FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation

José Ricardo Mello Viana

Laboratório de Computação Gráfica
LCG - PESC - UFRJ

July 27, 2007
Contents

- Introduction
- Lattice Shape Matching
- Linear-Time Fast Summation Algorithm
- Extensions
- Results
- Conclusion and Discussion
They propose *Lattice Shape Matching* that enables robust approximation of volumetric, large-deformation dynamics for real-time or large-scale offline simulations.

**Contributions**

- Lattice Shape Matching, a volumetric lattice-based formulation of deformable shape matching for robust dynamic deformation of embedded meshes;
- A linear-time fast summation algorithm for lattice shape matching (FastLSM);
- Practical enhancements such as for rotation estimation, hardware rendering, and fast-summation damping.
Conservatively voxelize the model to construct lattice of cubic cells containing the mesh;

The embedded mesh is then deformed using trilinear interpolation of lattice vertex positions;

Lattice deformation is controlled by unit point-mass particles placed at the lattice cell vertex;

Each particle $i$ contains:
- An equivalent vertex $i$;
- Static initial position $x_i^0$;
- Dynamic position $x_i$;
- Mass $m_i$;
- One-ring neighborhood list $N_i$. 
Figure: Voxelization and Embedding
Each particle $i$ is associated with a shape matching region comprised of a set of shape matching particles, $R_i$, which for half-width $w$ contains $i$ and all particles reachable by traversing not more than $w$ neighborhood lists from particle $i$; e.g., if $w = 1$ then $R_i = N_i$

- Allows irregular shape matching regions and handles boundary cases.

- For each particle $i$, the set of indices of all shape matching regions to which $i$ belongs is equivalent to $R_i$. 

At each time step, each region $r$ finds the best rigid transform from $(x_i^0)_{i \in R_r}$ to $(x_i)_{i \in R_r}$, determining a per-region rotation and translation of the rest positions, $T_r = [R_r t_r] \in R^{3 \times 4}$

Final goal position is computed for each particle $i$ as the average of regional goal positions $g_i = \langle T_r x_i^0 \rangle_{r \in R_i}$

Particle positions $x_i$ and velocities $v_i$ are updated using $g_i$:

$$v_i(t + h) = v(i) + \frac{g_i(t) - x_i(t)}{h} + h \frac{f_{ext}(t)}{m_i}$$

$$x_i(t + h) = x_i(t) + hv_i(t + h)$$
Figure: Increasing Shape-Matching region width increases stiffness
The system can have high-degree-off-freedom motions with minimal discontinuities, although utilizes only rigid transformations, because there are many regularly overlapping regions.

Figure: *Comparison of shape matching methods: Linear and Quadratic with a low number of regions; and LSM*
Figure 4: Cost complexity versus region width, \( w \), per simulation particle: the \( O(w^3) \) naïve brute-force approach; our \( O(w) \) bar-plate-cube approach; and our \( O(1) \) FastLSM approach. FastLSM speedups are noticeable even for small \( w \) values. Data is for solid buddha model. Cost is measured in units of FastLSM\((w=1) \approx \) FastLSM\((w)\), and illustrates hundredfold speedups over the naïve approach for moderate \( w \).
The method break down the particle lists $R_r$ into sub-summations that will be maximally reused between regions.

Figure: Decomposing a region summation into sub-summations for reuse across regions ($w = 1$ case)
The main idea of the algorithm can be illustrated for the simple case of cubical regions.

The summation of a particle-defined value $v$ is:

$$\text{SUM}_r = \sum_{k=z-w}^{z+w} \sum_{j=y-w}^{y+w} \sum_{i=x-w}^{x+w} v_{ijk} = \sum_{k=z-w}^{z+w} \left( \sum_{j=y-w}^{y+w} \left( \sum_{i=x-w}^{x+w} (v_{ijk}) \right) \right)$$

Can be broken for each location in the lattice:

$$X_{xyz} = \sum_{i=x-w}^{x+w} v_{iyz} \Rightarrow XY_{xyz} = \sum_{j=y-w}^{y+w} v_{xjz} \Rightarrow \text{SUM}_r = \sum_{k=z-w}^{z+w} v_{xyk}$$

- **Sum over $X$** $\rightarrow$ Bars
- **Sum over $Y$** $\rightarrow$ Plates
- **Sum over $Z$** $\rightarrow$ Cubes
The total cost for every index in lattice will be $3n(2w + 1)$ flops, as opposed to $n(2w + 1)^3$ for the naïve approach;

They can do even better than $O(w)$ flops per lattice index observing these summation recurrences:

\[
X_{xyz} = X_{(x-1)yz} - v_{(x-w-1)yz} + v_{(x+w)yz} \\
XY_{xyz} = XY_{(x(y-1)z} - X_{x(y-w-1)z} + X_{x(y+w)z} \\
SUM_{xyz} = SUM_{xy(z-1)} - XY_{xy(z-w-1)} + XY_{xy(z+w)}
\]

Using this definition, the summation requires constant time per lattice index: only 6 flops
Handling Irregular Regions

- For regions that are not perfect cubes, record region-specific sub-summation $X^r_{xyz}$ and $XY^r_{xyz}$;

  - These sums consist of the particles that would be in the corresponding generic sub-summations $X_{xyz}$ and $XY_{xyz}$ but are restricted to particles in $R_r$.

Fast Summation Operator

- To indicate the fast summation algorithm, the operator notation is defined:

\[
F_{i \in R_r} \{v_i\} \equiv \sum_{i \in R_r} v_i
\]
They describe efficient calculations for each region $r$ center of mass $c_r$ and least-squares rotation $R_r$ (which determine each regions optimal rigid transformation $T_r$) and each particle $i$ goal position $g_i$.

The deformed center of mass for each region $r$ is obtained as follows:

$$c_r = \frac{1}{M_r} \sum_{i \in R_r} \tilde{m}_i x_i$$

Where $M_R = \sum_{i \in R_r} \tilde{m}_i$ is precomputed effective region mass.
They estimate the least square rotation \( R_r \) from particles \( R_r \) using the rotational part of:

\[
A_r \equiv \sum_{i \in R_r} \tilde{m}(x_i - c_r)(x_i^0 - c_r^0)^T \in \mathbb{R}^{3 \times 3}
\]

Where \( c_r^0 = (\sum_{i \in R_r} \tilde{m}_i x_i^0) / M_r \) is the precomputed center of mass of region \( r \)'s undeformed particles. They obtain the rotational part using the polar decomposition \( A = RU \), where \( U \) is a unique 3-by-3 symmetric stretch matrix.

Isolating the dependencies and simplifying common terms:

\[
A_r = \sum_{i \in R_r} \tilde{m}_i x_i x_i^0^T - M_r c_r c_r^0^T
\]
Each regions least-squares rigid transformation of the rest positions $x_i^0$ is a rotation by $R_r$ and a translation that shifts the rotated $c_r^0$ to $c_r$, this is stored as the matrix:

$$T_r = [R_r(c_r - R_r c_r^0)] \in \mathbb{R}^{3 \times 4}$$

Each particles goal position $g_i$ can be restated as the transformation of the particles rest position, $x_i^0$, by the average rigid transformation over the regions the particle belongs to,

$$g_i = \frac{1}{|R_i|} \sum_{r \in R_i} T_r x_i^0$$
FASTLSM()
01 Precompute $M_r, c_r^0$ for all regions
02 while true
03 Calculate $F_{i \in R_r}\{\tilde{m}_i x_i\}, F_{i \in R_r}\{\tilde{m}_i x_i x_i^0 T\}$ for all $r$
04 for each region $r$
05 Calculate $c_r$ and $A_r$ using previous definitions
06 Polar decompose $A_r = R_r U_r$
07 Calculate $T_r = [R_r(c_r - R_r c_r^0)]$
08 Calculate $F_{r \in R_i}\{T_r\}$ for all $i$
09 for each particle $i$
10 Calculate $g_i$ and $v_i(t + h)$ using previous definitions
11 Perform damping
12 for each particle $i$
13 Calculate $x_i(t + h)$ using previous definition
Extensions
Fast Polar Decompositions and Damping

- **Fast Polar Decompositions**
  - They use cyclic Jacobi iterations to diagonalize $\mathbf{A}^T \mathbf{A} = \mathbf{U}^2 = \mathbf{V} \text{diag}(\lambda) \mathbf{V}^T$ to construct $\mathbf{R} = \mathbf{A} \mathbf{U}^{-1}$.

- **Damping**
  - As an alternative to global damping of nonrigid motion, they can apply damping on a per-region basis, bleeding off non-rigid motion of local regions. That makes this efficient is that the estimate of region $r$ rigid-body velocity is decomposable into fast-summation passes (reusing prior values).
  - Using fast summations, this damping model requires roughly the same number of flops as the shape matching operation.
Figure: Fracture
They therefore briefly outline an optimized vertex shader;

- Lattice particle positions are copied to uniform/constant GPU memory, so that each deformed vertex position \( x \) can be computed as the weighted combination of its eight lattice-cell positions;
Figure: High-Resolution examples (see video)
The system can handle a wide range of possible deformations, but may produce non-physical behavior in some circumstances as a result of its geometrically motivated approach. This makes it unsuitable for applications requiring precise or predictive modeling.

Future work includes exploring different particle frameworks, including tetrahedral lattices or irregular samplings.