How to find the inertia tensor (or other mass properties) of a 3D solid body represented by a triangle mesh

(Draft)
Jonathan Blow, Atman J Binstock
9 July 2004

Abstract

We show how to find the mass, center-of-mass and inertia tensor for a solid body of constant density represented by a polygonal mesh, using relatively simple math (no Green's Theorem or anything like that). This is a terse treatment of the algorithm, designed for readers who are already familiar with the basic ideas.

Overview of Algorithm

The algorithm works as follows:

1) Choose an arbitrary reference point (perhaps the origin)
2) For each triangle in the mesh:
    3) Form the tetrahedron defined by that triangle and the reference point
    4) Compute the covariance of the tetrahedron
        (by mapping a known canonical tetrahedron through an affine transform)
    5) Add this covariance to an accumulator
6) Convert the accumulator from covariance matrix to inertia tensor

The Inertia Tensor as Modified Covariance

The Tensor of Inertia of a solid body can be expressed as:

$$\mathbf{I} = \mathbf{1}_3 \operatorname{tr} \mathbf{C} - \mathbf{C} \tag{1}$$

where $\mathbf{1}_3$ is the 3x3 identity matrix and $\mathbf{C}$ is the mass-weighted covariance of the body:

$$\mathbf{C} = \sum_n m_n \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \tag{2}$$

We will focus on finding $\mathbf{C}$ for a closed, simple polyhedron. Given $\mathbf{C}$, we can easily convert to $\mathbf{I}$ using equation 1.

The Covariance of a Canonical Tetrahedron

The algorithm wants us to find the covariance of a set of arbitrary tetrahedra. The most straightforward way to compute the covariance of a tetrahedron is by solving a slightly messy integral equation. Instead, to simplify the math, we solve for the covariance of a canonical tetrahedron, for which the integral is simple. Then, we map that tetrahedron to the target via an affine transform, which yields the desired covariance.

Our canonical tetrahedron has 4 vertices $\mathbf{v}_0 = (0, 0, 0)$, $\mathbf{v}_1 = (1, 0, 0)$, $\mathbf{v}_2 = (0, 1, 0)$, $\mathbf{v}_3 = (0, 0, 1)$. To find its covariance, we convert equation 2 to continuous form:

$$\mathbf{C} = \int_V \rho\, \mathbf{x}\, \mathbf{x}^{\mathrm{T}} dV = \rho \int_V \mathbf{x}\, \mathbf{x}^{\mathrm{T}} dV \tag{3}$$

where ρ is the density of the body, which is constant, so it factors out of the integral. $\mathbf{C}$ is a 3x3 matrix, but because of the symmetry of the canonical tetrahedron, it contains only two unique scalars. To illustrate the effect of this symmetry, suppose we rotate our tetrahedron so that the x axis goes to where the y axis was, y goes to z, etc; after this rotation we have the same shape we started with, so the covariance must be the same.

Thus we need only to solve for two scalars, C_{xx} and C_{xy}. Taking the density of our canonical tetrahedron to be 1, we solve as follows:

$$C_{xx} = \int_0^1 \int_0^z \int_0^y x^2 dx dy dz = \frac{1}{60} \tag{4}$$

$$C_{xy} = \int_0^1 \int_0^z \int_0^{z-y} xy\, dx dy dz = \frac{1}{120} \tag{5}$$

thus

$$\mathbf{C}_{canonical} = \begin{bmatrix} \frac{1}{60} & \frac{1}{120} & \frac{1}{120} \\ \frac{1}{120} & \frac{1}{60} & \frac{1}{120} \\ \frac{1}{120} & \frac{1}{120} & \frac{1}{60} \end{bmatrix} \tag{6}$$

Transforming a Covariance Matrix

In order to map $\mathbf{C}_{\text{canonical}}$ to a target tetrahedron, we need the ability to manipulate covariance matrices in two ways. Firstly, we need to compute the matrix that would

result if we were to map each of **C**'s input points through a linear transformation. We'll call the transformation **A**:

$$\begin{aligned}\mathbf{C}' &= \rho\int_V (\mathbf{Ax})\ (\mathbf{Ax})^{\mathrm{T}} dV_{(\mathbf{A})} \\ &= \rho\int_V \mathbf{Axx}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} dV \det \mathbf{A} \\ &= (\det \mathbf{A})\mathbf{ACA}^{\mathrm{T}}\end{aligned} \tag{7}$$

The factor of det **A** comes from the fact that the transform by **A** distorts the volume element of the integral (which is why it is denoted $V_{(A)}$ in the first line; $dV_{(A)} = dV \det \mathbf{A}$.)

Secondly, we need to compute the matrix that would result from translating each of the input points by some offset $\mathbf{\Delta x}$:

$$\begin{aligned}\mathbf{C}' &= \mathrm{Translate}(\mathbf{C}, \mathbf{\Delta x}) \\ &= \rho\int_V (\mathbf{x}+\mathbf{\Delta x})(\mathbf{x}+\mathbf{\Delta x})^{\mathrm{T}} dV \\ &= \rho\int_V (\mathbf{xx}^{\mathrm{T}} + \mathbf{\Delta x x}^{\mathrm{T}} + \mathbf{x\Delta x}^{\mathrm{T}} + \mathbf{\Delta x\Delta x}^{\mathrm{T}}) dV \\ &= \rho\int_V \mathbf{xx}^{\mathrm{T}} dV + \rho\mathbf{\Delta x}\int_V \mathbf{x}^{\mathrm{T}} dV + \rho\int_V \mathbf{x} dV \mathbf{\Delta x}^{\mathrm{T}} + \rho\mathbf{\Delta x\Delta x}^{\mathrm{T}}\int_V dV \\ &= \mathbf{C} + m(\mathbf{\Delta x}\bar{\mathbf{x}}^{\mathrm{T}} + \bar{\mathbf{x}}\mathbf{\Delta x}^{\mathrm{T}} + \mathbf{\Delta x\Delta x}^{\mathrm{T}})\end{aligned} \tag{8}$$

where $\bar{\mathbf{x}}$ is the center of mass of the points used to construct **C**, and $m = \rho V$ is the total mass of those points.

Mapping the Canonical Tetrahedron to the Target Tetrahedron

We start with the canonical tetrahedron, map it through a linear transformation in order to make it the same shape as the target, then translate it so that it is in the same place. The vertices of the canonical tetrahedron are { $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ } as mentioned earlier. The vertices of the target tetrahedron are { $\mathbf{w}_0$, $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ }. To make the tetrahedra the same shape, we find a transform **A** such that

$$\mathbf{A}\left[\mathbf{v}_1-\mathbf{v}_0 \middle| \mathbf{v}_2-\mathbf{v}_0 \middle| \mathbf{v}_3-\mathbf{v}_0\right] = \left[\mathbf{w}_1-\mathbf{w}_0 \middle| \mathbf{w}_2-\mathbf{w}_0 \middle| \mathbf{w}_3-\mathbf{w}_0\right] \tag{9}$$

Since we have chosen the canonical tetrahedron so that

$$\left[\mathbf{v}_1-\mathbf{v}_0 \middle| \mathbf{v}_2-\mathbf{v}_0 \middle| \mathbf{v}_3-\mathbf{v}_0\right] = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{10}$$

we get

$$\mathbf{A} = \left[\mathbf{w}_1 - \mathbf{w}_0 \middle| \mathbf{w}_2 - \mathbf{w}_0 \middle| \mathbf{w}_3 - \mathbf{w}_0 \right] \tag{11}$$

So **C**' = (det **A**)**A**$\mathbf{C}_{canonical}\mathbf{A}^T$. Now we find **C**'' = Translate(**C**', $\mathbf{w}_0 - \mathbf{v}_0$) using equation 8. **C**'' is the covariance of our target tetrahedron. Since $\mathbf{v}_0 = \mathbf{0}$, **C**'' = Translate(**C**', $\mathbf{w}_0$). $\mathbf{w}_0$ is our arbitrary reference point, so if we choose it to be **0** also, we can eliminate the Translate step entirely and **A** becomes simpler also.

Other Mass Properties

We also want the mass and center-of-mass for each tetrahedron, but compared to covariance, these are trivial. The mass is just the density times the volume, where the volume is 1/6 the triple product of edges of the tetrahedron, that is, 1/6 det **A**. The center-of-mass is just the mean of the four vertex coordinates. (Note that for the purposes of evaluating Translate, we want to build a center-of-mass in which we pretend that $\mathbf{w}_0$ is the origin, then add the offset $\mathbf{w}_0$ afterward, because the covariance was computed relative to $\mathbf{w}_0$. See the sample code.)

Accumulating Results

We represent each tetrahedron as a "Body" which we call B, consisting of a triple of mass properties. $B_n = \{ \mathbf{C}_n, \mathbf{x}_n, m_n \}$ where **C** is the covariance, **x** is the center of mass, and *m* is the amount of mass.

We want to find $B_3 = B_1 + B_2$. Assuming $\mathbf{C}_1$ and $\mathbf{C}_2$ are computed about the same origin, $\mathbf{C}_3 = \mathbf{C}_1 + \mathbf{C}_2$. The center of mass $\bar{\mathbf{x}}_3 = (\bar{\mathbf{x}}_1 m_1 + \bar{\mathbf{x}}_2 m_2) / (m_1 + m_2)$. The total mass $m_3 = m_1 + m_2$.

We successively accumulate these vectors until we have $B_{total} = \Sigma B_i$. At this point, the covariance of C_{total} is still computed around our arbitrary reference point. We want to move it to the center of mass, since that is where a physics simulator is most likely to want it:

$$\mathbf{C}'_{total} = \text{Translate}(\mathbf{C}_{total}, \mathbf{w}_0 - \mathbf{x}_{total}) \tag{12}$$

Then we find $\mathbf{I}'_{total}$ from $\mathbf{C}_{total}$ using equation 1.

Why This Works

The **I**'$_{total}$ we have just computed is the correct inertia tensor for the mesh, but this may not be obvious at first, because we have summed together a group of tetrahedra that seems somewhat arbitrary – some tetrahedra may reach outside the mesh, and some may penetrate through it if the mesh is not convex.

So long as your mesh is a well-formed boundary that defines the surface of a solid body, separating inside from outside, this technique will work. Visualize a division of that solid body into convex pieces. Such a division exists for all meshes; you don't need to actually compute one, you just need to know it's there. Each convex piece is bounded by polygon faces. Each of those faces, along with our arbitrary reference point, defines a cone-like volume the likes of which we are adding in this algorithm.

If the reference point is inside the convex piece, all of the cone-like sub-pieces have positive volume, and together they cover every point inside the convex piece. Thus when we compute their covariances and sum them, we get the covariance of the convex piece.

If the reference point is outside the convex piece, some of the cones will have positive volume, and some will have negative volume. The positive-volume cones will exceed the bounds of the convex volume. But each point outside the convex volume that is inside a positive-volume cone is also covered, in a 1-to-1 correspondence, by a negative point, from one of the negative-volume cones (created from the polygon faces that are backfacing to the reference point).

This situation is a straightforward extension of the familiar case of barycentric coordinates.

[In future some illustrations will be placed here.]

[Source code is forthcoming.]

Note

Eberly's paper (in the references) describes a simplification and refactoring of Mirtich's method. We suspect that further refactoring would readily show an equivalence between this method and Eberly's. We think this method provides the advantage of being easier to understand.

References

"Efficient Texture Extraction for 2D/3D Objects in Mesh Representation", Cha Zhang and Tsuhan Chen, Carnegie Mellon University technical report. http://amp.ece.cmu.edu/Publication/Cha/icip01_Cha.pdf .

Brian Mirtich, "Fast and Accurate Computation of Polyhedral Mass Properties," Journal of Graphics Tools, volume 1, number 2, 1996.

David Eberly, "Polyhedral Mass Properties (Revisited)", http://www.magic-software.com/Documentation/PolyhedralMassProperties.pdf

Constantinos A. Balafoutis and Rajnikant V. Patel, *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*, Kluwer Academic Publishers, 1991.