Rigid Body Dynamics
Simulation and Modeling (CSCI 3010U)

Faisal Qureshi
Rigid Bodies

Particle

Rigid Body
Particle vs. Rigid Body Dynamics

- State of a particle
  - Position $\mathbf{p}$
  - Velocity $\mathbf{v}$

- State of a rigid body
  - Position $\mathbf{p}$
  - Velocity $\mathbf{v}$
  - Orientation $\theta$
  - Angular velocity $\omega$
Coordinate frames

- $[\mathbf{x}]_e$ is (1, 2) in coordinate frame described by $\mathbf{e}_1$ and $\mathbf{e}_2$
- What is $[\mathbf{x}]_u$, i.e., $[\mathbf{x}]_u$ expressed in $\mathbf{u}_1$ and $\mathbf{u}_2$?
Coordinate frames

- \( x = e_1 + 2e_2 \)
- \( x = [u_0]_e + c_1[u_1]_e + c_2[u_2]_e \)
- Note that \([x]_u = (c_1, c_2)\), so we are interested in finding values of \(c_1\) and \(c_2\).
Coordinate frames

\[
[u_0]_e + c_1[u_1]_e + c_2[u_2]_e = [x]_e
\]
\[
c_1[u_1]_e + c_2[u_2]_e = [x]_e - [u_0]_e
\]

\[
\begin{bmatrix}
[u_1]_e & [u_2]_e
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2
\end{bmatrix}
= 
\begin{bmatrix}
c_1 \\
c_2
\end{bmatrix}
= 
\begin{bmatrix}
[u_1]_e & [u_2]_e
\end{bmatrix}^{-1} ([x]_e - [u_0]_e)
\]

Change of basis

\[
[x]_u = \begin{bmatrix}
[u_1]_e & [u_2]_e
\end{bmatrix}^{-1} ([x]_e - [u_0]_e)
\]
Coordinate Frames Exercise - Part 1

What is $[x]_u$?
Coordinate Frames Exercise - Part 2

What is $[x]_u$?
Rigid Bodies in 2D

Angular Velocity

- $\omega = \frac{d\theta}{dt}$
- Units are radians per second
- Radian is the angle subtended by an arc whose length is equal to its radious: $\theta = \frac{l}{r}$

Linear Velocity at a Point on the Body

- $v = r\omega$
Rigid Bodies in 3D

- Unlike 2D, orientation in 3D cannot be described using any angle.
- There are many schemes for describing rotations in 3D.
- We will use a 3x3 rotation matrix $\mathbf{R}$.
- We need to find the relationship between angular velocity and rotation matrix.

*We will return to this later.*
Equations of Motions for Rigid Bodies

Force acting on a Rigid Body

- Net force acting on an object is the rate of change of its linear momentum.
  \[ \frac{d\mathbf{P}}{dt} = \mathbf{F} \]
- Linear momentum: \( \mathbf{P} = m\mathbf{v} \), where \( m \) is the mass of the object and \( \mathbf{v} \) is its linear velocity

Torque acting on a rigid body

- Net torque acting on an object (about point \( \mathbf{o} \)) is the rate of change of its angular momentum.
  \[ \frac{d\mathbf{L}}{dt} = \mathbf{N} \]
- Angular momentum: \( \mathbf{L} = \mathbf{I}\omega \), where \( \mathbf{I} \) is the inertia tensor and \( \omega \) is its angular velocity (about point \( \mathbf{o} \))
Torque

- Torque (in this example, clockwise or counter-clockwise):

\[ T = d \times F \]

- When force passes through the center of mass, the associated \( d \) vector is zero; therefore, this force produces no torque or rotational effect.
Center of Mass (COM)

- The center of mass is the mean location of all the mass of the body.
- The center of mass $\mathbf{r}$ is defined as

$$
\begin{align*}
    r_x &= \frac{1}{m} \int \rho(x, y, z) x dV \\
    r_y &= \frac{1}{m} \int \rho(x, y, z) y dV \\
    r_z &= \frac{1}{m} \int \rho(x, y, z) z dV
\end{align*}
$$

where $\rho(x, y, z)$ is the density at point $(x, y, z)$. 
COM as the origin of the body coordinate frame

- Selecting COM as the origin of the body coordinate frame greatly simplifies the equation of motions.
- Any force applied to (or passing through) the COM doesn’t induce rotation.

Force 1 results in translation only
Force 2 results in translation only
Force 3 results in both translation and rotation
Center of mass of a rectangular brick with point masses at its 8 vertices

- $x_i$, $y_i$, and $z_i$ are $i$th vertex location in the world coordinates.
- $m_i$ is the value of the point mass at vertex $i$.

- $(r_x, r_y, r_z)$ is the center of mass of the rectangular brick in the world coordinates.

$$r_x = \left( \sum_{i} m_i x_i \right) / \left( \sum_{i} m_i \right)$$

$$r_y = \left( \sum_{i} m_i y_i \right) / \left( \sum_{i} m_i \right)$$

$$r_z = \left( \sum_{i} m_i z_i \right) / \left( \sum_{i} m_i \right)$$
**COM - Example**

% MATLAB Code
% 5 ----- 6
% /       /|
% 4 ----- 3 7
% |       |
% 1 ----- 2
%

m = ones(1,8) / 8.;

r = zeros(8, 3);
r(1,:) = [1, 1, 1];
r(2,:) = r(1,:) + [w, 0, 0];
r(3,:) = r(2,:) + [0, h, 0];
r(4,:) = r(1,:) + [0, h, 0];
r(5,:) = r(4,:) + [0, 0, d];
r(6,:) = r(3,:) + [0, 0, d];
r(7,:) = r(2,:) + [0, 0, d];
r(8,:) = r(1,:) + [0, 0, d];

% compute center of mass first
M = sum(m);

com = ( m(1)*r(1,:) + ... 
       m(2)*r(2,:) + ... 
       m(3)*r(3,:) + ... 
       m(4)*r(4,:) + ... 
       m(5)*r(5,:) + ... 
       m(6)*r(6,:) + ... 
       m(7)*r(7,:) + ... 
       m(8)*r(8,:) ) / M
Inertia Tensor

Inertia tensor provides a concise description of the mass distribution around the center of mass

\[
I = \begin{bmatrix}
I_{xx} & -I_{xy} & -I_{xz} \\
-I_{xy} & I_{yy} & -I_{yz} \\
-I_{xz} & -I_{yz} & I_{zz}
\end{bmatrix}
\]

\[
I_{xx} = \int \rho(x, y, z)(y^2 + z^2) \, dV
\]
\[
I_{yy} = \int \rho(x, y, z)(x^2 + z^2) \, dV
\]
\[
I_{zz} = \int \rho(x, y, z)(y^2 + x^2) \, dV
\]
\[
I_{xy} = \int \rho(x, y, z)xy \, dV
\]
\[
I_{xz} = \int \rho(x, y, z)xz \, dV
\]
\[
I_{yz} = \int \rho(x, y, z)yz \, dV
\]
Inertia Tensor - Discretization

\[ I = \begin{bmatrix}
I_{xx} & -I_{xy} & -I_{xz} \\
-I_{yx} & I_{yy} & -I_{yz} \\
-I_{zx} & -I_{zy} & I_{zz}
\end{bmatrix} \]

- \( m_i \) are point masses.

\[ I_{xx} = \sum_i m_i (y_i^2 + z_i^2) \]
\[ I_{yy} = \sum_i m_i (z_i^2 + x_i^2) \]
\[ I_{zz} = \sum_i m_i (x_i^2 + y_i^2) \]
\[ I_{xy} = I_{yz} = \sum_i m_i x_i y_i \]
\[ I_{xz} = I_{zx} = \sum_i m_i x_i z_i \]
\[ I_{yz} = I_{zy} = \sum_i m_i y_i z_i \]
% CONTINUED FROM PREVIOUS.
% com - center of mass (1x3)
% r – vertex locations (8x3)

% now lets compute inertia tensor
rp = r - repmat(com, 8, 1);

I = zeros(3,3);

mrp = repmat(m',1,3) .* rp

I(1,1) = rp(:,2)' * mrp(:,2) + rp(:,3)' * mrp(:,3);
I(2,1) = - rp(:,1)' * mrp(:,2);
I(3,1) = - rp(:,1)' * mrp(:,3);
I(1,2) = - rp(:,2)' * mrp(:,3);
I(2,2) = rp(:,1)' * mrp(:,1) + rp(:,3)' * mrp(:,3);
I(3,2) = - rp(:,2)' * mrp(:,3);
I(1,3) = I(3,1);
I(2,3) = I(3,2);
I(3,3) = rp(:,1)' * mrp(:,1) + rp(:,2)' * mrp(:,2);
The inertia tensor $\mathbf{I}$ that we just computed is expressed in the world coordinate frame. Consequently it changes as the orientation of the rigid body changes.

- We can express the inertia tensor in the body coordinate frame.
- We refer to inertia tensor in the body coordinate frame as $\mathbf{I}_{body}$.
- $\mathbf{I}_{body}$ doesn’t change as the orientation of the body changes.
- $\mathbf{I}_{body}$ is diagonal, i.e.,

$$
\mathbf{I}_{body} = \begin{bmatrix}
I_{11} & 0 & 0 \\
0 & I_{22} & 0 \\
0 & 0 & I_{33}
\end{bmatrix}
$$

- Inverse of $\mathbf{I}_{body}$:

$$
\mathbf{I}_{body}^{-1} = \begin{bmatrix}
\frac{1}{I_{11}} & 0 & 0 \\
0 & \frac{1}{I_{22}} & 0 \\
0 & 0 & \frac{1}{I_{33}}
\end{bmatrix}
$$
Computing $I_{body}$

**Option 1**

**Diagonalize I**

- Compute eigenvectors and eigenvalues of $I$
- Eigenvalues form the diagonal matrix $I_{body}$
- Eigenvectors form the 3-by-3 rotation matrix $R$ that describes the orientation of the rigid body
- This is the preferred approach

**Option 2**

**Use 3-by-3 rotation matrix $R$ that describes the orientation of the rigid body**

- $I_{body} = R^T I R$
Inertia Tensor

- Inertia tensors are available for many canonical objects: rectangles, circles, spheres, etc.
- Efficient algorithms exist to compute inertia tensor, center of mass, body coordinate frames a given polygonal model of an object.
- Many tools exist to construct polygonal models of 2D/3D rigid objects.
Body coordinate frame

Attach a coordinate frame with each rigid body

- Origin = center of mass (defined in the world frame)
- Axes = defined in the world coordinate frame by a 3-by-3 rotation matrix $\mathbf{R}$. Columns of $\mathbf{R}$ define the $x$, $y$ and $z$ axes of the body coordinate frame
- Inertia tensor $\mathbf{I}_{\text{body}}$ is constant and diagonal in this frame

From body coordinate frame to world coordinate frame.

$$
\mathbf{p}_{\text{world}} = \mathbf{R}\mathbf{p}_{\text{body}} + \mathbf{x}
$$
World and Body Coordinate Frames

World coordinate frame

- Collision detection and response
- Display and visualization

Body coordinate frame

- Compute quantities such as inertia tensor once and store them for later use.
# Rigid Body Dynamics

<table>
<thead>
<tr>
<th>State variables</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Position</strong></td>
<td>( \mathbf{x} )</td>
<td>1 by 3 vector</td>
</tr>
<tr>
<td><strong>Orientation</strong></td>
<td>( \mathbf{R} )</td>
<td>3 by 3 rotation matrix</td>
</tr>
<tr>
<td><strong>Linear Momentum</strong></td>
<td>( \mathbf{P} )</td>
<td>1 by 3 vector</td>
</tr>
<tr>
<td><strong>Angular Momentum</strong></td>
<td>( \mathbf{L} )</td>
<td>1 by 3 vector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constants</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mass</strong></td>
<td>( m )</td>
<td>scalar</td>
</tr>
<tr>
<td><strong>Inertia tensor</strong></td>
<td>( \mathbf{I}_{body} )</td>
<td>3 by 3 matrix (in body frame)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Derived quantities</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear velocity</strong></td>
<td>( \mathbf{v} )</td>
<td>1 by 3 vector</td>
</tr>
<tr>
<td><strong>Angular velocity</strong></td>
<td>( \mathbf{\omega} )</td>
<td>1 by 3 vector</td>
</tr>
<tr>
<td><strong>Inertia tensor</strong></td>
<td>( \mathbf{I}^{-1} )</td>
<td>3 by 3 matrix (in world frame)</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total force</strong></td>
<td>( \mathbf{F} )</td>
<td>1 by 3 vector</td>
</tr>
<tr>
<td><strong>Total torque</strong></td>
<td>( \mathbf{T} )</td>
<td>1 by 3 vector</td>
</tr>
</tbody>
</table>
Rigid Body Dynamics

Linear effects

- $\frac{dx}{dt} = v$
- $\frac{dP}{dt} = F$
- $v = \frac{P}{m}$

Angular effects

- $\frac{dR}{dt} = \omega^* R$, where

$$
\begin{bmatrix}
0 & -\omega_x & \omega_y \\
\omega_z & 0 & -\omega_x \\
\omega_y & \omega_x & 0
\end{bmatrix}
$$

- $\frac{dL}{dt} = N$
- $\omega = I^{-1} L$
- $I^{-1} = R I_{body}^{-1} R^T$
Rigid Body Dynamics

// x - position
state[0] = x[0];
state[1] = x[1];
state[2] = x[2];

// R - orientation
state[3] = R[0][0];
state[4] = R[1][0];
state[5] = R[2][0];
state[6] = R[0][1];
state[7] = R[1][1];
state[8] = R[2][1];
state[9] = R[0][2];
state[10] = R[1][2];

// P - linear momentum
state[12] = P[0];
state[13] = P[1];
state[14] = P[2];

// L - angular momentum
state[15] = L[0];
state[16] = L[1];
state[17] = L[2];

// t - OSP needs it
state[18] = 0.0

// dx/dt = v
rate[0] = v[0];
rate[1] = v[1];
rate[2] = v[2];

// dr/dt = w* R
double[][] Rdot = mult(star(omega), R);
rate[3] = Rdot[0][0];
rate[4] = Rdot[1][0];
rate[5] = Rdot[2][0];
rate[6] = Rdot[0][1];
rate[7] = Rdot[1][1];
rate[8] = Rdot[2][1];
rate[9] = Rdot[0][2];
rate[10] = Rdot[1][2];

// dP/dt = force
rate[12] = force[0];
rate[13] = force[1];
rate[14] = force[2];

// dL/dt = torque
rate[15] = torque[0];
rate[16] = torque[1];
rate[17] = torque[2];

// dt/dt = 1
rate[18] = 1;

odeSolver.step();

// x
x[0] = state[0];
x[1] = state[1];
x[2] = state[2];

// R
R[0][0] = state[3];
R[1][0] = state[4];
R[2][0] = state[5];
R[0][1] = state[6];
R[1][1] = state[7];
R[2][1] = state[8];
R[0][2] = state[9];
R[1][2] = state[10];
R[2][2] = state[11];
R = orthonormalize(R);

// P
P[0] = state[12];
P[1] = state[13];
P[2] = state[14];

// L
L[0] = state[15];
L[1] = state[16];
L[2] = state[17];
Iinv = mult(R, mult(IbodyInv, transpose(R)));
omega = mult(Iinv, l);

Init: Flatten state variables into a state vector
Rate[] encodes 1st order ODE for our system
Let ODE solve the state and then copy the state back to our state variables x, R, L and T.
Rigid Body Dynamics: Numerical Considerations

- Over time numerical errors accumulate in rotation matrix $\mathbf{R}$
- This effects our computation of $\mathbf{I}$ and $\omega$
- Orthonormalize $\mathbf{R}$ after every timestep

Orthonormalization

1. Normalize $\mathbf{R}_1$ (excluding last column)
2. $\mathbf{R}_1 \times \mathbf{R}_2 = \mathbf{R}_3$ (normalize)
3. $\mathbf{R}_3 \times \mathbf{R}_1 = \mathbf{R}_2$ (normalize)

Here $\mathbf{R}_i$ represent the $i$-th row of matrix $\mathbf{R}$

Errors were shifted in the matrix
Representing Rotations

- We chose to represent rotations as 3-by-3 rotation matrices
- Quaternions can be used to represent rotations
- Most rigid body dynamics systems use quaternions
- See Ch. 17 of the textbook
Representing Rigid Bodies

- A rigid body has a shape that does not change over time
- It can translate through space and rotate
- A rigid body occupies a volume of space
- The distribution of its mass over this volume determines its motion or dynamics
- Shape representation is studied extensively in computer graphics and some areas of mechanical engineering and mathematics
- There are many ways of representing shape, each with a different set of advantages and disadvantages
- We will stick to polygons
Shape Representation using Polygons

- The surface of the object is represented by a collection of polygons.
- The polygons are connected across their edges to form a continuous surface.
- In order to have a well-behaved representation we need to constrain our polygons.
- First all of the polygons must be convex, note that we can always convert a concave polygon into two or more convex ones.

Convex Polygon
Concave Polygon
Non-planar Polygon
Self-intersecting Polygon
Triangle
Shape Representation using Triangles

- We will stick to triangles
- Any convex polygon can be converted to a collection of triangles

Advantages

- We are only dealing with one type of polygon, a uniform representation
- Triangles are the simplest polygon, makes our algorithms simpler
- Many modeling programs allow us to construct polygonal models
- Easy to display
- Many efficient algorithms exist for manipulating triangles

Disadvantages

- Not a compact representation
- Not a good approximation for curved surfaces
Other Types of Dynamics

- Articulated figures
  - Rigid bodies connected by joints and hinges
  - Used to model the dynamics of human figures
- Vehicle dynamics used to model the dynamics of various kinds of vehicles
- Deformable objects
  - Cloth, soft toys, etc.
- These are more complicated than what we have seen so far
Readings

- Ch. 17 of the textbook