CS 188 Robotics Week 2

Aidan Jan

April 8, 2025

Rigid Body Motions

Representing Position

[FILL 11]

2D Transformation: Translation

Translate the point p to p' with $T = (\mathrm{d}x, \mathrm{d}y)$:

$$p' = T + p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

[FILL 12, graph only]

2D Transformation: Rotation

$$p' = R \cdot p$$

Here we are doing a counter-clockwise rotation [FILL 13] The triangle here helps us visualize the rotation. However, we are still considering one 2D point p.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Combining Rotation and Transformation

$$p' = R \cdot p + T$$

In general, a matrix multiplication lets us linearly combine components of a vector.

- It is sufficient for representing rotation, but we can't add a constant :(

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

Homogeneous Coordinates

- The solution? Stick a "1" at the end of every vector.
- Now, we can do rotation AND translation
- This is called "homogeneous coordinates"

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- Our old way of representing point is called "Cartesian coordinate system"

Cartesian and Homogeneous Coordinate

- A point in cartesian coordinate $\langle x, y \rangle$ can be represented by $\langle sx, sy, s \rangle$ in homogeneous coordinate, where s is any scalar number.
  - For example, $\langle 2, 3 \rangle$ in cartesian coordinate can be represented as $\langle 2, 3, 1 \rangle$ or $\langle 4, 6, 2 \rangle$, or $\langle 1, 1.5, 0.5 \rangle$, etc. in homogeneous coordinates
  - A point in homogeneous coordinate $\langle x, y, z \rangle$ can be converted to cartesian coordinates by dividing the last element $\langle x/z, y/z \rangle$
  - Similarly for higher dimensions

Transformation Matrices

Representing rotation and translation homogeneous coordinates

- 2D Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 2D Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Now we can represent both the rotation and translation operation with one transformation matrix.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Note: Following the matrix multiplication rule, a transformation matrix always apply rotation first, then translation.

- Matrix multiplication is *not* commutative.

3D Transformation

Our examples so far were all in 2D, but we often want a 3D representation [FILL 21]

Right Hand Rule

[FILL 24] Most of robotics system's coordinate system follows the right hand rule

- Not always true (e.g., in some graphics and physics engine directX Unity)
- Therefore, be careful!

3D Transformation: Translation

A 3D point (x, y, z), translation by t_x, t_y, t_z:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

[FILL 26]

3D Transformation: Rotation

- A rotation in 2D is around a point
- A rotation in 3D is around an <u>axis</u> (a line with direction)
  - rotation direction also follows right hand rule (thumb points to the axis direction, other fingers points towards the **positive** rotation direction)
  - It is a 3D space, not just 1D
  - most common choices for rotation axes are the x, y, z-axes (Euler angle representation)

  [FILL 27]

3D Rotation Matrices

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

Reference Frames (Coordinate System)

- Up to now we have look at transformation in a single reference frame. However, in a complex robotic system we often need to define many reference frames.
- The same 3D point might have different coordinate if we use different reference frames, next we will learn how to transform between different reference frames.

Example: green dot's coordinate is $(2, 1)$ in blue reference frame, but its coordinate is $(4, 4)$ in red reference frame. [FILL 30] Changing coordinate frame is like translating between two different languages that describes the same thing.

Examples:

[FILL 31, full]

Changing Reference Frames

- We define two coordinate frames A and B
- A Point P:
  - P's coordinate in Frame A is ${}^{A}P = (4, 4)$
  - P's coordinate in Frame B is ${}^{B}P = (2, 1)$
- Transformations between reference frames we will use the notation ${}^{A}T_{B}$ (FROM frame is in the bottom right and the TO frame is in the top left.)
- To transform ${}^{B}P$'s reference frame from B to A, we just need to apply ${}^{A}T_{B}$ to ${}^{B}P$.

$$ {}^{A}P = {}^{A}T_{B} \cdot {}^{B}P $$

How do we compute ${}^{A}T_{B}$?

- Suppose the point P is rigidly attached to reference Frame B.
- No matter where the reference B, point P is its coordinates with respect to Frame B is always given by ${}^{B}P = (2, 1)$.

[FILL 37, edited, 2x2 grid] First, let's make Frame B identical to Frame A. Now, ${}^{A}P = {}^{B}P = (2, 1)$. Now, simply translate Frame B together with $d = (2, 3)$, we will get the ${}^{A}P = {}^{B}P + d$.
Therefore in this case,

$$ {}^{A}T_{B} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} $$

(There is no rotation in this case, only translation)

- If there is a rotation, first rotate the frame so it is aligned with the target, then do a translation.

[FILL 47, edited] If we combine this rotation and translation into one transformation matrix, we get:

$$ T = \begin{bmatrix} R_{\theta} & d \\ 0_{n} & 1 \end{bmatrix} $$

This is the transformation ${}^{A}T_{B}$ that change the coordinate frame from B to A.

- However, geometrically it describes the motion from Frame A to B.
- ${}^{A}T_{B}$ also describes Frame B's "pose" in Frame A, where the rotation component R describes the B's orientation in Frame A, and the translation represents B's position in Frame A.

[FILL 52]

Change of Basis Summary

What is ${}^{A}T_{B}$?

- ${}^{A}T_{B}$ is a rigid transformation matrix (3x3 matrix in 2D, 4x4 in 3D)
- ${}^{A}T_{B}$ represents the transform that **change the coordinate frame from B to A:** ${}^{A}P = {}^{A}T_{B}\, {}^{B}P$
- ${}^{A}T_{B}$ geometrically describes the motion from Frame A to B.
- ${}^{A}T_{B}$ is also the pose of coordinate frame (B) in the coordinate frame (A); that describes the position and orientation of Frame B in Frame A.

Composing Transformation

[FILL 54, fill]

Chained 3D Rotation

We can chain a sequence of Euler angle rotations (multiple sequence of rotation matrix) to get a general 3D rotation.

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha\cos\alpha & 0 & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

There are a few things to note when writing down the sequence of rotation:

1. Rotation matrix is non-commutative - order matters!
2. Be aware of which sequence convention you are using when describing the 2nd and 3rd rotations: **extrinsic** rotation (fixed global frame), or **intrinsic** rotation? (last rotated coordinate system) - they are different.

Extrinsic vs. Intrinsic Rotation

[FILL 57, 58, incl. text, edited] The final rotation R is the same. However, the order of describing rotation sequence is opposite in each convention.

- (Use premultiply!)

Rotation Matrix

Rotation matrix has a number of highly useful properties:

- R is an orthonormal matrix: Its columns are orthogonal unit vectors. ($R^{-1} = R^T$)
  - This does not apply to general transformation matrices.
- determinant of the matrix $|R| = 1$
- The length of the vector is unchanged after transformation

Other 3D Rotation Representations

There are many ways to specify rotation

- Rotation matrix
- Euler angles: 3 angles about 3 axes
- Axis-angle representation
- Quaternions

Axis Angle Representation

Parameterize a 3D rotation by two quantities: a unit vector e indicating the direction of an axis of rotation, and an angle θ describing the magnitude of the rotation about the axis.

- Euler's rotation theorem: any rotation or sequence of rotations of a rigid body in a three-dimensional space is equivalent to a single rotation about a single fixed axis.

[FILL 63]

Quaternions

Uses a unit four-dimensional vector (x, y, z, w) to represent rotation.

- If the rotation is (v_1, v_2, v_3, θ) in angle-axis representation, it can be written in quaternion as:

$$
\begin{aligned}
x &= v_1 \sin \frac{\theta}{2} \\
y &= v_2 \sin \frac{\theta}{2} \\
z &= v_3 \sin \frac{\theta}{2} \\
w &= \cos \frac{\theta}{2}
\end{aligned}
$$

$$x^2 + y^2 + z^2 + w^2 = 1$$

- the above is a 4-dimensional vector on a 4D sphere.

Quaternions are a very popular parameterization due to the following properties:

- More <u>compact</u> than the matrix representation (4 numbers instead of 9 numbers)
- The quaternion elements vary <u>continuously</u> over the unit sphere in $\mathbb{R}^4$ as the orientation changes, avoiding <u>discontinuous</u> jumps (it is important for many optimization or learning algorithms).

To inverse a quaternion:

- keep the rotation axis, rotate backward
- Inverse of (x, y, z, w) is $(x, y, z, -w)$
- (x, y, z, w) is equivalent to $(-x, -y, -z, -w)$