Dartmouth COSC 89/189 Computational Methods for Physical Systems, Fall 2019

Instructor: Bo Zhu (bo.zhu@dartmouth.edu)

Course Notes 5: Rigid Body

2D Rigid Body Let's consider a 2D rigid body as a particle enhanced by rotation. The particle has mass m, inertial tensor I, position $\vec{x} = (x, y)^T$, velocity $\vec{u} = (u, v)^T$, orientation θ, and angular velocity ω. In 2D, we can use a scalar θ to represent the orientation as a rotation angle from the local frame to the world frame. That is, for an arbitrary vector $\vec{r}^0$ defined in the local frame, we can get its world coordinates as:

$$\vec{r} = R(\theta)\vec{r}^0,$$

with $R(\theta)$ defined as a 2D rotation matrix:

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

The state of the rigid body can be written as a 3-dimensional vector composed of position and orientation $(x, y, \theta)^T$, and its phase-space vector can be described by a 6-dimensional vector composed of the states and its time derivative $(x, y, \theta, u, v, \omega)^T$.

For a time instant t, once we calculate $\vec{x}$ and θ, the state of the rigid body is fully determined. For example, for an arbitrary point $\vec{r}_i^0$ defined in the local frame, its position in the world frame can be calculated as

$$\vec{x}_i = \vec{x} + R(\theta)\vec{r}_i^0$$

If we further define $\vec{r}_i = R(\theta)\vec{r}_i^0$, we will get

$$\vec{x}_i = \vec{x} + \vec{r}_i$$

Here the distance vector $\vec{r}_i$ is a vector pointing from the mass center to the point's current position: $\vec{r}_i = \vec{x}_i - \vec{x}$. The coordinates $\vec{x}_i$, $\vec{x}$, and $\vec{r}_i$ are all defined in the world frame.

The velocity of the point in the world frame can be calculated as the synthesis of its linear motion and its angular motion:

$$\vec{u}_i = \vec{u} + \vec{r}_i \times \omega.$$

The dynamics of a 2D rigid body can be written as:

$$\frac{d}{dt}\begin{bmatrix}\vec{x}\\ \theta\\ \vec{u}\\ \omega\end{bmatrix} = \begin{bmatrix}\vec{u}\\ \omega\\ m^{-1}\vec{f}\\ I^{-1}\tau\end{bmatrix}$$

with $\vec{f}$ as the net force and τ as the net torque. The net force is calculated as the sum of all the forces applied on the body's mass center:

$$\vec{f} = \sum_i \vec{f}_i$$

And the net torque is computed as the sum of all the torques applied on the body, with each torque calculated as the cross product between the distance vector and the force:

$$\tau = \sum_i \tau_i = \sum_i \vec{r}_i \times \vec{f}_i$$

Simulation: The simulation algorithm for a 2D rigid body can be written as:

1) Initialize $m, I, \vec{x}, \vec{u}, \theta, \omega$

2) For each time step:

$$\vec{u}^{n+1} = \vec{u}^n + \Delta t \vec{f}^n / m$$

$$\omega^{n+1} = \omega^n + \Delta t\, \tau / I$$

$$\vec{x}^{n+1} = \vec{x}^n + \Delta t\, \vec{u}^{n+1}$$

$$\theta^{n+1} = \theta^n + \Delta t\, \omega^{n+1}$$

3D Rigid Body Most of the concepts in 2D rigid body can be applied directly to the 3D case. The main difference between 3D and 2D rigid bodies is that the orientation needs to be represented using a rotation matrix R instead of an angle θ. Accordingly, the torque τ and angular velocity $\vec{\omega}$ are 3D vectors instead of scalars, and the inertial tensor is a 3×3 matrix instead of a scalar.

The 3D rigid body dynamics equation can be modified from 2D as:

$$\frac{d}{dt}\begin{bmatrix}\vec{x}\\ R\\ \vec{u}\\ \vec{\omega}\end{bmatrix} = \begin{bmatrix}\vec{u}\\ \omega \times R\\ m^{-1}\vec{f}\\ I^{-1}(\vec{\tau} - \vec{\omega} \times I\vec{\omega})\end{bmatrix}$$

Here we leverage two important facts from rigid body mechanics to describe the temporal evolution of 3D orientation and 3D rotation (the second and fourth row). First, the temporal derivative of orientation can be described as the cross product between the angular velocity and orientation:

$$\frac{dR}{dt} = \vec{\omega} \times R$$

Second, the time derivative of angular velocity due to the external torques has an addition term $\vec{\omega} \times I\vec{\omega}$, which can be written as (known as the Euler equation):

$$I\dot{\vec{\omega}} + \vec{\omega} \times I\vec{\omega} = \vec{\tau}$$

The details of derivation of these two equations can be seen in the course slides.

Simulation: The simulation algorithm for a 3D rigid body can be summarized as:

1) Initialize m, I, $\vec{x}$, $\vec{u}$, θ, ω

2) For each time step:

$$\vec{u}^{n+1} = \vec{u}^n + \Delta t \vec{f}^n / m$$

$$\vec{\omega}^{n+1} = \vec{\omega}^n + \Delta t \, I^{-1}(\vec{\tau} - \vec{\omega} \times I\vec{\omega})$$

$$\vec{x}^{n+1} = \vec{x}^n + \Delta t \, \vec{u}^{n+1}$$

$$R^{n+1} = R^n + \Delta t(\vec{\omega} \times R^n)$$

Notice that due to the accumulation of numerical error, R might deviate from a rotation matrix after a number of time steps. So a normalization step is needed to get R back to a rotation matrix every a few time steps.