

Physically Based Animation 2008

Implicit Data Structure for Rope and Cloth Simulation

Imaging Media Research Center

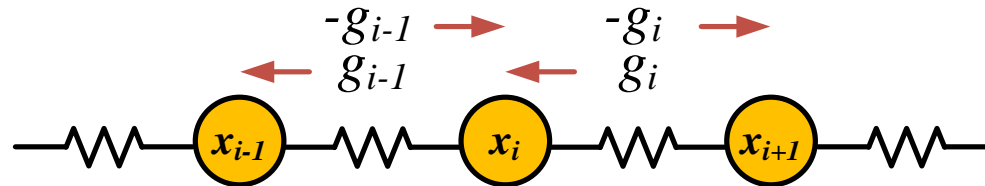
KIST

Jinwook Kim

Implicit Data Structure

Rope Modeling

- Network of particles



- Spring force between particle i and $i+1$ acting on particle i

$$g_i = k_i \frac{(\|x_{i+1} - x_i\| - l_0)}{\|x_{i+1} - x_i\|} (x_{i+1} - x_i)$$

- Net force on particle i

$$f_i = g_i - g_{i-1}$$

Implicit Data Structure

Systems of Dynamics Equations

- For an n -particle system, we have n numbers of dynamics equations

$$\begin{cases} m_1 \ddot{x}_1 = f_1 \\ m_2 \ddot{x}_2 = f_2 \\ \vdots \\ m_n \ddot{x}_n = f_n \end{cases}$$

- Concisely,

$$\ddot{x} = \frac{1}{m} f$$

where $x = [x_1^T \ \cdots \ x_n^T]^T \in \mathfrak{R}^{3n \times 1}$, $f = [f_1^T \ \cdots \ f_n^T]^T \in \mathfrak{R}^{3n \times 1}$

and $m = m_1 = \cdots = m_n$

Implicit Data Structure

Implicit Euler Integration

$$\Delta v = \left(I - \frac{h^2}{m} \frac{\partial f}{\partial x} \right)^{-1} \left(\frac{h}{m} f + \frac{h^2}{m} \frac{\partial f}{\partial x} v \right)$$

$$\Delta x = h(v + \Delta v)$$

$$v^{(n+1)} = v^{(n)} + \Delta v$$

$$x^{(n+1)} = x^{(n)} + \Delta x$$

- Need to evaluate a **BIG** Jacobian matrix $\frac{\partial f}{\partial x} \in \mathbb{R}^{3n \times 3n}$ and solve a **BIG** linear system

$$\left(I - \frac{h^2}{m} \frac{\partial f}{\partial x} \right) \Delta v = \left(\frac{h}{m} f + \frac{h^2}{m} \frac{\partial f}{\partial x} v \right)$$

Implicit Data Structure

Jacobian

- Take a look at f

$$f_i(x_{i-1}, x_i, x_{i+1}) = g_i(x_i, x_{i+1}) - g_{i-1}(x_{i-1}, x_i)$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & 0 & \dots & 0 \\ -\frac{\partial g_1}{\partial x_1} & -\frac{\partial g_1}{\partial x_2} + \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} & \dots & 0 \\ 0 & -\frac{\partial g_2}{\partial x_2} & -\frac{\partial g_2}{\partial x_3} + \frac{\partial g_3}{\partial x_3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -\frac{\partial g_{n-1}}{\partial x_n} \end{bmatrix}$$

Implicit Data Structure

Jacobian

- Since

$$\frac{\partial g_i}{\partial x_i} = -\frac{\partial g_i}{\partial x_{i+1}} = k_i \left(\frac{l_0 - \|x_{i+1} - x_i\|}{\|x_{i+1} - x_i\|} I - \frac{(x_{i+1} - x_i)(x_{i+1} - x_i)^T}{\|x_{i+1} - x_i\|^2} \right)$$

Jacobian is

$$\frac{\partial f}{\partial x} = \begin{bmatrix} G_1 & -G_1 & 0 & \cdots & 0 \\ -G_1 & G_1 + G_2 & -G_2 & \cdots & 0 \\ 0 & -G_2 & G_2 + G_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & G_{n-1} \end{bmatrix}$$

where $G_i = \frac{\partial g_i}{\partial x_i}$

Implicit Data Structure

Implicit Representation of Jacobian

- Store only G_i
 - Take a full advantage of sparsity of the Jacobian
 - Smallest memory footprint

- Recall why we need to evaluate Jacobian?

- To solve
$$\left(I - \frac{h^2}{m} \frac{\partial f}{\partial x} \right) \Delta v = \left(\frac{h}{m} f + \frac{h^2}{m} \frac{\partial f}{\partial x} v \right)$$

- We can use Conjugate Gradient method

- Note that is $I - \frac{h^2}{m} \frac{\partial f}{\partial x}$ positive definite and strictly diagonally dominant
- Without preconditioning, CG will work fine

Implicit Data Structure

Conjugate Gradient Method

- i^{th} iteration

$$\alpha^{(i)} = \frac{r^{(i)T} r^{(i)}}{d^{(i)T} Ad^{(i)}}$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$

$$r^{(i+1)} = r^{(i)} - \alpha^{(i)} Ad^{(i)}$$

$$\beta^{(i)} = \frac{r^{(i+1)T} r^{(i+1)}}{r^{(i)T} r^{(i)}}$$

$$d^{(i+1)} = r^{(i+1)} + \beta^{(i)} d^{(i)}$$

- Note that only operation related to the matrix is Matrix-vector multiplication

Implicit Data Structure

Jacobian-Vector Multiplication

- Due to the special structure of the Jacobian, matrix-vector multiplication breaks down into a few of 3X3 matrix-3 vector multiplications

$$\begin{bmatrix} G_1 & -G_1 & 0 & \cdots & 0 \\ -G_1 & G_1 + G_2 & -G_2 & \cdots & 0 \\ 0 & -G_2 & G_2 + G_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & G_{n-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} G_1(v_1 - v_2) \\ G_2(v_2 - v_3) \\ G_3(v_3 - v_4) \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ G_1(v_1 - v_2) \\ G_2(v_2 - v_3) \\ \vdots \\ G_{n-1}(v_{n-1} - v) \end{bmatrix}$$

- Did you find a room for improvement?

Implicit Data Structure

General 3X3 Symmetric Matrix

- Naïve implementation

$$G_i v = \left(\frac{k_i (l_0 - \|x_{i+1} - x_i\|)}{\|x_{i+1} - x_i\|} I - \frac{k_i (x_{i+1} - x_i)(x_{i+1} - x_i)^T}{\|x_{i+1} - x_i\|^2} \right) v$$

- Storage: 6 floating point values
- Construction: 11 multiplications, 3 additions
- multiplication: 9 multiplications, 6 additions

Implicit Data Structure

Implicit Representation

- $G_i = \{ k, l, \alpha, d \}$

where $k = k_i$, $d = x_{i+1} - x_i$, $l = \|d\|$, $\alpha = \frac{k(l_0 - l)}{l}$

- Then matrix-vector multiplication is

$$Gv = \alpha v - \frac{k(d \cdot v)}{l^2} d$$

- Storage: 6 floating point values
 - Already stored somewhere at spring force computation
- Construction: nothing
- multiplication: 12 multiplications, 5 additions

Implicit Data Structure

Modified CG

- To Solve
$$\left(I - \frac{h^2}{m} \frac{\partial f}{\partial x} \right) \Delta v = \left(\frac{h}{m} f + \frac{h^2}{m} \frac{\partial f}{\partial x} v \right)$$

- Given
$$J = \frac{\partial f}{\partial x}$$

- Calculate
$$b = \left(\frac{h}{m} f + \frac{h^2}{m} Jv \right)$$

using efficient Jacobian-vector multiplication

Implicit Data Structure

Modified CG

$$\alpha^{(i)} = \frac{r^{(i)T} r^{(i)}}{d^{(i)T} A d^{(i)}}$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$

$$r^{(i+1)} = r^{(i)} - \alpha^{(i)} A d^{(i)}$$

$$\beta^{(i)} = \frac{r^{(i+1)T} r^{(i+1)}}{r^{(i)T} r^{(i)}}$$

$$d^{(i+1)} = r^{(i+1)} + \beta^{(i)} d^{(i)}$$



$$e^{(i)} = d^{(i)} - \frac{h^2}{m} J d^{(i)}$$

$$\alpha^{(i)} = \frac{r^{(i)} \cdot r^{(i)}}{d^{(i)} \cdot e^{(i)}}$$

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$$

$$r^{(i+1)} = r^{(i)} - \alpha^{(i)} e^{(i)}$$

$$\beta^{(i)} = \frac{r^{(i+1)} \cdot r^{(i+1)}}{r^{(i)} \cdot r^{(i)}}$$

$$d^{(i+1)} = r^{(i+1)} + \beta^{(i)} d^{(i)}$$

Implicit Data Structure

Cloth Modeling

- Force against stretch
 - Linear spring in u and v directions
 - High stiffness
- Area preserving force
 - Resist to skew
 - Moderate stiffness
- Bending force
 - Resist to bending in u and v directions
 - Moderate stiffness

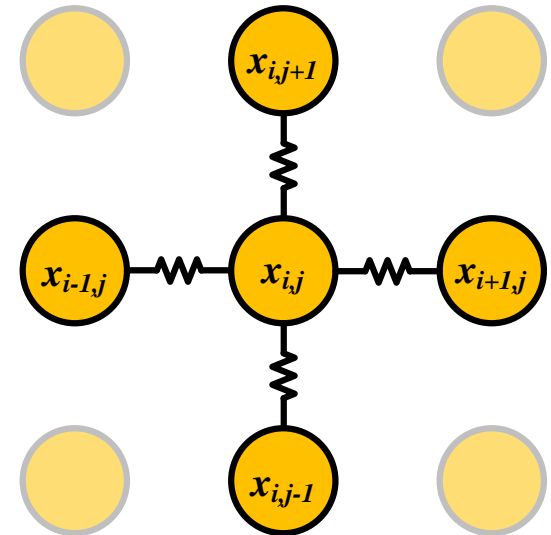
Implicit Data Structure

Area Preserving Force

$$x_u = \frac{1}{2\Delta l} (x_{i+1,j} - x_{i-1,j})$$

$$x_v = \frac{1}{2\Delta l} (x_{i,j+1} - x_{i,j-1})$$

$$g_{ij} = \|x_u\|^2 \|x_v\|^2 - (x_u \cdot x_v)^2$$



$$f_{i+1,j}^a = -f_{i-1,j}^a = \frac{k_a (g_{ij} - 1) \Delta l}{2} (\|x_v\|^2 x_u - (x_u \cdot x_v) x_v)$$

$$f_{i,j+1}^a = -f_{i,j-1}^a = \frac{k_a (g_{ij} - 1) \Delta l}{2} (\|x_u\|^2 x_v - (x_u \cdot x_v) x_u)$$

Implicit Data Structure

Bending Force in u-direction

- For all j
$$\kappa_{ij} = \frac{2 \| c_i \|}{\Delta l \| e_i \|}$$

$$f_{i+1,j}^b = k_b \kappa_{ij} \Delta l \left(\frac{\kappa_{ij}}{\| c_i \|^2} c_i - \frac{\kappa_{ij}}{\| e_i \|^2} e_i \right)$$

$$f_{i-1,j}^b = k_b \kappa_{ij} \Delta l \left(\frac{\kappa_{ij}}{\| c_i \|^2} c_i + \frac{\kappa_{ij}}{\| e_i \|^2} e_i \right)$$

$$f_{i,j}^b = -f_{i-1,j}^b - f_{i+1,j}^b$$

where $e_i = x_{i+1,j} - x_{i-1,j}$ $c_i = x_{i+1,j} - 2x_{i,j} + x_{i-1,j}$

- Note that $f_{i,j}^b = f_{i-1,j}^b = f_{i+1,j}^b = 0$ if $c_i = 0$
- Do the same for all i to compute bending force in v-direction

Implicit Data Structure

Systems of Dynamics Equations

- For an $n \times n$ -particle cloth system, we have $n \times n$ numbers of dynamics equations

$$\begin{cases} m\ddot{x}_{11} = f_{11} \\ m\ddot{x}_{21} = f_{21} \\ \vdots \\ m\ddot{x}_{n1} = f_{n1} \end{cases} \quad \begin{cases} m\ddot{x}_{12} = f_{12} \\ m\ddot{x}_{22} = f_{22} \\ \vdots \\ m\ddot{x}_{n2} = f_{n2} \end{cases} \quad \dots \quad \begin{cases} m\ddot{x}_{1n} = f_{1n} \\ m\ddot{x}_{2n} = f_{2n} \\ \vdots \\ m\ddot{x}_{nn} = f_{nn} \end{cases}$$

- Concisely,

$$\ddot{x} = \frac{1}{m} f$$

where $x = \begin{bmatrix} x_{11}^T & \dots & x_{n1}^T & x_{12}^T & \dots & x_{n2}^T & \dots & x_{1n}^T & \dots & x_{nn}^T \end{bmatrix}^T \in \mathbb{R}^{9n^2 \times 1}$

and $f \in \mathbb{R}^{9n^2 \times 1}$

Implicit Data Structure

Rearrange State

$$\begin{aligned}
 x = \begin{bmatrix} x_{11} \\ \vdots \\ x_{n1} \\ x_{12} \\ \vdots \\ x_{n2} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{nn} \end{bmatrix} &= \begin{bmatrix} | \\ y_1 \\ | \\ | \\ y_2 \\ | \\ \vdots \\ | \\ y_n \\ | \end{bmatrix} = \begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & 0 & 0 & \vdots & 0 & \vdots & 1 & \vdots & 0 \\ 0 & 1 & 0 & 0 & \vdots & 0 & \vdots & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & 0 & 0 & \vdots & 0 & \vdots & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \vdots & 1 & 0 & \vdots & 0 & \vdots & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \\ x_{21} \\ \vdots \\ x_{2n} \\ \vdots \\ x_{n1} \\ \vdots \\ x_{nn} \end{bmatrix} = \Pi \begin{bmatrix} | \\ z_1 \\ | \\ | \\ z_2 \\ | \\ \vdots \\ | \\ z_n \\ | \end{bmatrix}
 \end{aligned}$$

Implicit Data Structure

Separate and Rearrange Force

$$\begin{aligned}
 \mathbf{f} = & \begin{bmatrix} f_{11} \\ \vdots \\ f_{n1} \\ f_{12} \\ \vdots \\ f_{n2} \\ \vdots \\ f_{1n} \\ \vdots \\ f_{nn} \end{bmatrix} = \begin{bmatrix} f_{11}^u \\ \vdots \\ f_{n1}^u \\ f_{12}^u \\ \vdots \\ f_{n2}^u \\ \vdots \\ f_{1n}^u \\ \vdots \\ f_{nn}^u \end{bmatrix} + \begin{bmatrix} f_{11}^v \\ \vdots \\ f_{n1}^v \\ f_{12}^v \\ \vdots \\ f_{n2}^v \\ \vdots \\ f_{1n}^v \\ \vdots \\ f_{nn}^v \end{bmatrix} = \begin{bmatrix} f_{11}^u \\ \vdots \\ f_{n1}^u \\ f_{12}^u \\ \vdots \\ f_{n2}^u \\ \vdots \\ f_{1n}^u \\ \vdots \\ f_{nn}^u \end{bmatrix} + \Pi \begin{bmatrix} f_{11}^v \\ \vdots \\ f_{1n}^v \\ f_{21}^v \\ \vdots \\ f_{2n}^v \\ \vdots \\ f_{n1}^v \\ \vdots \\ f_{nn}^v \end{bmatrix} = \begin{bmatrix} | \\ p_1 \\ | \\ | \\ p_2 \\ | \\ \vdots \\ | \\ p_n \\ | \end{bmatrix} + \Pi \begin{bmatrix} | \\ q_1 \\ | \\ | \\ q_2 \\ | \\ \vdots \\ | \\ q_n \\ | \end{bmatrix}
 \end{aligned}$$

Implicit Data Structure

Huge Jacobian BUT

$$\frac{\partial f^u}{\partial x} = \begin{bmatrix} \frac{\partial p_1}{\partial y_1} & 0 & \dots & 0 \\ 0 & \frac{\partial p_2}{\partial y_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial p_n}{\partial y_n} \end{bmatrix} \quad \frac{\partial f^v}{\partial x} = \Pi^{-1} \begin{bmatrix} \frac{\partial q_1}{\partial z_1} & 0 & \dots & 0 \\ 0 & \frac{\partial q_2}{\partial z_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial q_n}{\partial z_n} \end{bmatrix} \Pi$$

- Note that $\frac{\partial p_i}{\partial y_i} = U_i$, $\frac{\partial q_i}{\partial z_i} = V_i$ are same to the Jacobian developed in rope modeling
- $\Pi^{-1} = \Pi^T$ since Π is a permutation matrix
- Our cloth model is equivalent to $2n$ threads weaved horizontally and vertically

Implicit Data Structure

Jacobian-Vector Multiplication

- Permute v first $\bar{v} = \Pi v$
- Then

$$Jv = \begin{bmatrix} U_1 v_1 \\ U_2 v_2 \\ \vdots \\ U_n v_n \end{bmatrix} + \Pi^T \begin{bmatrix} V_1 \bar{v}_1 \\ V_2 \bar{v}_2 \\ \vdots \\ V_n \bar{v}_n \end{bmatrix}$$

- Same to independent Jacobian-vector multiplications for each thread
- Note that multiplying Π^T is an inverse permutation operation