way coupling
  - Apply inverse impulse to object



Surface Effects

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





Surface Extraction

- Extract iso-surface of density field
- Marching cubes

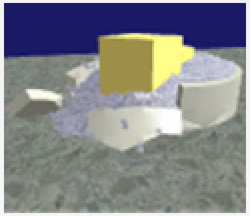
Extensions

- Adaptive Sampling [Adams et al 08]

- Incompressible flow [Zhu et al 05]

- Multiphase flow [Mueller et al 05]

- Interaction with deformables [Mueller et al 04]



- Interaction with porous materials [Lenaerts et al 08]

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

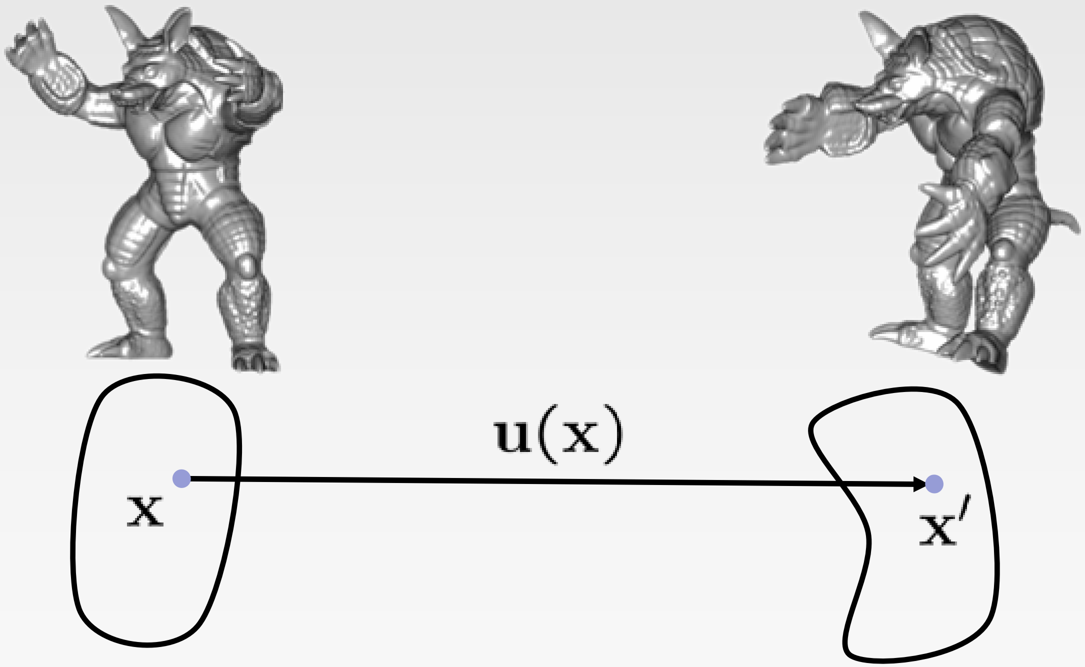
elastic, plastic, fracturing

~

highly detailed surfaces

Elasticity Model

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



Elasticity Model

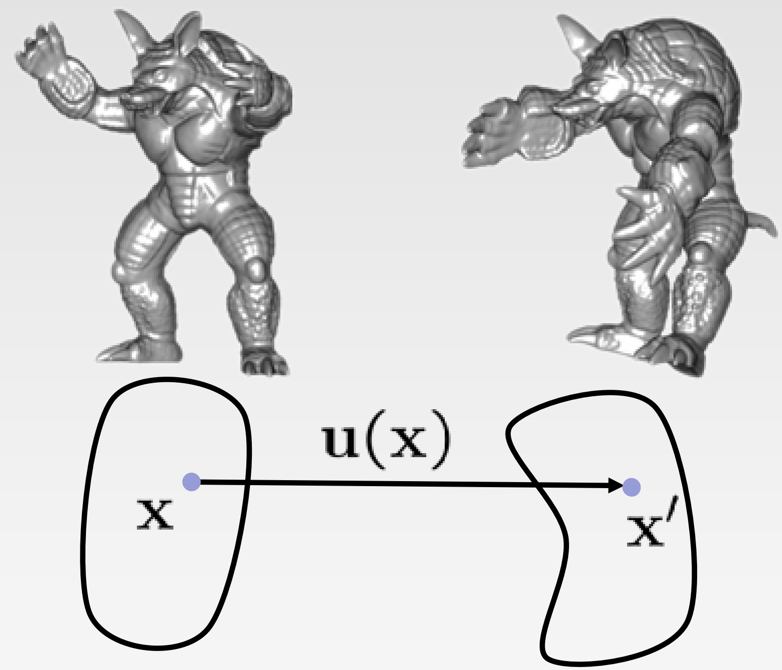
Displacement field

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

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

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

$$w(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R}$$



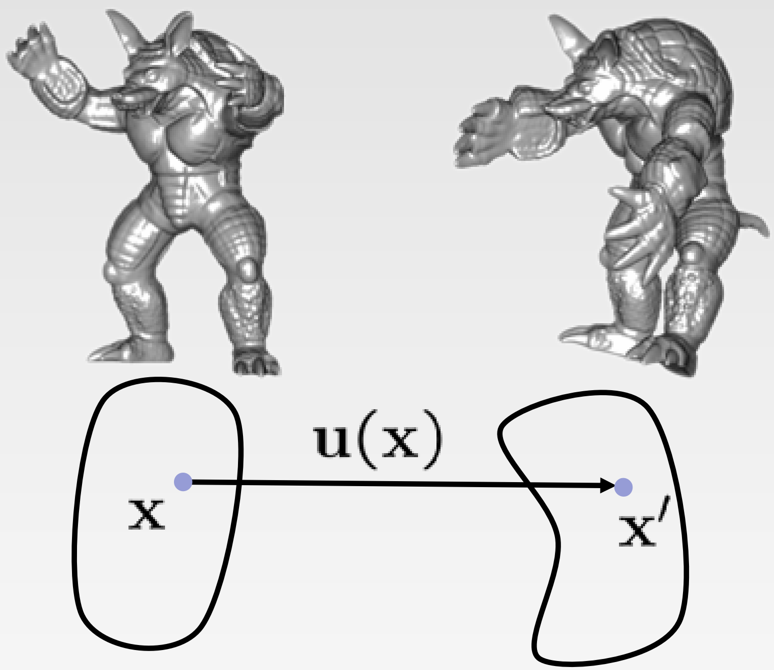
Elasticity Model

Gradient of
displacement field

$$\mathbf{u}(\mathbf{x}) = (u, v, w)^T : \mathbb{R}^3 \to \mathbb{R}^3$$

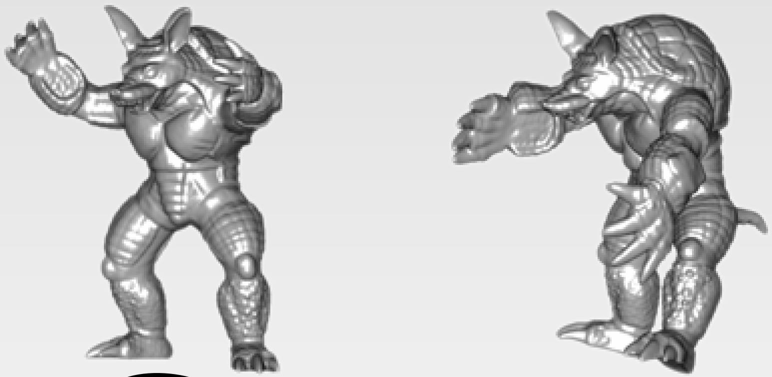
$$\Big\downarrow \nabla$$

$$\nabla \mathbf{u} = \begin{bmatrix} u_{,x} & u_{,y} & u_{,z} \\ v_{,x} & v_{,y} & v_{,z} \\ w_{,x} & w_{,y} & w_{,z} \end{bmatrix}$$

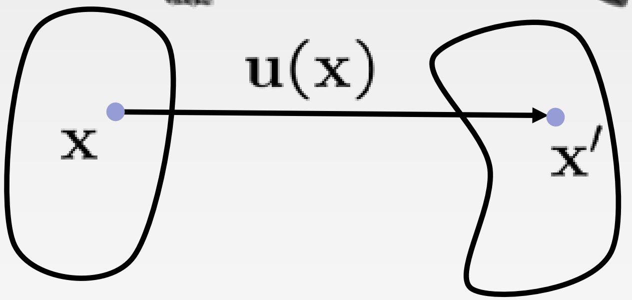


Elasticity Model

Green-Saint-Venant
non-linear strain tensor



$$\epsilon = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T + \nabla \mathbf{u}^T \nabla \mathbf{u})$$



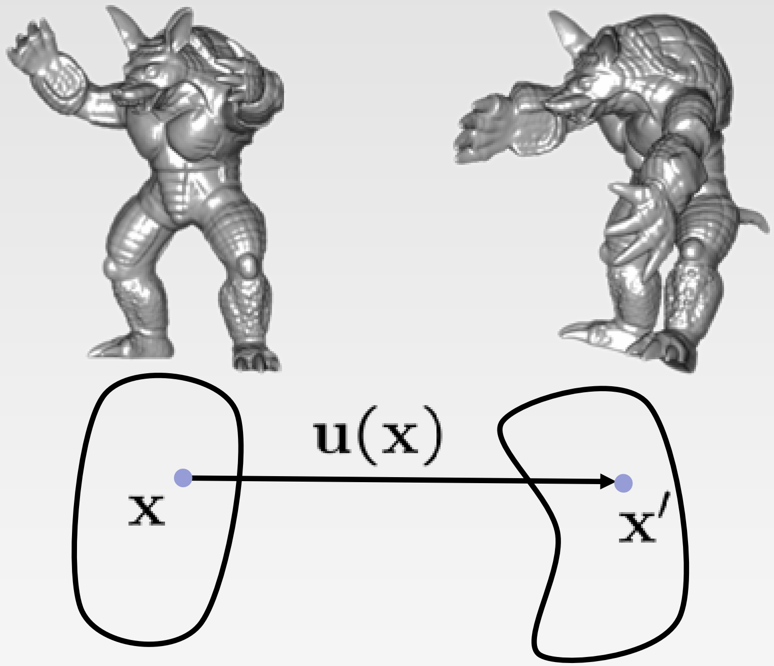
symmetric 3x3 matrix

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 v

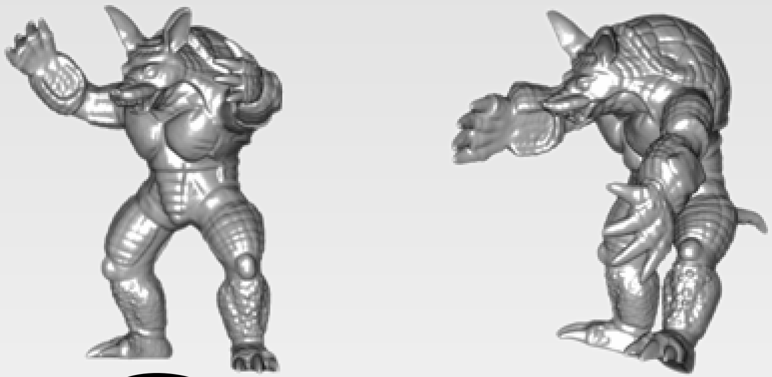
Elasticity Model

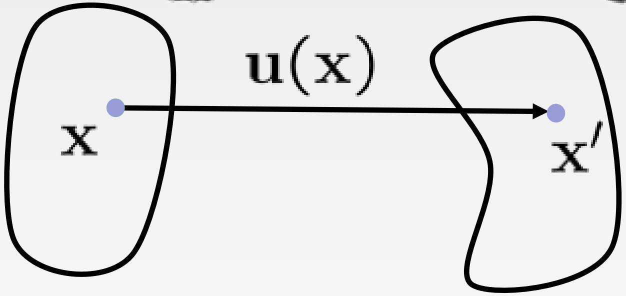
Strain energy density

$$U = \frac{1}{2}\epsilon \cdot \boldsymbol{\sigma}$$

Elastic force

$$\mathbf{f}^{\text{elastic}} = -\nabla_{\mathbf{u}} U$$

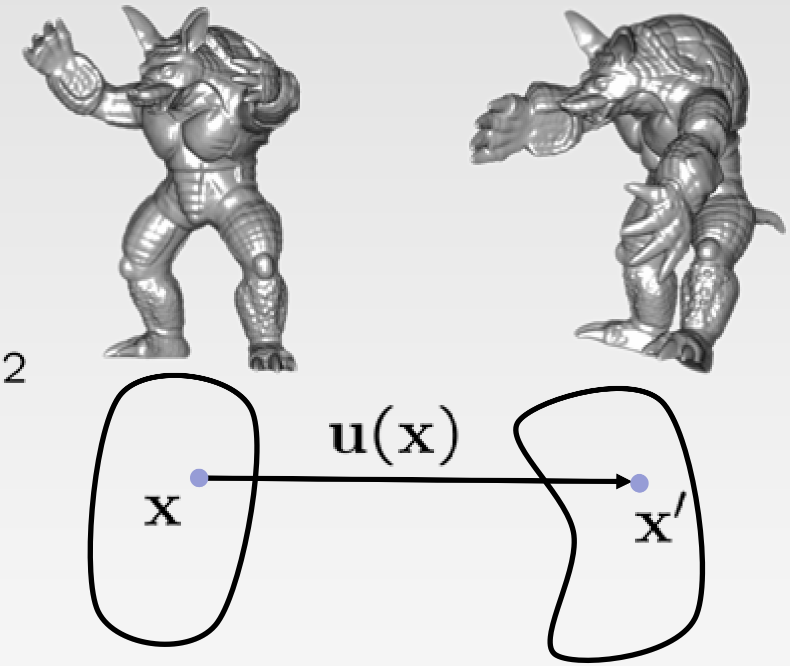




Elasticity Model

Volume conservation force

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



prevents undesirable shape inversions

Elasticity Model

Final PDE

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

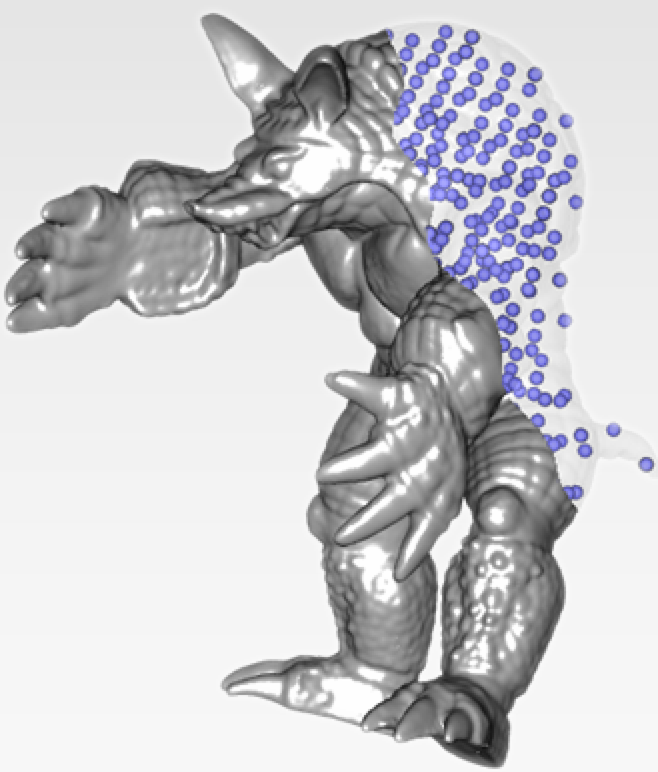
$$\mathbf{f}^{\text{elastic}} = -\frac{1}{2} \nabla_{\mathbf{u}} \epsilon \cdot \boldsymbol{\sigma}$$

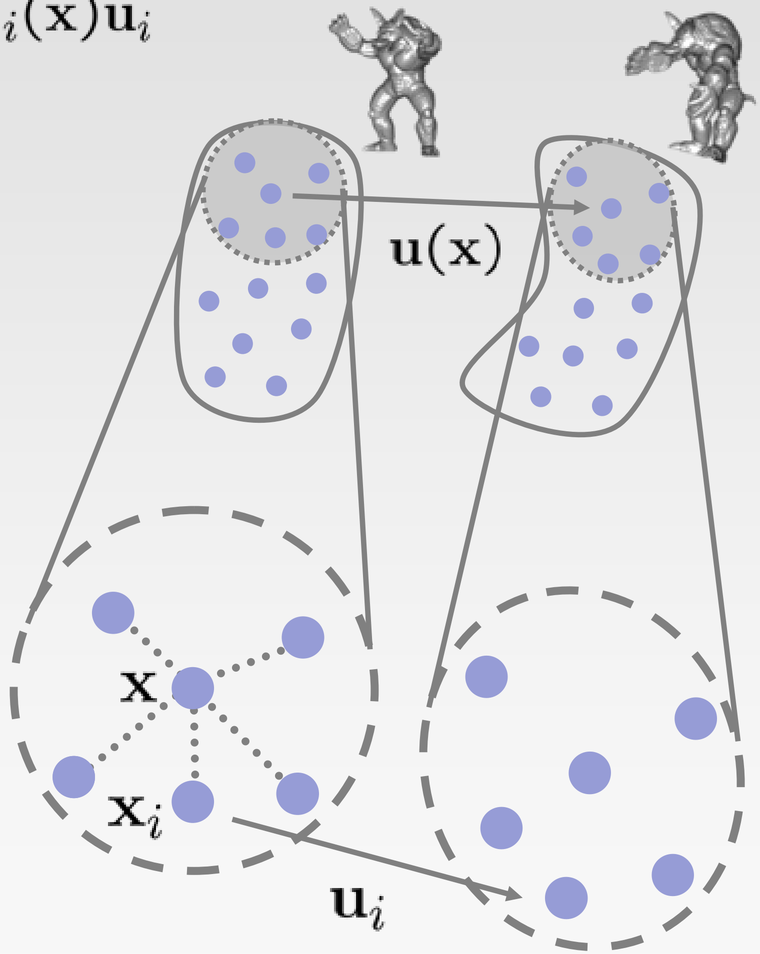
$$\mathbf{f}^{\text{volume}} = -\frac{k_v}{2} \nabla_{\mathbf{u}} (|\mathbf{I} + \nabla \mathbf{u}(\mathbf{x})| - 1)^2$$

$$\mathbf{f}^{\text{body}} = \rho \mathbf{g}$$

Particle Discretization

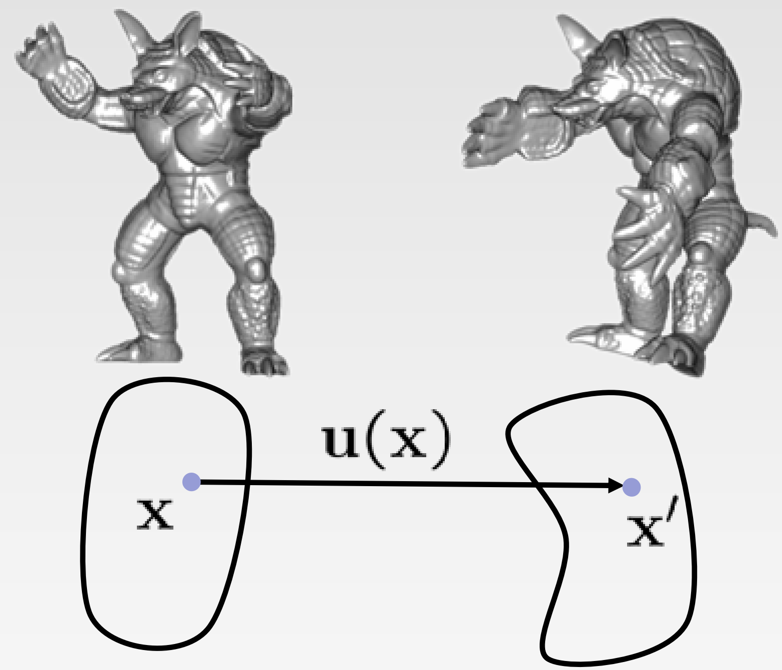
$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_i$$





Simulation Loop

$$\begin{tikzcd}
\mathbf{f}_{ext} \arrow[d] & \\
\mathbf{f}_t \arrow[d] & U_{t+\Delta t} \arrow[l] \\
\mathbf{u}_{t+\Delta t} \arrow[d] & \sigma_{t+\Delta t} \arrow[u] \\
\nabla \mathbf{u}_{t+\Delta t} \arrow[r] & \epsilon_{t+\Delta t} \arrow[u]
\end{tikzcd}$$



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

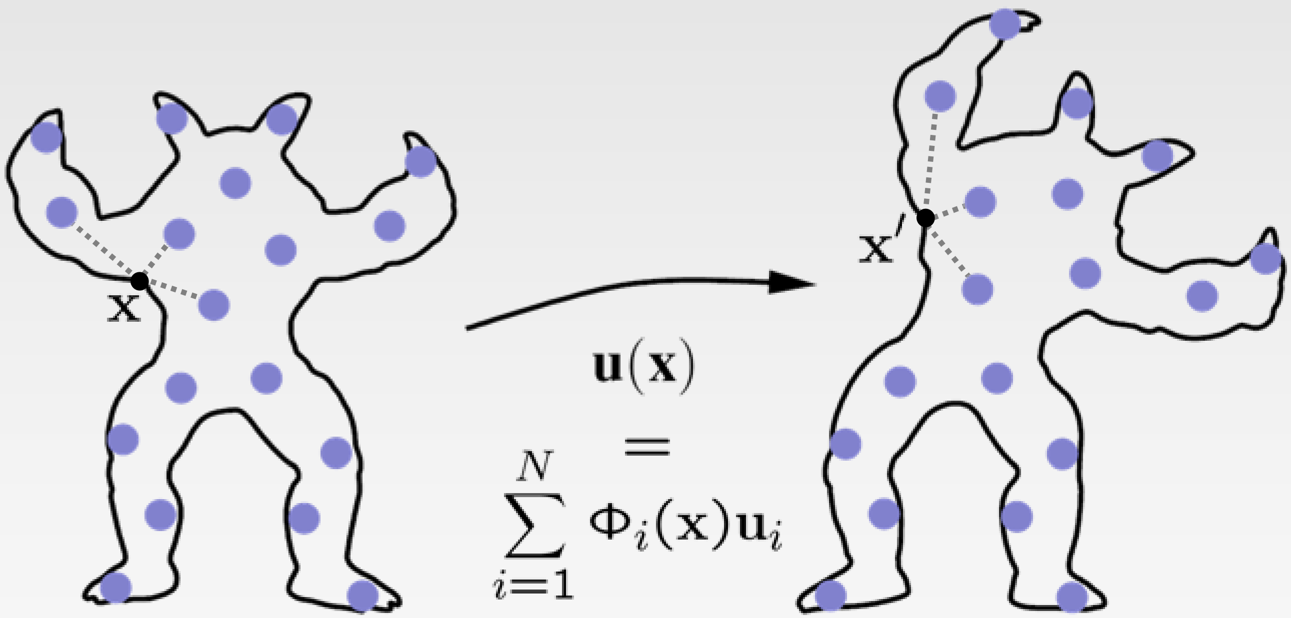
Surface Animation

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

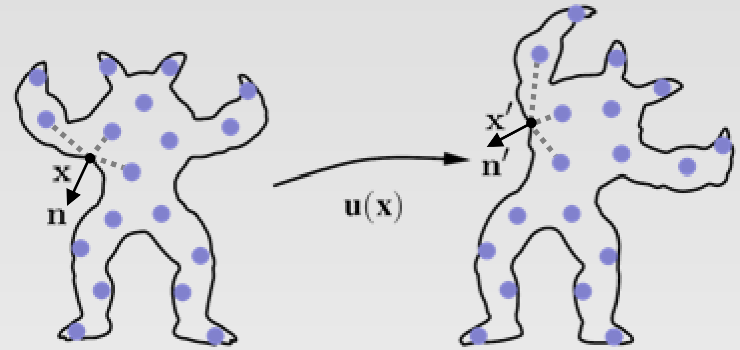


Can use whatever representation:
triangle meshes, point clouds, …

Surface Animation – Alternative 1

Vertex position update

$$\mathbf{x}' = \mathbf{x} + \mathbf{u}(\mathbf{x})$$



Approximate normal update

- first-order Taylor for displacement field at normal tip

$$\mathbf{u}(\mathbf{x}+\mathbf{n}) \approx \mathbf{u}(\mathbf{x}) + \nabla\mathbf{u}(\mathbf{x})^T\mathbf{n}$$

- tip is transformed to

$$\begin{aligned}(\mathbf{x}+\mathbf{n})' &= \mathbf{x}+\mathbf{n}+\mathbf{u}(\mathbf{x})+\nabla\mathbf{u}(\mathbf{x})\mathbf{n}\\ \mathbf{x}'+\mathbf{n}' &= \mathbf{x}'+\mathbf{n}+\nabla\mathbf{u}(\mathbf{x})\mathbf{n}\\ \mathbf{n}' &= \mathbf{n}+\nabla\mathbf{u}(\mathbf{x})\mathbf{n}\end{aligned}$$

Surface Animation – Alternative 1

Easy GPU Implementation

$$\mathbf{x}' = \mathbf{x} + \mathbf{u}(\mathbf{x}) = \mathbf{x} + \sum_{i=1}^{N} \boxed{\Phi_i(\mathbf{x})}\mathbf{u}_i$$

scalars
remain constant

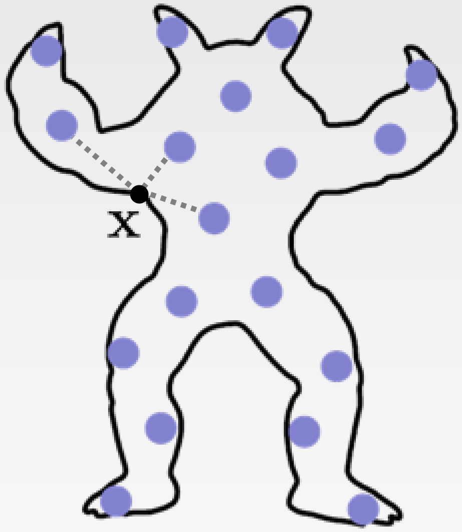
$$\mathbf{n}' = \mathbf{n} + \nabla\mathbf{u}(\mathbf{x})^T\mathbf{n} = \mathbf{n} + \sum_{i=1}^{N} \boxed{(\nabla\Phi_i^T(\mathbf{x})\mathbf{n})}\mathbf{u}_i$$

→ only have to send particle deformations to the GPU

Surface Animation – Alternative 2

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

$$\tilde{\mathbf{u}}(\mathbf{x}) = \sum_j \bar{\omega}_{ij} \left(\mathbf{u}_j + \nabla \mathbf{u}(\mathbf{x}_j)^T (\mathbf{x} - \mathbf{x}_j) \right)$$



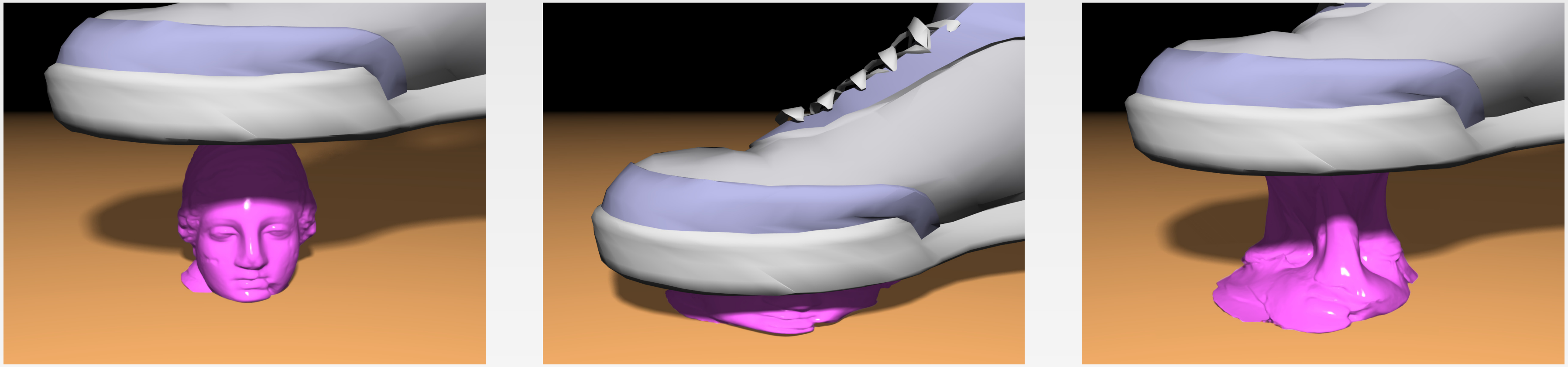
Updated vertex position

$$\mathbf{x}' = \mathbf{x} + \tilde{\mathbf{u}}(\mathbf{x})$$

→ 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

$$\boldsymbol{\epsilon}_i^{\text{elastic}} = \boldsymbol{\epsilon}_i - \boxed{\boldsymbol{\epsilon}_i^{\text{plastic}}}$$

strain state variable

$$\mathbf{f}_i^{\text{elastic}} = -\frac{1}{2}\nabla_{\mathbf{u}_i}\boldsymbol{\epsilon}_i^{\text{elastic}} \cdot \boldsymbol{\sigma}_i$$

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:

$$\text{if } \|\epsilon_i^{\text{plastic}}\| > c_{\text{max}} \text{ then } \epsilon_i^{\text{plastic}} \leftarrow \epsilon_i^{\text{plastic}} \cdot c_{\text{max}} / \|\epsilon_i^{\text{plastic}}\|$$

Plasticity

Update the reference shape and
store the plastic strain state variables

$$
\begin{aligned}
\boldsymbol{\epsilon}_i^{\mathsf{plastic}} &\leftarrow \boldsymbol{\epsilon}_i^{\mathsf{plastic}} - \boldsymbol{\epsilon}_i, \\
\mathbf{x}_i &\leftarrow \mathbf{x}_i + \mathbf{u}_i, \\
\mathbf{u}_i &\leftarrow 0.
\end{aligned}
$$

Ductile Fracture

Initial statistics:

2.2k nodes

134k surfels

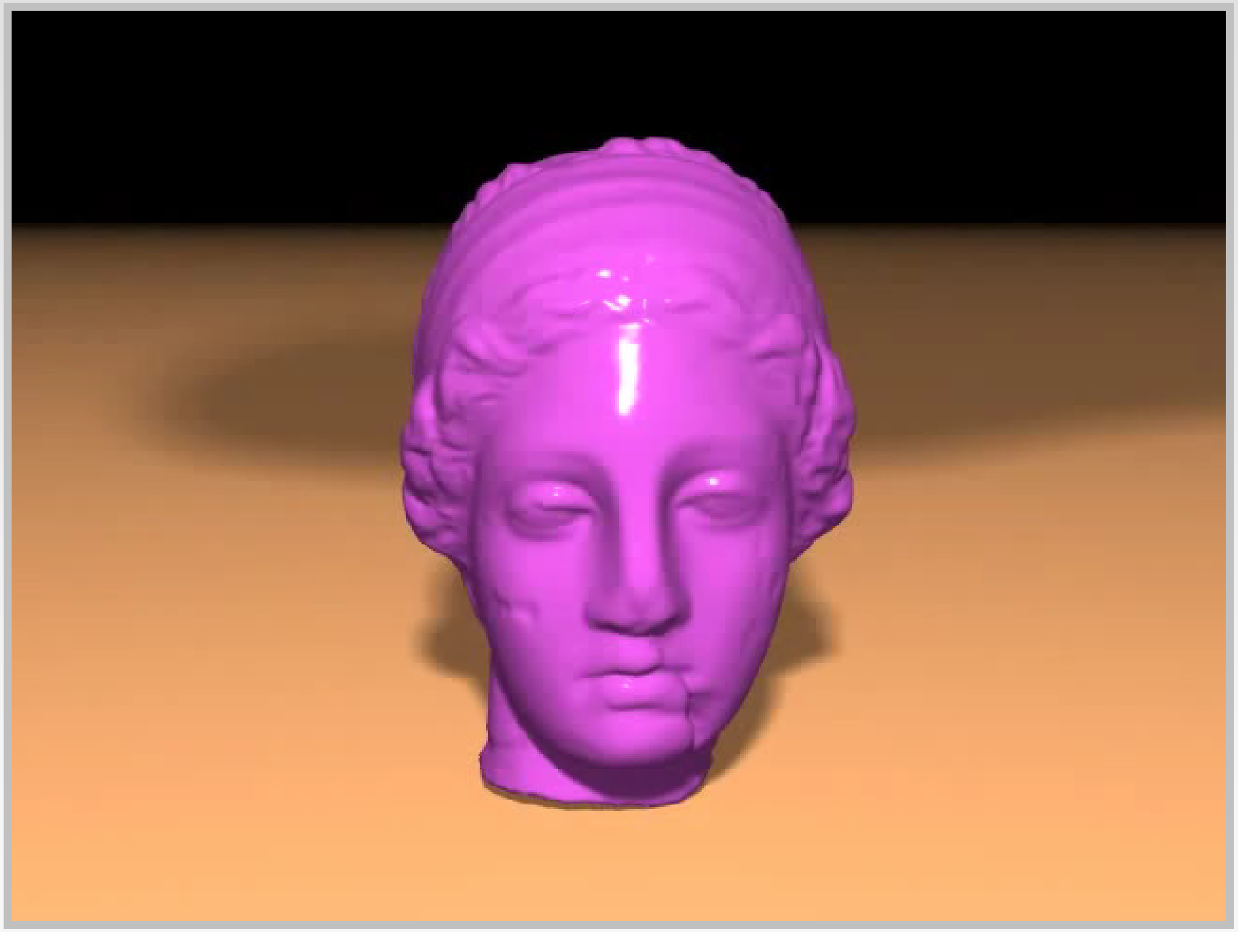
Final statistics:

3.3k nodes

144k surfels

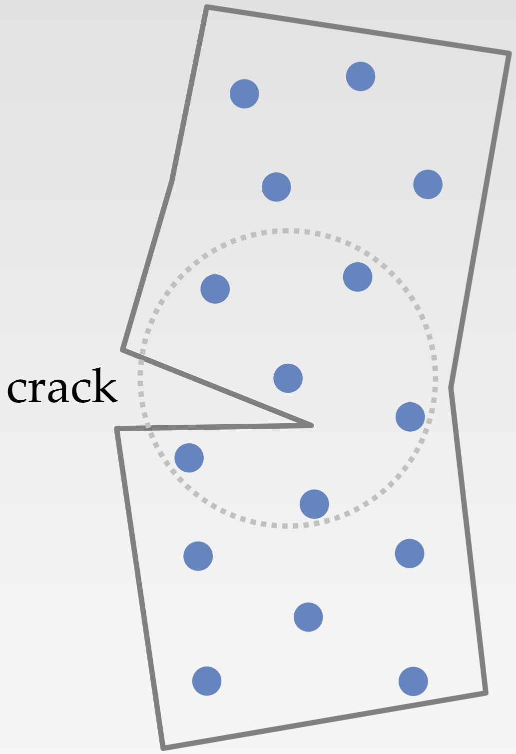
Simulation time:

23 sec/frame



Modeling Discontinuities

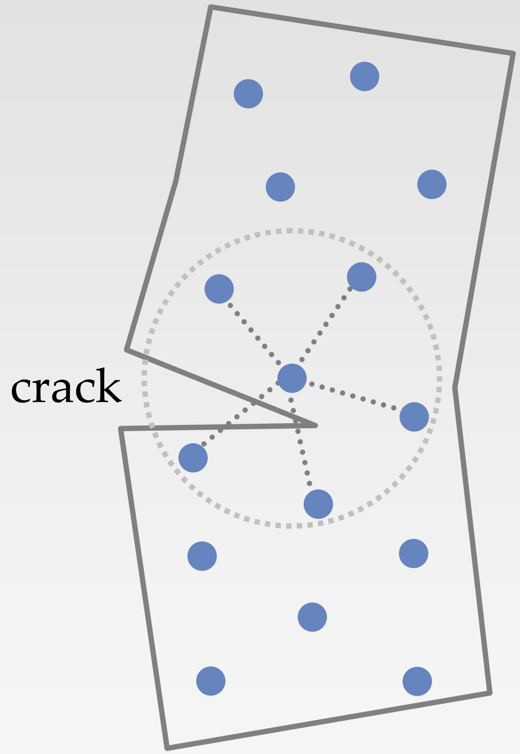
Only visible nodes should interact



Modeling Discontinuities

Only visible nodes should interact

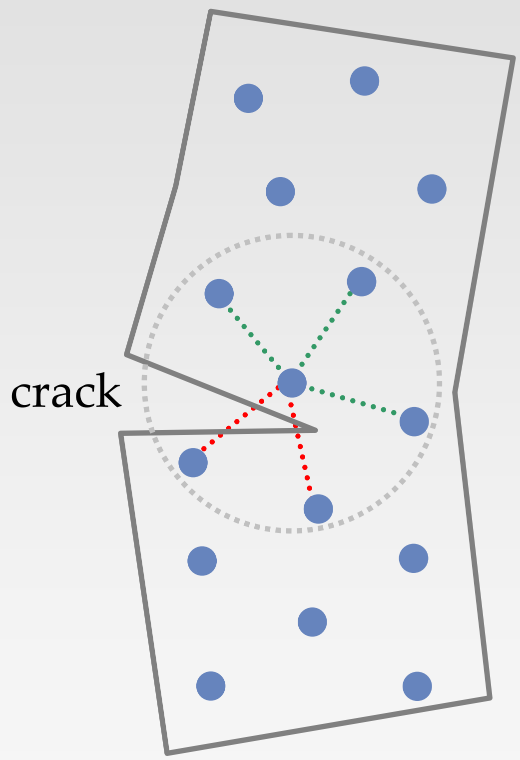
- collect nearest neighbors



Modeling Discontinuities

Only visible nodes should interact

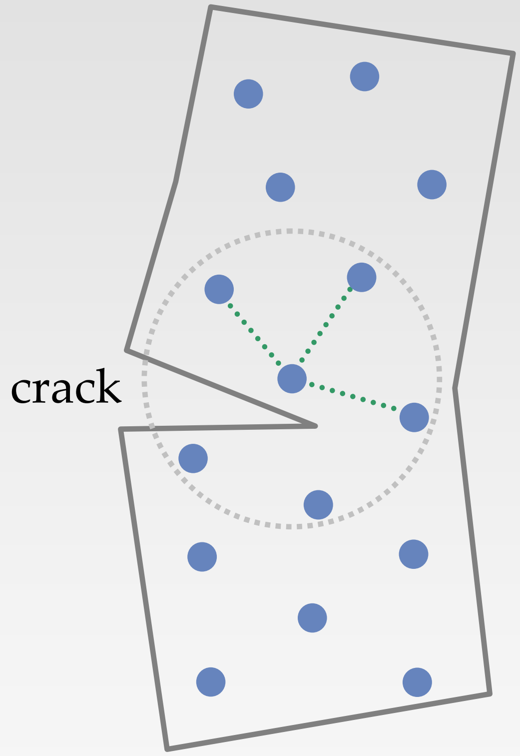
- collect nearest neighbors
- perform visibility test



Modeling Discontinuities

Only visible nodes should interact

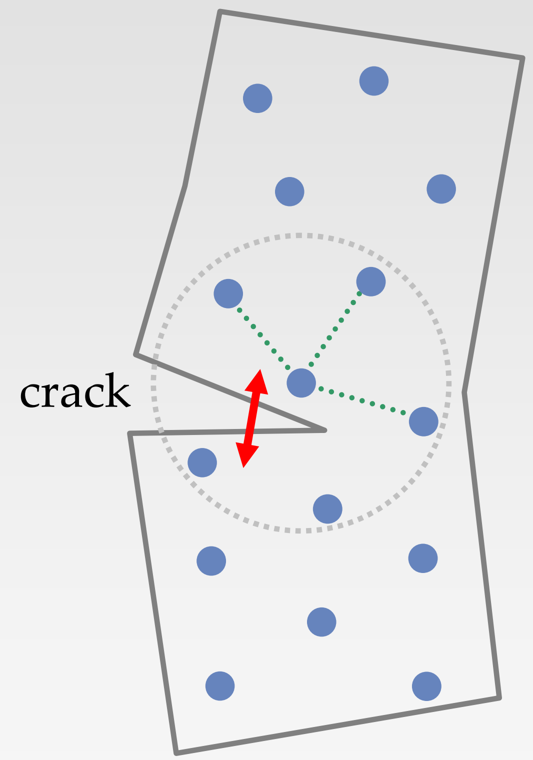
- collect nearest neighbors
- perform visibility test



Modeling Discontinuities

Only visible nodes should interact

Discontinuity along the crack surfaces



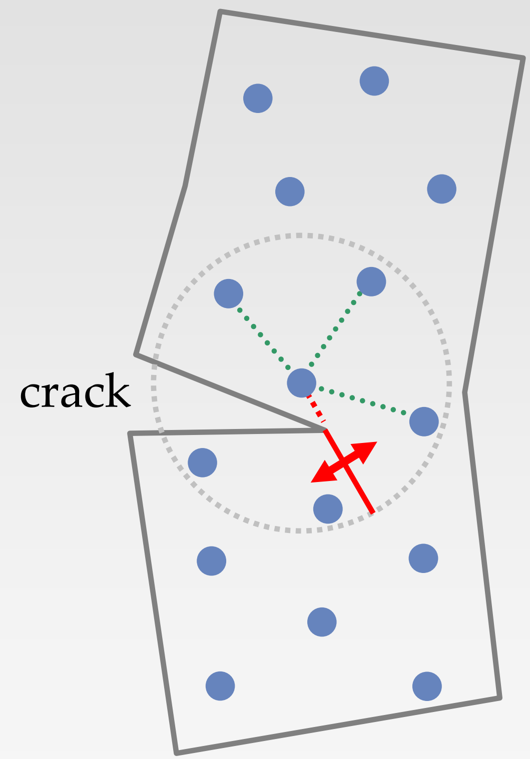
Modeling Discontinuities

Only visible nodes
should interact

Discontinuity along the
crack surfaces

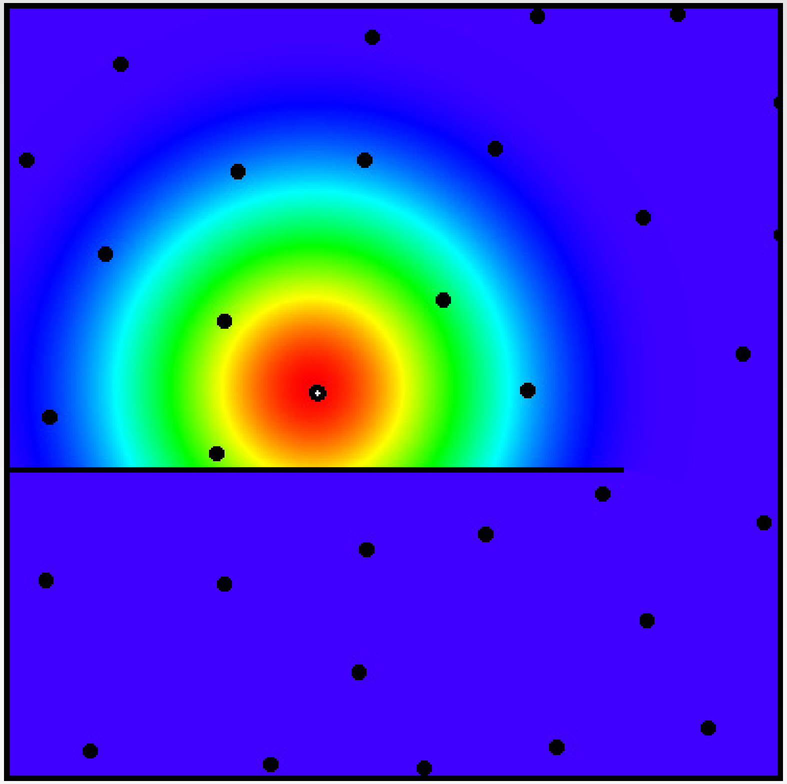
But also within the domain

→ undesirable!

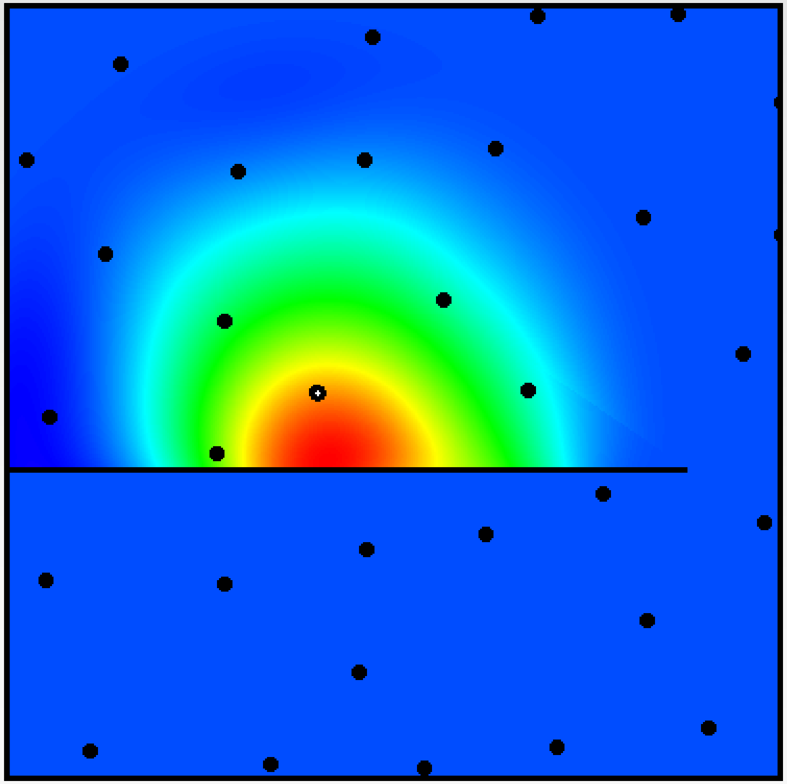


Modeling Discontinuities

Visibility Criterion



Weight function



Shape function

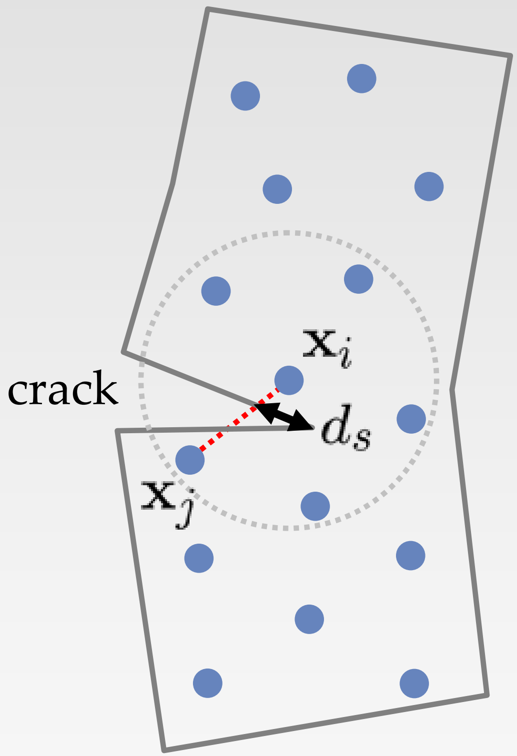
Modeling Discontinuities

Solution: transparency method[1]

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

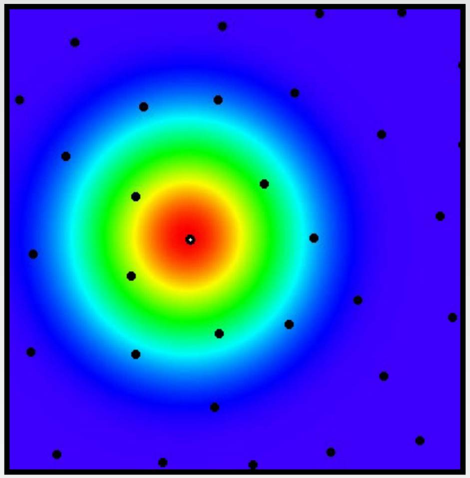
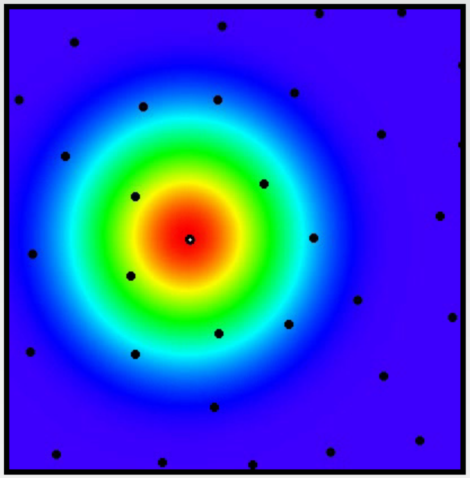
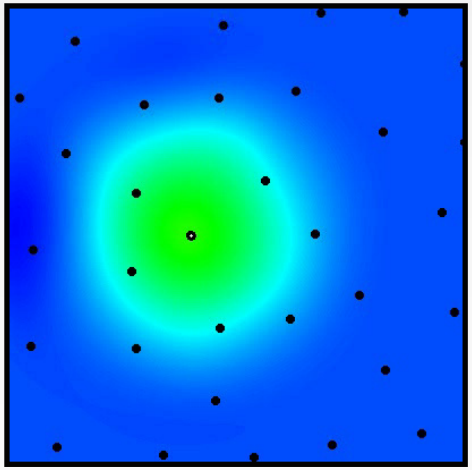
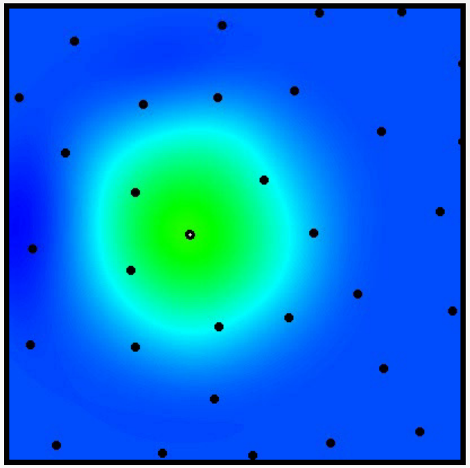
$$\omega'_{ij} = \omega(\|\mathbf{x}_j - \mathbf{x}_i\|/h_i + (d_s/\kappa h)^2)$$

$\rightarrow$ crack becomes transparent near the crack tip



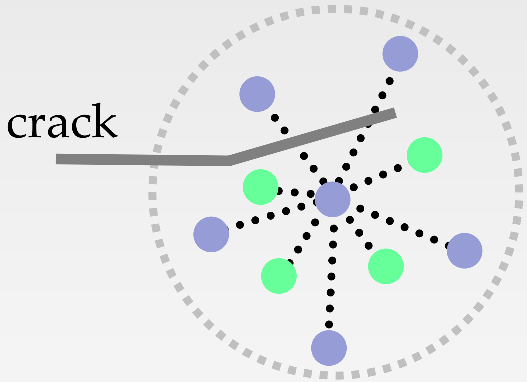
[1] Organ et al.: *Continuous Meshless Approximations for Nonconvex Bodies by Diffraction and Transparency*, Comp. Mechanics, 1996

Modeling Discontinuities

	Visibility Criterion	Transparency Method
Weight function		
Shape function		

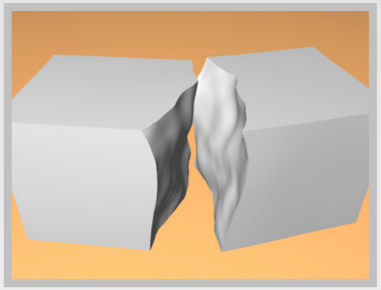
Re-sampling

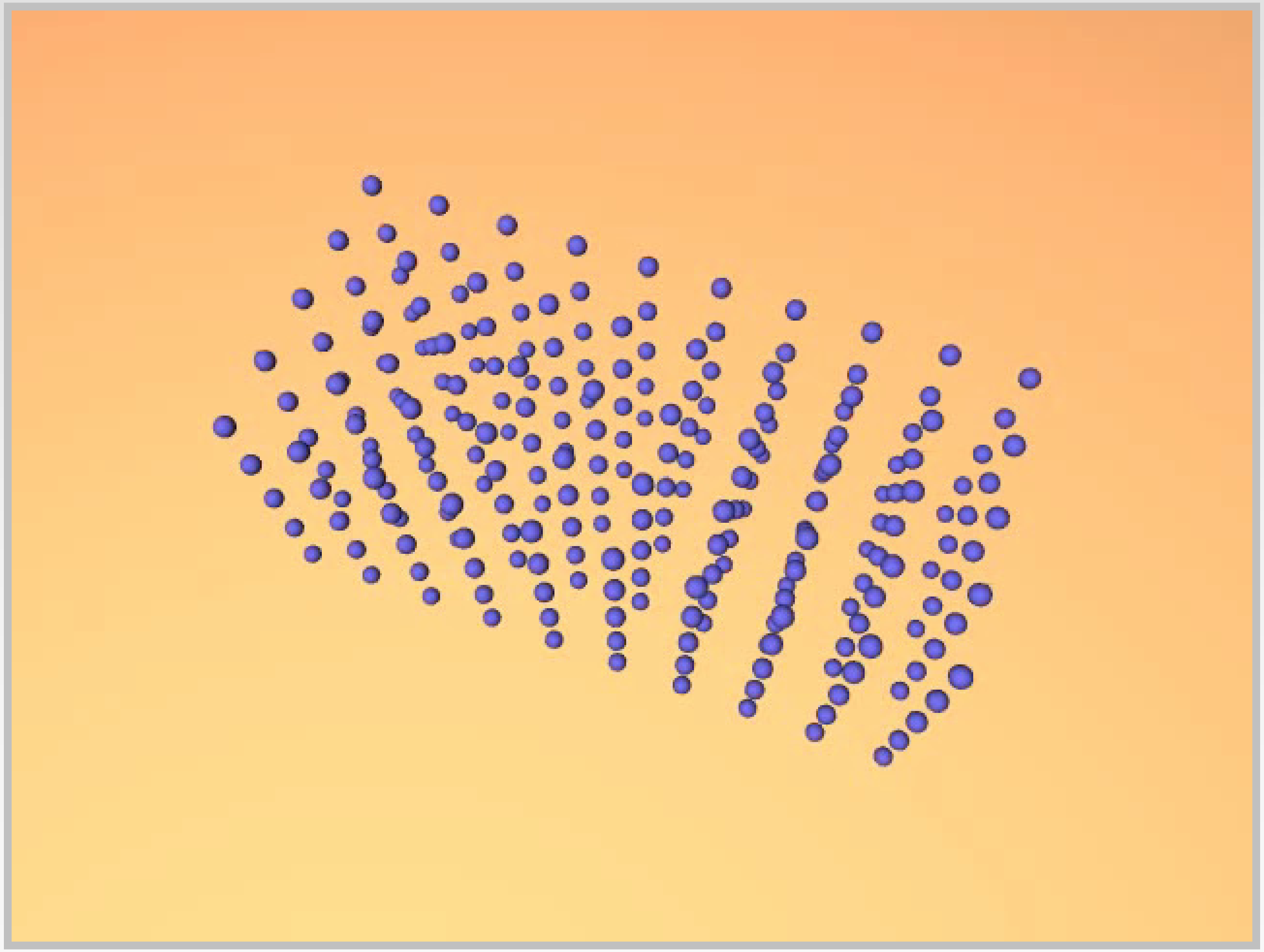
- Add simulation nodes when number of neighbors too small
- Local re-sampling of the domain of a node
  - distribute mass
  - adapt support radius
  - interpolate attributes



- Shape functions adapt automatically!

Re-sampling: Example





Brittle Fracture

Initial statistics:

4.3k nodes

249k surfels

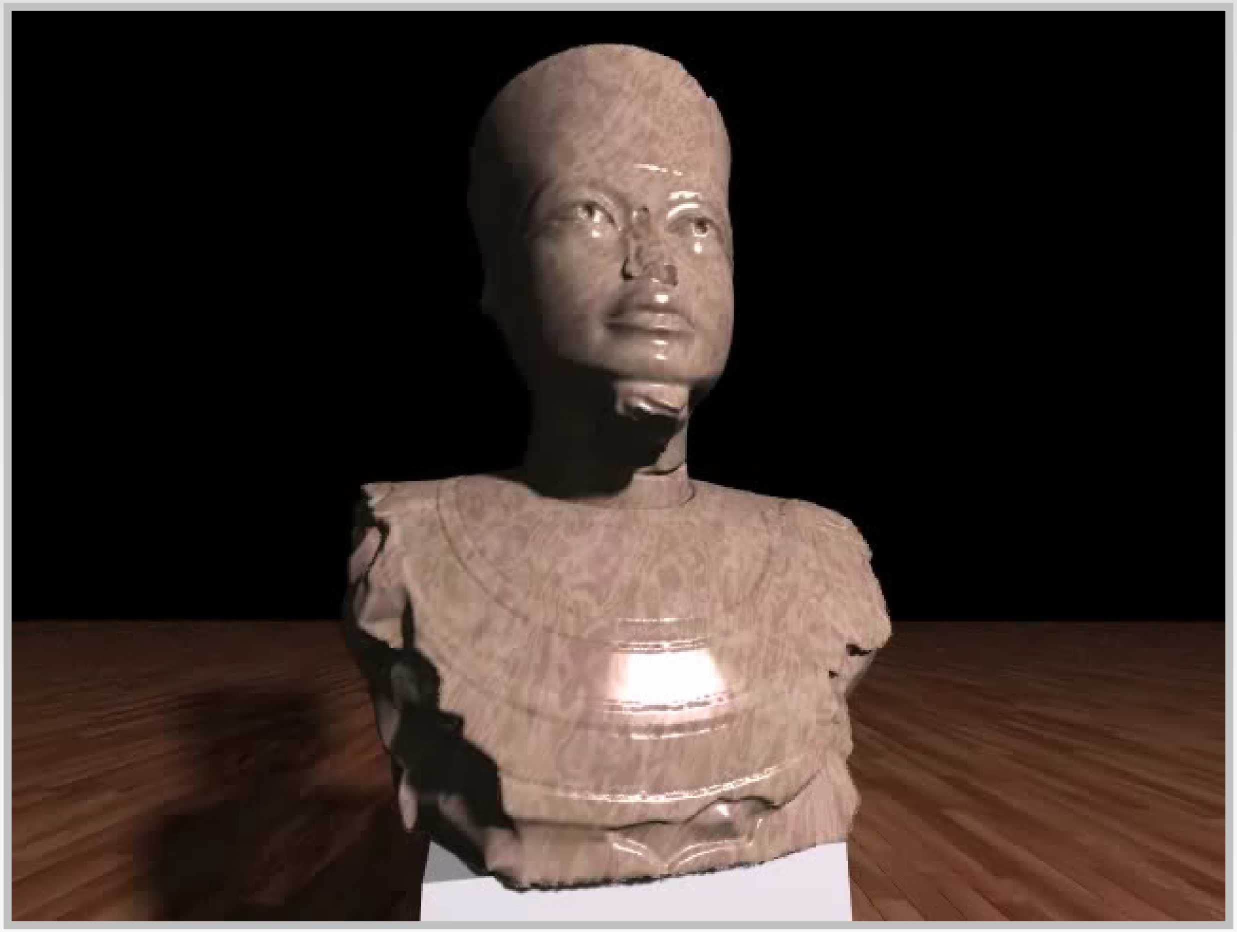
Final statistics:

6.5k nodes

310k surfels

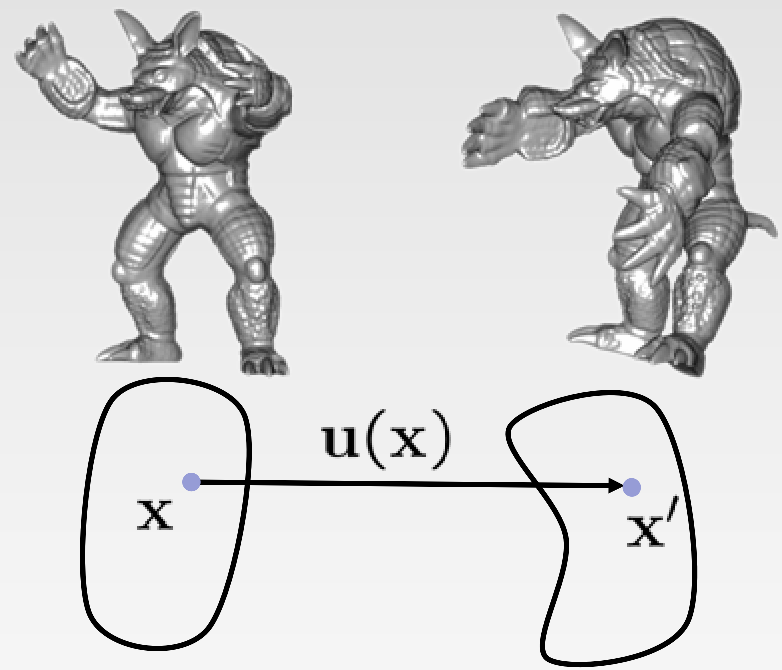
Simulation time:

22 sec/frame



Summary

$$\begin{tikzcd}
\mathbf{f}_{ext} \arrow[d] & \\
\mathbf{f}_t \arrow[d] & U_{t+\Delta t} \arrow[l] \\
\mathbf{u}_{t+\Delta t} \arrow[d] & \sigma_{t+\Delta t} \arrow[u] \\
\nabla \mathbf{u}_{t+\Delta t} \arrow[r] & \epsilon_{t+\Delta t} \arrow[u]
\end{tikzcd}$$



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

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

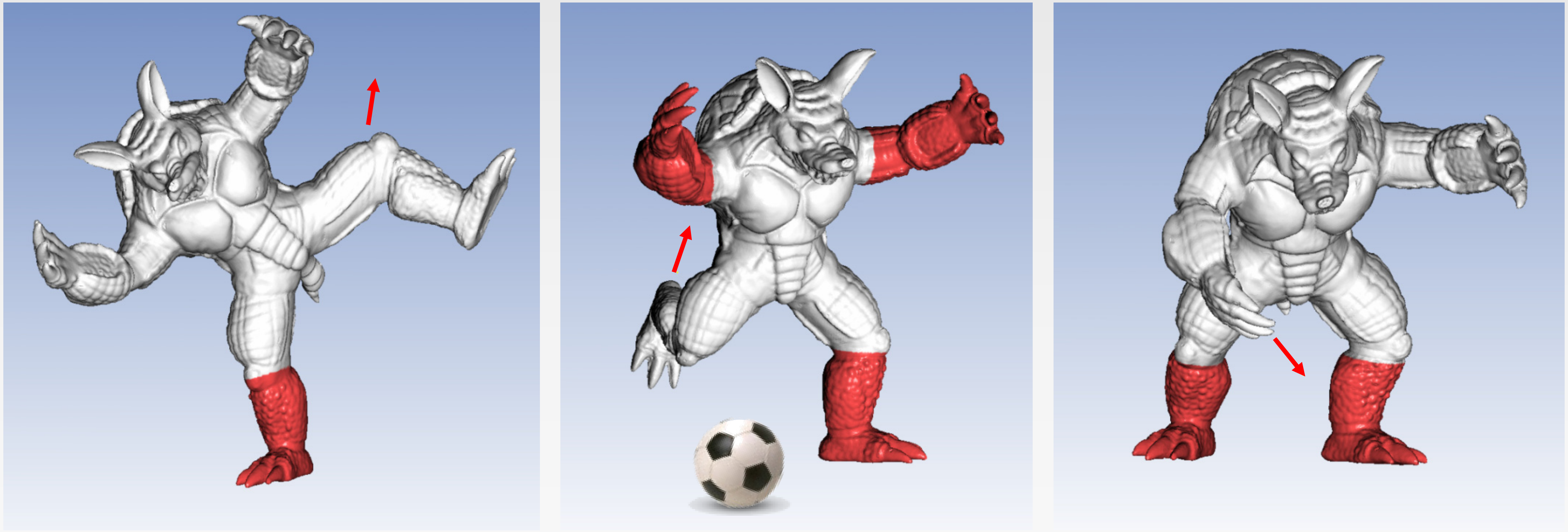
Application 3:

Shape & Motion Modeling

Shape Deformations

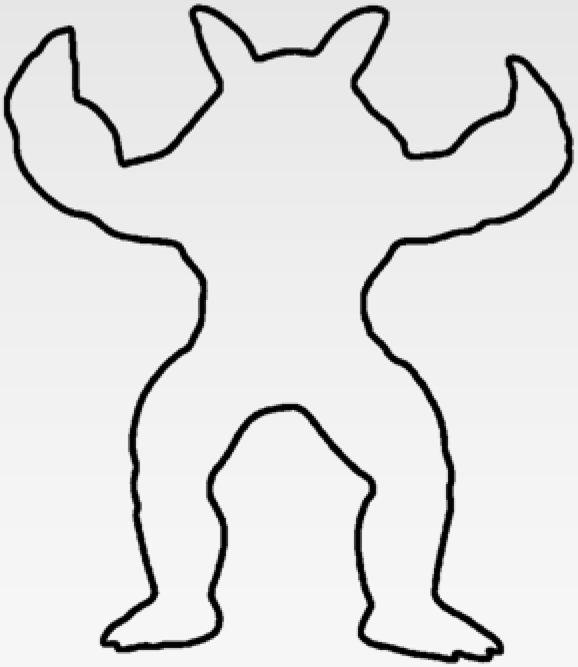
Shape Deformations: Objective

Find a realistic shape deformation given the user's input constraints.

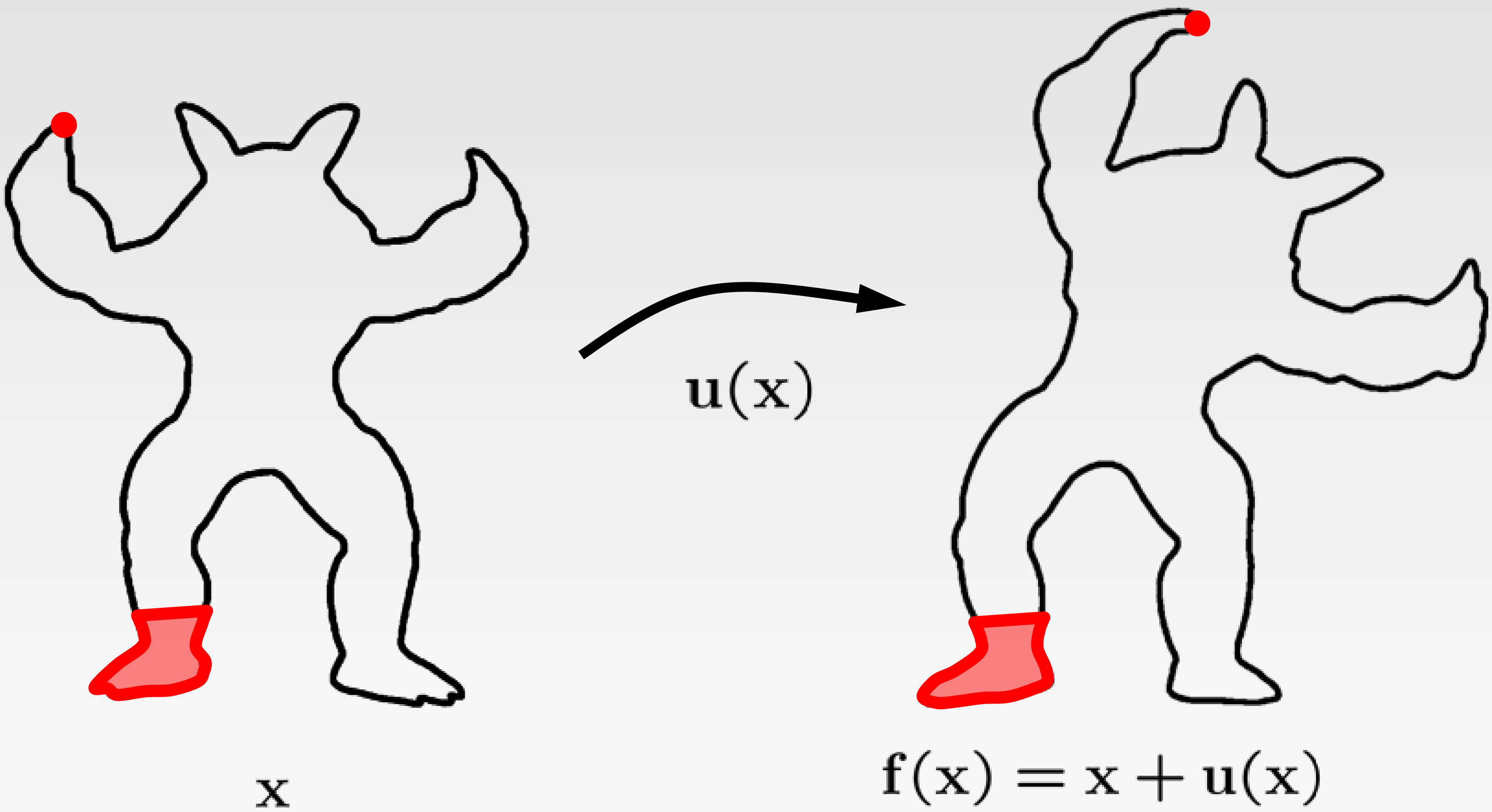


Shape Deformations

Shape Deformations

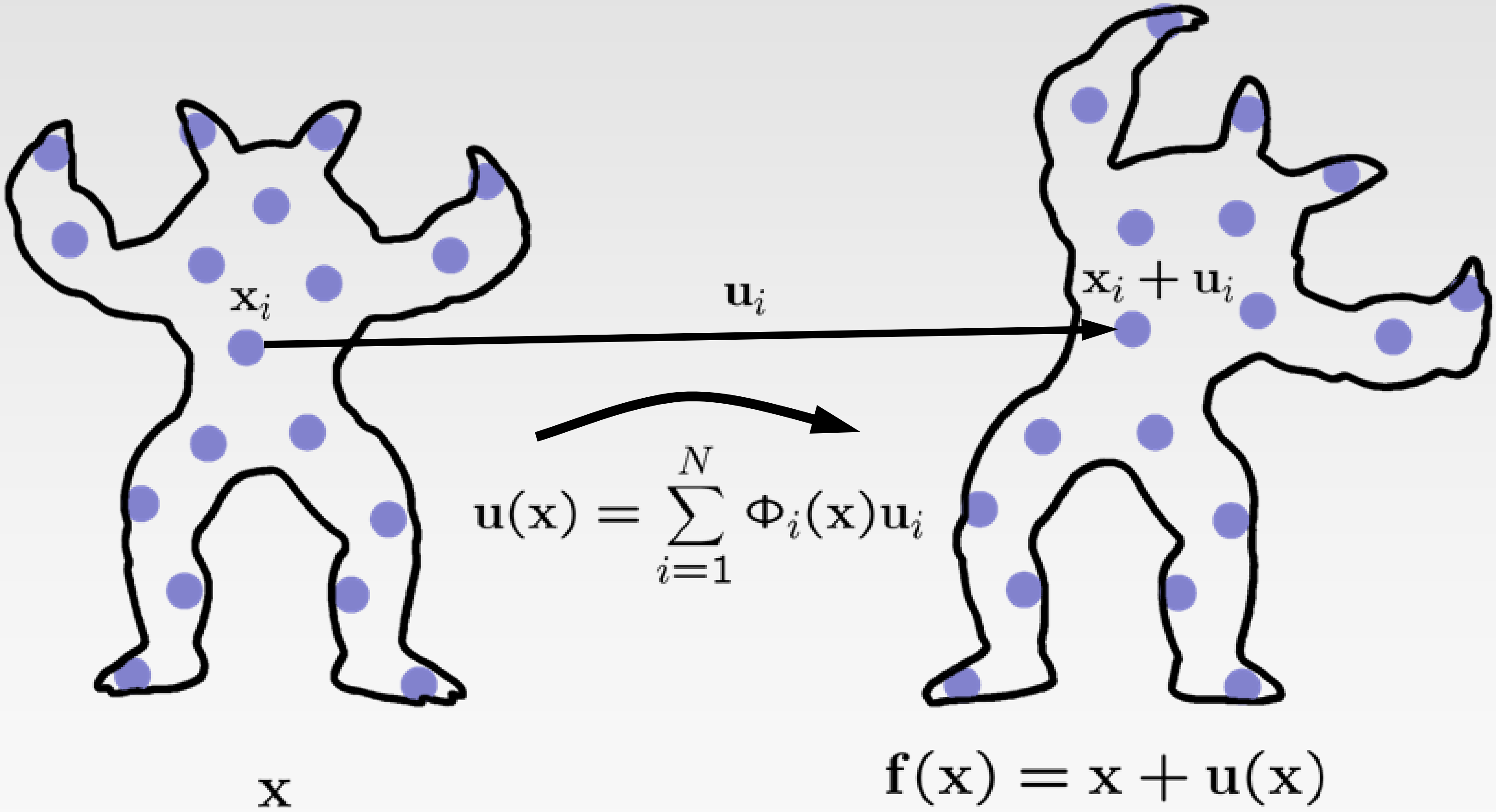


Shape Deformations



Deformation Field Representation

Use meshless shape functions to define
a continuous deformation field.



Deformation Field Representation

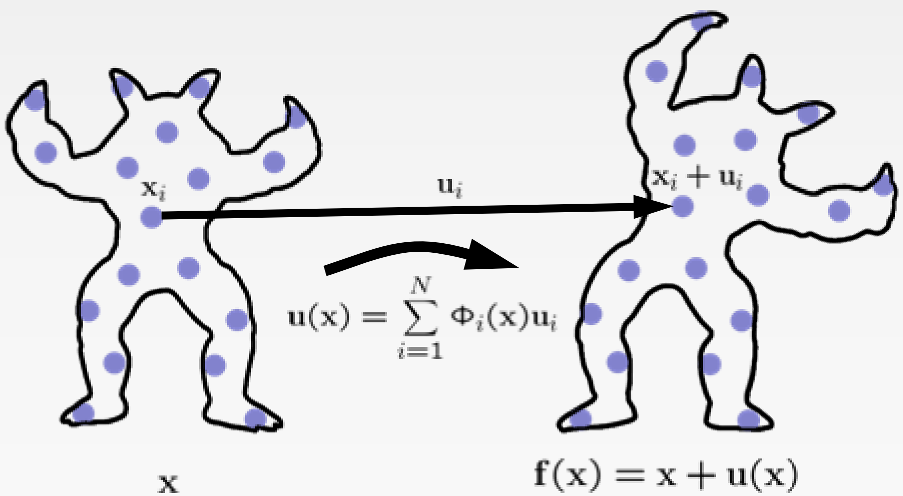
$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\mathbf{u}_i$$

Precompute for every node and neighbor

$$\Phi_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|/h_i)\mathbf{p}(\mathbf{x})^T \mathbf{M}(\mathbf{x})^{-1}\mathbf{p}(\mathbf{x}_i)$$

Complete linear basis in 3D

$$\mathbf{p}(\mathbf{x}) = [1 \;\; x \;\; y \;\; z]^T$$

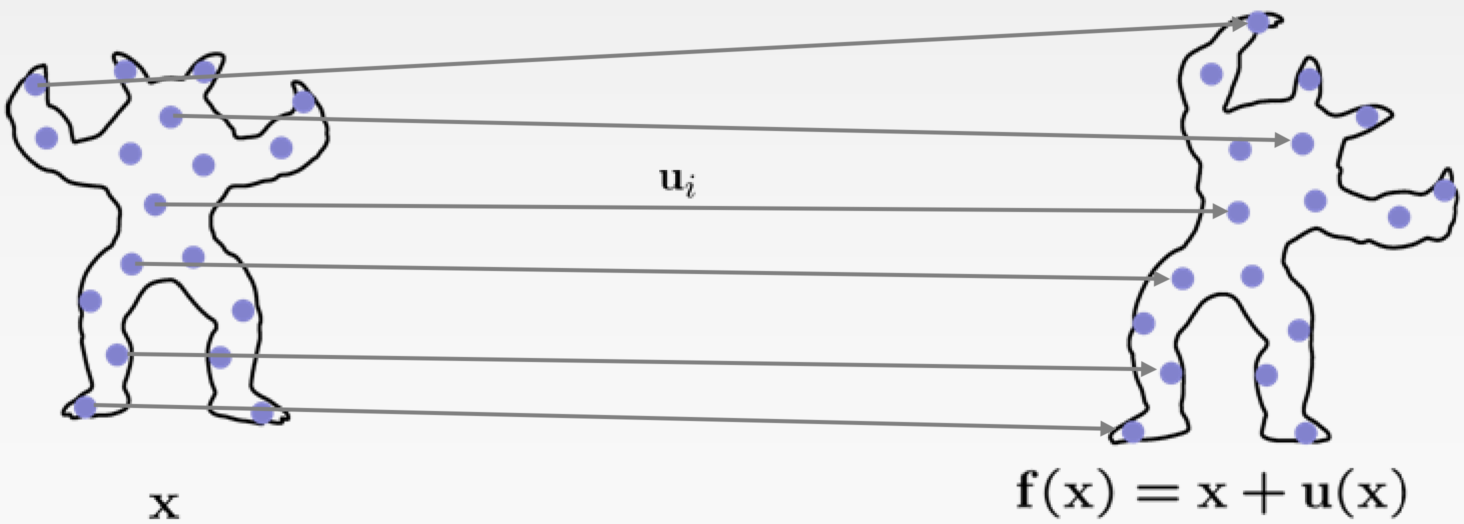


Deformation Field Optimization

We are optimizing the displacement field

$$\mathbf{f}(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x})$$
$$= \mathbf{x} + \sum_{i=1}^{N} \Phi_i(\mathbf{x}) \boxed{\mathbf{u}_i}$$

nodal deformations
unknowns to solve for



Deformation Field Optimization

The displacement field should
satisfy the input constraints.

Position constraint

$$\|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|^2 \rightarrow \text{min}$$

$$\|\mathbf{x} + \mathbf{u}(\mathbf{x}) - \mathbf{y}\|^2 \rightarrow \text{min}$$

$$\|\mathbf{x} + \sum_i \Phi_i(\mathbf{x})\mathbf{u}_i - \mathbf{y}\|^2 \rightarrow \text{min}$$

→ quadratic in the unknowns

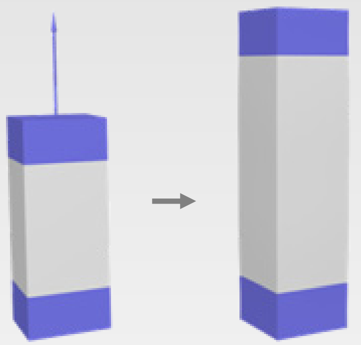
Deformation Field Optimization

The displacement should be realistic.

Locally rigid (minimal strain)

$$\{\nabla \mathbf{f}^T(\mathbf{x})\nabla \mathbf{f}(\mathbf{x}) - \mathbf{I}\}^2 \rightarrow \text{min}$$

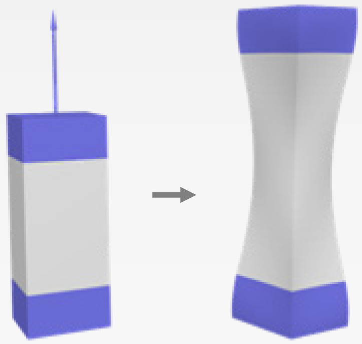
$$\{\nabla \mathbf{u}(\mathbf{x}) + \nabla \mathbf{u}^T(\mathbf{x}) + \nabla \mathbf{u}^T(\mathbf{x})\nabla \mathbf{u}(\mathbf{x})\}^2 \rightarrow \text{min}$$



Volume preserving

$$\{|\nabla \mathbf{f}(\mathbf{x})| - 1\}^2 \rightarrow \text{min}$$

$$\{|\mathbf{I} + \nabla \mathbf{u}(\mathbf{x})| - 1\}^2 \rightarrow \text{min}$$



$\rightarrow$ degree 6 in the unknowns
$\rightarrow$ non-linear problem

Deformation Field Optimization

The total energy to minimize

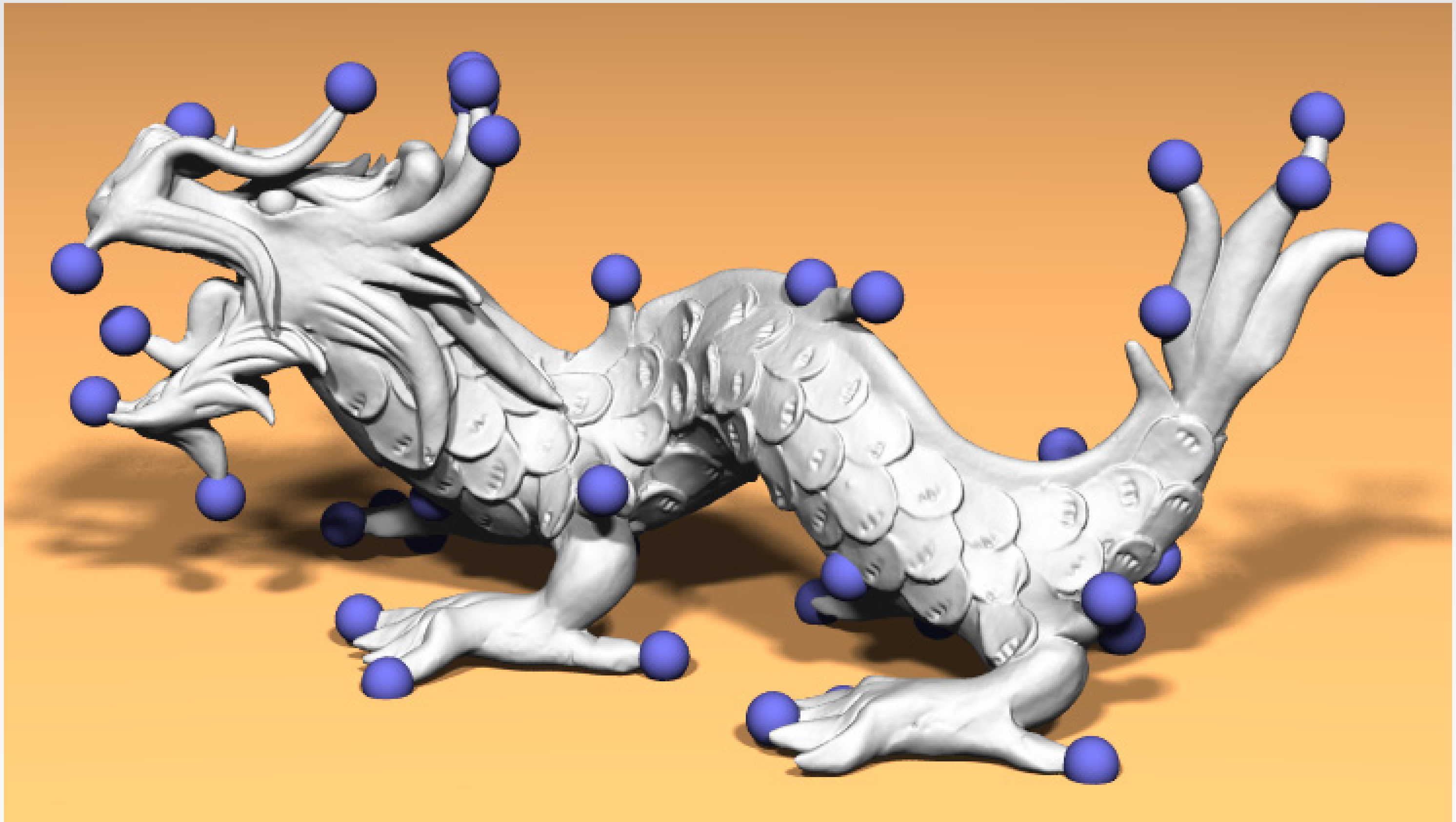
$$E = \sum_{constraints} \|\mathbf{x} + \sum_i \Phi_i(\mathbf{x})\mathbf{u}_i - \mathbf{y}\|^2$$
$$+ \sum_{nodes\ i} \|\nabla \mathbf{f}^T(\mathbf{x}_i)\nabla \mathbf{f}(\mathbf{x}_i) - \mathbf{I}\|_F^2$$
$$+ \sum_{nodes\ i} (|\nabla \mathbf{f}(\mathbf{x}_i)| - 1)^2$$

Solve using LBFGS

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

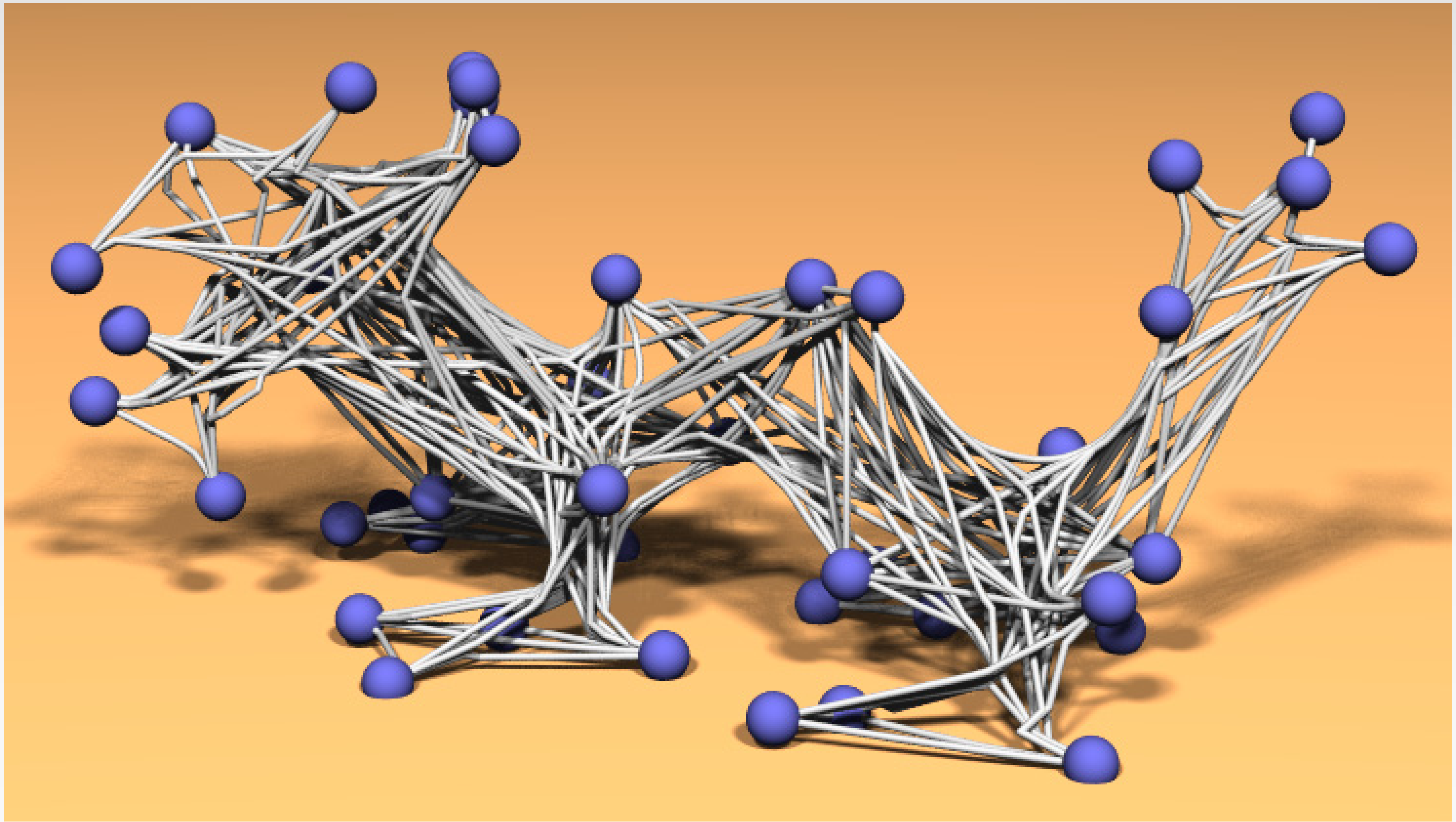
Nodal Sampling & Coupling

Keep number of unknowns as low as possible.



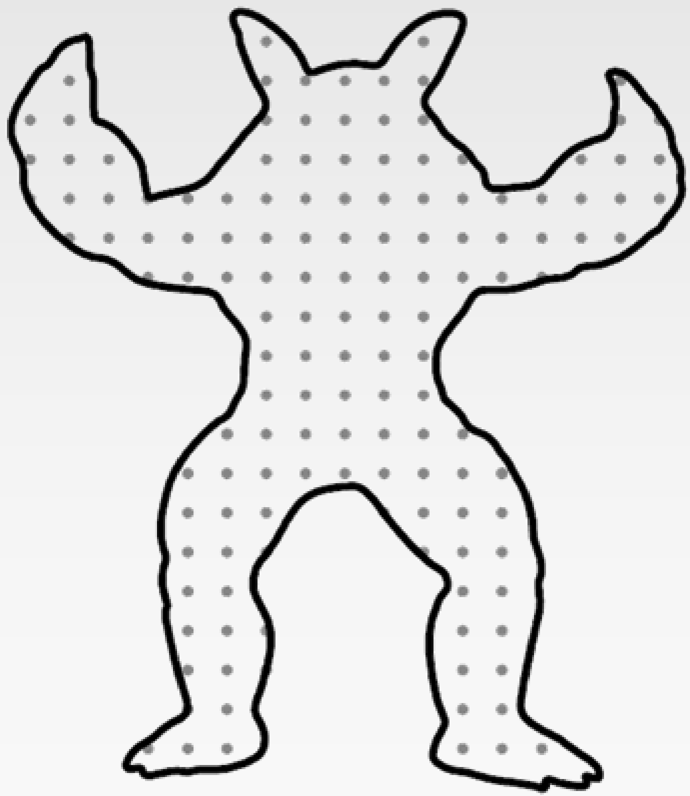
Nodal Sampling & Coupling

Ensure proper coupling by using material distance in weight functions.



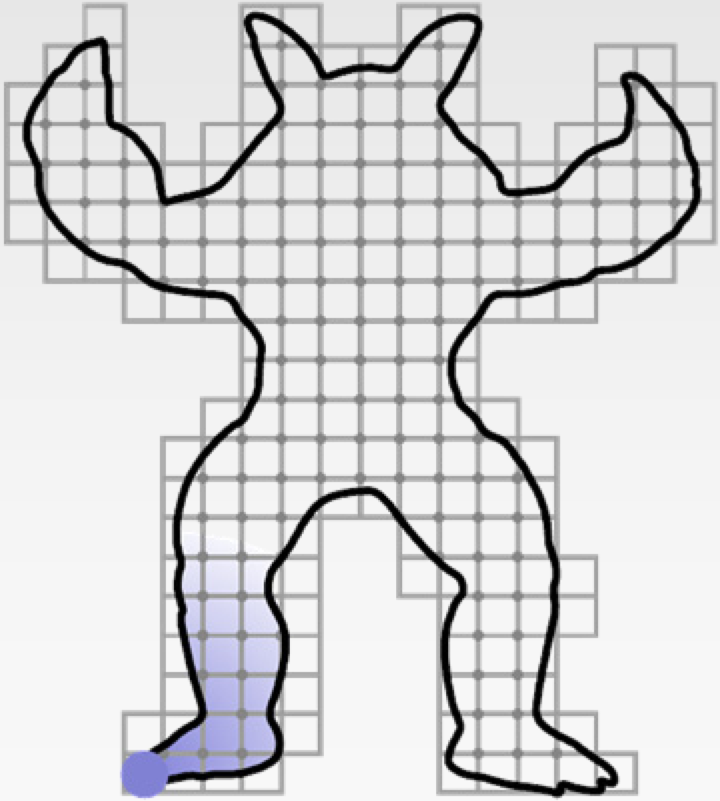
Nodal Sampling & Coupling

Set of candidate points:
vertices and interior set of dense grid points



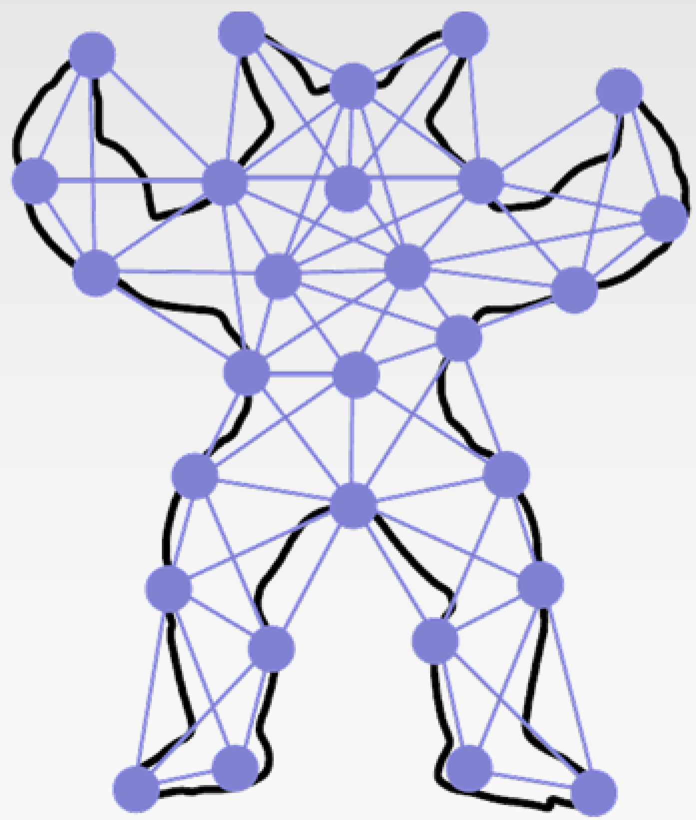
Nodal Sampling & Coupling

Grid-based fast marching to compute material distances.



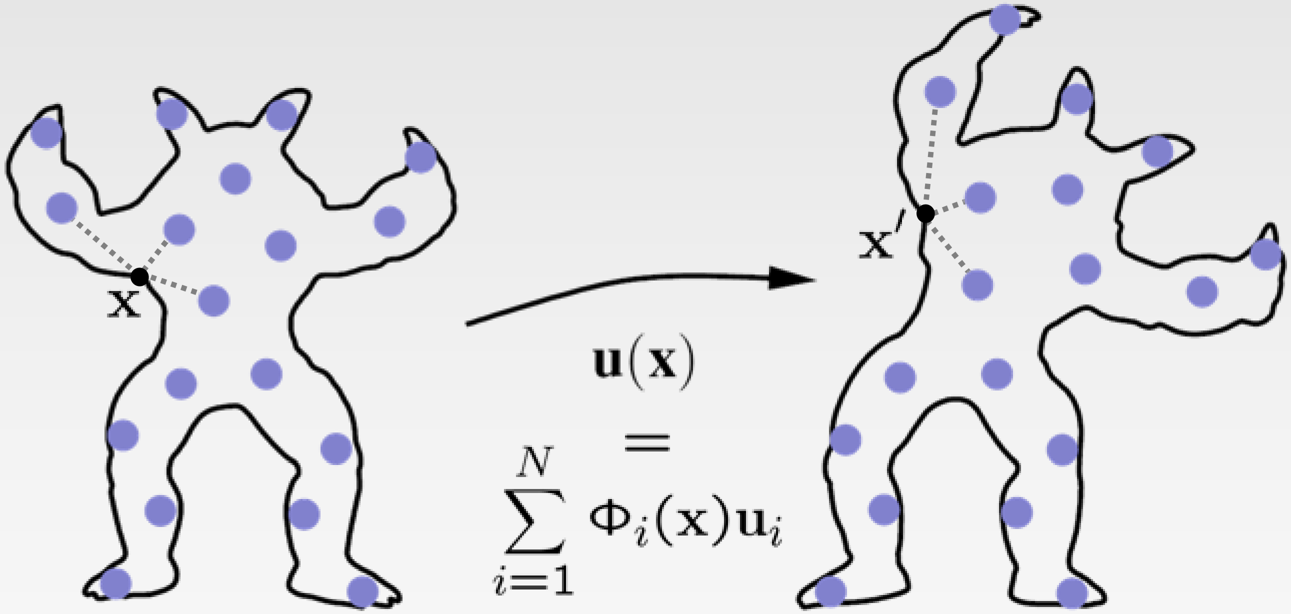
Nodal Sampling & Coupling

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



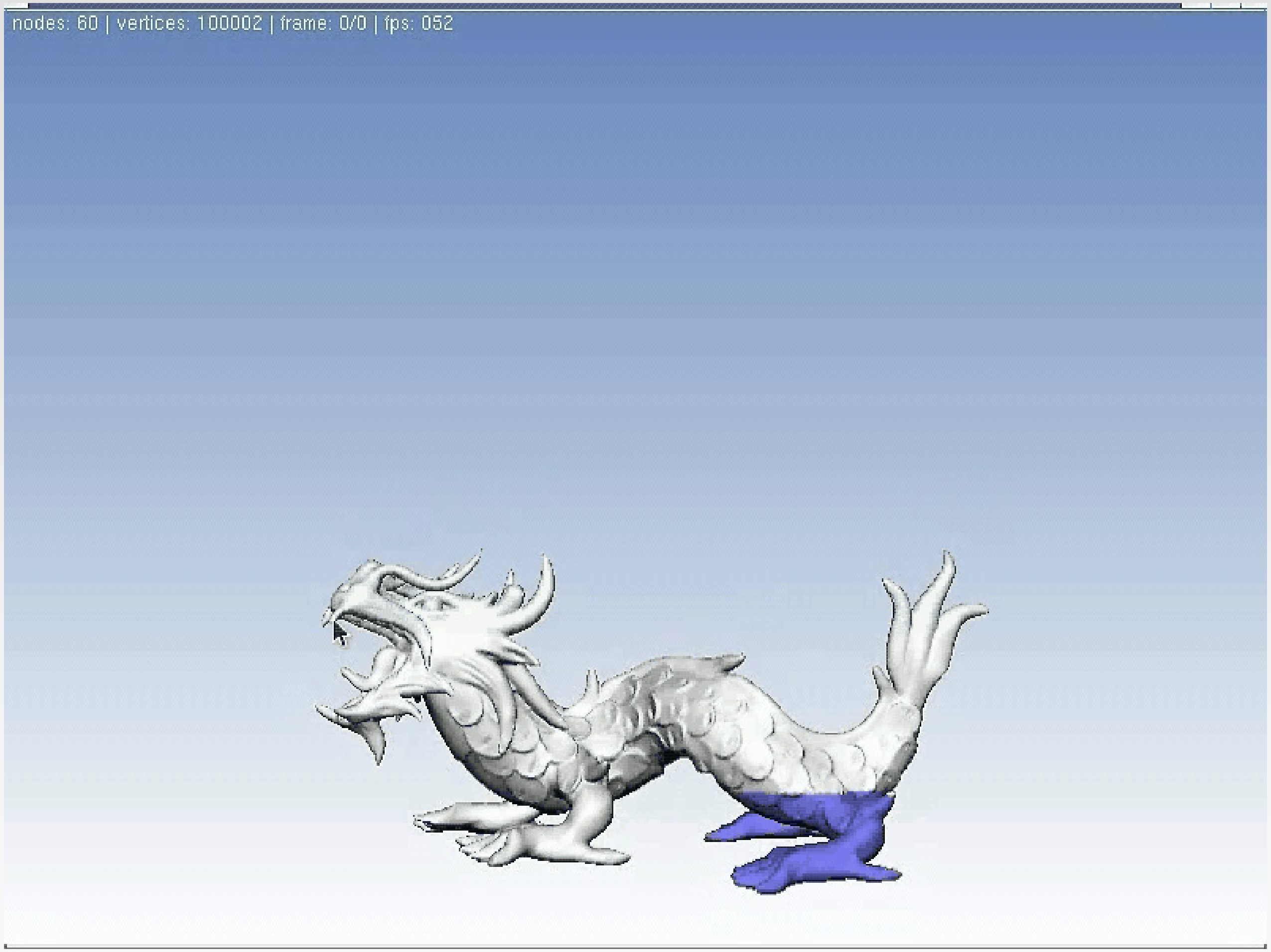
Surface Deformation

Use Alternative 1 of the surface animation algorithms discussed before



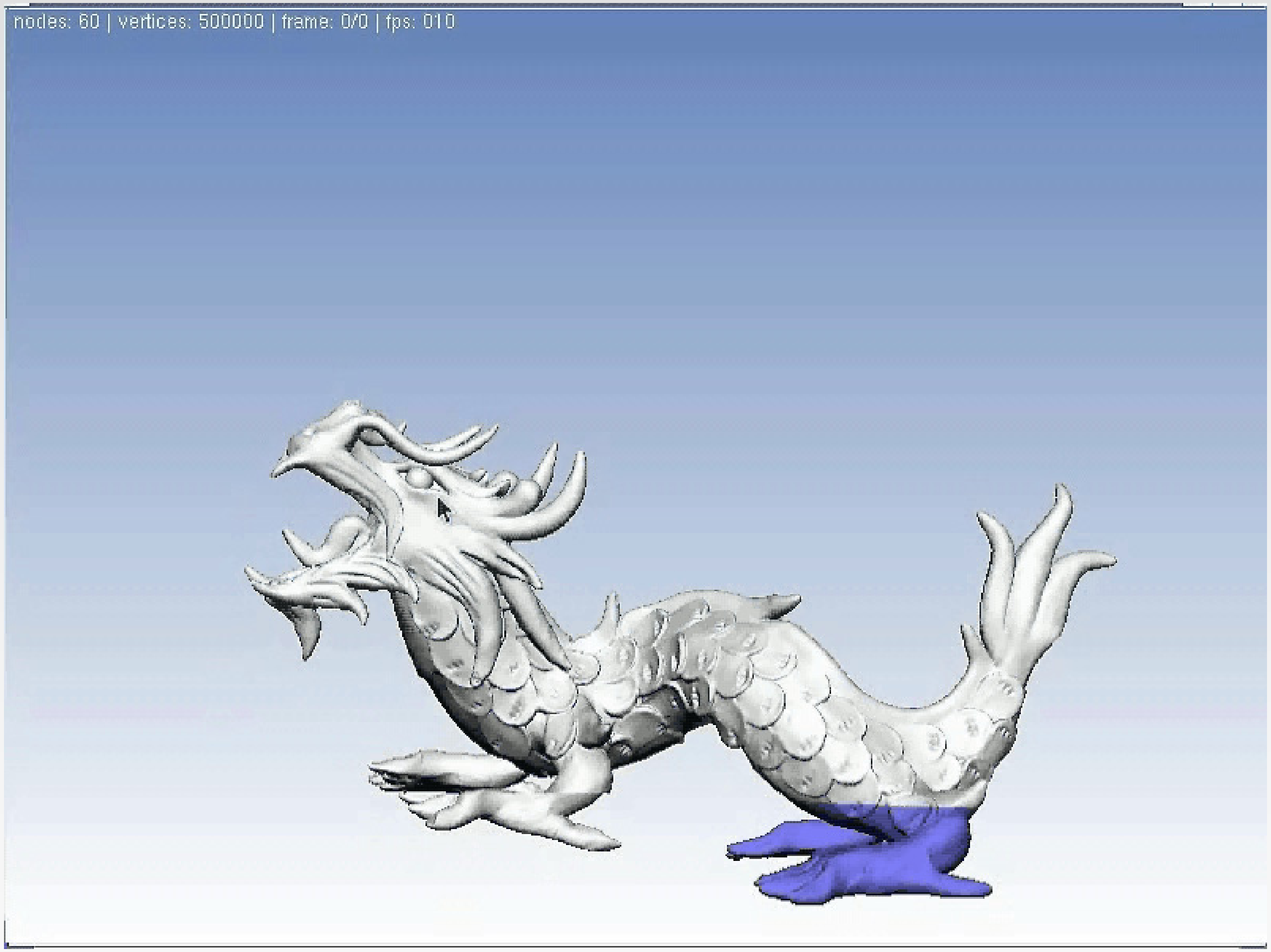
Vertex positions and normals updated on the GPU

Shape Deformations



100k vertices, 60 nodes → 55 fps

Shape Deformations

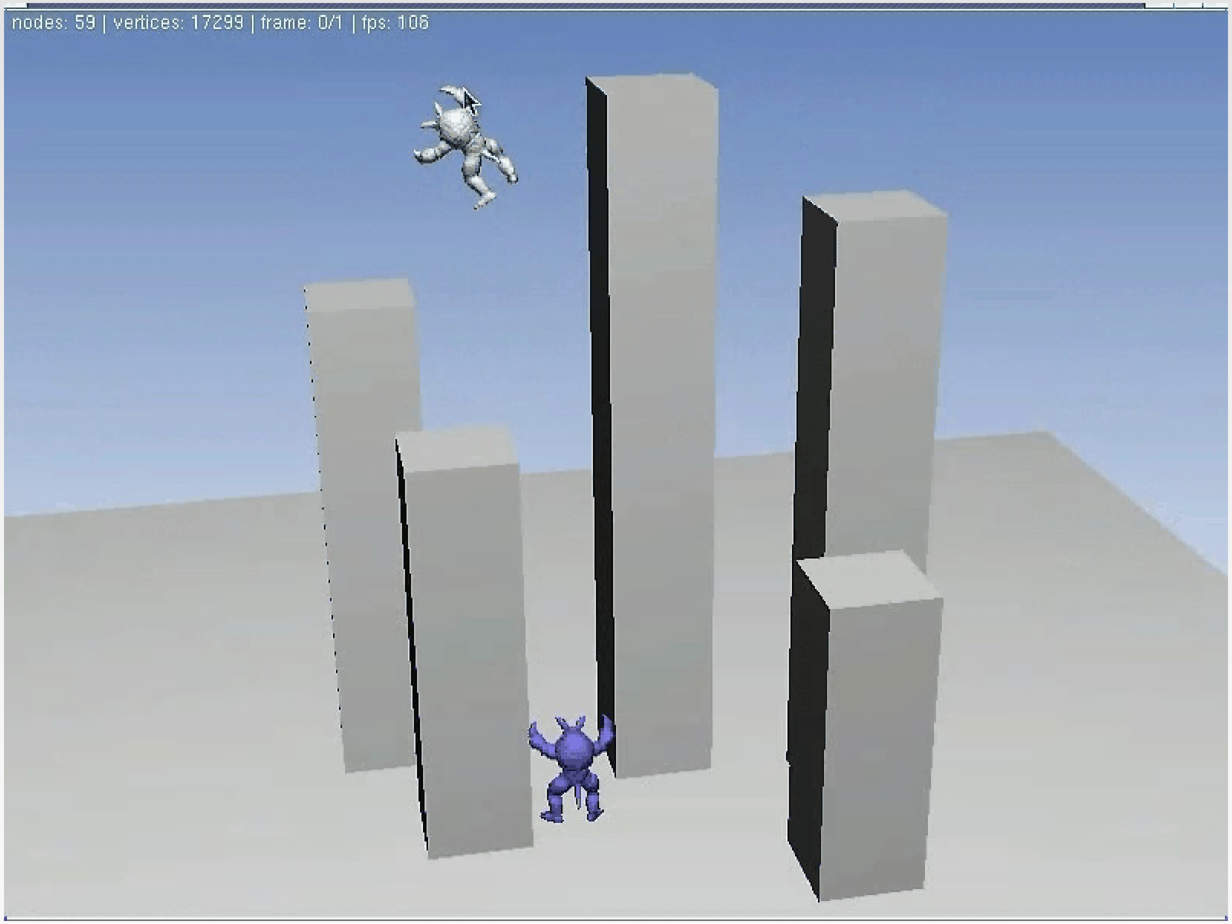


500k vertices, 60 nodes → 10 fps

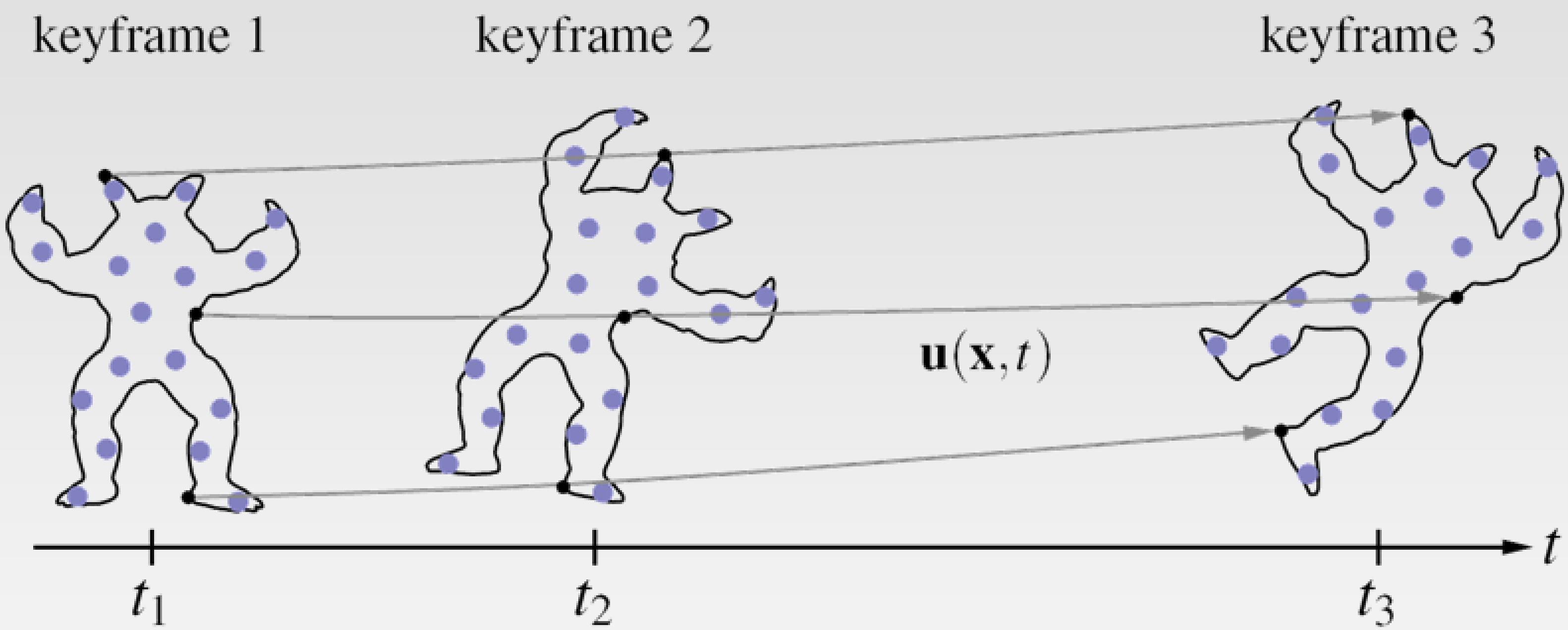
Deformable Motions

Deformable Motions: Objective

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



Deformation Field Representation



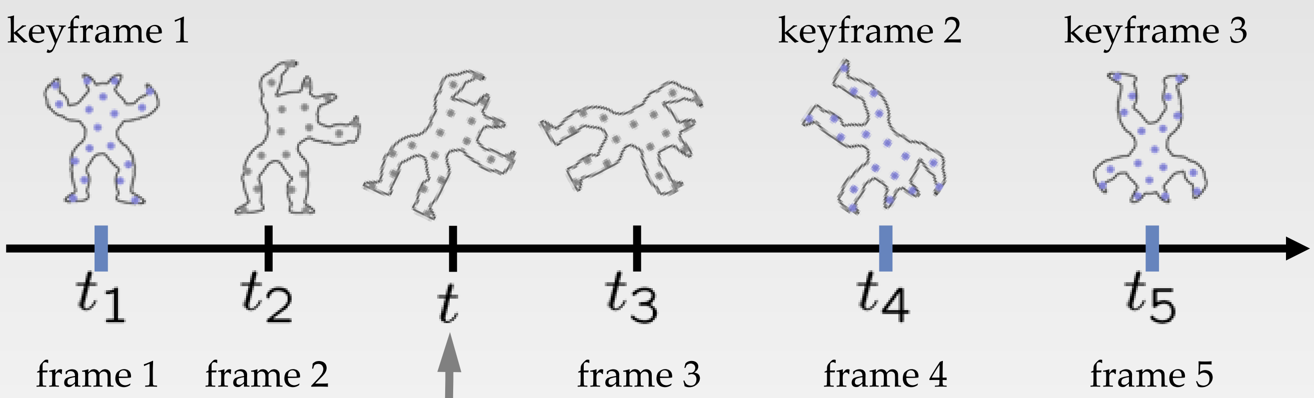
$$\mathbf{u}(\mathbf{x}, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}$$

shape functions in space

shape functions in time

Deformation Field Representation

Frames: discrete samples in time



Solve only at discrete frames: nodal displacements $\mathbf{u}_{i,t_j}$

Use meshless approximation to define continuous displacement field

$$\mathbf{u}(\mathbf{x}, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}$$

Deformation Field Representation

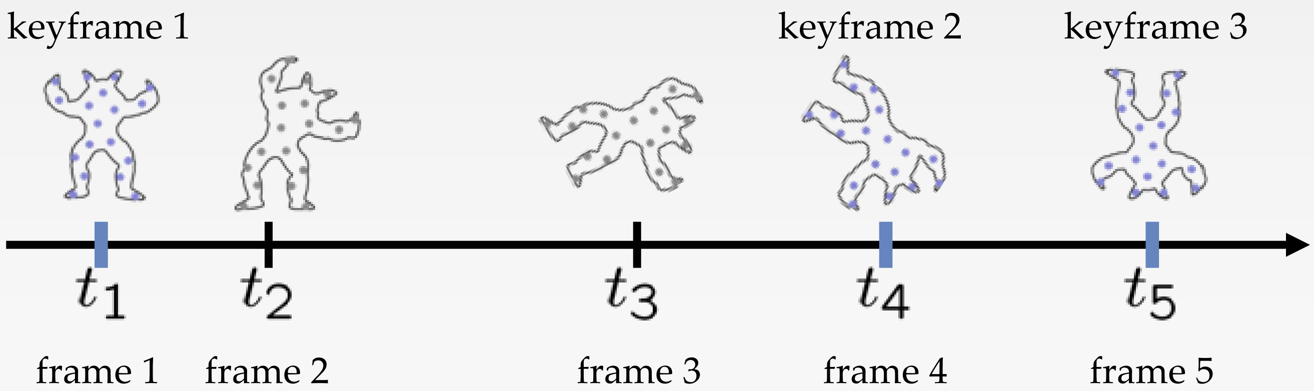
$$\mathbf{u}(\mathbf{x}, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}$$

Precompute for each frame for every neighboring frame

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

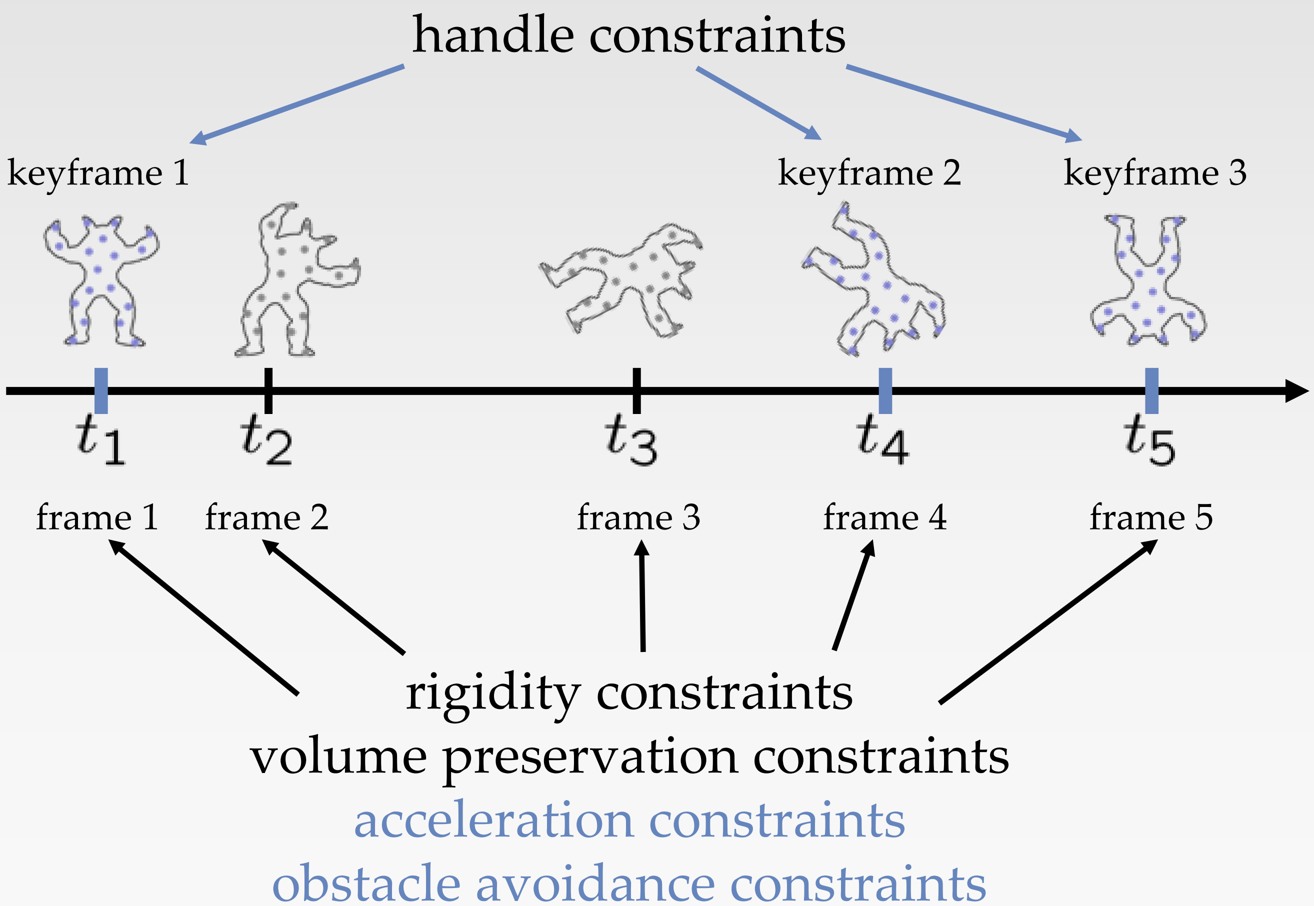
Complete quadratic basis in 1D

$$\mathbf{p}(t) = [1 \;\; t \;\; t^2]^T$$



Deformation Field Optimization

We want a realistic motion interpolating the keyframes.



Deformation Field Optimization

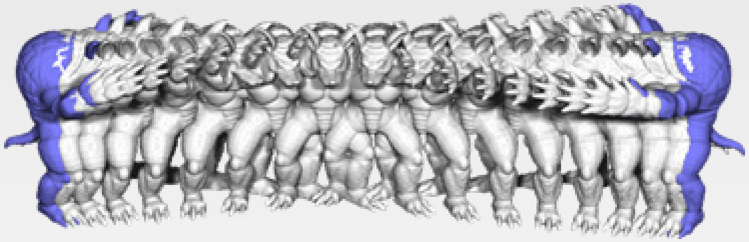
We want a smooth motion.

Acceleration constraint

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

$$\Big\downarrow \; \mathbf{u}(\mathbf{x}, t) = \sum_{j=1}^{T} \sum_{i=1}^{N} \Phi_j(t)\Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}$$

$$\| \sum_{j=1}^{T} \sum_{i=1}^{N} \frac{\partial^2 \Phi_j(t)}{\partial t^2} \Phi_i(\mathbf{x})\mathbf{u}_{i,t_j}\|^2 \rightarrow \min$$



for all nodes in all frames

Deformation Field Optimization

We want a collision free motion.

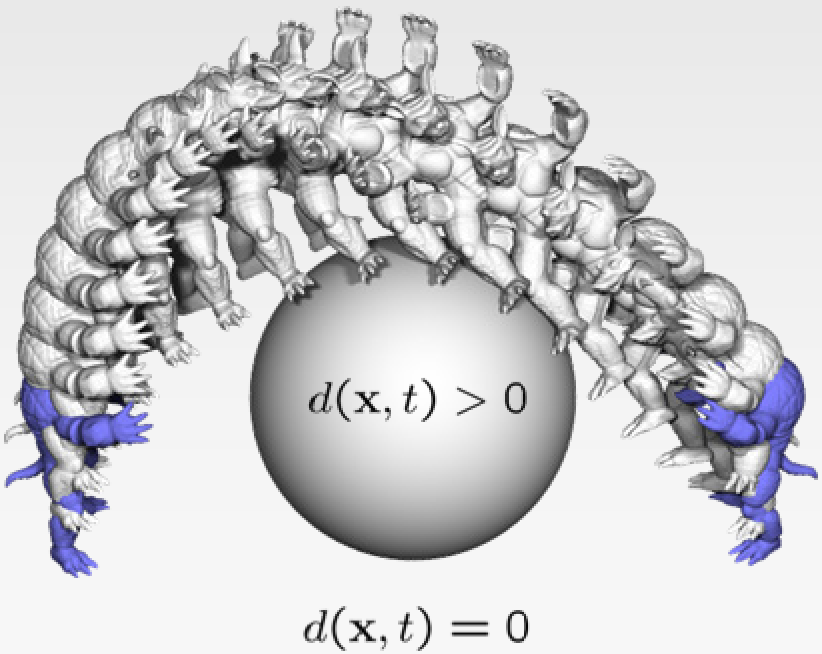
Obstacle avoidance constraint

$$d^2(\mathbf{f}(\mathbf{x}, t), t) \rightarrow \text{min}$$

$$\Big\downarrow \ \mathbf{f}(\mathbf{x}, t) = \mathbf{x} + \mathbf{u}(\mathbf{x}, t)$$

$$d^2(\mathbf{x} + \mathbf{u}(\mathbf{x}, t), t) \rightarrow \text{min}$$

for all nodes in all frames



Deformable Motions

59 nodes

500k vertices

2 keyframes

solve time: 10 seconds, 25 frames

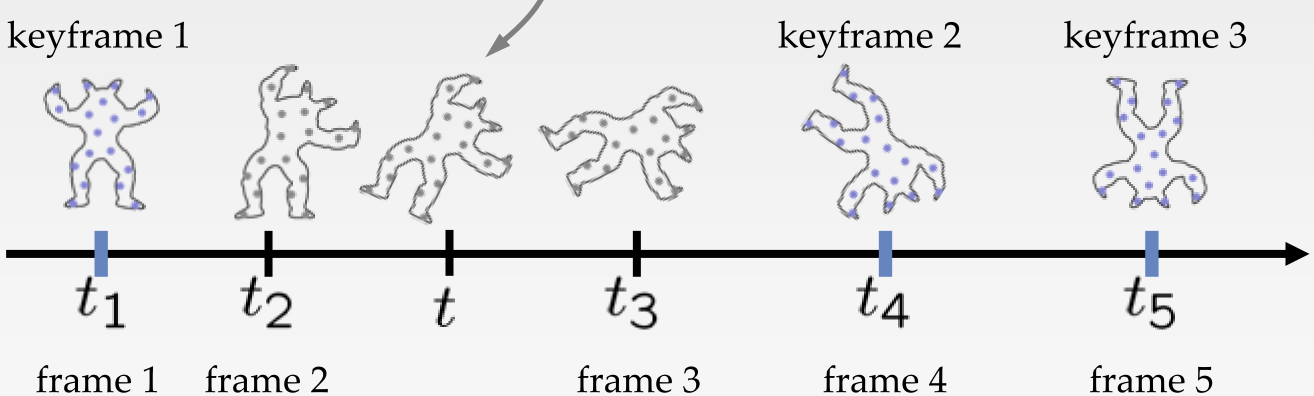
Adaptive Temporal Sampling

Number of unknowns to solve for: $3NT$

$\rightarrow$ keep as low as possible!

Constraints only imposed at frames

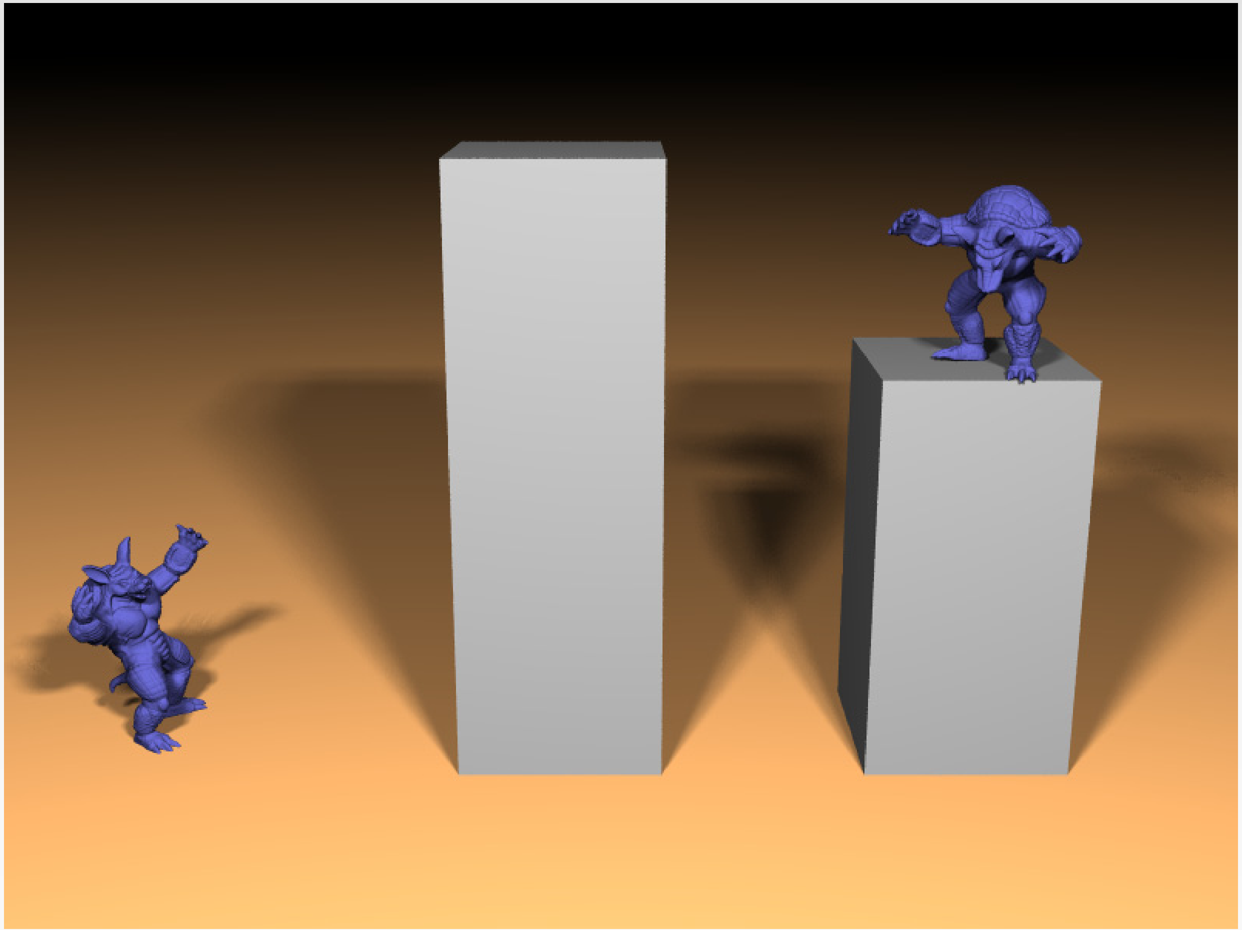
$\rightarrow$ what at interpolated frames?



Adaptive temporal sampling algorithm

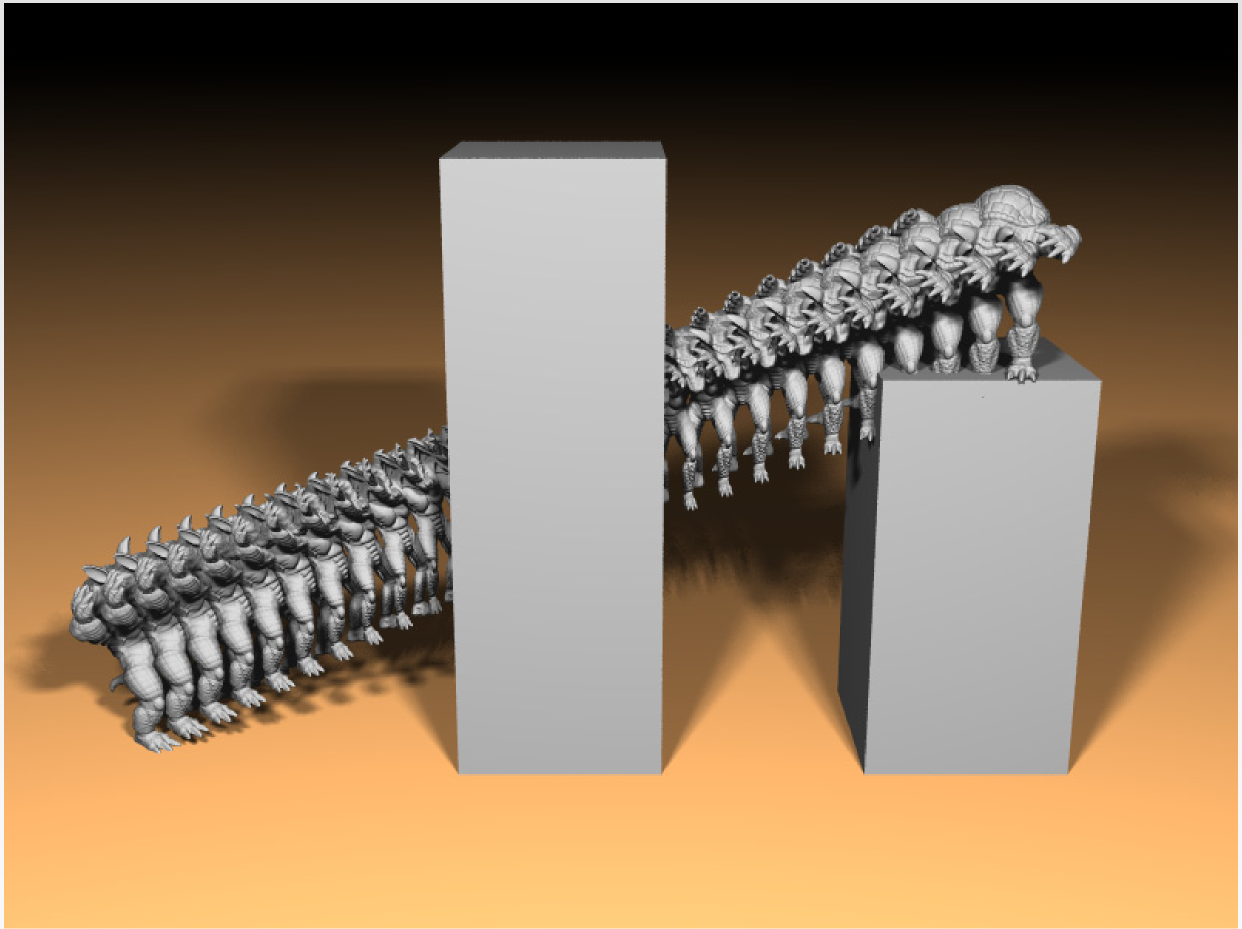
Adaptive Temporal Sampling

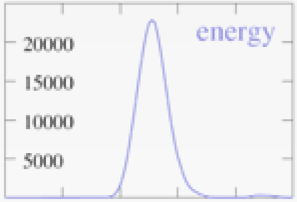
Solve only at the key frames.



Adaptive Temporal Sampling

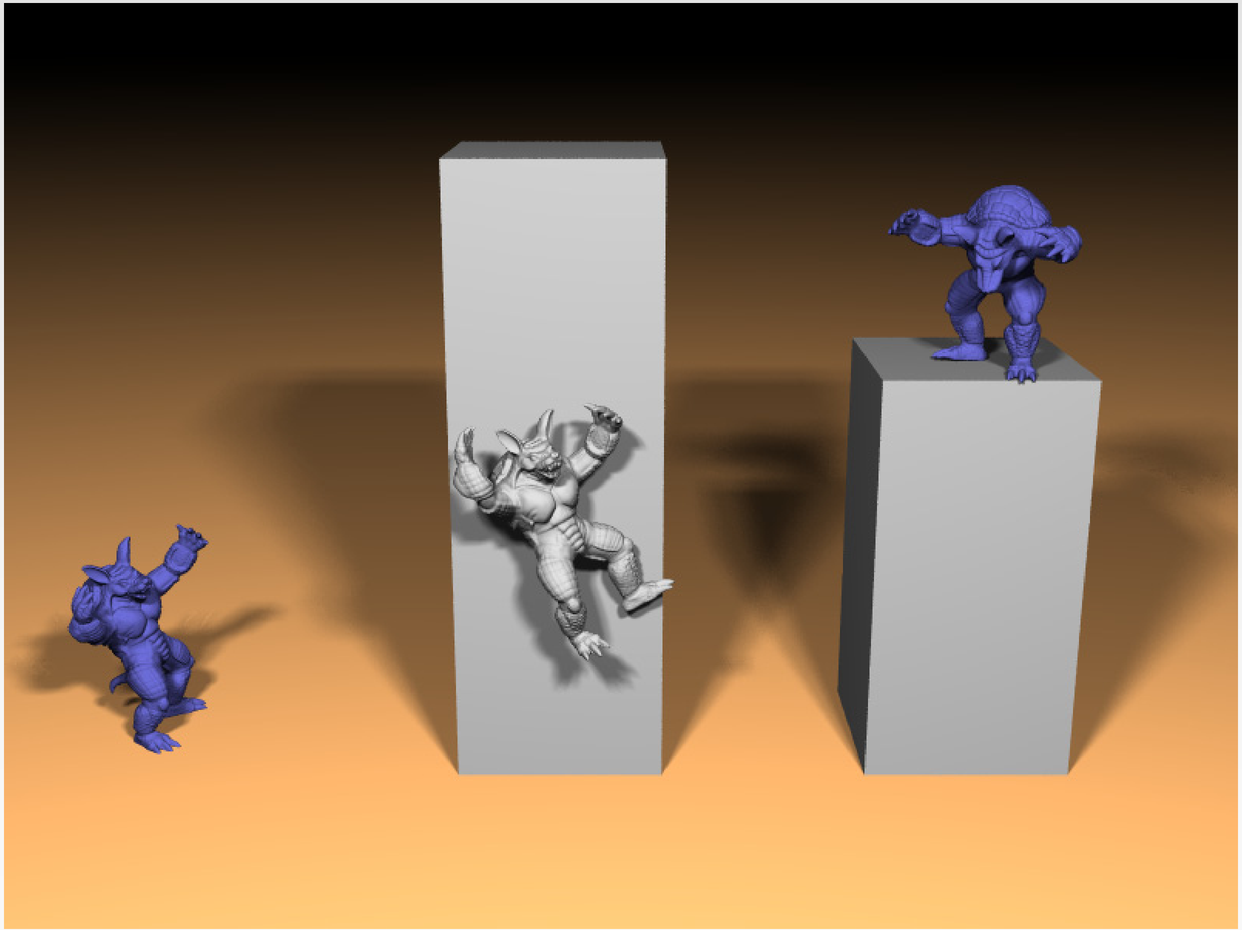
Evaluate over whole time interval.





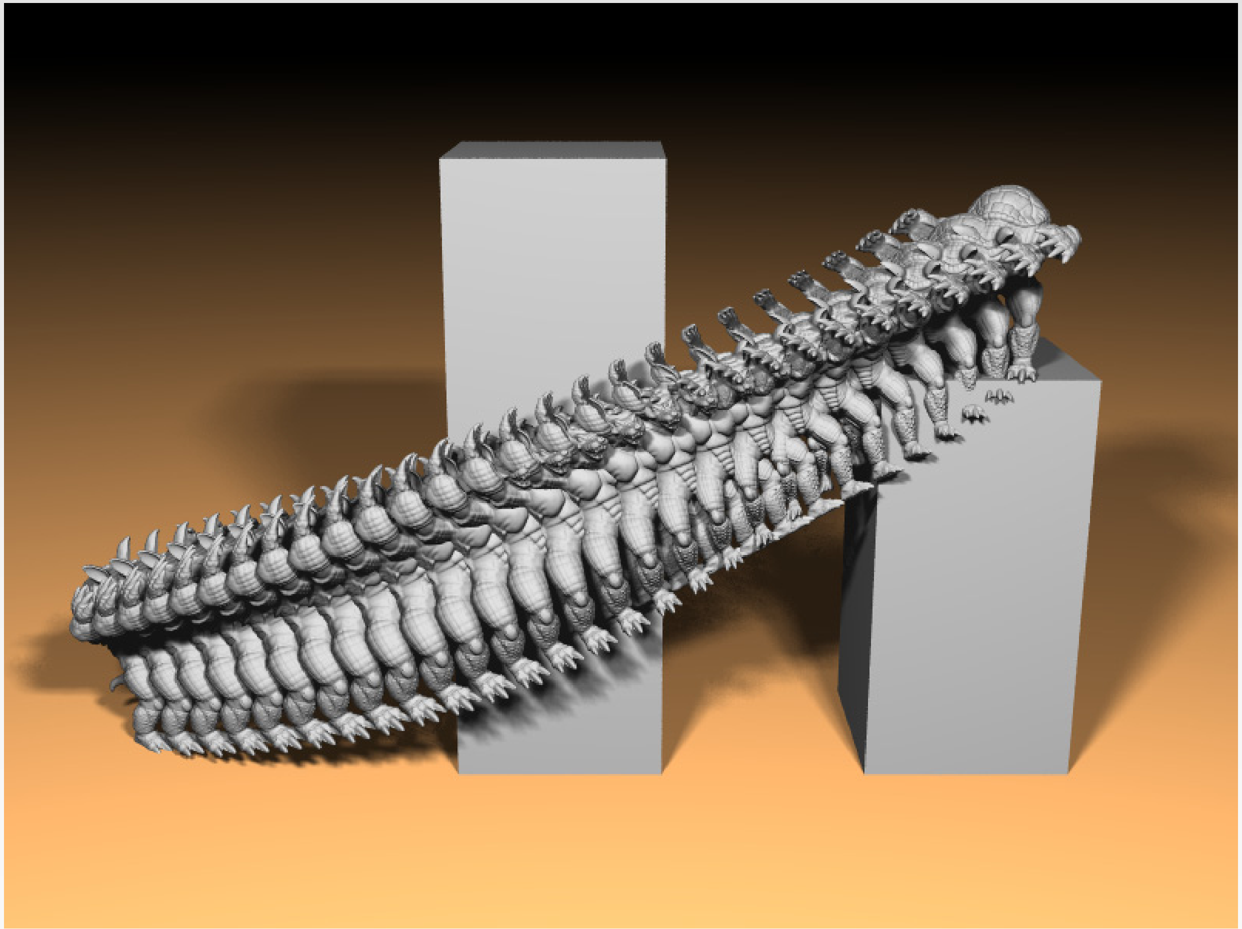
Adaptive Temporal Sampling

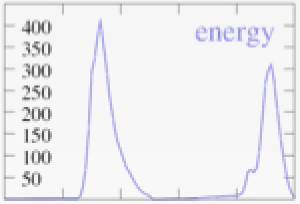
Introduce new frame where energy highest and solve.



Adaptive Temporal Sampling

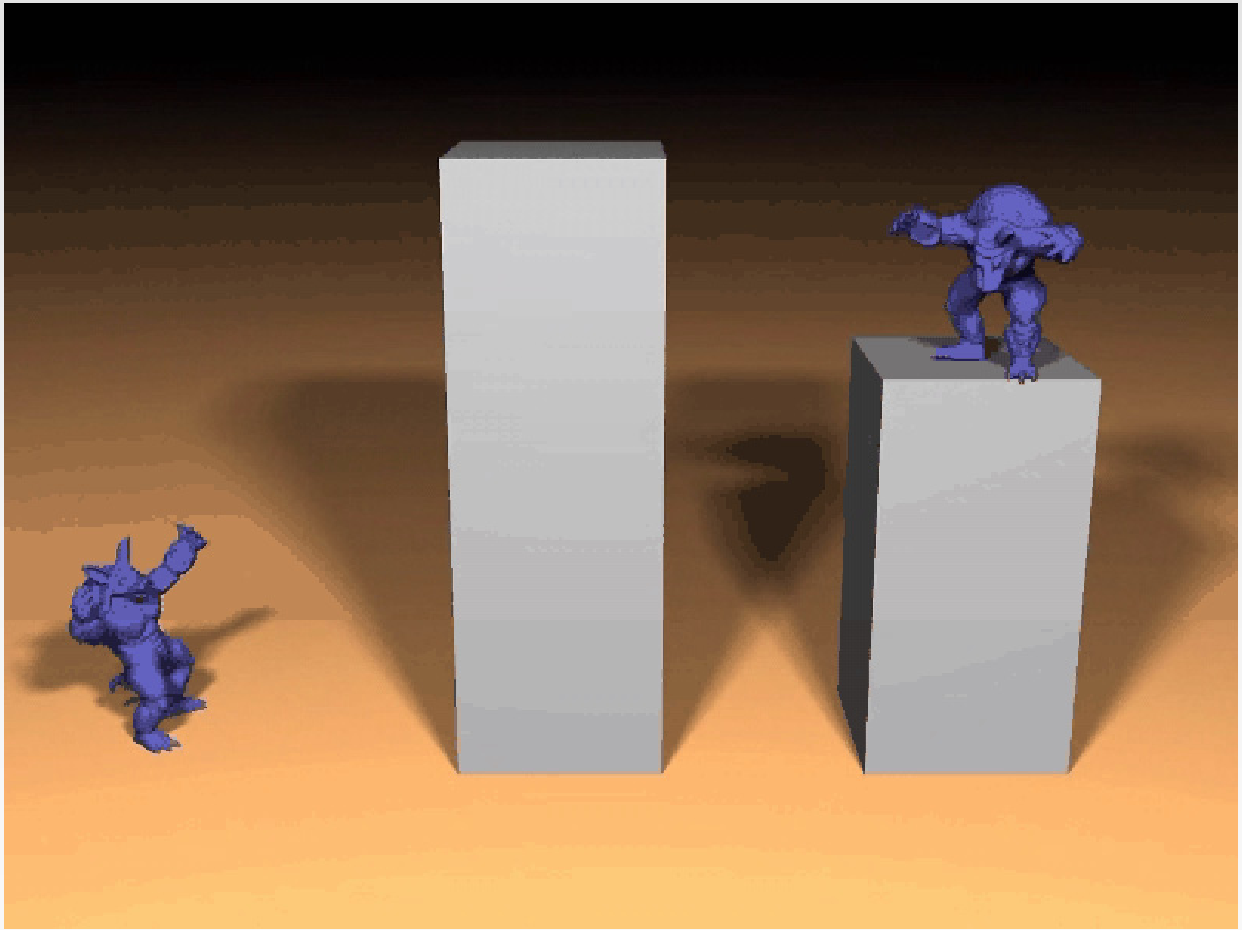
Evaluate over whole time interval.





Adaptive Temporal Sampling

Iterate until motion is satisfactory.

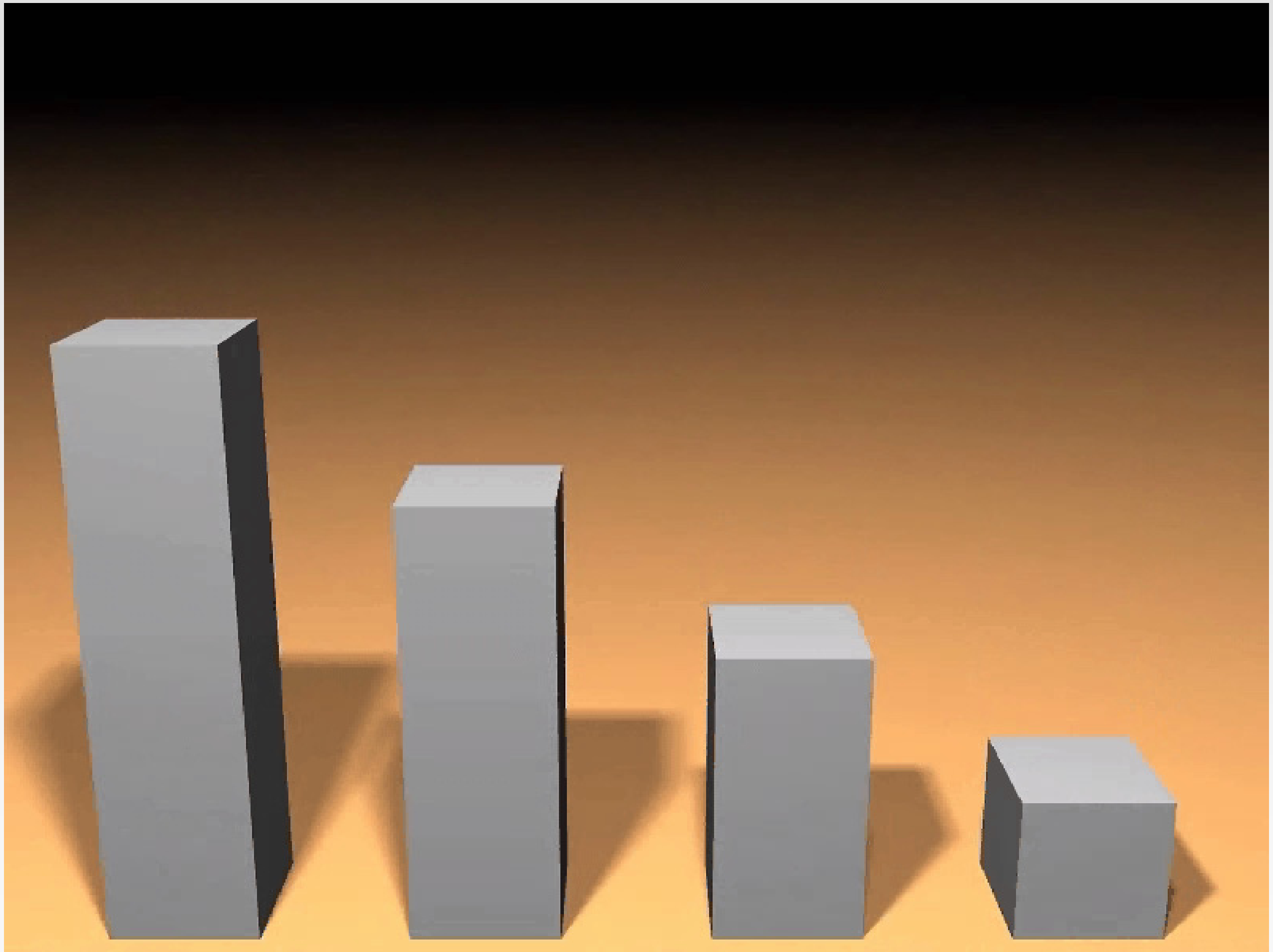


Deformable Motions

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