Random Notes

wxgopher

January 14, 2021

Contents

. . . .

1 Coordinate system transformation

Notations:

Local coordinate system is denoted by C_L, and global (world) coordinate system is denoted by C_W.

We define

$$C_L = \begin{pmatrix} e_1 & e_2 & e_3 & 0 \\ & 0 & & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}, \tag{1}$$

which translates a homogeneous coordinate x_L in C_L to x_W in C_W.

In C_L, we define two transformations (rotation, translation, scaling) $T_1^L, T_2^L \in \mathbb{R}^{4\times 4}$ of **local coordinate system C_L w.r.t. global coordinate system** C_W, and T_1 is applied before T_2. It's easy to derive in world space C_W, $T_1^{W\mapsto W} = C_L T_1^L C_L^{-1}$, and this is a mapping from global coordinate to global coordinate. Similarly, $T_1^{L\mapsto W} = T_1^{W\mapsto W} C_L = C_L T_1^L$.

Consider combining these transformations, we have (this time C_L is actually $C_L T_1^L$, because we've moved our local coordinate system)

$$T_2 \circ T_1 \equiv (T_2 \circ T_1)^{L\mapsto W} = (C_L T_1^L) T_2 (C_L T_1^L)^{-1} C_L T_1^L = C_L T_1^L T_2^L, \tag{2}$$

and to be clear, this mapping is applied on local coordinate system and produce global coordinate.

2 Simple linear elasticity and implementation

We use a variant of classical linear elasticity:

$$E(x) = \sum_t \frac{\mu}{2} \|D_s - D_m\|_{\mathrm{F}}^2 \tag{3}$$

where D_s and D_m are shape matrices, D_m is a constant matrix, $D_s = G_t \cdot x$ where G_t is a linear mapping (tensor) . Let $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^{3\mathrm{n}}$. We denote restpose with x_m and deformed pose with x_s.

If we are to use a general optimizer like LBFGS (implemented in master branch), we need the gradient:

$$\nabla E(x_i) = \mathrm{tr}\left[(D_s - D_m)^{\mathrm{T}} \cdot \frac{\partial D_s}{\partial x_i} \right] \tag{4}$$

The constraints is given by fixed boundaries on certain DoFs.

2.1 Quadratic minimization

Since E is a quadratic form, we can convert this problem to a quadratic minimization. This is done as follows:

For every tet, we define $vec(D_s) = S_t \otimes I_3 \cdot x_s = G x_s \in \mathbb{R}^{9\times 1}$, where S_t is a selector matrix, defined by: $S_{t,ij} = 1$ iff $i = 1, 2, 3$ and j is the index of that point, and $S_{t,ij} = -1$ iff $i = 4$ and j is the index of that point.

G is a $9 \times 3n$ matrix, and vec is a vectorization denoted as $vec((a_1, a_2, \cdots, a_n)) = (a_1, a_2, \cdots, a_n)^{\mathrm{T}}$.

Also note that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$, $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$, and $\mathrm{tr}(A^{\mathrm{T}}) = \mathrm{tr}(A)$, then

$$\begin{aligned} E(x) &= \sum \frac{\mu}{2} \|D_s - D_m\|_F^2 = \sum \frac{\mu}{2} \cdot \mathrm{tr}((D_s - D_m) \cdot (D_s - D_m)^{\mathrm{T}}) &\text{(5a)} \\ &= \sum \frac{\mu}{2} \mathrm{tr}(D_s^{\mathrm{T}} D_s - 2 D_s^{\mathrm{T}} D_m + D_m D_m^{\mathrm{T}}) &\text{(5b)} \end{aligned}$$

It's easy to derive:

$$\mathrm{tr}(D_s^{\mathrm{T}} D_s) = x^{\mathrm{T}} G^{\mathrm{T}} G x, \tag{6a}$$
$$\mathrm{tr}(D_s^{\mathrm{T}} D_m) = x^{\mathrm{T}} G^{\mathrm{T}} \cdot vec(D_m) = x^{\mathrm{T}} G^{\mathrm{T}} G x_m. \tag{6b}$$

Thus we have

$$E(x) = \frac{1}{2} x^{\mathrm{T}} A x - x^{\mathrm{T}} b + c, \tag{7}$$

where

$$A = \mu \sum G^{\mathrm{T}} G \geq 0, \tag{8a}$$
$$b = \mu \sum G^{\mathrm{T}} G x_m, \tag{8b}$$
$$c = \frac{\mu}{2} \sum \mathrm{tr}(D_m D_m^{\mathrm{T}}). \tag{8c}$$

The stationary point of E can be obtained by solving $Ax = b$.

2.2 Constraints

Two ways to impose m constraints:
The constraint can be written as $Qx = q \Rightarrow x^{\mathrm{T}} Q^{\mathrm{T}} - q^{\mathrm{T}} = 0$, where $Q \in R^{3m \times 3n}$, $q \in R^{3m \times 1}$, $m \leq n$.

2.2.1 Soft constraints

Let w become a constraint parameter, the objective becomes

$$E(x) = \frac{1}{2} x^{\mathrm{T}} A x - x^{\mathrm{T}} b + c + w(x^{\mathrm{T}} Q^{\mathrm{T}} Q x - 2x^{\mathrm{T}} Q^{\mathrm{T}} q + q^{\mathrm{T}} q). \tag{9}$$

The final linear system to minimize E becomes solving

$$(A + 2wQ^{\mathrm{T}} Q)x = b + 2wQ^{\mathrm{T}} q. \tag{10}$$

2.2.2 Lagrangian multipliers

The Lagrangian of E is

$$L(x, \lambda) = \frac{1}{2} x^{\mathrm{T}} A x - b^{\mathrm{T}} x + c + \lambda^{\mathrm{T}} (Qx - q) = \frac{1}{2} x^{\mathrm{T}} A x + (\lambda^{\mathrm{T}} Q - b^{\mathrm{T}}) x + c - \lambda^{\mathrm{T}} q. \tag{11}$$

Let

$$\frac{\partial L}{\partial x} = Ax + Q^{\mathrm{T}} \lambda - b = 0. \tag{12}$$

Hence the dual form of the problem:

$$\begin{pmatrix} A & Q^{\mathrm{T}} \\ Q & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ q \end{pmatrix} \tag{13}$$

Tips for implementation: Creating sparse matrices in a loop is costly. An efficient evaluation of A:

$$A = \mu \sum_i (G_i^{\mathrm{T}} G_i) = \sum_i (S_i \otimes I_3)^{\mathrm{T}} (S_i \otimes I_3) = (\sum_i S_i^{\mathrm{T}} S_i) \otimes I_3. \tag{14}$$

Precomputing $(\sum_i S_i^{\mathrm{T}} S_i)$ can be done via coo_matrix (in scipy) on (i, j, k) tuples.

3 Position-based Dynamics (PBD) related

3.1 Angle-based triangle bending constraint

Just came across an implementation in Magica (a Unity plugin).

References