CS-184: Computer Graphics
Lecture #5: 3D Transformations and Rotations

Prof. James O'Brien
University of California, Berkeley

1

Today

- Transformations in 3D
- Rotations
  - Matrices
  - Euler angles
  - Exponential maps
  - Quaternions

2
## 3D Transformations

- Generally, the extension from 2D to 3D is straightforward
  - Vectors get longer by one
  - Matrices get extra column and row
  - SVD still works the same way
  - Scale, Translation, and Shear all basically the same
  - Rotations get interesting

### Translations

<table>
<thead>
<tr>
<th>( \tilde{A} )</th>
<th>( \tilde{A} )</th>
</tr>
</thead>
</table>
| \[
1 0 t_x \\
0 1 t_y \\
0 0 1
\]
| \[
1 0 0 t_x \\
0 1 0 t_y \\
0 0 1 t_z \\
0 0 0 1
\]

*For 2D*  
*For 3D*
### Scales

\[
\tilde{A} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{For 2D}
\]

\[
\tilde{A} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{For 3D}
\]

(Axis-aligned!)

### Shears

\[
\tilde{A} = \begin{bmatrix} 1 & h_{xy} & 0 \\ h_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{For 2D}
\]

\[
\tilde{A} = \begin{bmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{For 3D}
\]

(Axis-aligned!)
Shears

\[ \tilde{A} = \begin{bmatrix}
1 & h_{xy} & h_{xz} & 0 \\
 h_{yx} & 1 & h_{yz} & 0 \\
 h_{zx} & h_{zy} & 1 & 0 \\
 0 & 0 & 0 & 1
\end{bmatrix} \]

Shears y into x

Rotations

- 3D Rotations fundamentally more complex than in 2D
  - 2D: amount of rotation
  - 3D: amount and axis of rotation
Rotations

- Rotations still orthonormal
- \( \text{Det}(R) = 1 \neq -1 \)
- Preserve lengths and distance to origin
- 3D rotations DO NOT COMMUTE!
- Right-hand rule
- Unique matrices

Axis-aligned 3D Rotations

- 2D rotations implicitly rotate about a third out of plane axis
11

Axis-aligned 3D Rotations

- 2D rotations implicitly rotate about a third out of plane axis

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

\[
\hat{R} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Note: looks same as \( \hat{R} \)

12

Axis-aligned 3D Rotations

\[
R_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
\]

\[
R_y = \begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix}
\]

\[
R_z = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

“Z is in your face”
Axis-aligned 3D Rotations

\( R_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix} \)

Also right handed "Zup"

\( R_y = \begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix} \)

\( R_z = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix} \)

Also known as "direction-cosine" matrices
Arbitrary Rotations

- Can be built from axis-aligned matrices:
  \[ \mathbf{R} = \mathbf{R}_z \cdot \mathbf{R}_y \cdot \mathbf{R}_x \]
- Result due to Euler... hence called Euler Angles
- Easy to store in vector
- But NOT a vector:
  \[ \mathbf{R} = \text{rot}(x, y, z) \]
Arbitrary Rotations

- Allows tumbling
- Euler angles are non-unique
- Gimbal-lock
- Moving -vs- fixed axes
  - Reverse of each other

Exponential Maps

- Direct representation of arbitrary rotation
- AKA: axis-angle, angular displacement vector
- Rotate $\theta$ degrees about some axis
- Encode $\theta$ by length of vector

\[
\theta = |\vec{r}|
\]
Exponential Maps

• Given vector \( \mathbf{r} \), how to get matrix \( \mathbf{R} \)

• Method from text:
  1. rotate about \( x \) axis to put \( \mathbf{r} \) into the x-y plane
  2. rotate about \( z \) axis align \( \mathbf{r} \) with the x axis
  3. rotate \( \theta \) degrees about x axis
  4. undo #2 and then #1
  5. composite together

Exponential Maps

• Vector expressing a point has two parts
  • \( \mathbf{x}_\perp \) does not change
  • \( \mathbf{x}_\perp \) rotates like a 2D point
Exponential Maps

\[ x' = x_0 + x_\perp \sin(\theta) + x_\parallel \cos(\theta) \]

Exponential Maps

- Rodriguez Formula

\[ x' = \hat{r} (\hat{r} \cdot x) + \sin(\theta) (\hat{r} \times x) - \cos(\theta) (\hat{r} \times (\hat{r} \times x)) \]

Linear in \( x \)

Actually a minor variation ...
**Exponential Maps**

- Building the matrix

\[ \mathbf{x}' = ((\hat{\mathbf{r}} \hat{\mathbf{r}}) + \sin(\theta)(\hat{\mathbf{r}} \times) - \cos(\theta)(\hat{\mathbf{r}} \times)(\hat{\mathbf{r}} \times)) \mathbf{x} \]

\[
(\hat{\mathbf{r}} \times) = \begin{bmatrix}
0 & -\hat{r}_x & \hat{r}_y \\
\hat{r}_x & 0 & -\hat{r}_z \\
-\hat{r}_y & \hat{r}_z & 0
\end{bmatrix}
\]

Antisymmetric matrix
\[ (a \times b) = a \times b \]
Easy to verify by expansion

---

**Exponential Maps**

- Allows tumbling
- No gimbal-lock!
- Orientations are space within $\pi$-radius ball
- Nearly unique representation
- Singularities on shells at $2\pi$
- Nice for interpolation
Exponential Maps

• Why exponential?
  - Instead of rotating once by $\theta$, let's do $n$ small rotations of $\theta/n$.
  - Now the angle is small, so the rotated $x$ is approximately:
    $$x + (\theta/n)\hat{r} \times x$$
    $$= \left(1 + \frac{(\hat{r} \times \theta)}{n}\right)x$$
  - Do it $n$ times and you get:
    $$x' = \left(1 + \frac{(\hat{r} \times \theta)}{n}\right)^n x$$

• Remind you of anything?
  - $\lim_{n \to \infty} \left(1 + \frac{a}{n}\right)^n$ is a definition of $e^a$.

• So the rotation we want is the exponential of $(\hat{r} \times \theta)$!
• In fact you can just plug it into the infinite series...
Exponential Maps

- Why exponential?
- Recall series expansion of $e^x$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

---

Exponential Maps

- Why exponential?
- Recall series expansion of $e^x$
- Euler: what happens if you put in $i\theta$ for $x$

$$e^{i\theta} = 1 + \frac{i\theta}{1!} + \frac{-\theta^2}{2!} + \frac{-i\theta^3}{3!} + \frac{\theta^4}{4!} + \cdots$$

$$= \left(1 + \frac{-\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right) + i\left(\frac{\theta}{1!} + \frac{-\theta^3}{3!} + \cdots\right)$$

$$= \cos(\theta) + i\sin(\theta)$$
### Exponential Maps

- Why exponential?

\[
e^{(\hat{r} \times \theta)} = I + \frac{(\hat{r} \times \theta)}{1!} + \frac{(\hat{r} \times \theta)^2}{2!} + \frac{(\hat{r} \times \theta)^3}{3!} + \frac{(\hat{r} \times \theta)^4}{4!} + \cdots
\]

But notice that: \((\hat{r} \times)^3 = - (\hat{r} \times)\)

\[
e^{(\hat{r} \times \theta)} = I + \frac{(\hat{r} \times \theta)}{1!} + \frac{(\hat{r} \times \theta)^2}{2!} - \frac{(\hat{r} \times \theta)^3}{3!} + \frac{(\hat{r} \times \theta)^4}{4!} + \cdots
\]

### Exponential Maps

\[
e^{(\hat{r} \times \theta)} = I + \frac{(\hat{r} \times \theta)}{1!} + \frac{(\hat{r} \times \theta)^2}{2!} + \frac{(\hat{r} \times \theta)^3}{3!} + \frac{(\hat{r} \times \theta)^4}{4!} + \cdots
\]

\[
e^{(\hat{r} \times \theta)} = (\hat{r} \times) \left( \frac{\theta}{1!} + \frac{\theta^3}{3!} + \cdots \right) + I + (\hat{r} \times)^2 \left( \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots \right)
\]

\[
e^{(\hat{r} \times \theta)} = (\hat{r} \times) \sin(\theta) + I + (\hat{r} \times)^2 (1 - \cos(\theta))
\]
Quaternions

- More popular than exponential maps
- Natural extension of \( e^{\theta} = \cos(\theta) + i\sin(\theta) \)
- Due to Hamilton (1843)
  - Interesting history
  - Involves “hermaphroditic monsters”

\[ e^{i\theta} = \cos(\theta) + i\sin(\theta) \]

\[ i^2 = j^2 = k^2 = -1 \]

\[ ij = k \quad ji = -k \]
\[ jk = i \quad kj = -i \]
\[ ki = j \quad ik = -j \]

\[ q = (z_1, z_2, z_3, s) = (z, s) \]

\[ q = iz_1 + jz_2 + kz_3 + s \]
### Quaternions

- Multiplication natural consequence of defn.
  \[ q \cdot p = (z_q s_p + z_p s_q + z_p \times z_q, s_p s_q - z_p \cdot z_q) \]
- Conjugate
  \[ q^* = (-z, s) \]
- Magnitude
  \[ ||q||^2 = z \cdot z + s^2 = q \cdot q^* \]

- Vectors as quaternions
  \[ v = (v, 0) \]
- Rotations as quaternions
  \[ r = (\hat{r} \sin \frac{\theta}{2}, \cos \frac{\theta}{2}) \]
  - Rotating a vector
    \[ x' = r \cdot x \cdot r^* \]
  - Composing rotations
    \[ r = r_1 \cdot r_2 \quad \text{Compare to Exp. Map} \]
Quaternions

- No tumbling
- No gimbal-lock
- Orientations are “double unique”
- Surface of a 3-sphere in 4D \[ ||\mathbf{q}|| = 1 \]
- Nice for interpolation

Interpolation
Rotation Matrices

• Eigen system
  • One real eigenvalue
  • Real axis is axis of rotation
  • Imaginary values are 2D rotation as complex number

• Logarithmic formula

\[ \hat{x} = \ln(R) = \frac{\theta}{2 \sin \theta} (R - R^T) \]
\[ \theta = \cos^{-1}\left(\frac{\text{Tr}(R) - 1}{2}\right) \]

Similar formulae as for exponential...

Rotation Matrices

• Consider:

\[ R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \]

\[ RI = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

• Columns are coordinate axes after
  (true for general matrices)
• Rows are original axes in original system
  (not true for general matrices)
Scene Graphs

- Draw scene with pre-and-post-order traversal
  - Apply node, draw children, undo node if applicable
- Nodes can do pretty much anything
  - Geometry, transformations, groups, color, switch, scripts, etc.
  - Node types are application/implementation specific
- Requires a stack to implement “undo” post children
- Nodes can cache their children
- Instances make it a DAG, not strictly a tree
- Will use these trees later for bounding box trees