Tetrahedron constraint gradients

In position-based dynamics (Müller et al. 2006), simulation solves constraints by directly updating the positions of mass particles. Several constraints are mentioned in the paper, including distance constraint between two particles, bending constraint (cloth), volume constraint and collision constraint. Particles are moved along negative gradient of the constraint with respect to each particle position. For example, distance constraint

$$C(p_0, p_1) = \|p_0 - p_1\| - d$$

where p_0 and p_1 are positions ($\in \mathbb{R}^3$), $\| \ \|$ is the Euclidean norm and d is the rest distance between them, has gradients

$$\nabla_{p_0} C(p_0, p_1) = \frac{p_0 - p_1}{\|p_0 - p_1\|}$$
$$\nabla_{p_1} C(p_0, p_1) = -\frac{p_0 - p_1}{\|p_0 - p_1\|}$$

I wanted to use this approach on soft body simulation with tetrahedra. Having a distance constraint for each edge of the tetrahedral mesh is not enough, since tetrahedra can be inverted and will stay inverted (distance constraints are satisfied either way).

My first idea was to add four additional constraints per tetrahedron, which would keep vertices from crossing the planes of the other three vertices. The constraint looks identical to the triangle collision constraint from the paper:

$$C(p_0, p_1, p_2, p_3) = (p_1 - p_0) \cdot \frac{(p_2 - p_0) \times (p_3 - p_0)}{\|(p_2 - p_0) \times (p_3 - p_0)\|} - d$$

where we are trying to keep p_1 at a distance $|d|$ from the plane of p_0, p_2, p_3. d can be negative depending on the order of the vertices. This is important, since now an inverted tetrahedron will work to un-invert itself.

My vector calculus skills are terrible (as of late 2014), it took me quite a while to derive the gradients of this constraint. It's helpful to translate the problem so that $p_0 = 0$:

$$C(p_0, p_1, p_2, p_3) = p_1 \cdot \frac{p_2 \times p_3}{\|p_2 \times p_3\|} - d$$
$$n = \frac{p_2 \times p_3}{\|p_2 \times p_3\|}$$

$$\nabla_{p_1} C = n$$
$$\nabla_{p_2} C = \frac{p_3 \times p_1 + (n \times p_3)(n \cdot p_1)}{\|p_2 \times p_3\|}$$
$$\nabla_{p_3} C = -\frac{p_2 \times p_1 + (n \times p_2)(n \cdot p_1)}{\|p_2 \times p_3\|}$$
$$\nabla_{p_0} C = -\nabla_{p_1} - \nabla_{p_2} - \nabla_{p_3}$$

This constraint worked well, even without the edge distance constraints. It is a bit expensive in my implementation though.

Volume constraint

I also tried a volume constraint:

$$C(p_0, p_1, p_2, p_3) = \frac{1}{6}(p_1 - p_0) \cdot ((p_2 - p_0) \times (p_3 - p_0)) - V$$

with gradients (after translating to $p_0 = 0$):

$$\nabla_{p_1} C = \frac{1}{6} p_2 \times p_3$$
$$\nabla_{p_2} C = -\frac{1}{6} p_1 \times p_3$$
$$\nabla_{p_3} C = \frac{1}{6} p_1 \times p_2$$
$$\nabla_{p_0} C = -\nabla_{p_1} - \nabla_{p_2} - \nabla_{p_3}$$

which also works well (and only one volume constraint per tetrahedron), but must be combined with edge distance constraints. This volume constraint is also signed (rest volume V can be negative), and therefore works to un-invert the tetrahedra.

References

Müller, Matthias, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. “Position Based Dynamics.”