MEAM 520
Introduction to Robotics

Katherine J. Kuchenbecker, Ph.D.
Mechanical Engineering and Applied Mechanics Department
SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 1: August 28, 2013

What is this class about?

Robots, particularly robot arms and some mobile robots

Tentative list of topics:

- Translation and Rotation in 2D and 3D
- Forward Kinematics of Serial Manipulators
- Trajectory Planning
- Inverse Kinematics
- Velocity Kinematics
- Independent Joint Dynamics and Control
- Manipulator Dynamics and Control
- Haptic Interfaces
- Teleoperation
- Simple Computer Vision
- Mobile Robots

Piazza Use

- We will post all announcements, course materials, and assignments via Piazza.
- You should use Piazza for all questions and comments on this class.
- You can post anonymously (hidden name) if you don't want other students to identify you.
- You can post privately (only to the instructors) if you think your question may disclose important information about an assignment.
- Use email only for sensitive personal topics.

Activity

1. Find a partner. Sit together.
2. Obtain a blindfold, a cup, and some candy.
3. Devise a general non-linguistic auditory communication scheme that enables a sighted partner to **direct a blindfolded partner to pick up a piece of candy and put it in the cup** without knocking over the cup or touching anything else.
4. Test your scheme, using different cup and candy locations and switching roles.

5. Blindfold one partner. No peeking.
6. Use the same communication scheme to direct your partner to do a different secret task:

MEAM 520

Background

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 2: September 3, 2013

Robotics is difficult
or, somewhat equivalently,
humans are very good
at what they do.

Preface of Spong, Hutchinson, and Vidyasagar

Robotics is a subset of Mechatronics,
the synergistic integration of mechanics,
electronics, controls, and computer science.

The robot is the ultimate mechatronic system.

Preface of Spong, Hutchinson, and Vidyasagar

What applications can robots do?

manufacturing
household cleaning
lawn mowing
undersea exploration
planetary exploration
satellite retrieval and repair
defusing explosive devices
radioactive work
prosthetics
surgery
medical training

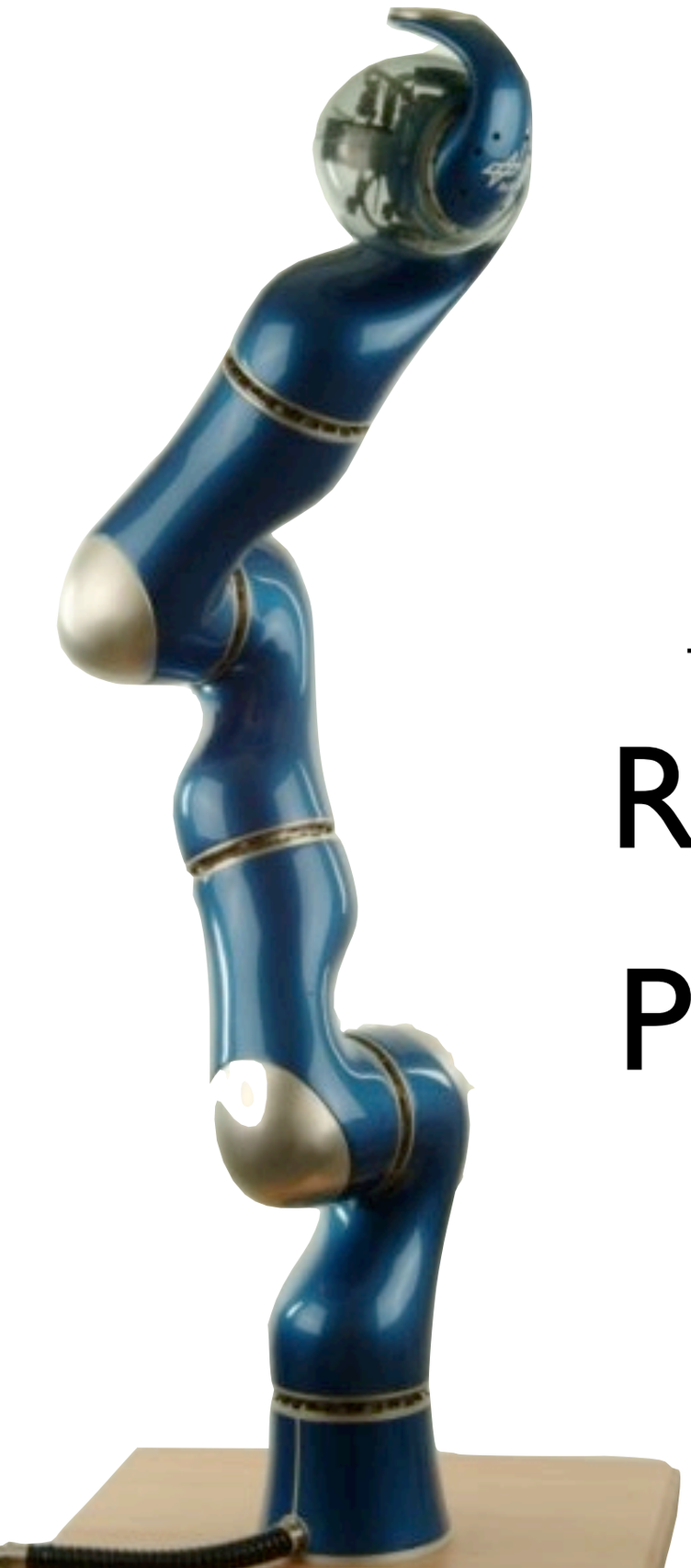
Chapter 1 of Spong, Hutchinson, and Vidyasagar

What is a robot?

a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks

- Robot Institute of America (RIA)

Chapter 1 of Spong, Hutchinson, and Vidyasagar



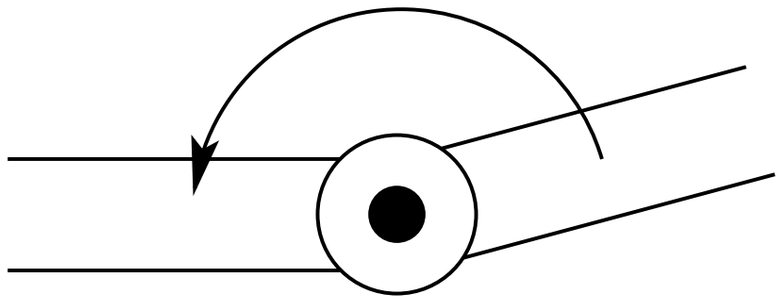
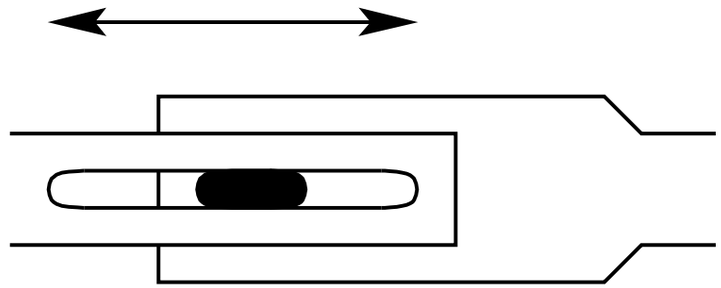
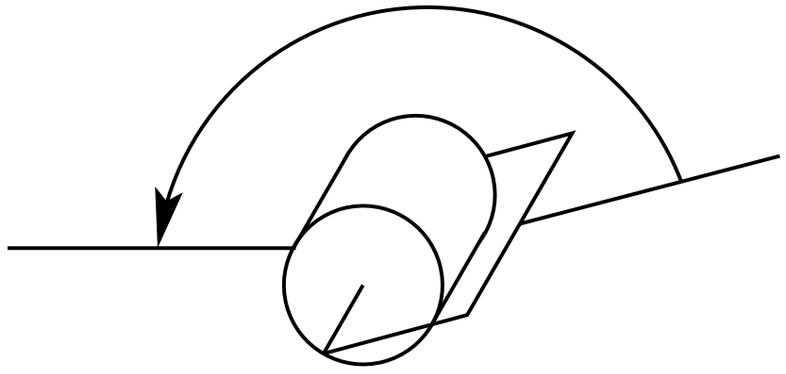
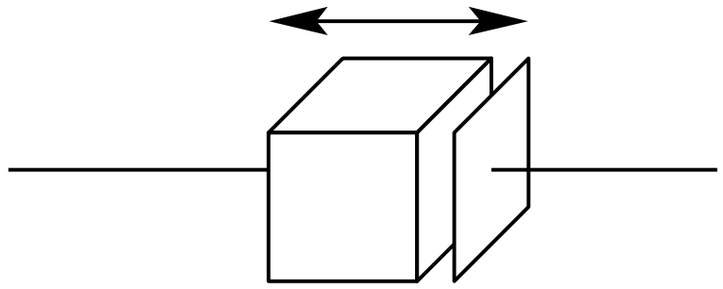
Robot manipulators are composed of:

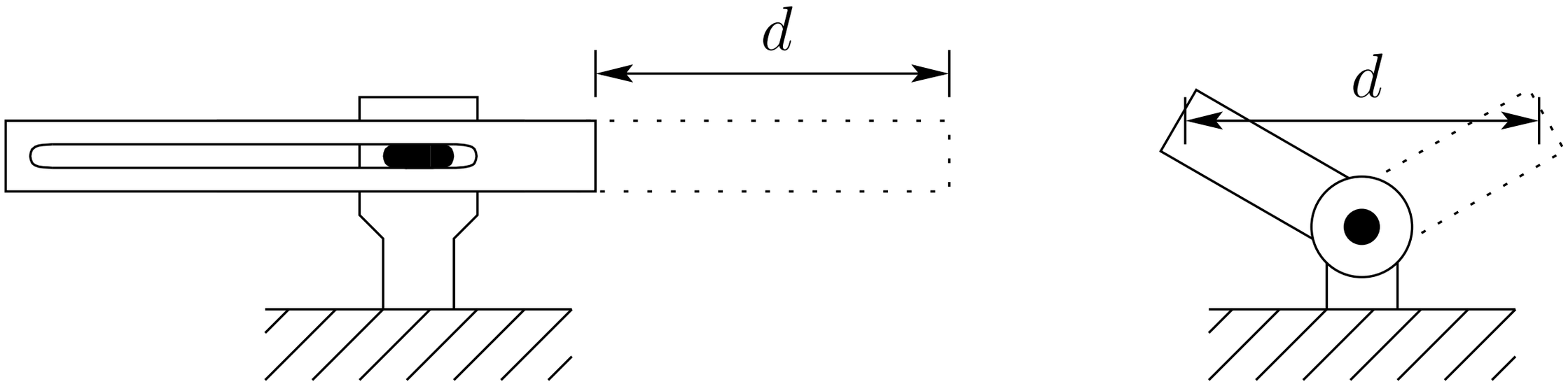
- Rigid **links**
- Connected by **joints**
- To form a **kinematic chain**

There are two types of **joints**:

R
- **Revolute** (rotary), like a hinge, allows relative rotation between two links

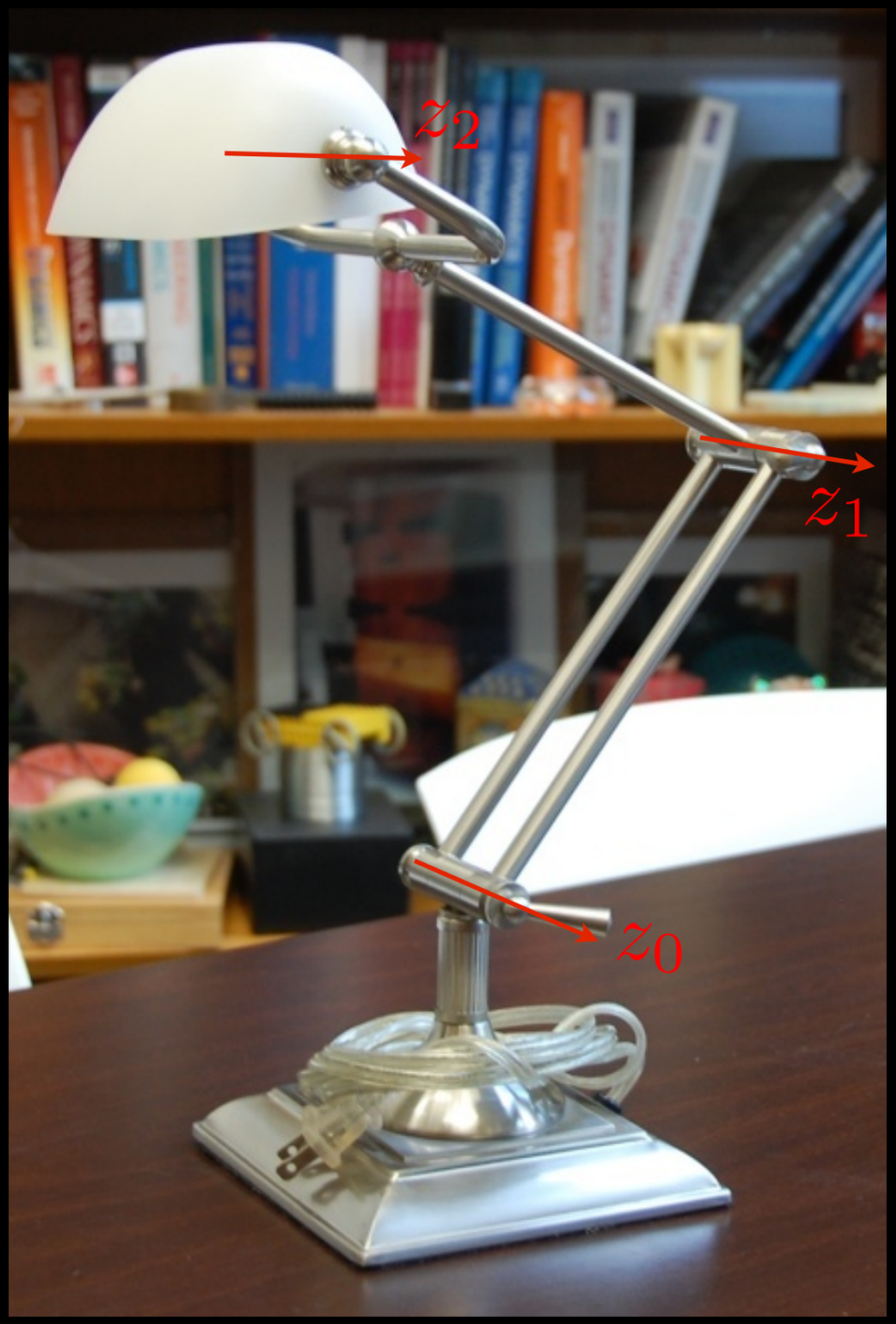
P
- **Prismatic** (linear), like a slider, allows a relative linear motion (translation) between two links

	Revolute	Prismatic
2D		
3D		



Prismatic joints are easier to model and analyze,
but they are bulkier.

In other words, they take up more space to achieve
the same amount of movement.

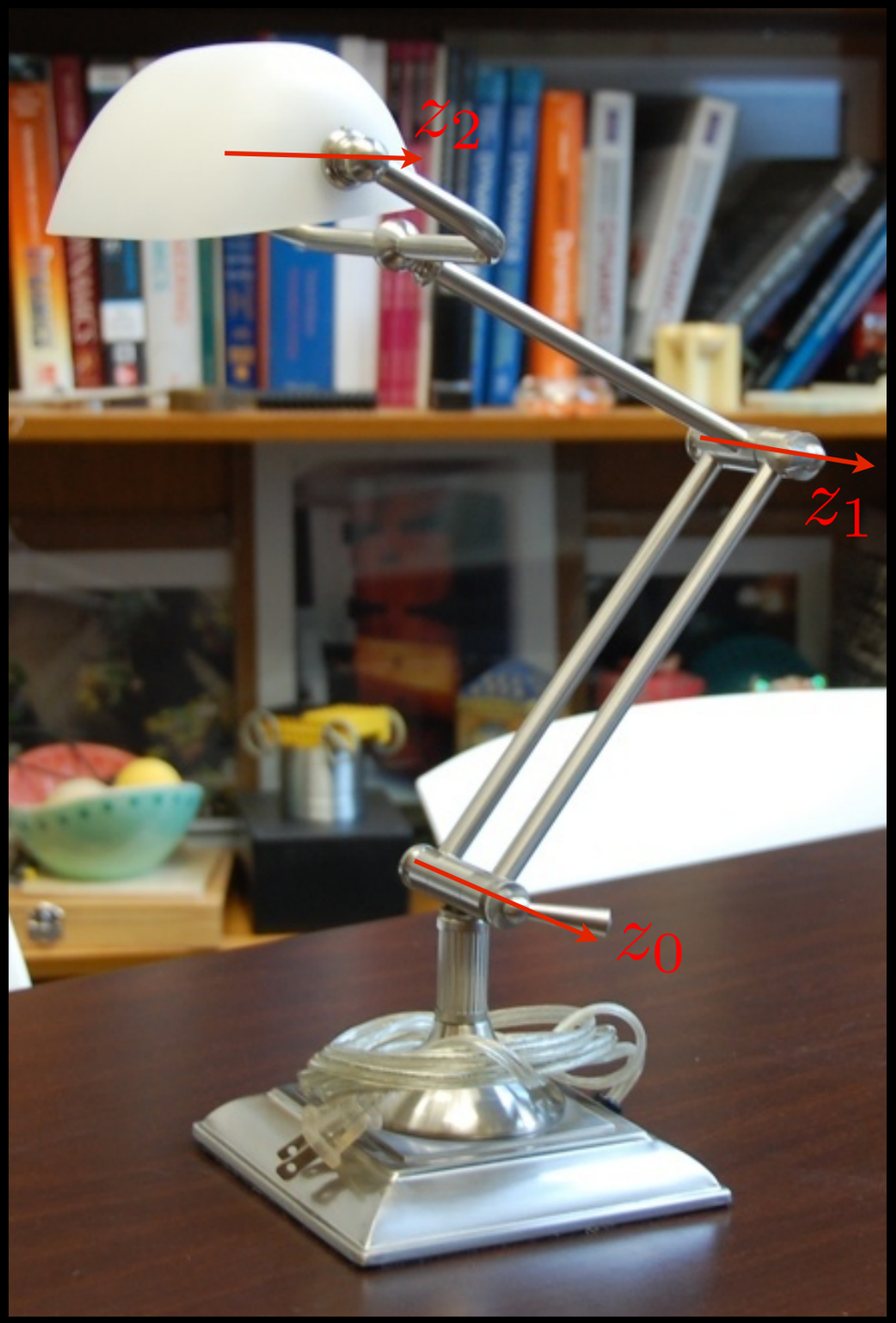


Which **parts** are we describing, and which are we neglecting?

You need to see it **move** in order to know where the joints are.

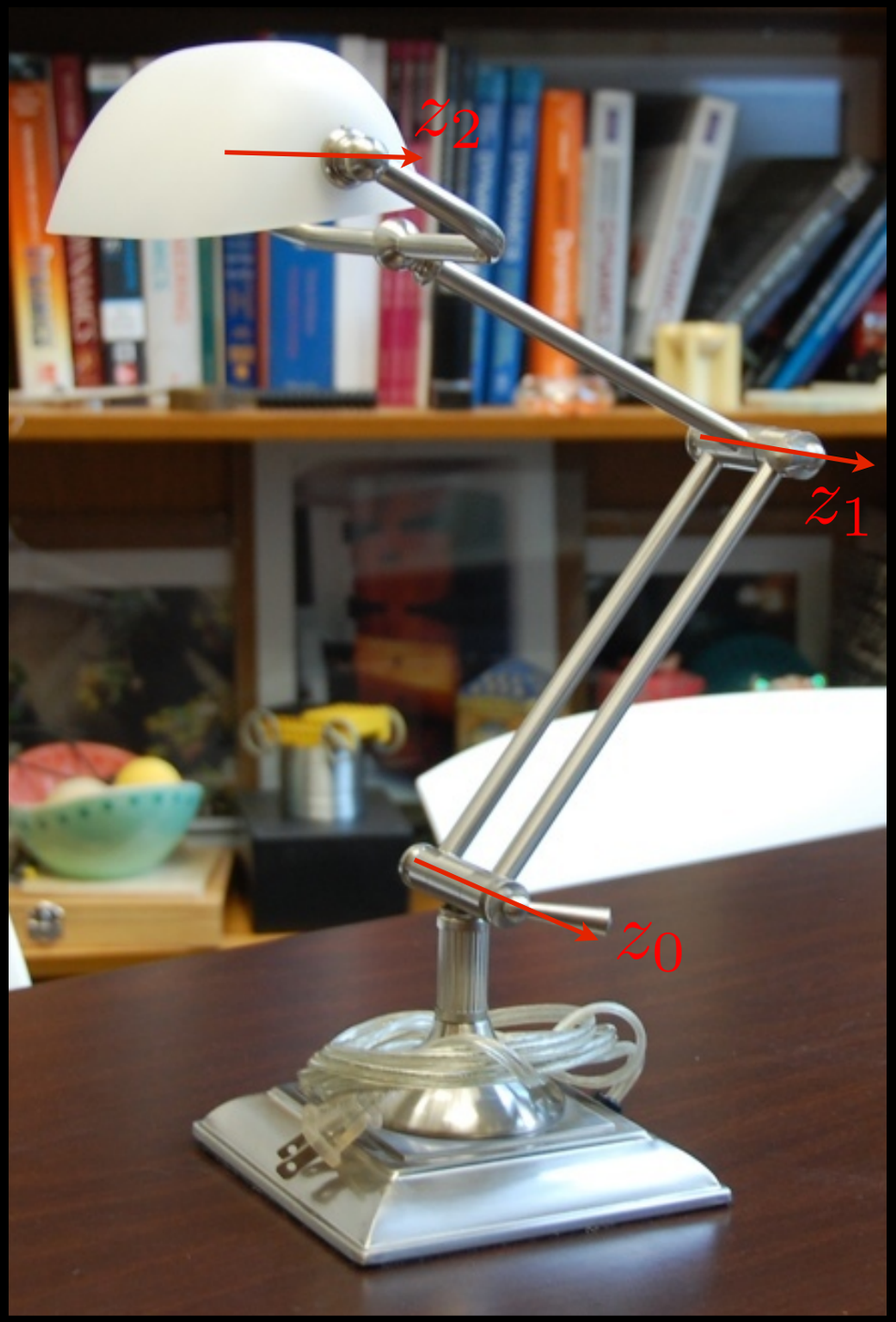
If we keep the **base fixed** (stationary) and treat the white glass shade as the **end-effector**, what type of robot is the lamp?

- **RRR** manipulator
- All rotational axes are **parallel**
- **Planar** mechanism



A manipulator's **configuration** is a complete specification of the location of every point on the manipulator.

- We use a **joint variable** to denote each joint's position.
- Value defines the joint's displacement from a **zero configuration**.
- Use **θ** for revolute joints.
- Use **d** for prismatic joints.
- Axis orientation defines the **positive direction**.
- Knowing all joint variable values defines the configuration.



A manipulator's **configuration space** is the set of all configurations.

How many **configurations** does this lamp have?

- Infinitely many!

How many **numbers** do I need to define the lamp's configuration?

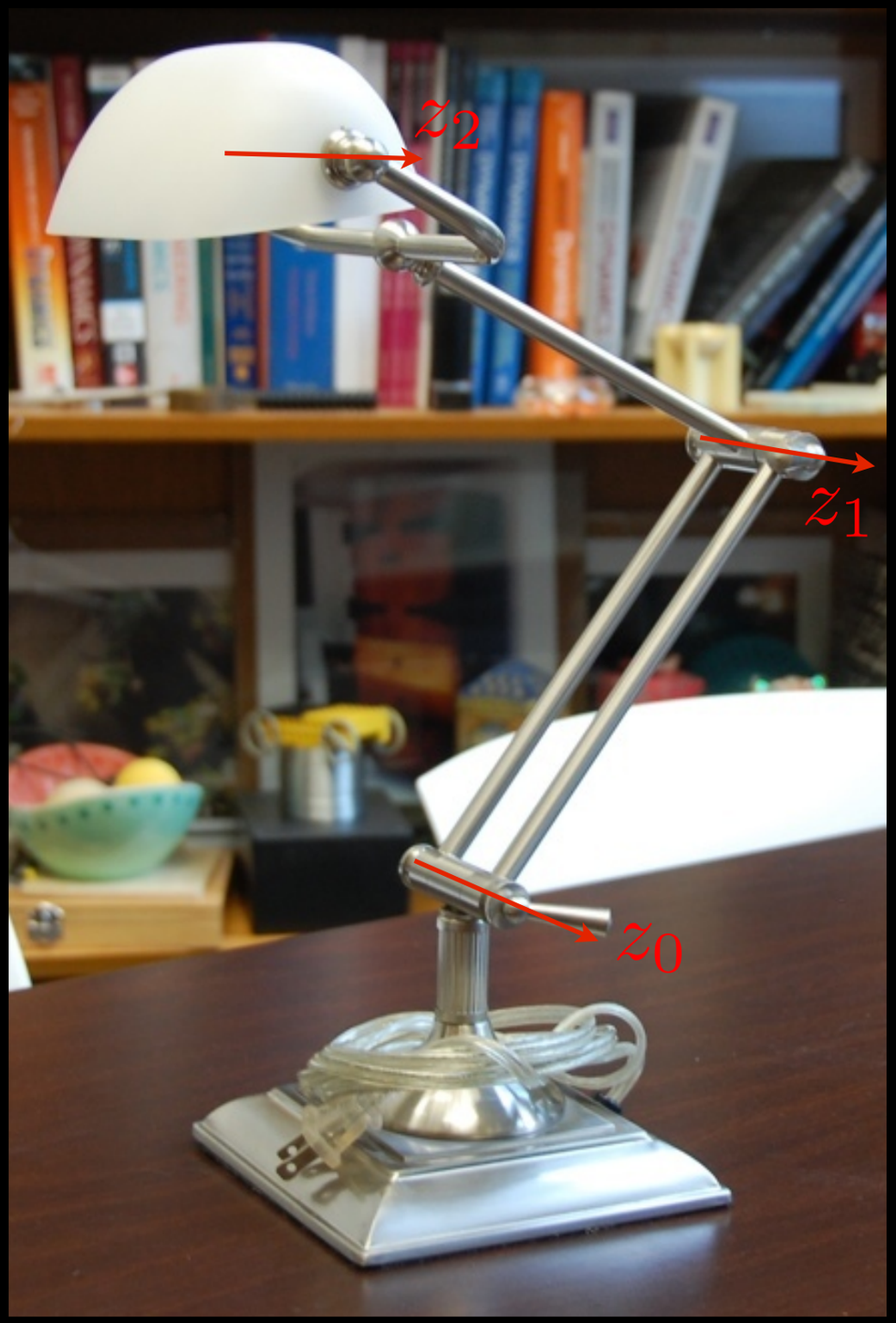
- Three: θ_1, θ_2, θ_3

This value is the **degrees of freedom (DOF)** of the robot.

How many **DOF** does the lamp have if I lock the joints and move it around in three dimensions?

- Six
- Three for positioning
- Three for orientation

Robots need at least 6 joints **(6 DOF)** for the end-effector to reach every point in its workspace with arbitrary orientation.

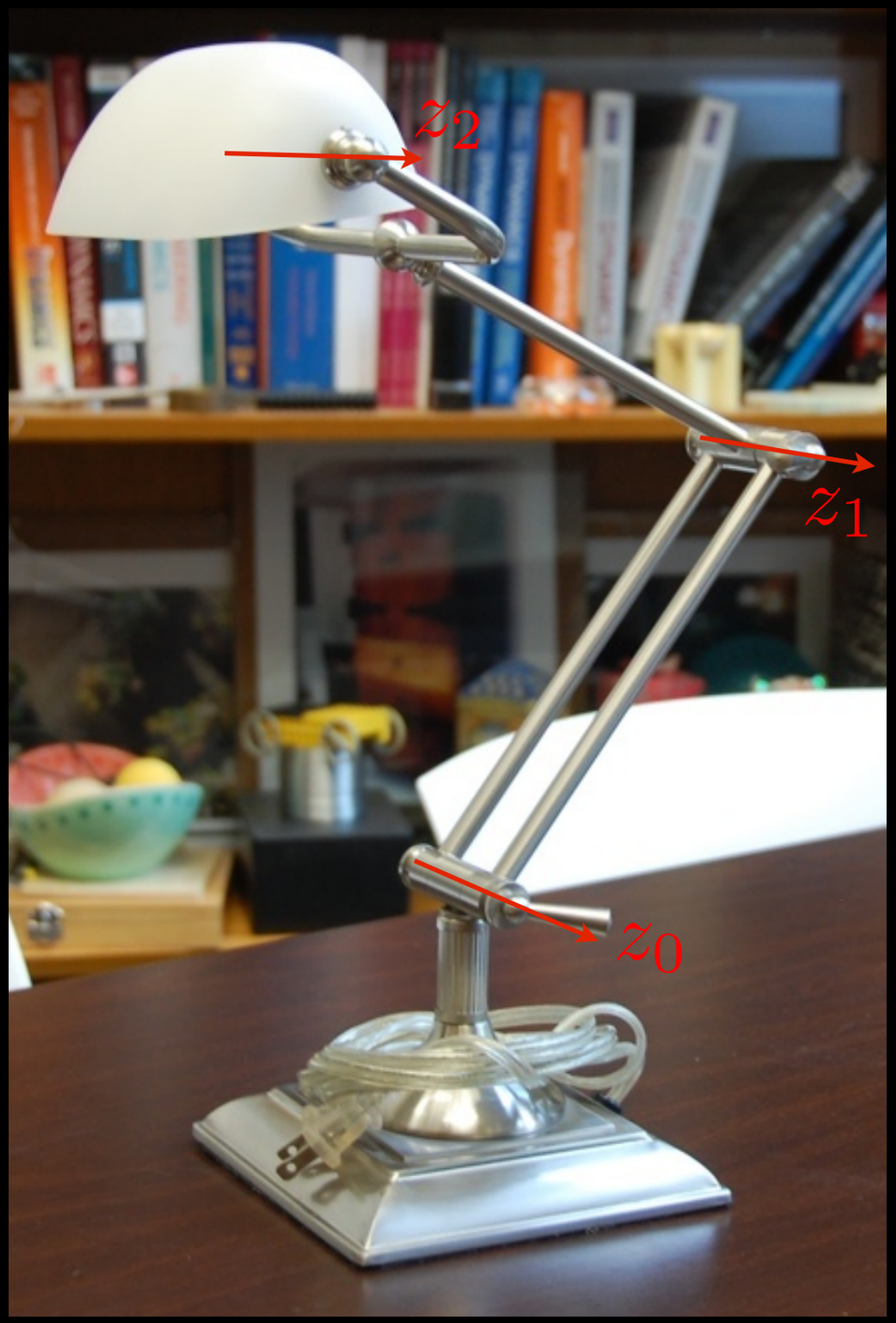


The **workspace** of a manipulator is the total volume swept out by the end-effector as the manipulator executes all possible motions.

- Depends on robot geometry as well as mech. limits on the joints.
- Choose a particular point to represent end-effector location.

The **reachable workspace** is the entire set of points that the end-effector can reach.

The **dexterous workspace** is the set of points that the manipulator can reach with arbitrary orientation.



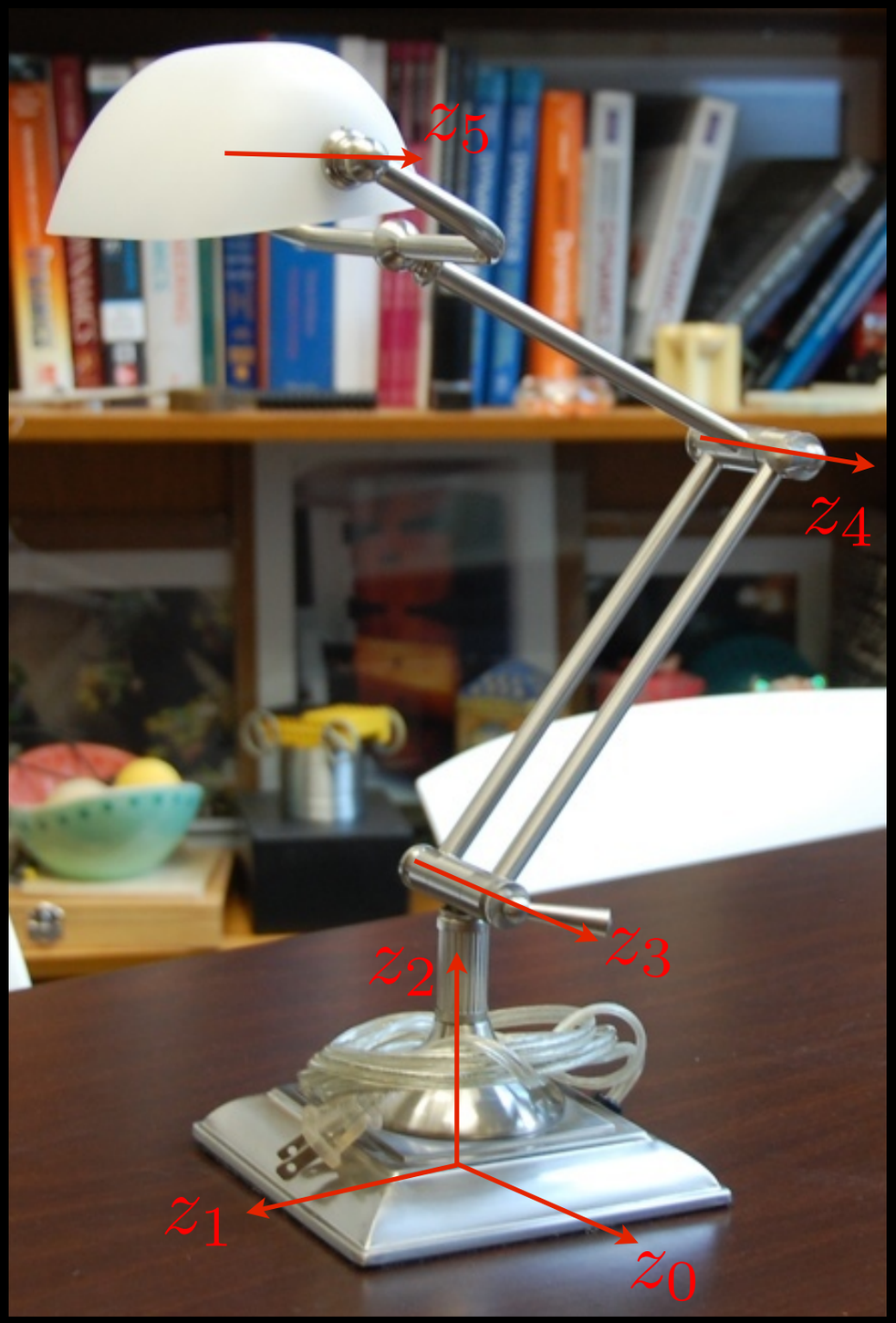
Define a point on the shade along the final axis (z_2) as the end-effector location.

What is the **reachable workspace** of the lamp?

- Everywhere the point can go when joints are rotated.
- This is a two-dimensional shape.

What is the **dexterous workspace** of the lamp?

- If we consider motion only in 2D, dexterous is the same as reachable workspace.
- Considering 3D, there is none!



What if we let the **base move and rotate** on the table. What type of robot is the lamp now?

- **PPRRRR** manipulator
- Not all rotational axes are **parallel**
- **Spatial** mechanism

What is the **reachable workspace** of the lamp?

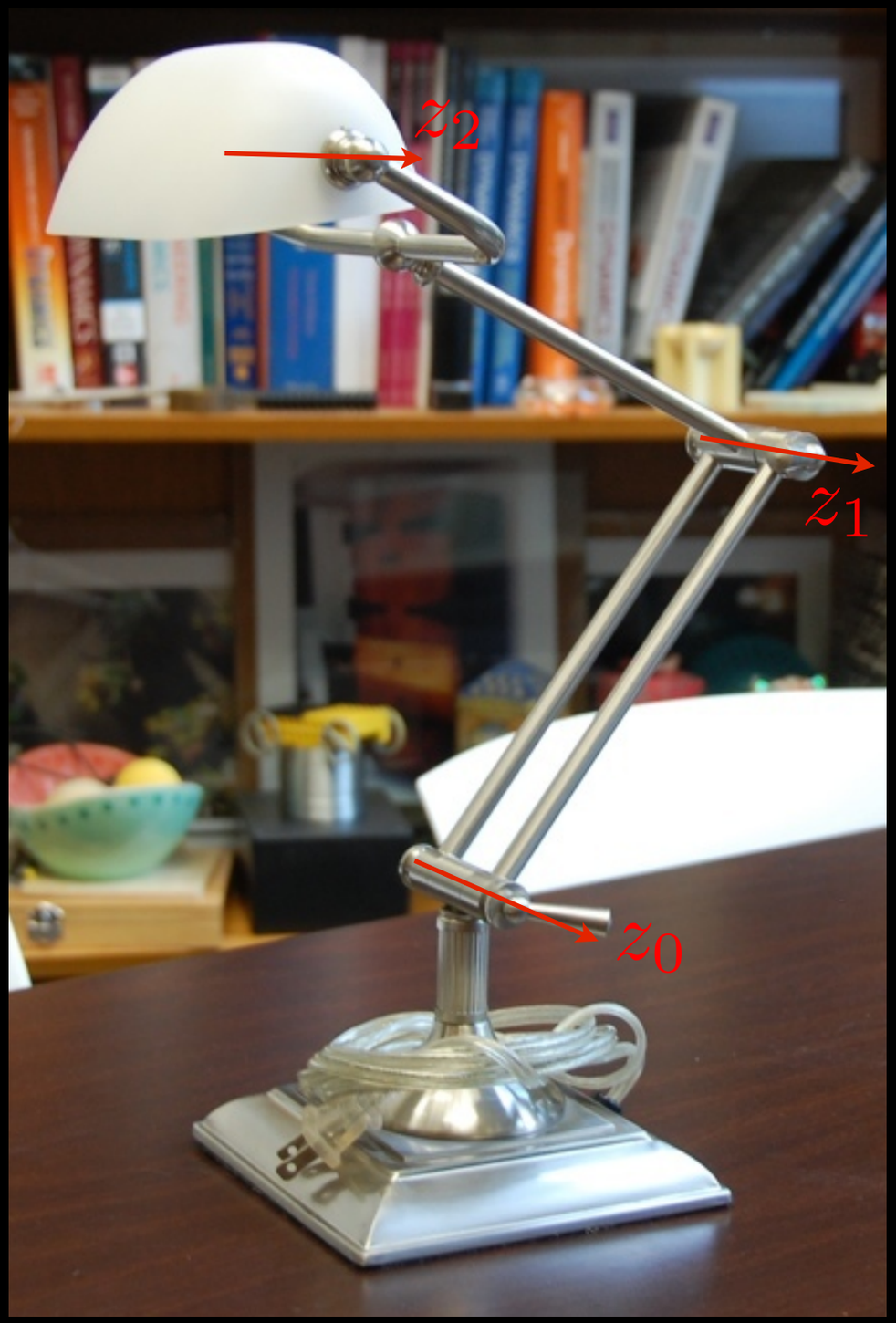
- Everywhere the shade can go when joints rotate, base slides.

What is the **dexterous workspace** of the lamp?

- There is none! The shade can't tilt left/right; z_5 always horizontal.

Can we apply the same terminology of revolute and prismatic joints to the **Roomba**?

- It's sort-of a **PPR** manipulator.
- With chassis lift, **PPRP**.
- But it can't move in any direction at any time.
- This is a **non-holonomic constraint**, and it makes mobile robots with wheels more challenging to model and control.



Does the **configuration** of a manipulator fully define how it will move in the future?

- No. It only gives you an **instantaneous description** of the geometry.
- The manipulator's **state** is a set of variables that is sufficient to tell you its future time response when combined with dynamics and future inputs.
- State requires both the joint variables and their **derivatives**.

Articulated Manipulator (RRR)

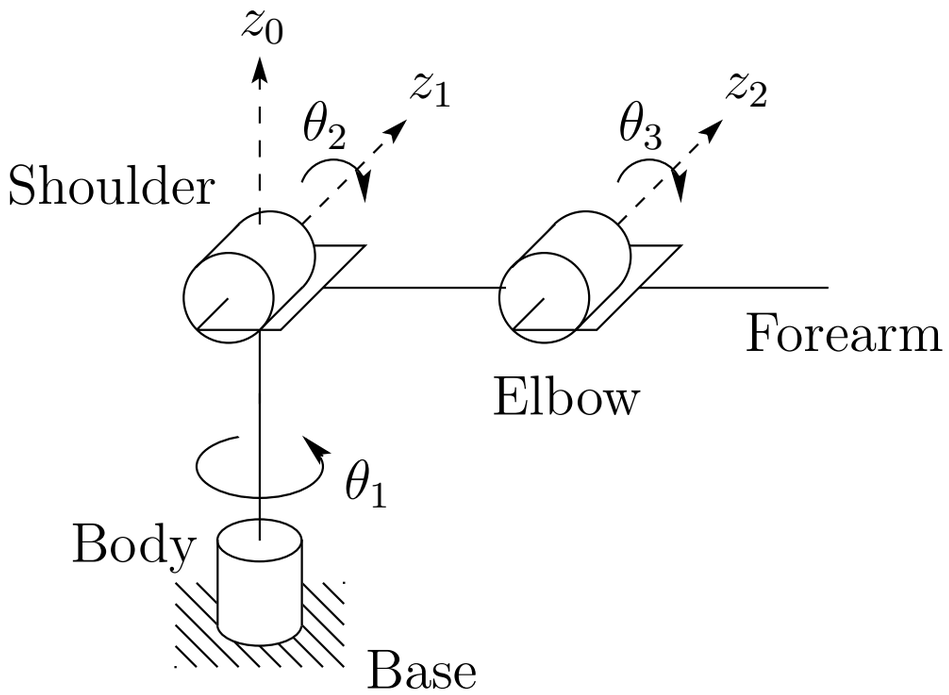


Fig. 1.8 Structure of the elbow manipulator.

Fig. 1.6 The ABB IRB1400 Robot. Photo courtesy of ABB.

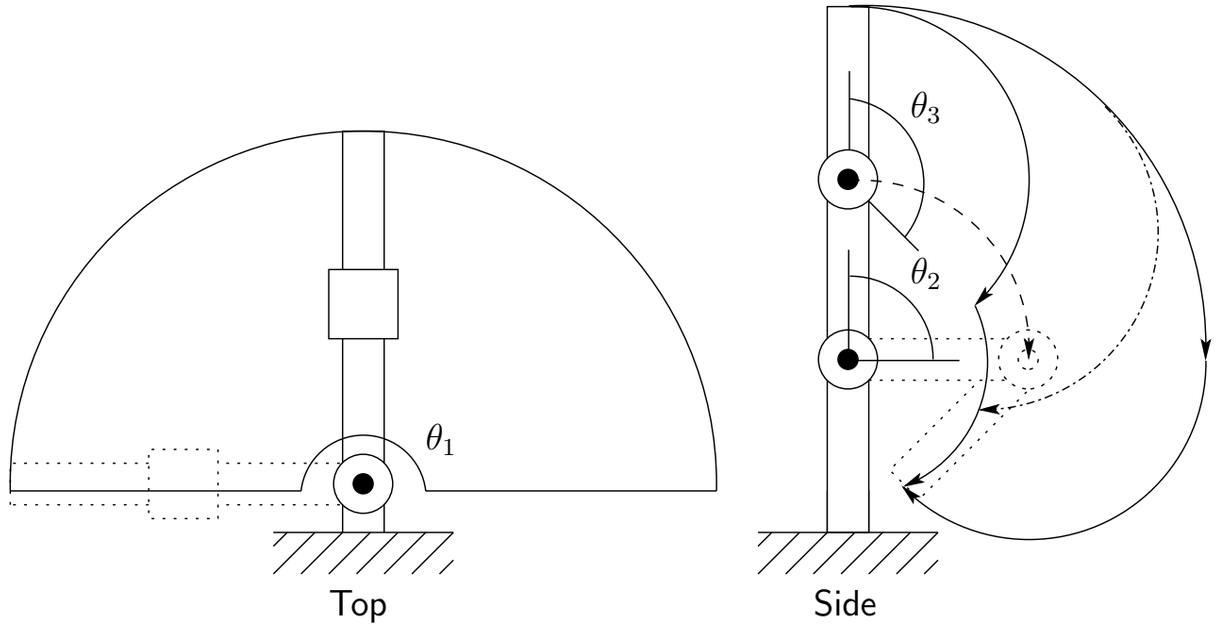


Fig. 1.9 Workspace of the elbow manipulator.

a.k.a. Revolute, Elbow, or Anthropomorphic

Spherical Manipulator (RRP)

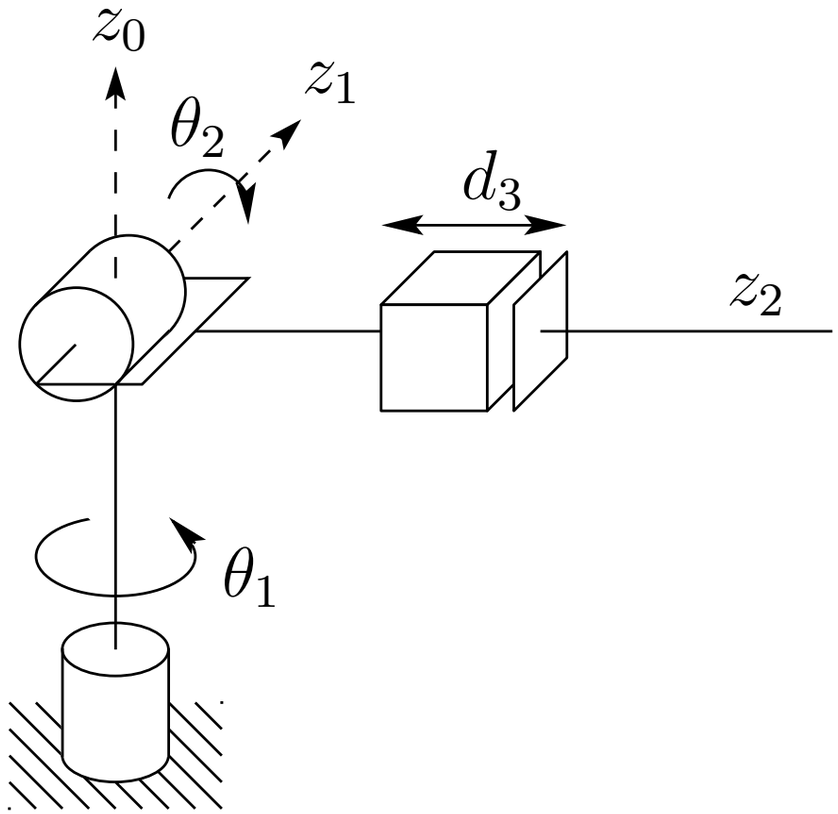


Fig. 1.10 The spherical manipulator.

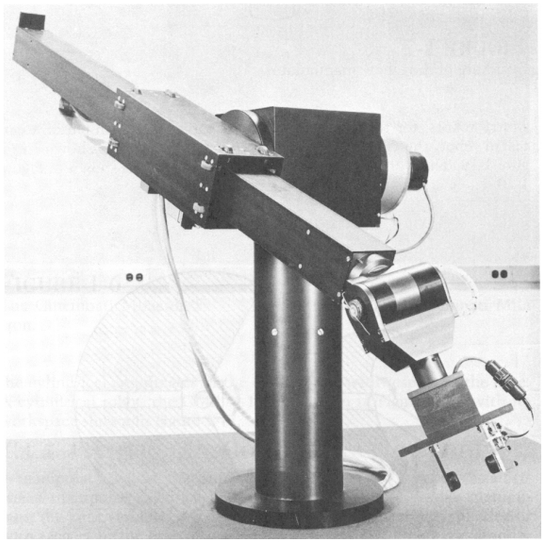


Fig. 1.11 The Stanford Arm. Photo courtesy of the Coordinated Science Lab, University of Illinois at Urbana-Champaign.

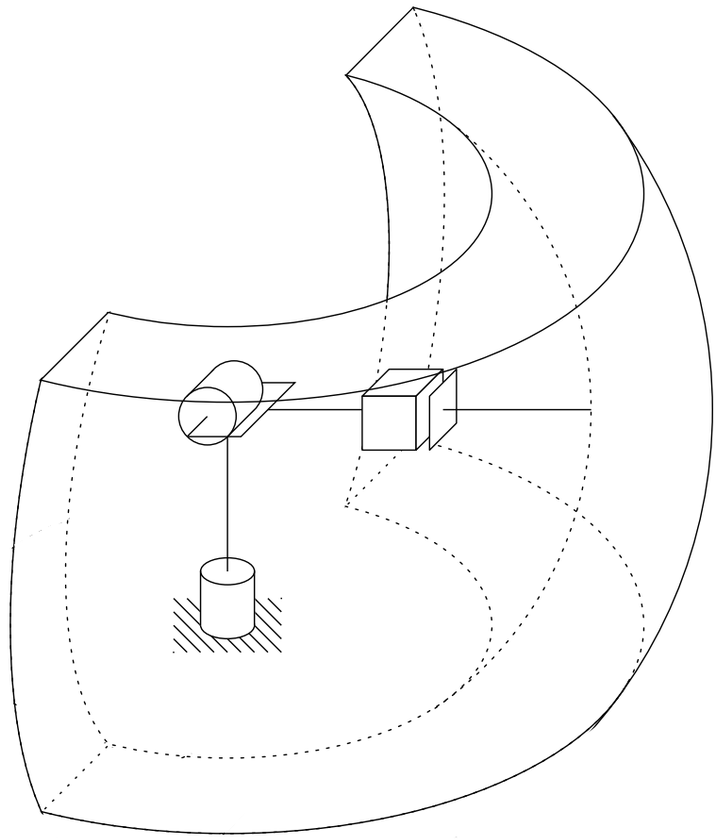


Fig. 1.12 Workspace of the spherical manipulator.

SCARA Manipulator (RRP)

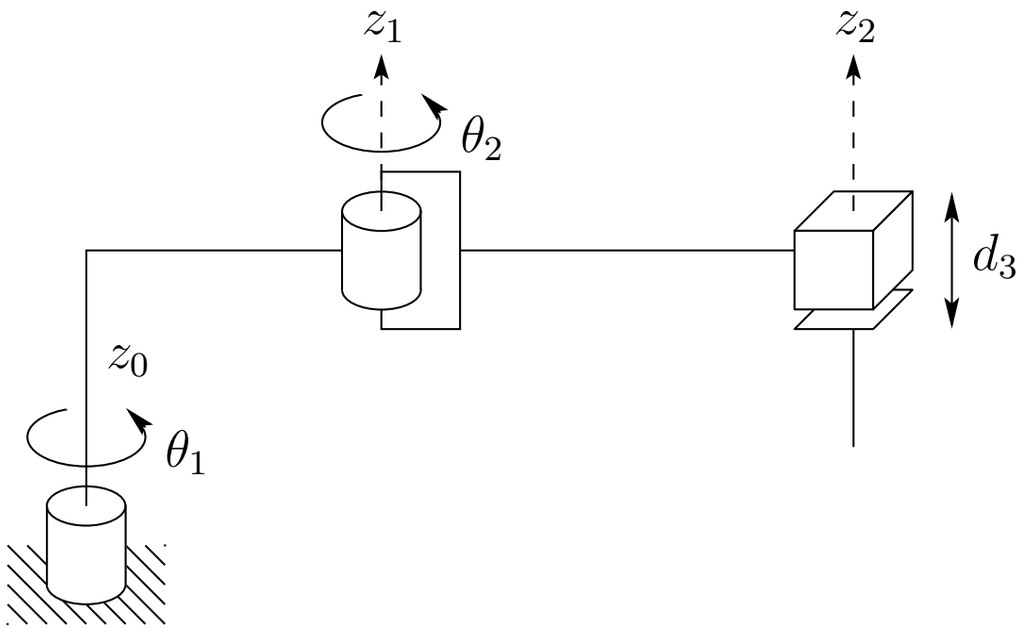


Fig. 1.13 The SCARA (Selective Compliant Articulated Robot for Assembly).

Fig. 1.14 The Epson E2L653S SCARA Robot. Photo Courtesy of Epson.

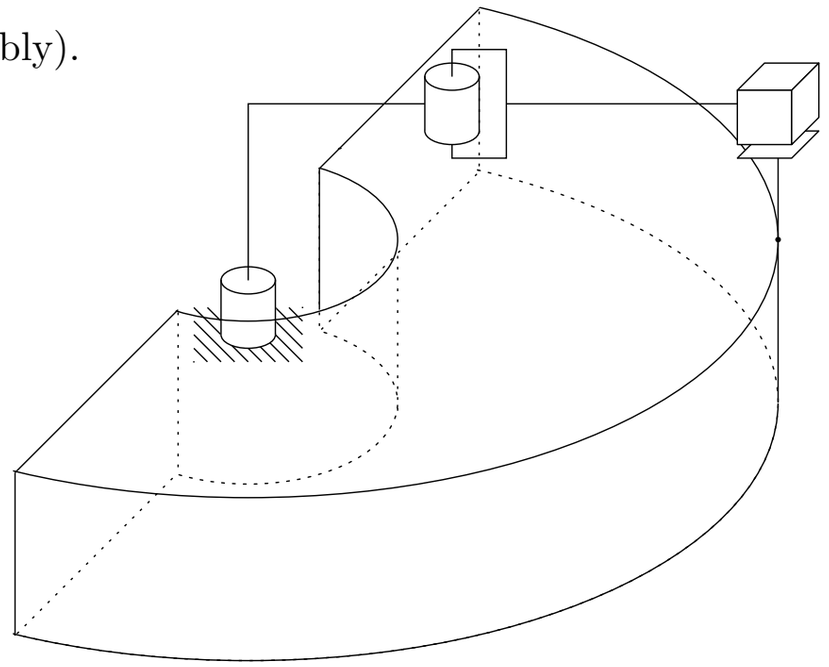


Fig. 1.15 Workspace of the SCARA manipulator.

Cylindrical Manipulator (RPP)

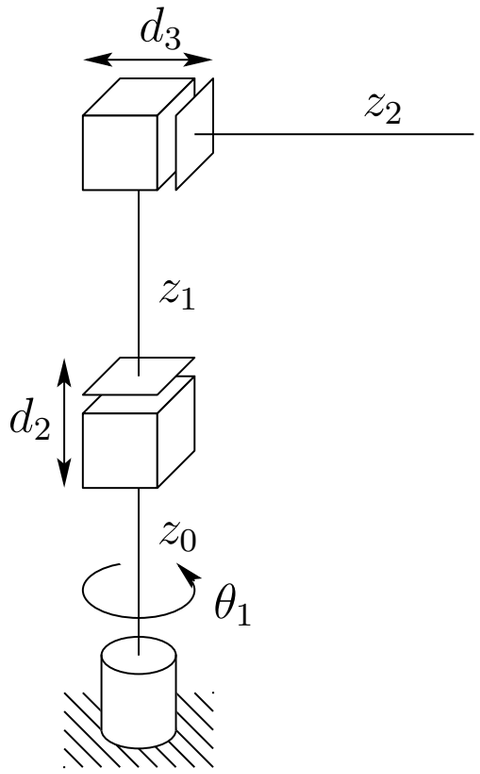


Fig. 1.16 The cylindrical manipulator.

Fig. 1.17 The Seiko RT3300 Robot. Photo courtesy of Seiko.

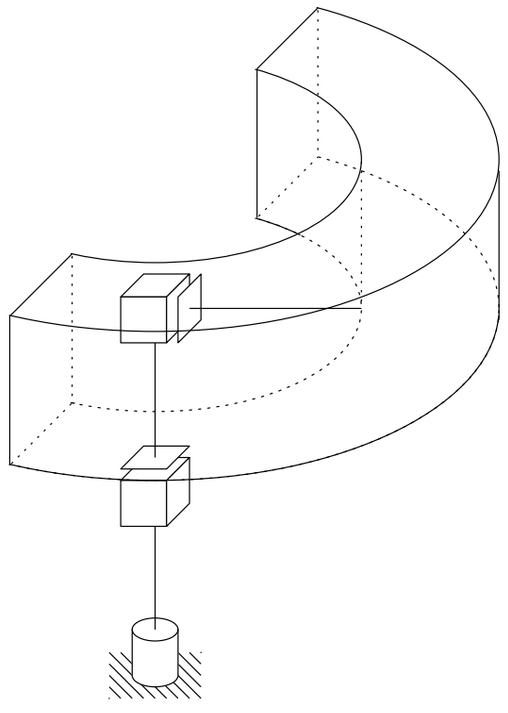


Fig. 1.18 Workspace of the cylindrical manipulator.

Cartesian Manipulator (PPP)

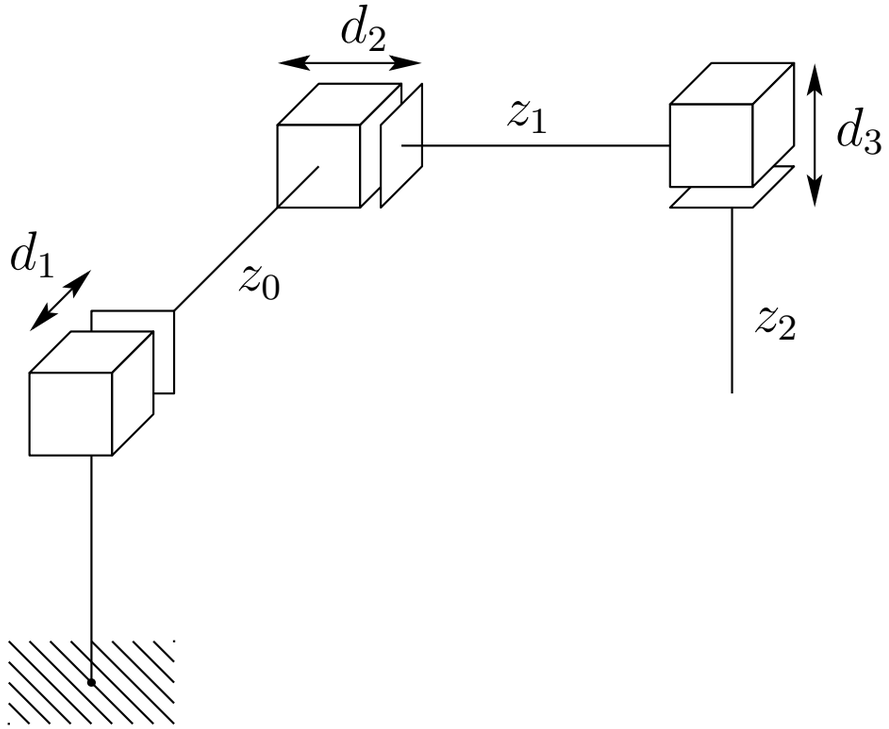


Fig. 1.19 The Cartesian manipulator.

Fig. 1.20 The Epson Cartesian Robot. Photo courtesy of Epson.

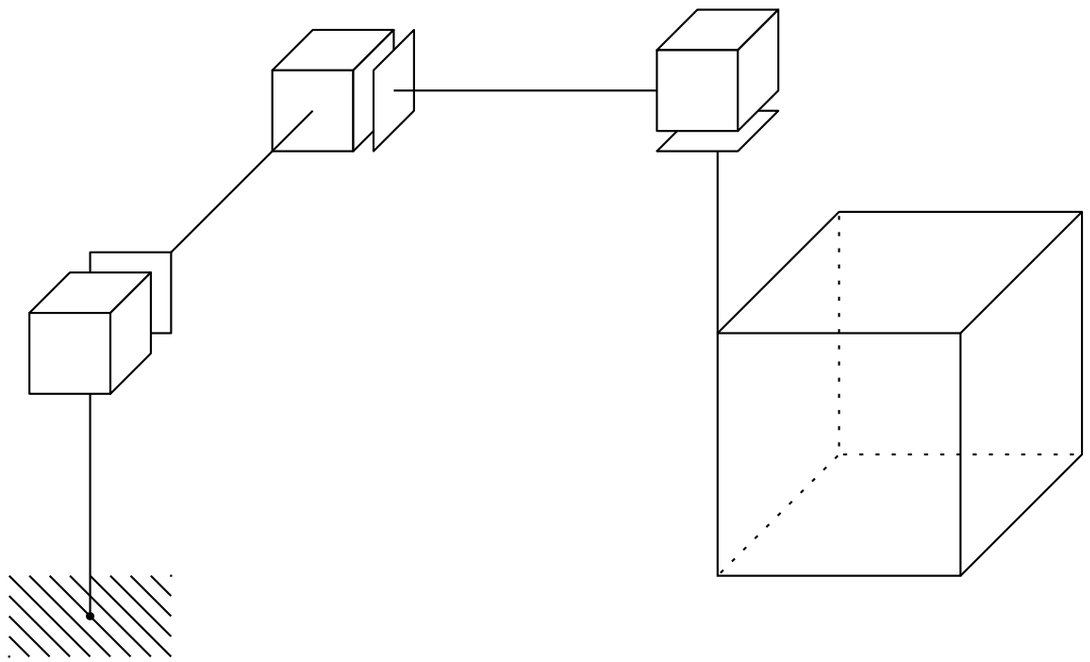


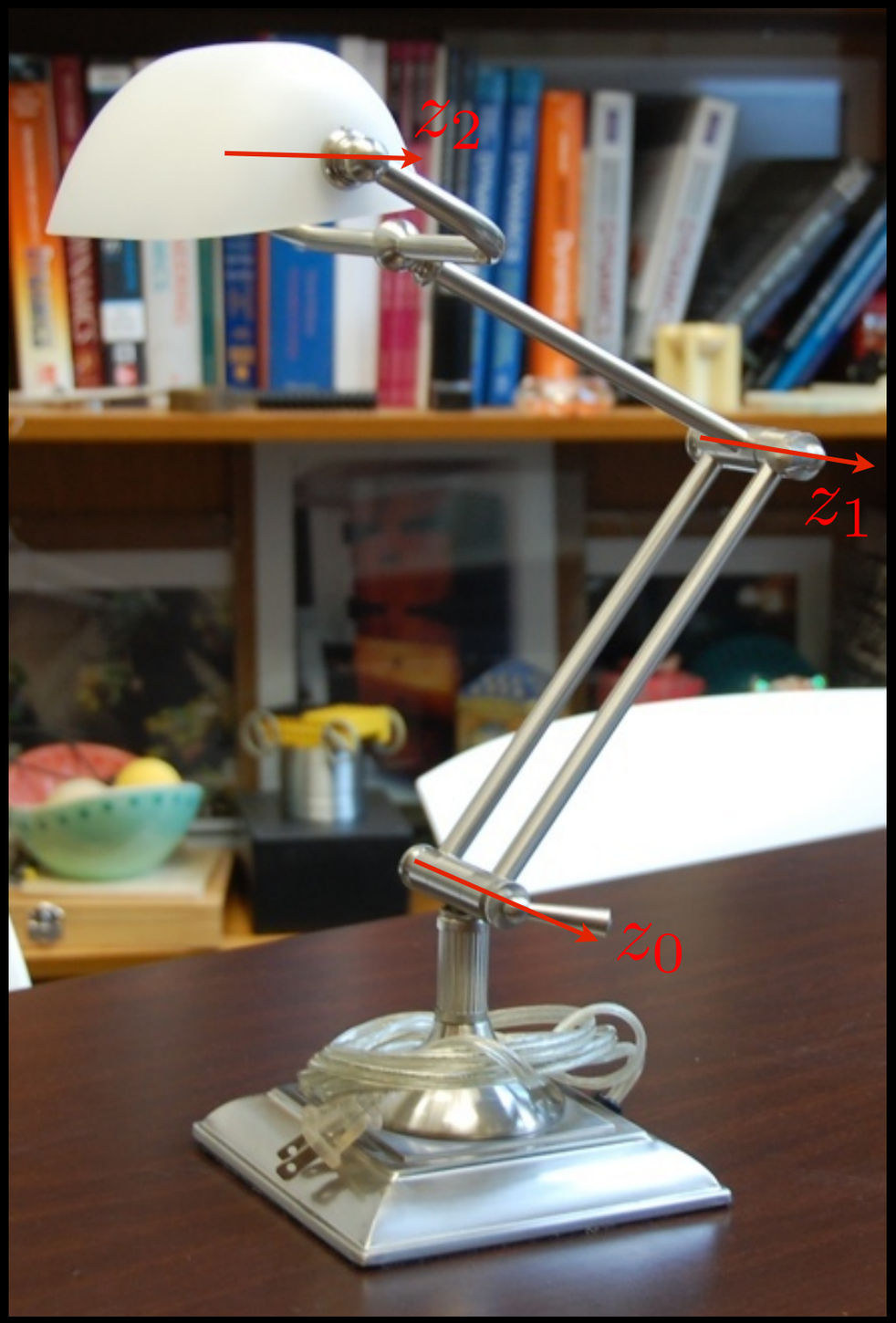
Fig. 1.21 Workspace of the Cartesian manipulator.

MEAM 520
Rotation Matrices

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 3: September 5, 2013



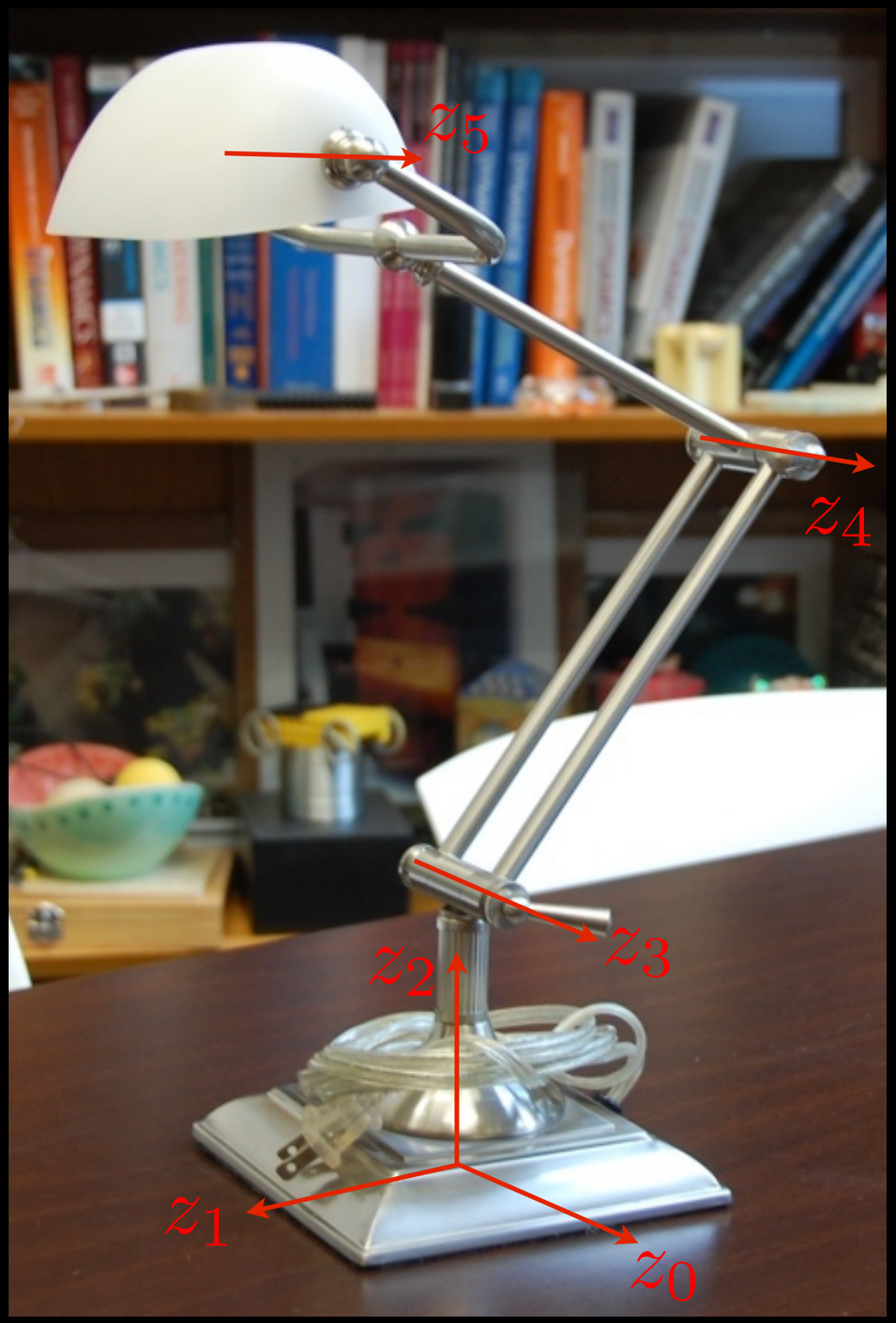
Define a point on the shade along the final axis (z_2) as the end-effector location.

What is the **reachable workspace** of the lamp?

- Everywhere the point can go when joints are rotated.
- This is a two-dimensional shape.

What is the **dexterous workspace** of the lamp?

- If we consider motion only in 2D, dexterous is the same as reachable workspace.
- Considering 3D, there is none!



What if we let the **base move and rotate** on the table. What type of robot is the lamp now?

- **PPRRRR** manipulator
- Not all rotational axes are **parallel**
- **Spatial** mechanism

What is the **reachable workspace** of the lamp?

- Everywhere the shade can go when joints rotate, base slides.

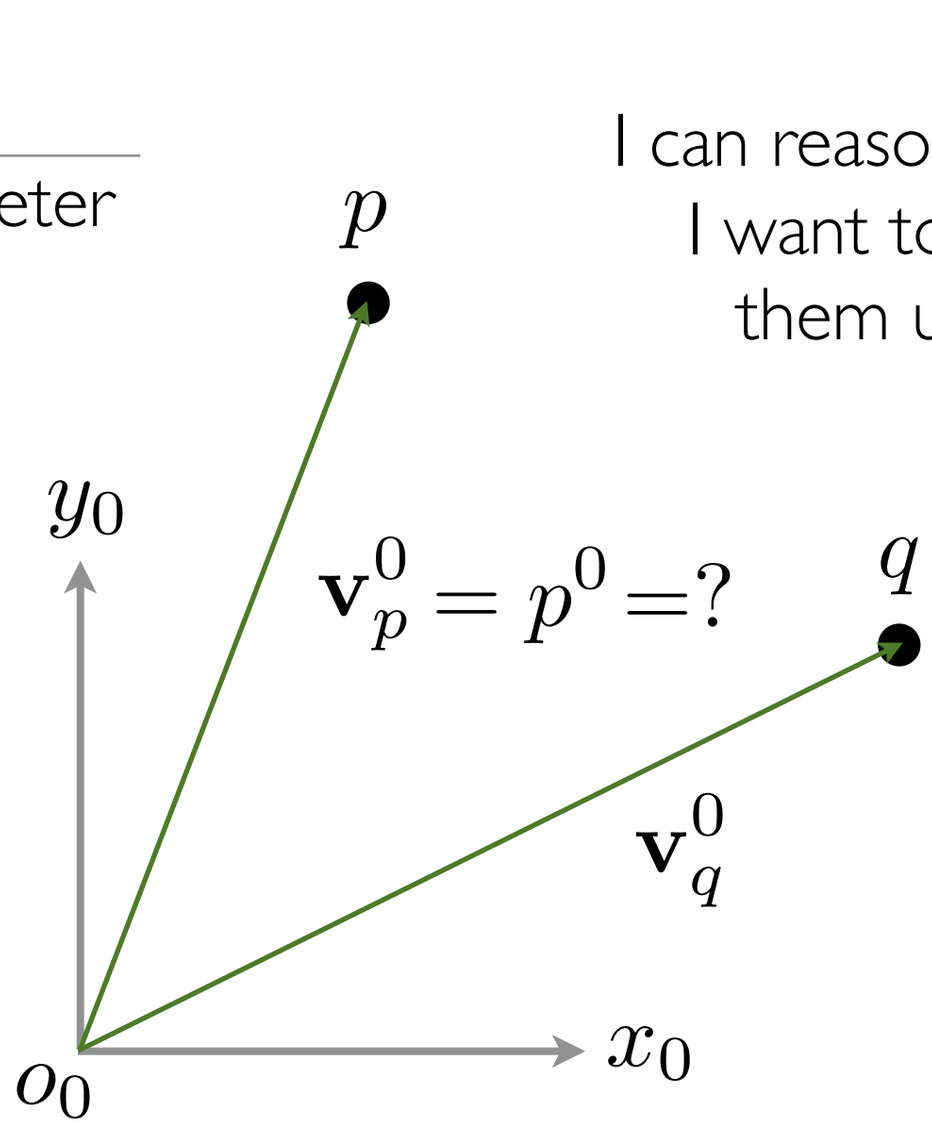
What is the **dexterous workspace** of the lamp?

- There is none! The shade can't tilt left/right; z_5 always horizontal.

Representing positions

A **point** exists in space as a geometric entity

I can reason directly about these points, but if I want to **analyze** them, I must represent them using coordinates or equations.



The coordinate frame being used is designated using **superscript** notation

"It's *super* to specify the frame!"

What if we were doing this in **3D**.
How would the coordinate frame change?
How would the coordinates change?

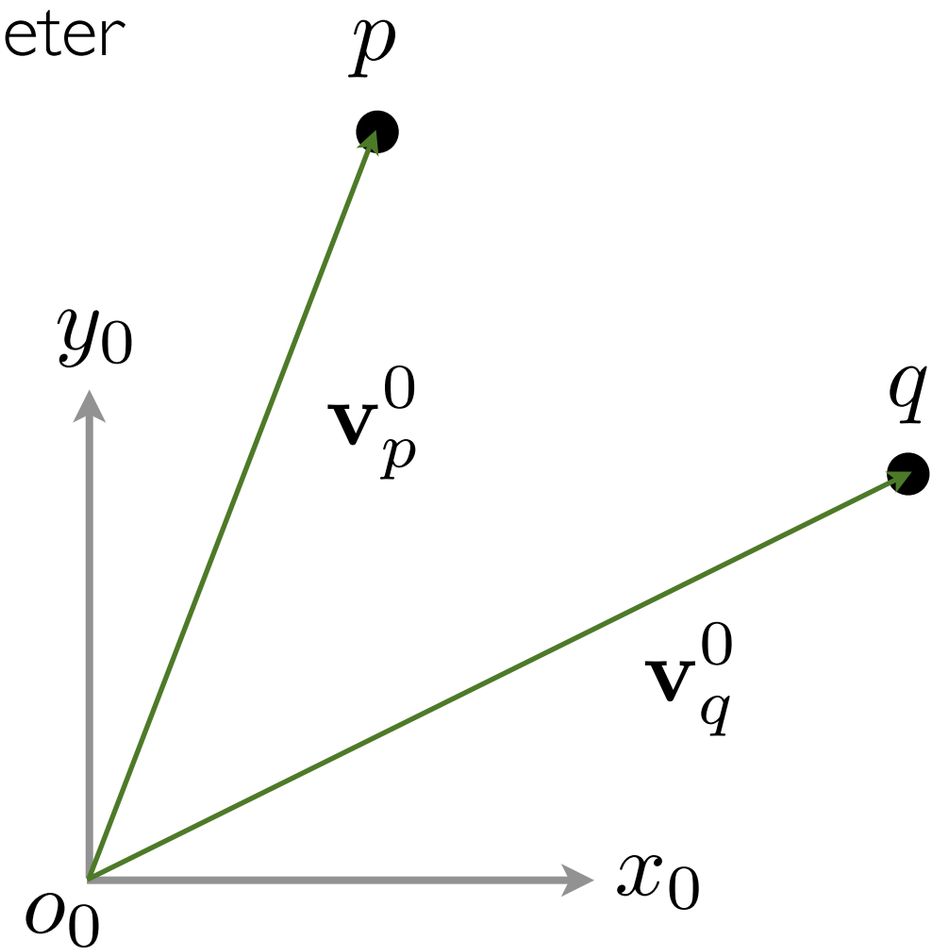
Coordinate Frame

Needs an origin (a single point in space)
and two or three orthogonal coordinate axes

Vectors

How do I find the **magnitude** or **length** or **norm** of a vector?

$$\mathbf{v}_p^0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{v}_p^0 = [x, y, z]^T$$



$$||\mathbf{v}_p^0|| = \sqrt{x^2 + y^2 + z^2}$$

$$||\mathbf{v}_p^0|| = (x^2 + y^2 + z^2)^{\frac{1}{2}}$$

$$||\mathbf{v}_p^0|| = \left((\mathbf{v}_p^0)^T \mathbf{v}_p^0\right)^{\frac{1}{2}}$$

$$||\mathbf{v}_p^0|| = \langle \mathbf{v}_p^0, \mathbf{v}_p^0 \rangle^{\frac{1}{2}}$$

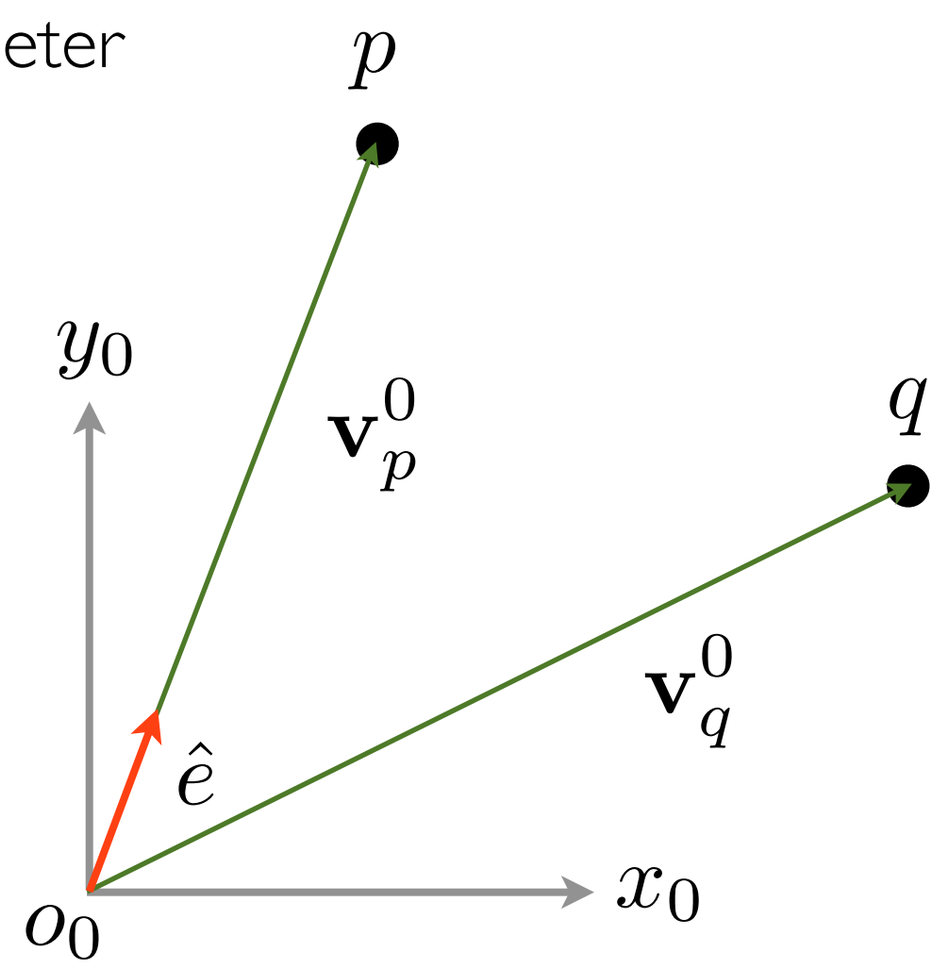
In MATLAB?

Vectors

How do I represent the **direction** of a vector?

$$\mathbf{v}_p^0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{v}_p^0 = [x, y, z]^T$$

1 meter



Create a **unit vector**.

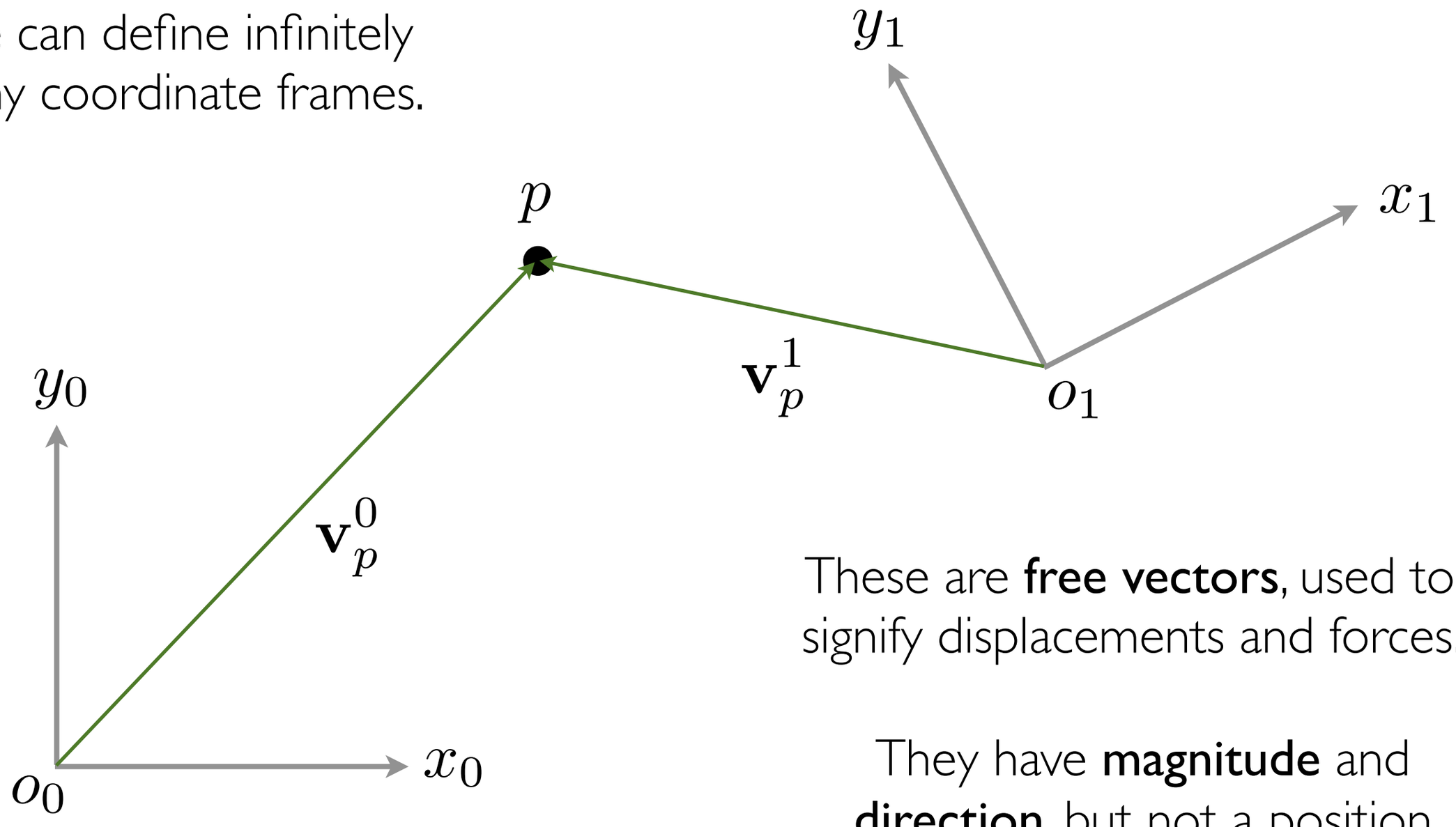
$$\hat{e} = \frac{\mathbf{v}_p^0}{||\mathbf{v}_p^0||}$$

In MATLAB?

Multiple coordinate frames

The **superscript** should make you turn your head to consider world from the orientation of the coordinate frame.

We can define infinitely many coordinate frames.

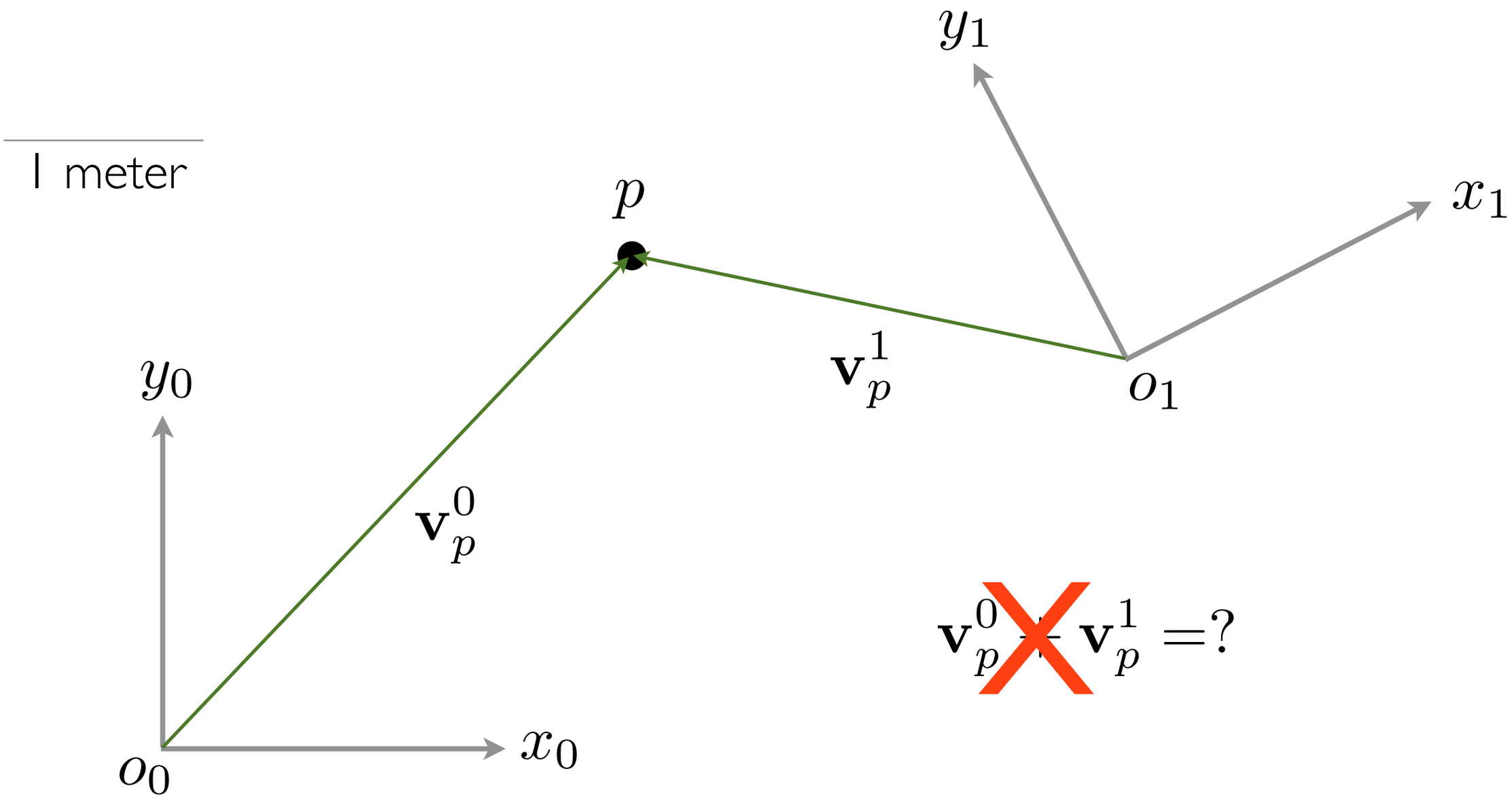


These are **free vectors**, used to signify displacements and forces.

They have **magnitude** and **direction**, but not a position from which they start.

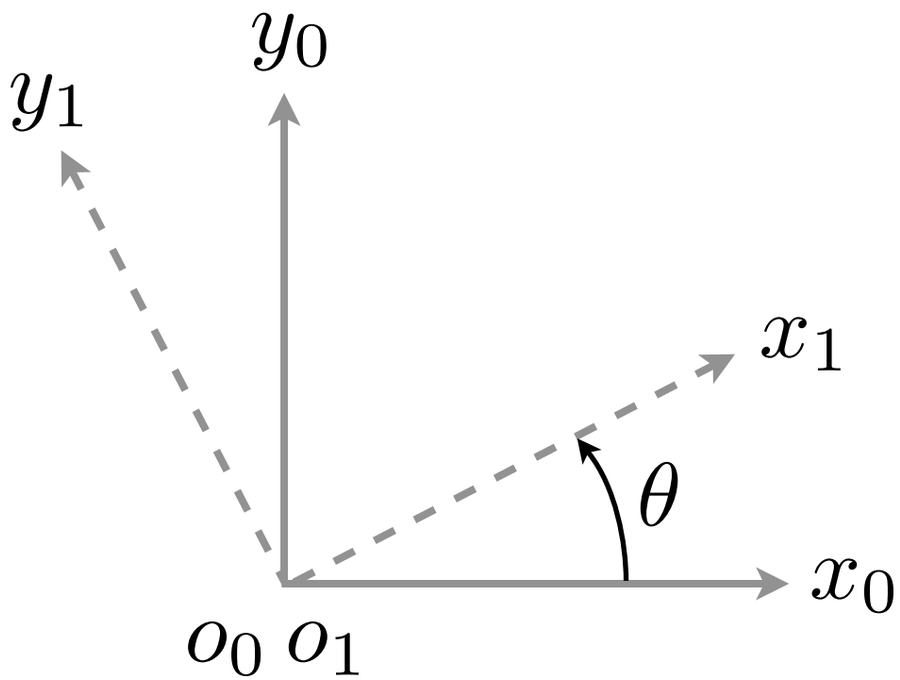
We draw them at different locations for convenience.

Multiple coordinate frames



To perform algebraic manipulation, you must express vectors in the **same frame** or in **parallel frames**

Planar Coordinate Rotations



project frame 1 into frame 0

$$\mathbf{x}_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

$$\mathbf{y}_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

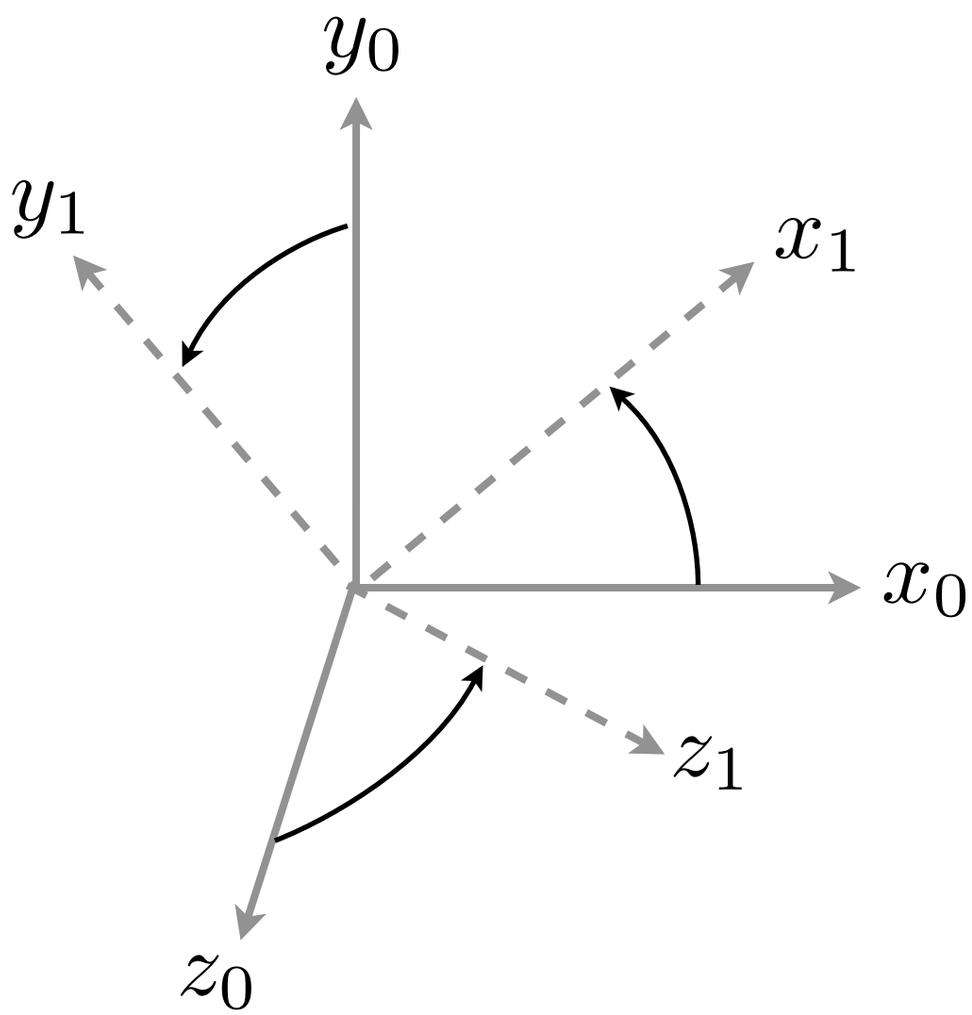
which can be expressed as a **rotation matrix**

$$\mathbf{R}_1^0 = \begin{bmatrix} \mathbf{x}_1^0 & \mathbf{y}_1^0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

the inverse of which is the matrix transpose

$$\mathbf{R}_0^1 = (\mathbf{R}_1^0)^\top = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Three-Dimensional Coordinate Rotations



The **basic rotation matrices** define rotations about the three coordinate axes

SO(3)
Special Orthogonal group of order 3

Represent orientation of one frame with respect to another frame

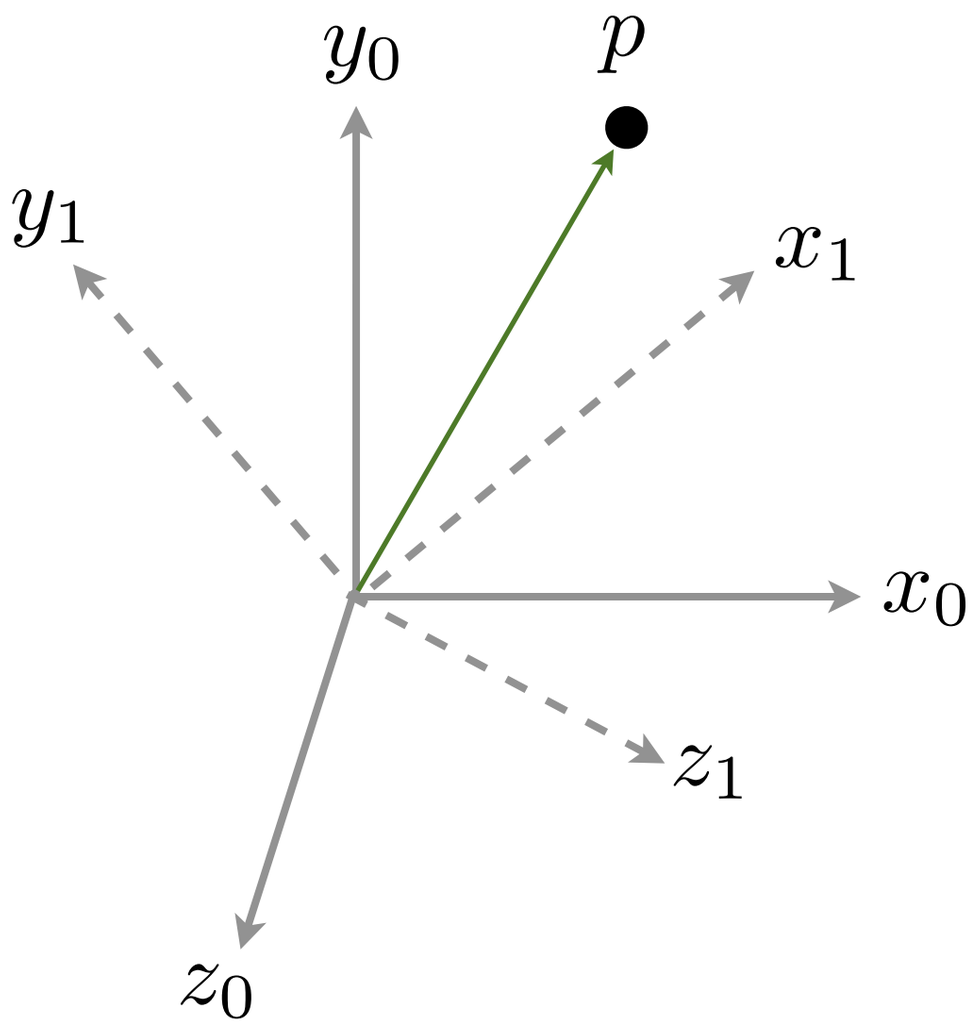
$$\mathbf{R}_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

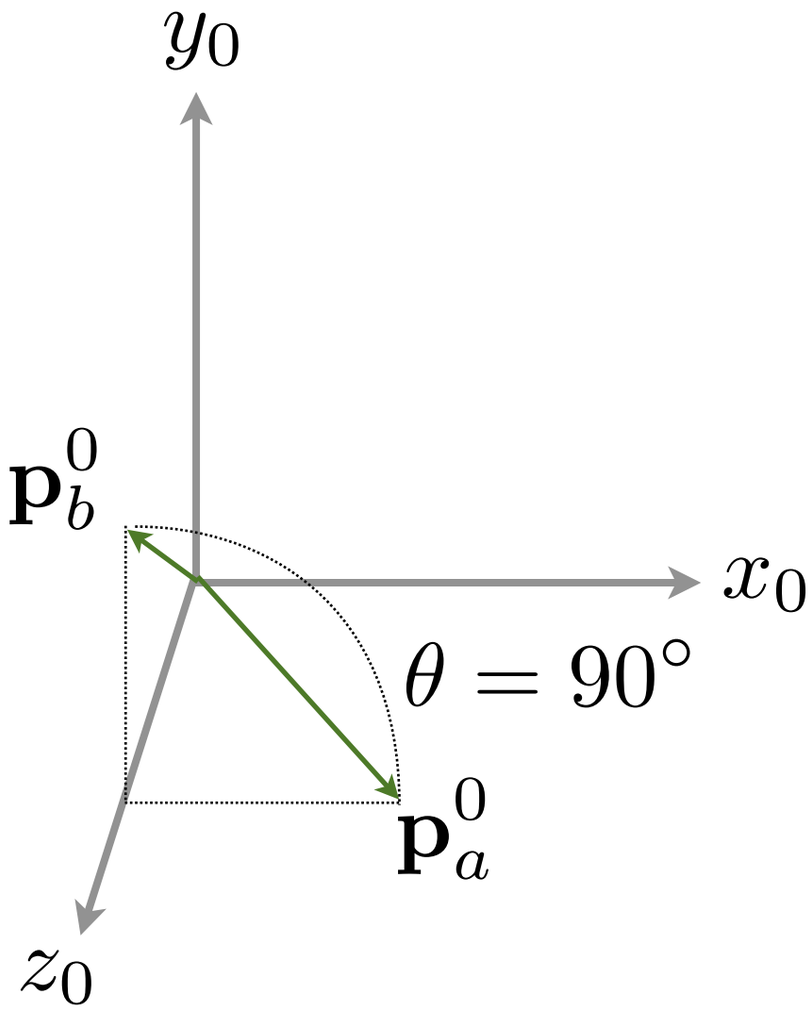
Rotational Transformations



For pure coordinate rotation, a point in frame 1 can be expressed in frame 0 using the rotation matrix

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \, \mathbf{v}_p^1$$

Rotational Transformations



The rotation matrix can also be used to perform rotations on vectors

$$\mathbf{p}_b^0 = \mathbf{R}\,\mathbf{p}_a^0$$

$$\mathbf{p}_a^0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

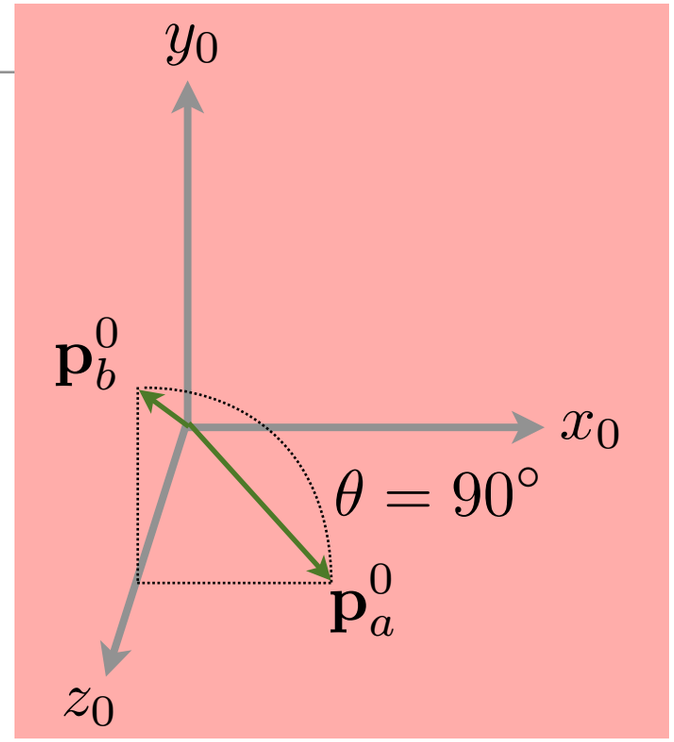
$$\mathbf{R}_{z,90°} = ?$$

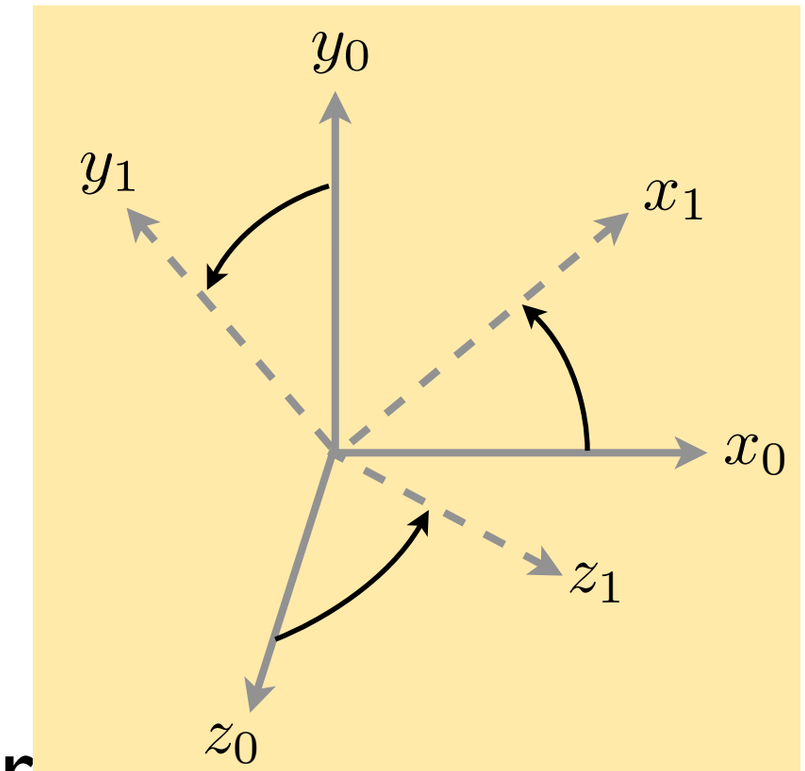
$$\mathbf{p}_b^0 = \mathbf{R}_{z,\theta}\ \mathbf{p}_a^0 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

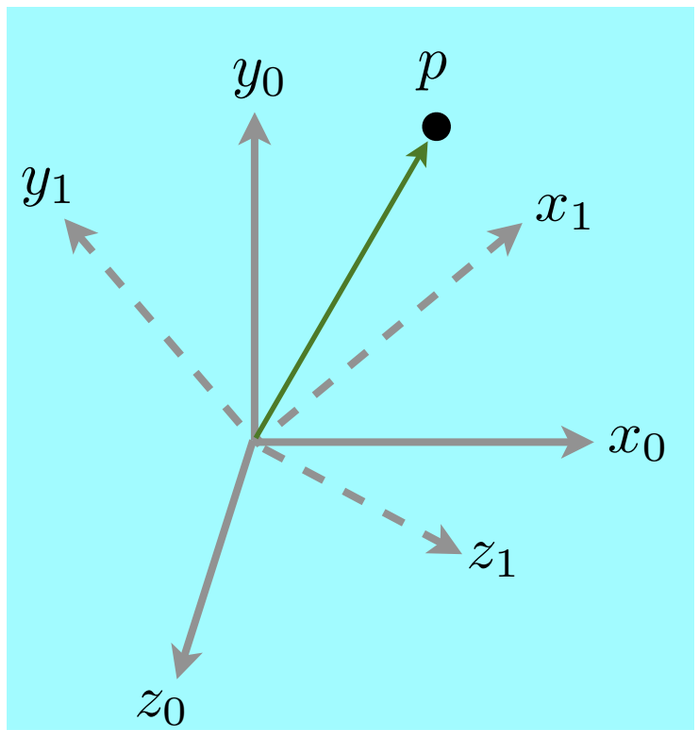
In MATLAB?

Rotation matrices serve three purposes (p. 47 in SHV):

1. Coordinate transformation relating the coordinates of a point p in two different frames
2. Orientation of a transformed coordinate frame with respect to a fixed frame
3. Operator taking a vector and rotating it to yield a new vector in the same coordinate frame







MEAM 520

More Rotation Matrices

Katherine J. Kuchenbecker, Ph.D.

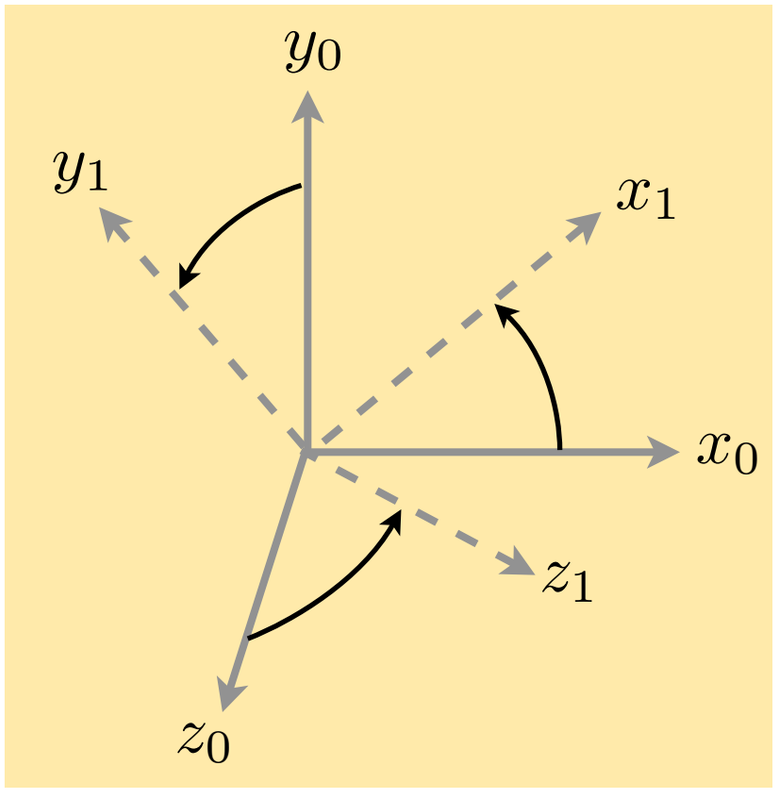
General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 4: September 10, 2013

Rotation Matrices - Interpretation 1 of 3



Represents the orientation of one coordinate frame with respect to another frame

$$\mathbf{R}_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

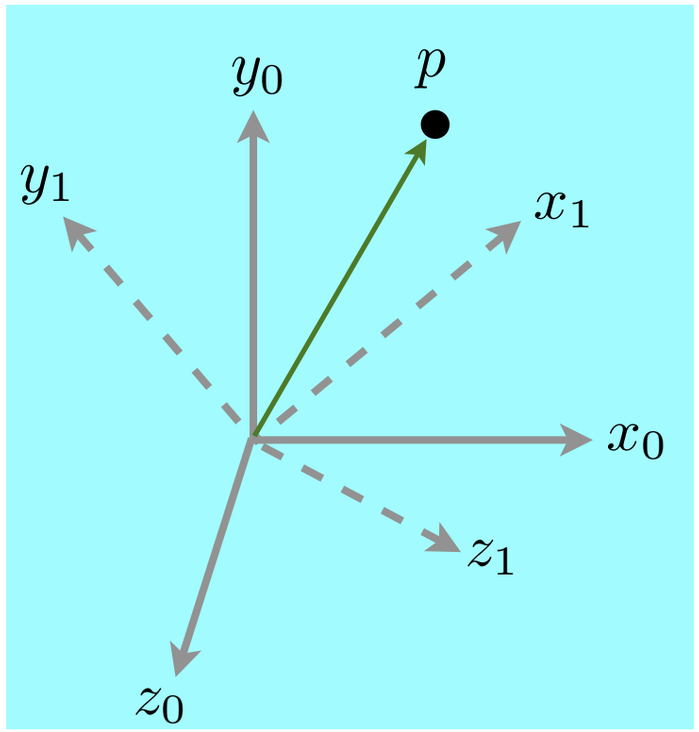
Orientation of frame 1 w.r.t. frame 0

columns are of unit length
columns are mutually orthogonal
$\det \mathbf{R}_1^0 = +1$

$\mathbf{R}_0^1 = ?$ $\quad \mathbf{R}_0^1 = \left(\mathbf{R}_1^0\right)^T$

$\left(\mathbf{R}_1^0\right)^T = \left(\mathbf{R}_1^0\right)^{-1}$

Rotation Matrices - Interpretation 2 of 3



Coordinate transformation relating the coordinates of a point p in two different frames

$\mathbf{R}_1^0 \qquad \mathbf{v}_p^1$

$\mathbf{v}_p^0 = ?$

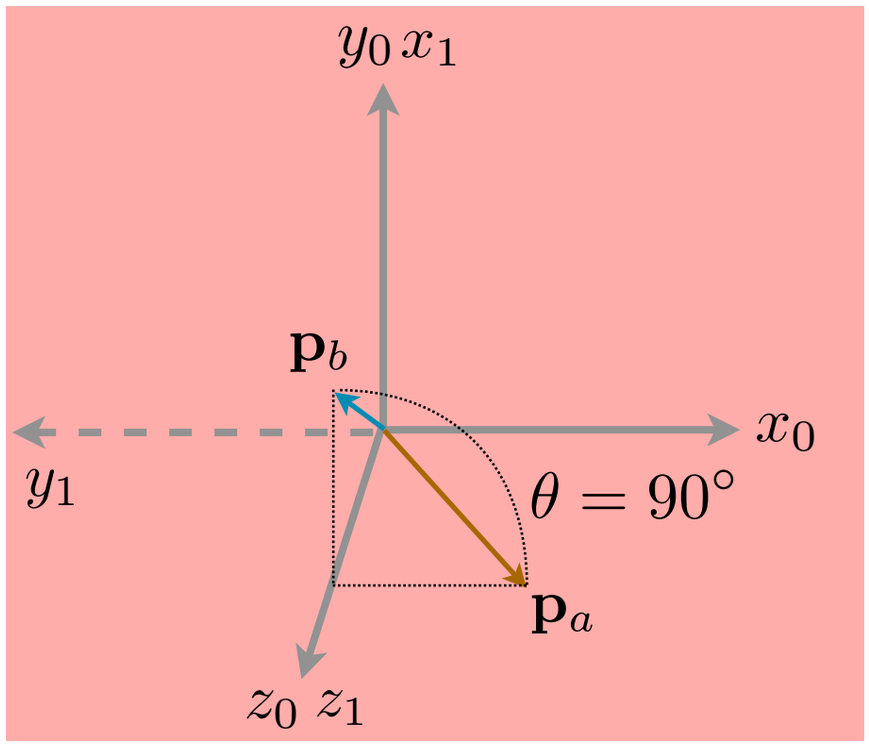
$$\boxed{\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1}$$

Subscript and superscript cancel

$\mathbf{v}_p^1 = ?$ $\qquad \left(\mathbf{R}_1^0\right)^{-1} \mathbf{v}_p^0 = \cancel{\left(\mathbf{R}_1^0\right)^{-1} \mathbf{R}_1^0} \mathbf{v}_p^1$ $\qquad \mathbf{v}_p^1 = \left(\mathbf{R}_1^0\right)^T \mathbf{v}_p^0$

$$\boxed{\mathbf{v}_p^1 = \mathbf{R}_0^1 \mathbf{v}_p^0}$$

Rotation Matrices - Interpretation 3 of 3



Operator taking a vector and rotating it to yield a new vector in the same coordinate frame

$$\mathbf{p}_a^0 \qquad\qquad \mathbf{R}_1^0$$

$$\mathbf{p}_b^0 = ?$$

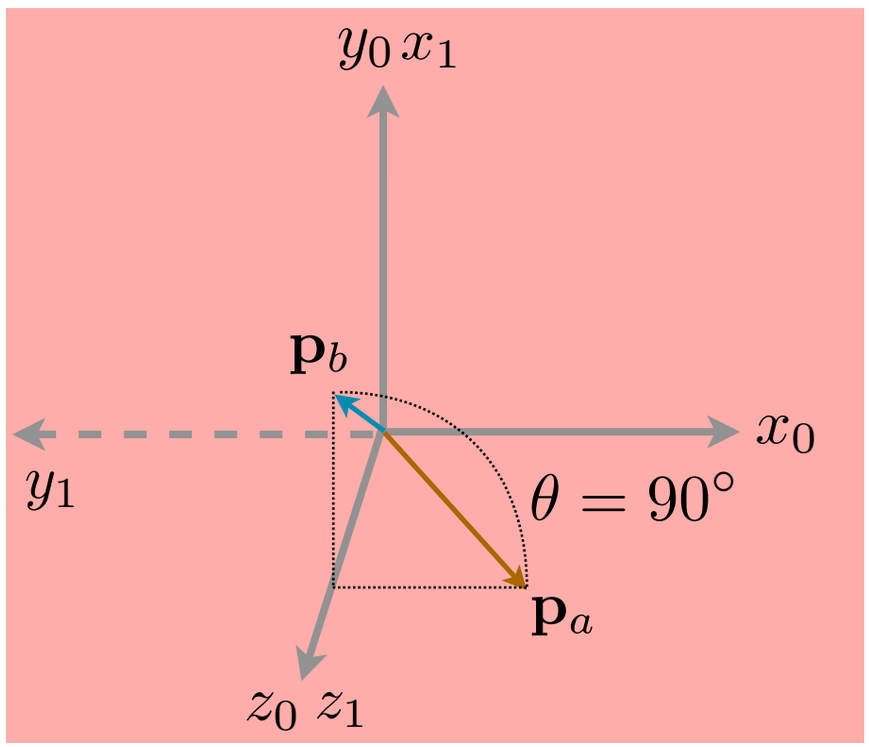
$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_b^1$$

$$\mathbf{p}_b^1 = \mathbf{p}_a^0$$

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_a^0 \qquad \boxed{\mathbf{p}_b^0 = \mathbf{R}\, \mathbf{p}_a^0}$$

Rotation Matrices - Interpretation 3 of 3

What if I want to apply another rotation to this vector?
Method depends on which axis I want to rotate around.



Operator taking a vector and rotating it to yield a new vector in the same coordinate frame

$$\mathbf{p}_a^0 \qquad\qquad \mathbf{R}_1^0$$

$$\mathbf{p}_b^0 = ?$$

For example:
rotate 45° around y_0
vs.
rotate 45° around y_1

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_b^1$$

$$\mathbf{p}_b^1 = \mathbf{p}_a^0$$

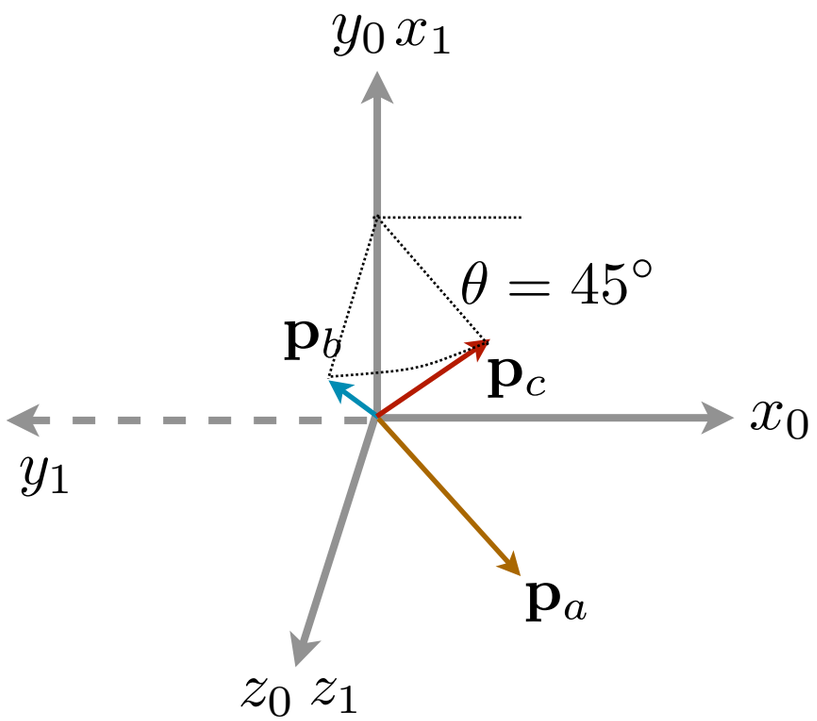
$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_a^0$$

$$\boxed{\mathbf{p}_b^0 = \mathbf{R}\, \mathbf{p}_a^0}$$

The **order** in which a sequence of rotations is performed is crucial.

Thus, the **order** in which the rotation matrices are multiplied together is crucial.

What if I want to apply another rotation to this vector? Method depends on which axis I want to rotate around.



$$\mathbf{p}_b^0 = \mathbf{R}\,\mathbf{p}_a^0 \qquad \mathbf{R}'$$

$$\mathbf{p}_c^0 = ?$$

$$\boxed{\mathbf{p}_c^0 = \mathbf{R}'\mathbf{p}_b^0}$$

$$\boxed{\mathbf{p}_c^0 = \mathbf{R}'\,\mathbf{R}\,\mathbf{p}_a^0}$$

For example:

rotate 45° around y_0

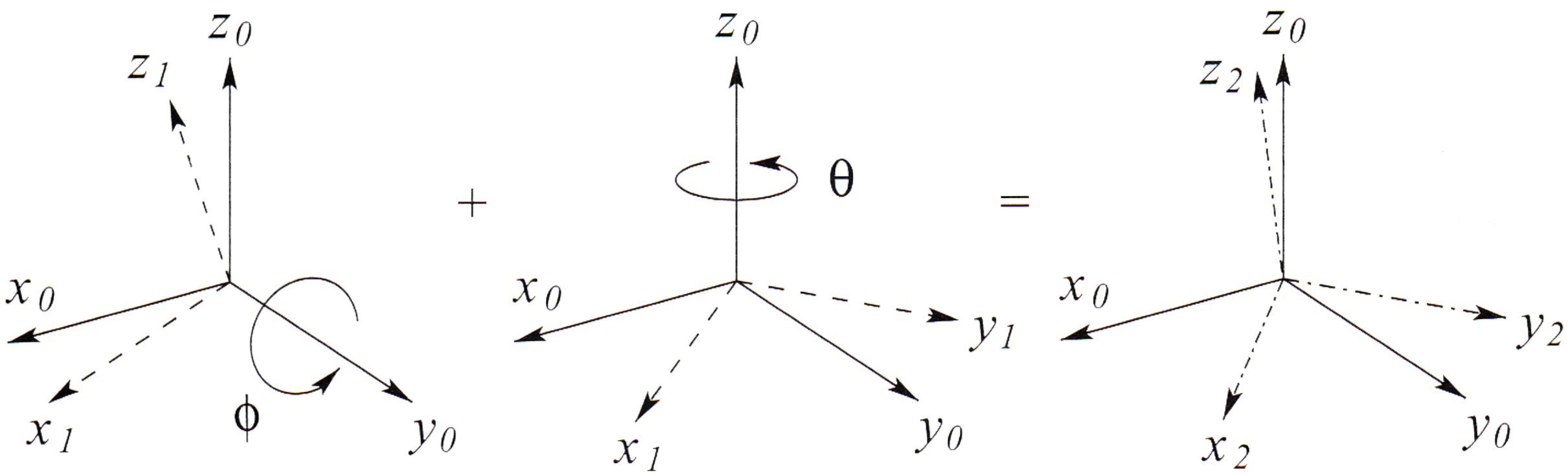
vs.

rotate 45° around y_1

Composition of Rotations with Respect to a Fixed Frame

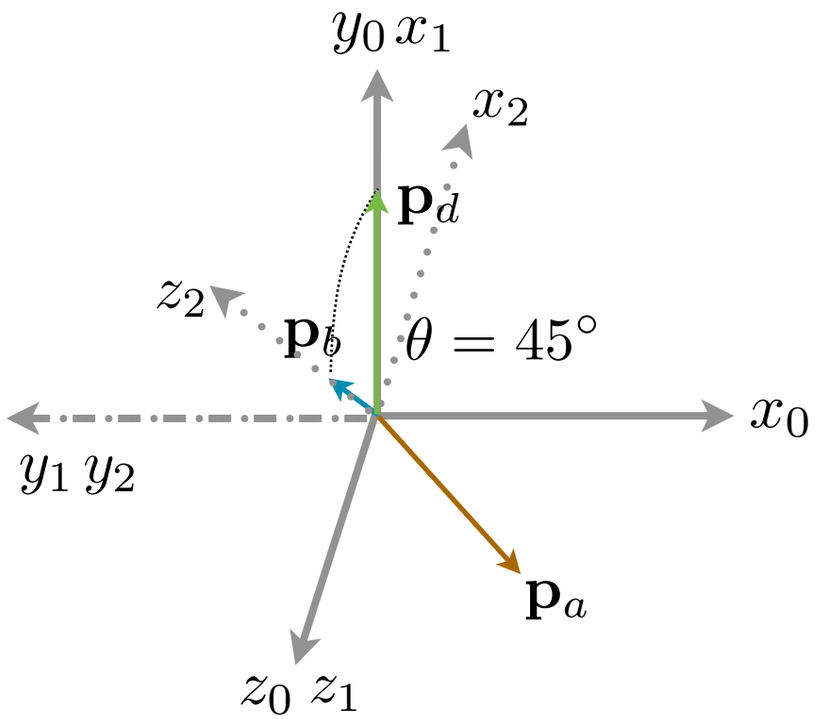
the result of a successive rotation about a fixed frame can be found by **pre-multiplying** by the corresponding rotation matrix

$$\mathbf{R}_2^0 = \mathbf{R}\,\mathbf{R}_1^0$$



Note that $\mathbf{R}$ is a rotation about the original frame

What if I want to apply another rotation to this vector? Method depends on which axis I want to rotate around.

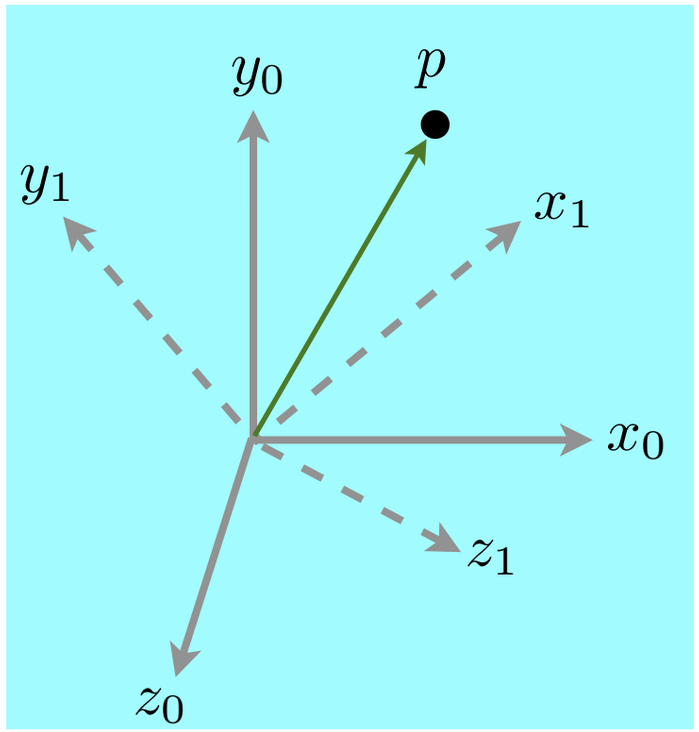


$$\mathbf{p}_d^0 = ?$$

For example:
rotate 45° around y_0
vs.
rotate 45° around y_1

Rotation Matrices - Interpretation 2 of 3

Remember...



Coordinate transformation relating the coordinates of a point p in two different frames

$$\mathbf{R}_1^0 \qquad \mathbf{v}_p^1$$

$$\mathbf{v}_p^0 = ?$$

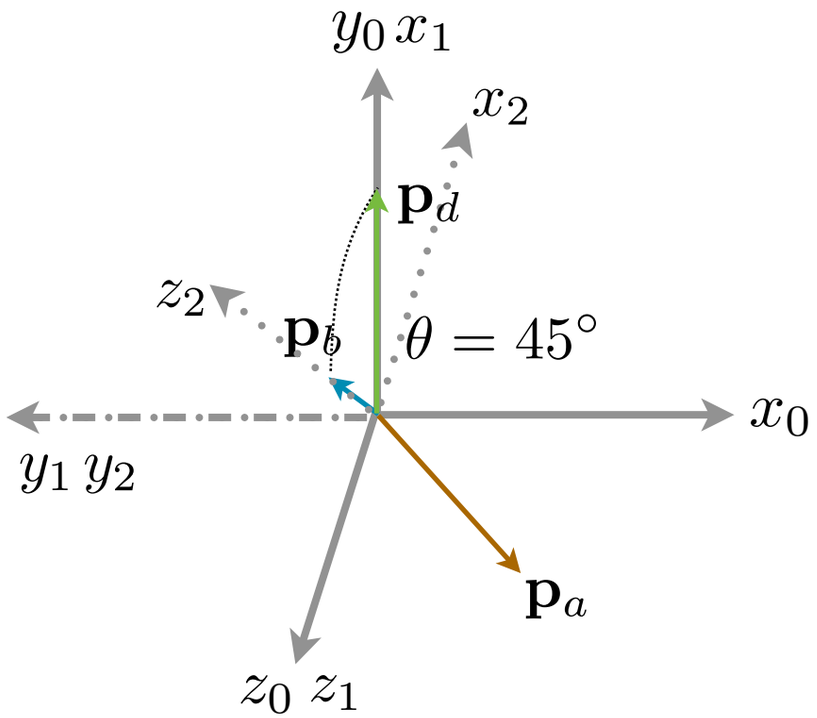
$$\boxed{\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1}$$

Subscript and superscript cancel

$$\mathbf{v}_p^1 = ? \qquad \left(\mathbf{R}_1^0\right)^{-1} \mathbf{v}_p^0 = \cancel{\left(\mathbf{R}_1^0\right)^{-1} \mathbf{R}_1^0} \mathbf{v}_p^1 \qquad \mathbf{v}_p^1 = \left(\mathbf{R}_1^0\right)^T \mathbf{v}_p^0$$

$$\boxed{\mathbf{v}_p^1 = \mathbf{R}_0^1 \mathbf{v}_p^0}$$

What if I want to apply another rotation to this vector? Method depends on which axis I want to rotate around.



$$\mathbf{p}_d^2 = \mathbf{p}_a^0$$

$$\mathbf{p}_d^0 = ?$$

$$\boxed{\mathbf{p}_d^0 = \mathbf{R}_1^0\, \mathbf{R}_2^1\, \mathbf{p}_a^0}$$

interp. 3 (operator)

$$\mathbf{p}_d^0 = \mathbf{R}_2^0\, \mathbf{p}_d^2$$

$$\mathbf{p}_d^0 = \mathbf{R}_1^0\, \mathbf{p}_d^1$$

$$\mathbf{p}_d^1 = \mathbf{R}_2^1\, \mathbf{p}_d^2$$

$$\boxed{\mathbf{R}_2^0 = \mathbf{R}_1^0\, \mathbf{R}_2^1}$$

$$\boxed{\mathbf{p}_d^0 = \mathbf{R}_1^0\, \mathbf{R}_2^1\, \mathbf{p}_d^2}$$

interp. 2 (frames)

For example:

rotate 45° around y_0

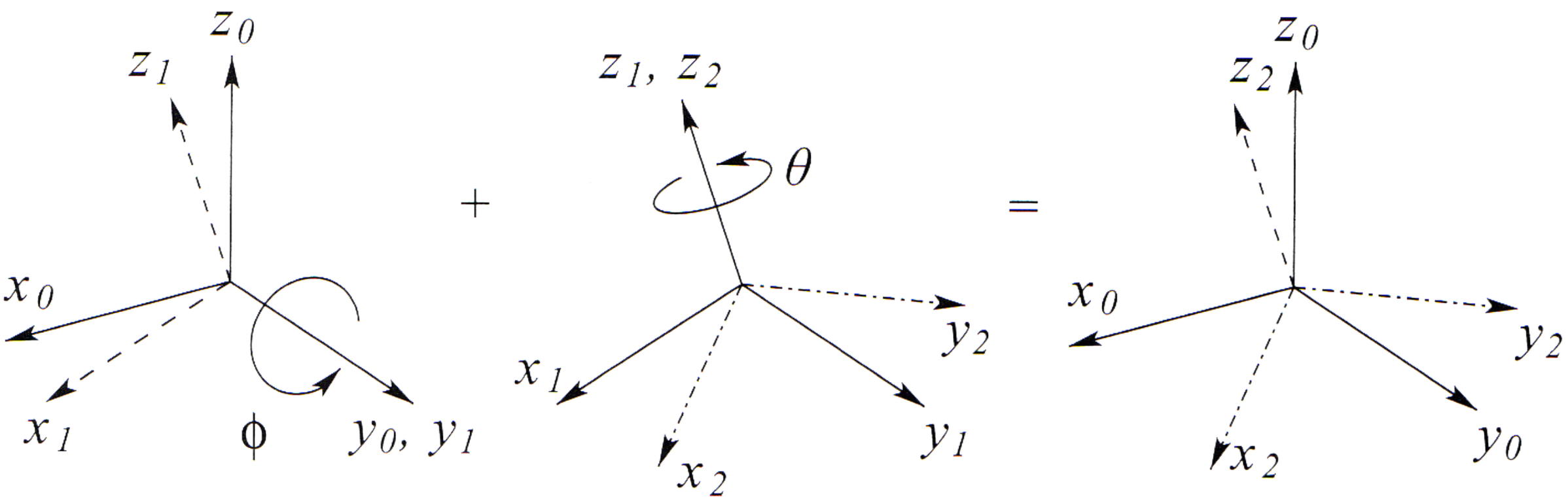
vs.

rotate 45° around y_1

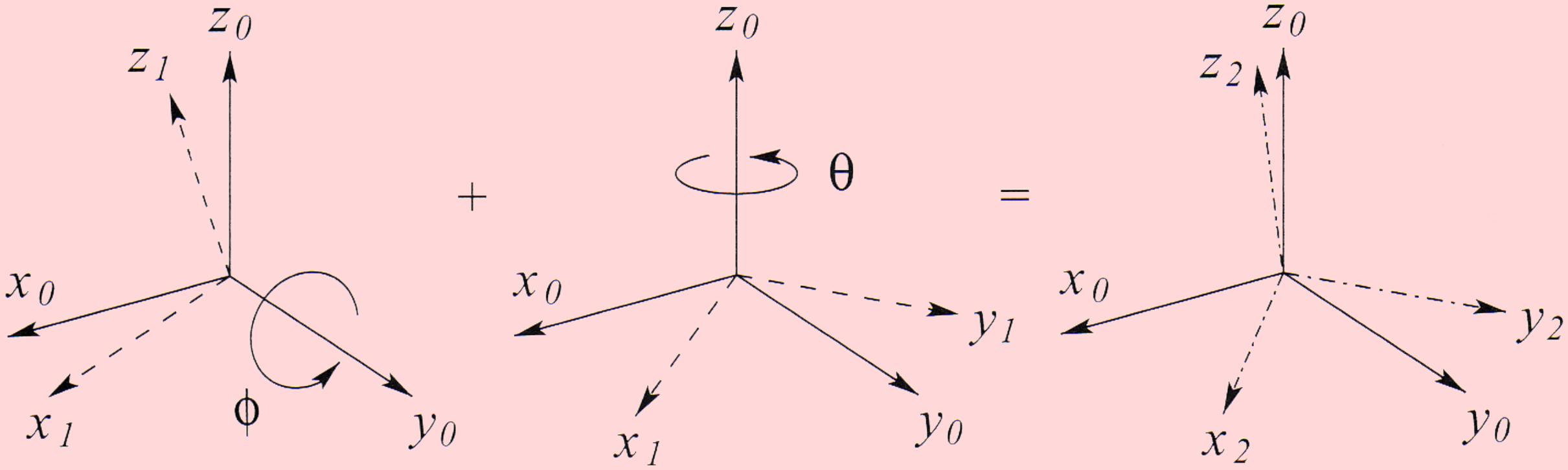
Composition of Rotations with Respect to the Current Frame

the result of a successive rotation about the current (intermediate) frame can be found by **post-multiplying** by the corresponding rotation matrix

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \, \mathbf{R}_2^1$$



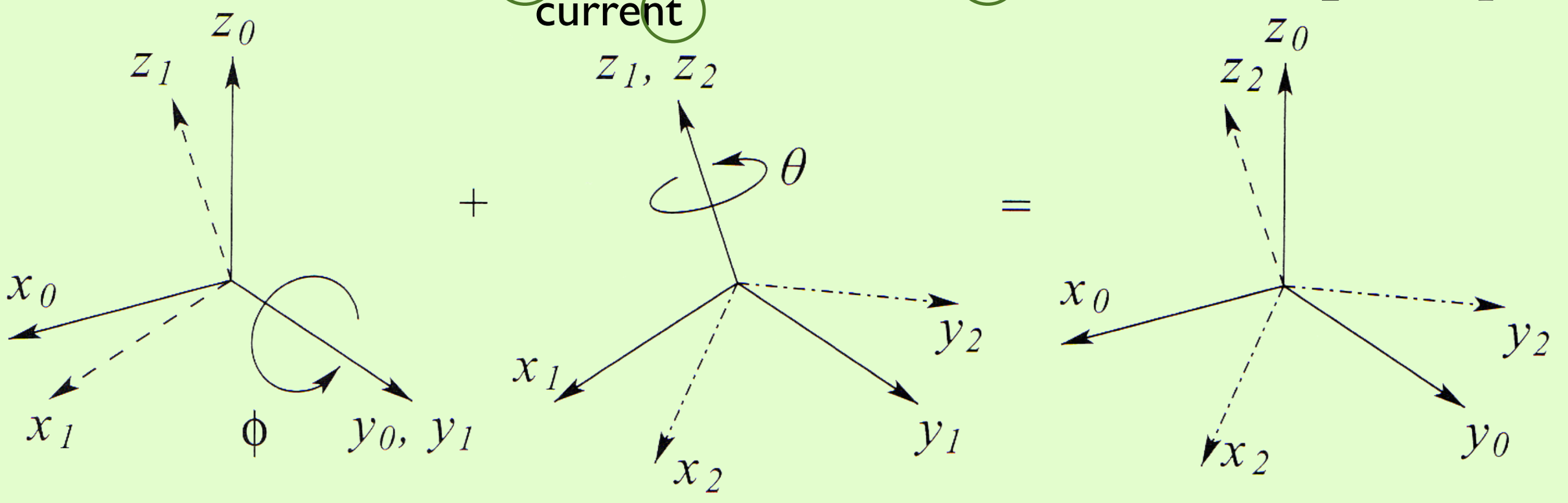
successive rotation about fixed frame? **pre-multiply** $\mathbf{R}_2^0 = \mathbf{R}\,\mathbf{R}_1^0$



Which of these is more commonly used in robotics? post multiply!

successive rotation about intermediate frame? **post-multiply** $\mathbf{R}_2^0 = \mathbf{R}_1^0\,\mathbf{R}_2^1$

current



Parameterization of Rotations

$$\mathbf{R}_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

in three dimensions, no more than 3 values are needed to specify an arbitrary rotation

which means that the 9-element rotation matrix has at least 6 redundancies

---

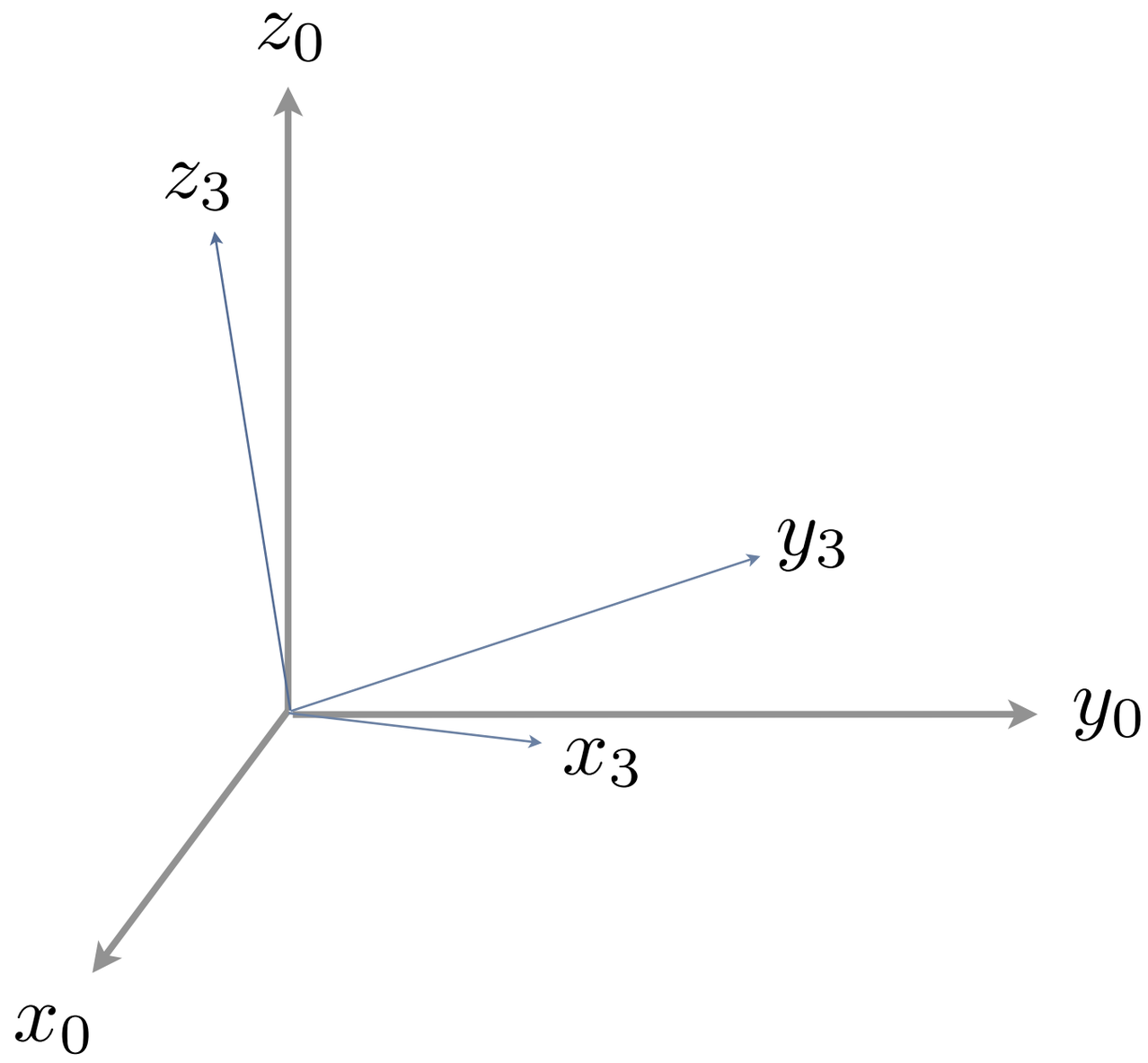
numerous methods have been developed to represent rotation/orientation with less redundancy

Euler Angles Roll, Pitch, Yaw Angles Axis/Angle Representation

Conventions vary, so always check definitions!

Euler Angles

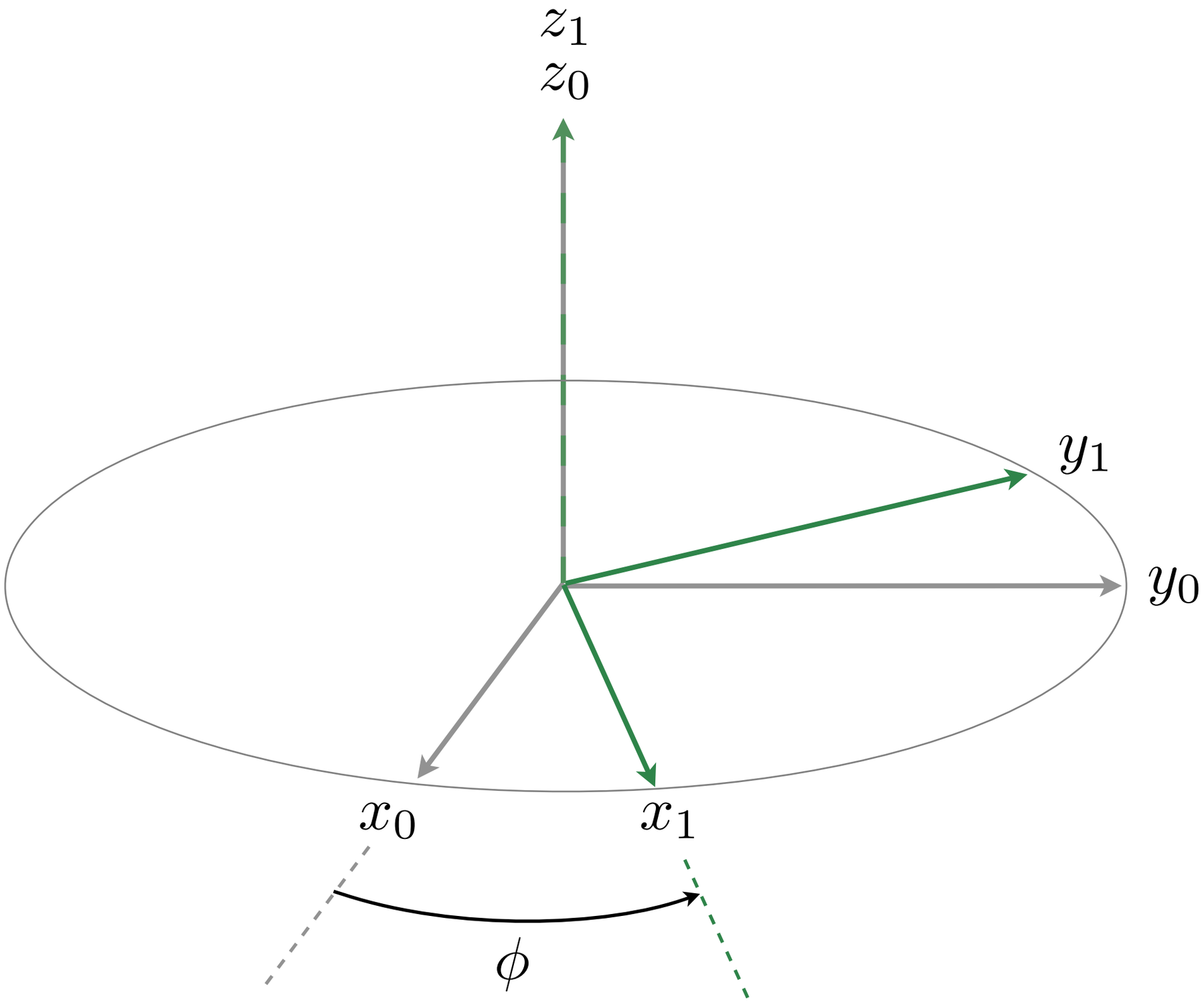
Define a set of three **intermediate** angles, ϕ, θ, ψ , to go from $0 \rightarrow 3$



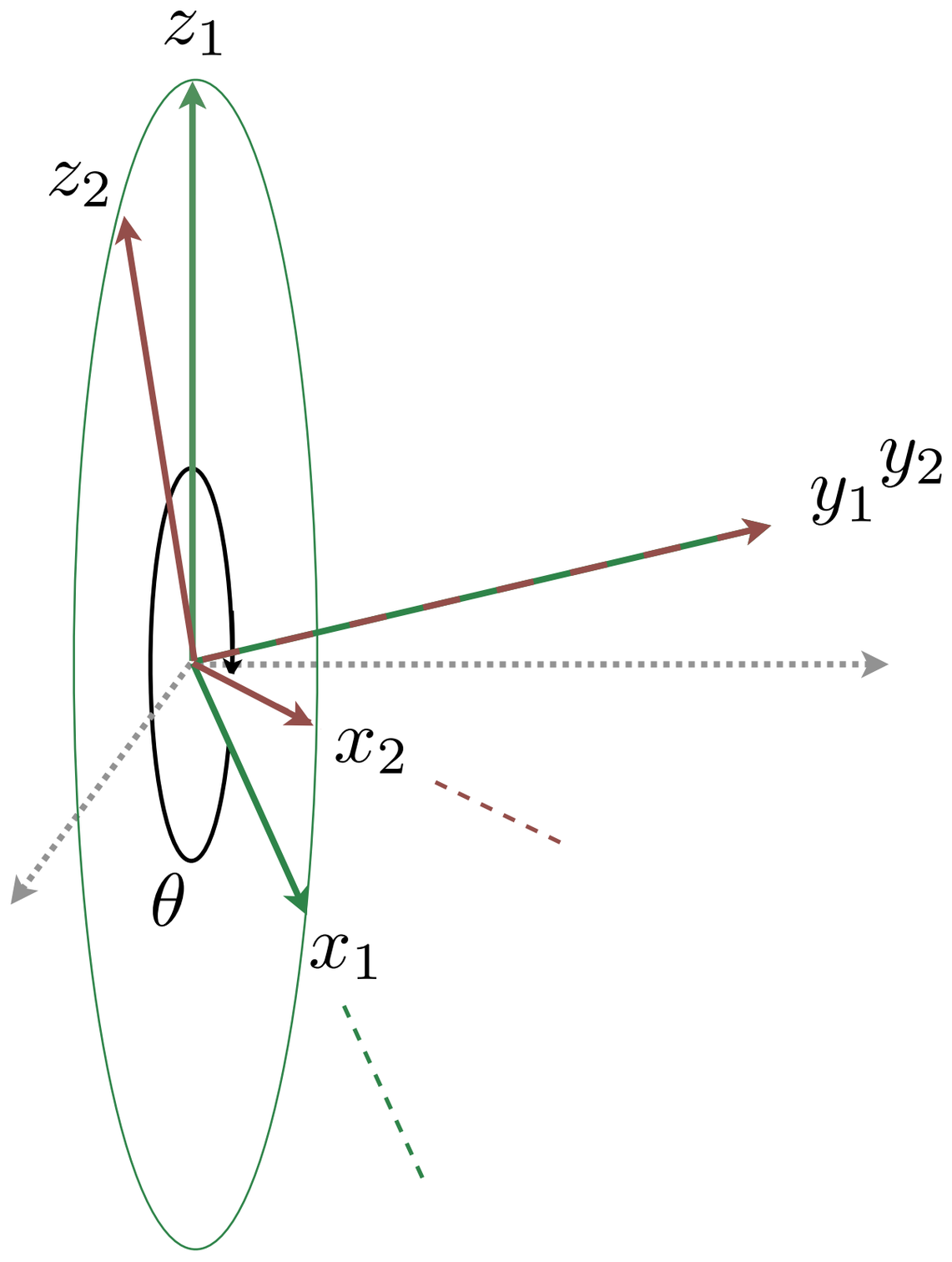
Our book uses a Z-Y-Z convention for Euler angles.

Euler Angles

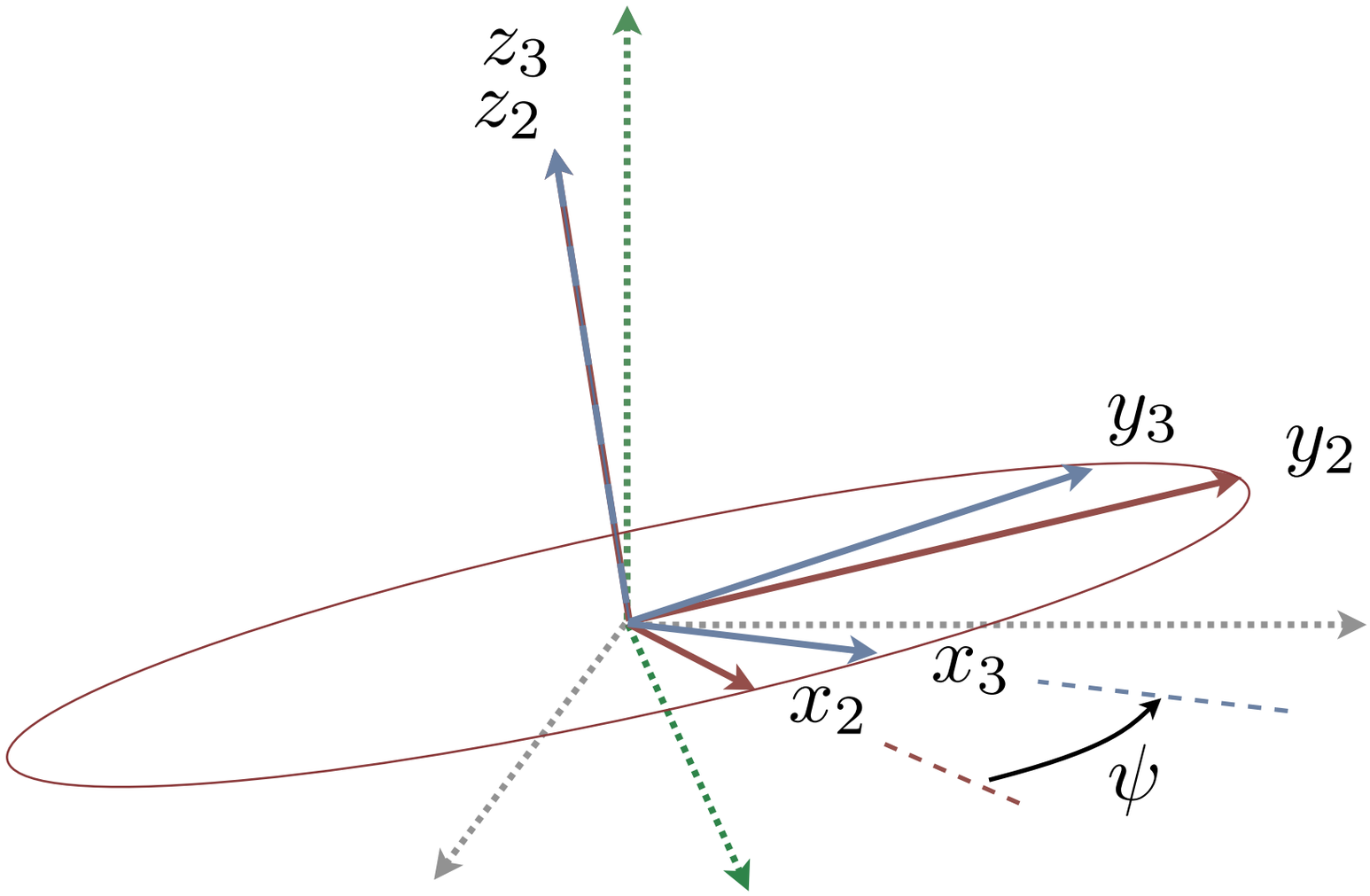
step 1: rotate by ϕ about z_0



step 2: rotate by θ about y_1

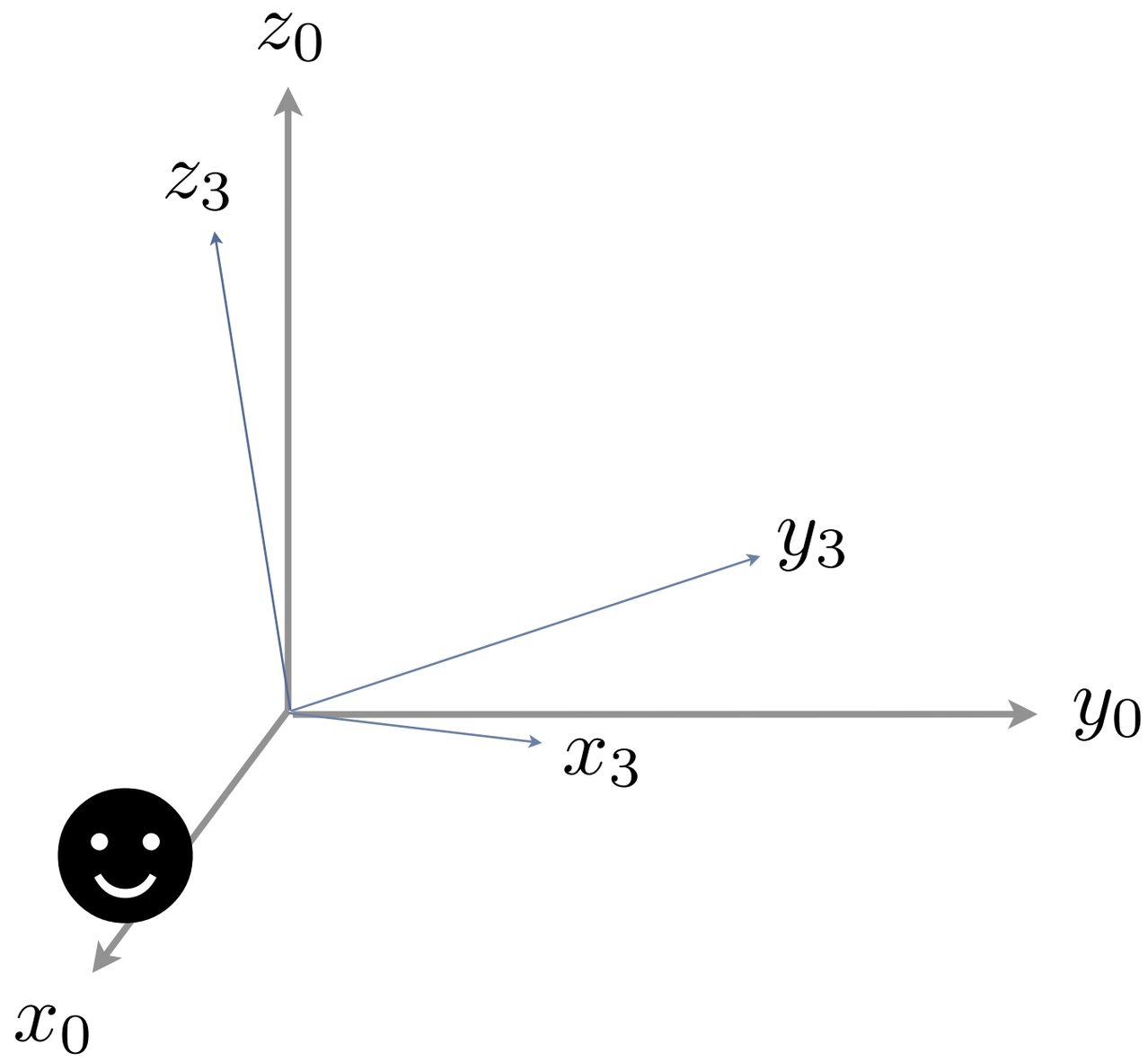


step 3: rotate by ψ about z_2



Euler Angles

Define a set of three **intermediate** angles, ϕ, θ, ψ , to go from $0 \rightarrow 3$



Think about looking out the x-axis with your head, looking left/right, then tilting up/down, then spinning your head around its long axis.

Should we pre- or post-multiply the successive rotations?

Euler Angles to Rotation Matrix

(**post**-multiply using the **basic rotation matrices**)

$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{z,\psi} \qquad s_\theta = \sin\theta \qquad c_\theta = \cos\theta$$

$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

Rotation Matrix to Euler Angles?

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Use atan2 to determine ϕ for both θ options

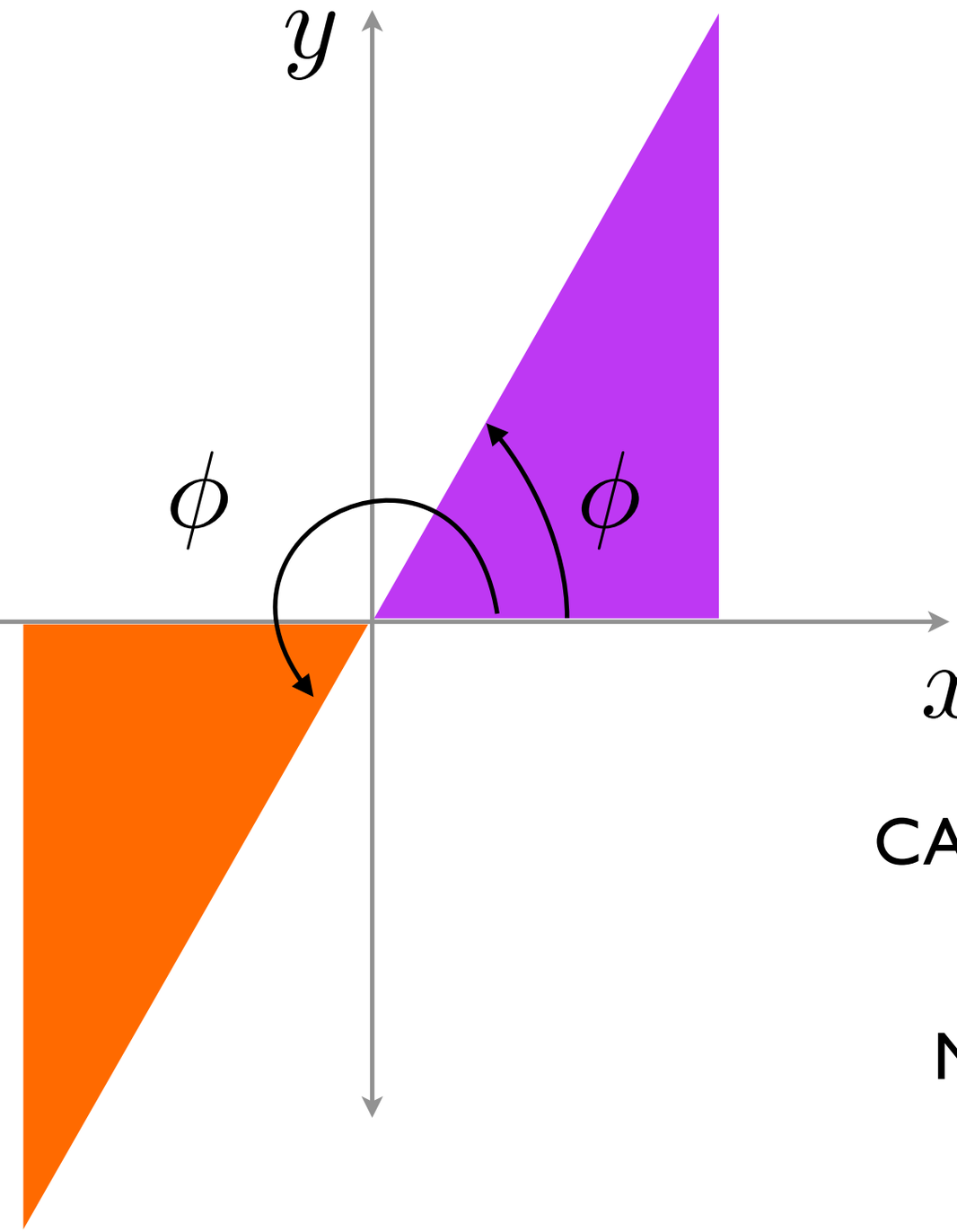
$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

Use atan2 to determine ψ for both θ options

Two solutions for θ because sign of s_θ is not known.

$$\phi = ? \quad \theta = ? \quad \psi = ?$$

atan2



atan2 is the two-argument inverse tangent function.

It preserves the signs of the numerator and denominator, returning angles in all four quadrants.

Read Appendix A.1 to learn more.

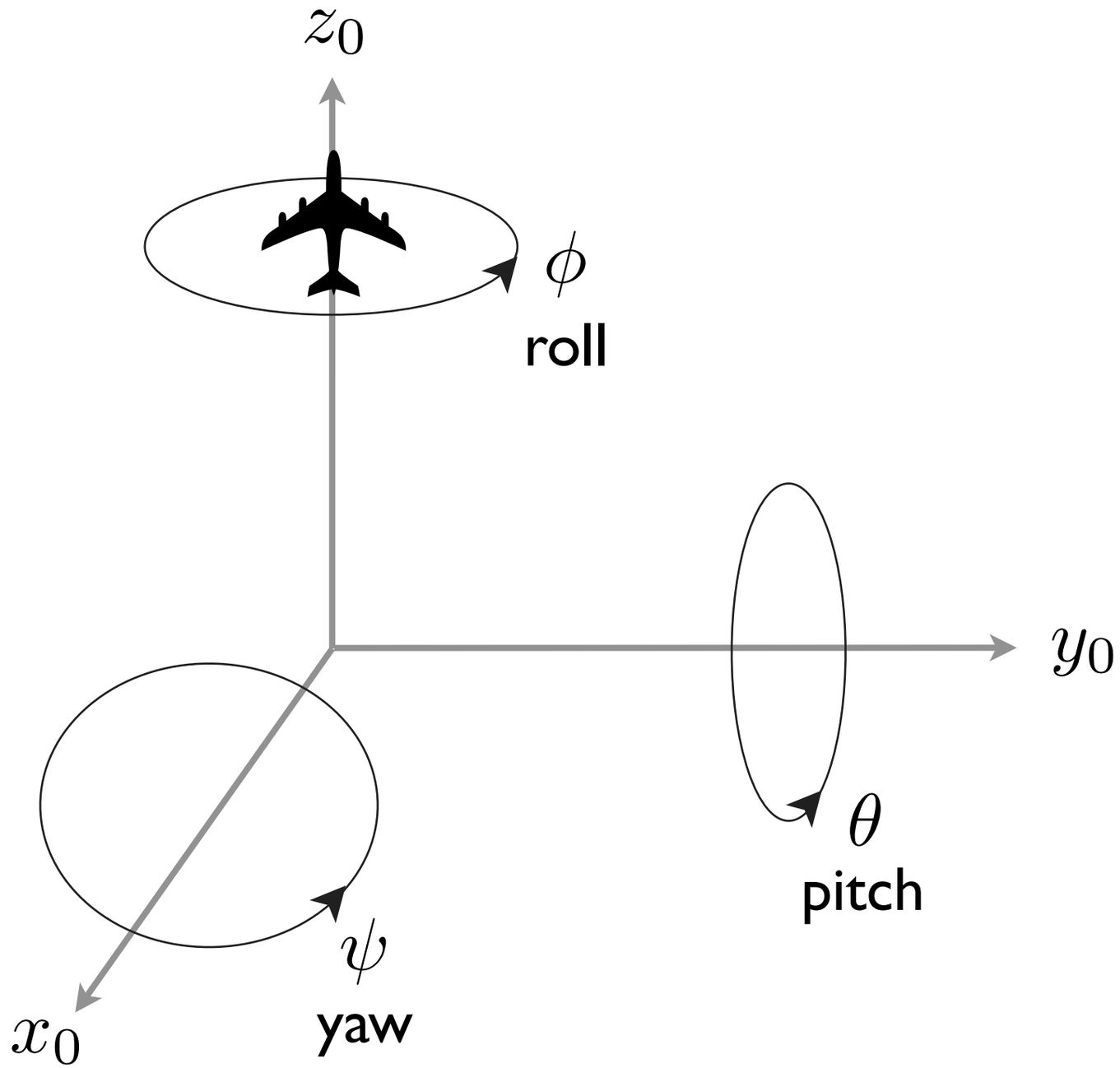
CAUTION: Our book lists atan2 arguments in the opposite order of MATLAB.

MATLAB expects atan2(num, den) or similarly atan2(y, x).

SHV uses atan2(den, num) or similarly atan2(x, y).

Roll, Pitch, Yaw Angles

defined as a set of three angles about a **fixed** reference



Our book uses an X-Y-Z convention for Yaw, Pitch, Roll angles.

Think about being a plane flying in the z-axis direction. Yaw is turning left/right, pitch is tilting up/down, and roll rotates around travel direction.

Should we pre- or post-multiply the successive rotations?

Roll, Pitch, Yaw Angles to Rotation Matrices

(**pre**-multiply using the **basic rotation matrices**)

$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{x,\psi}$$

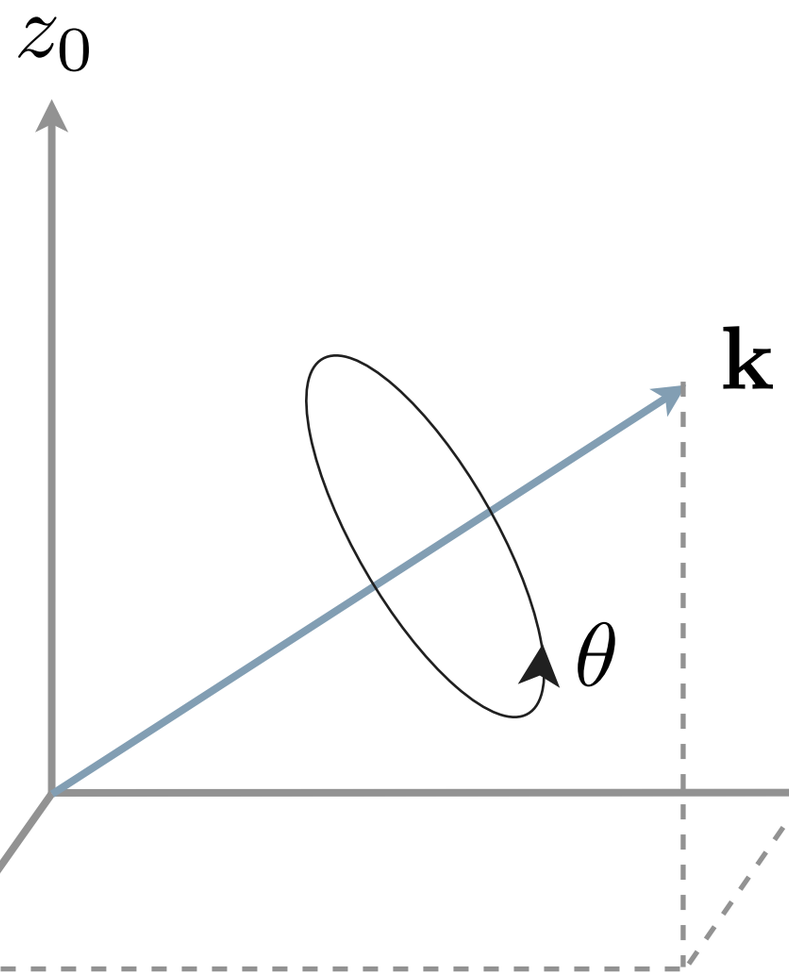
$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$$

You can convert from a rotation matrix to these angles in a manner similar to the procedure for Euler angles.

Axis/Angle Representation

rotation by an angle about an axis in space



$$\mathbf{k} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix}$$

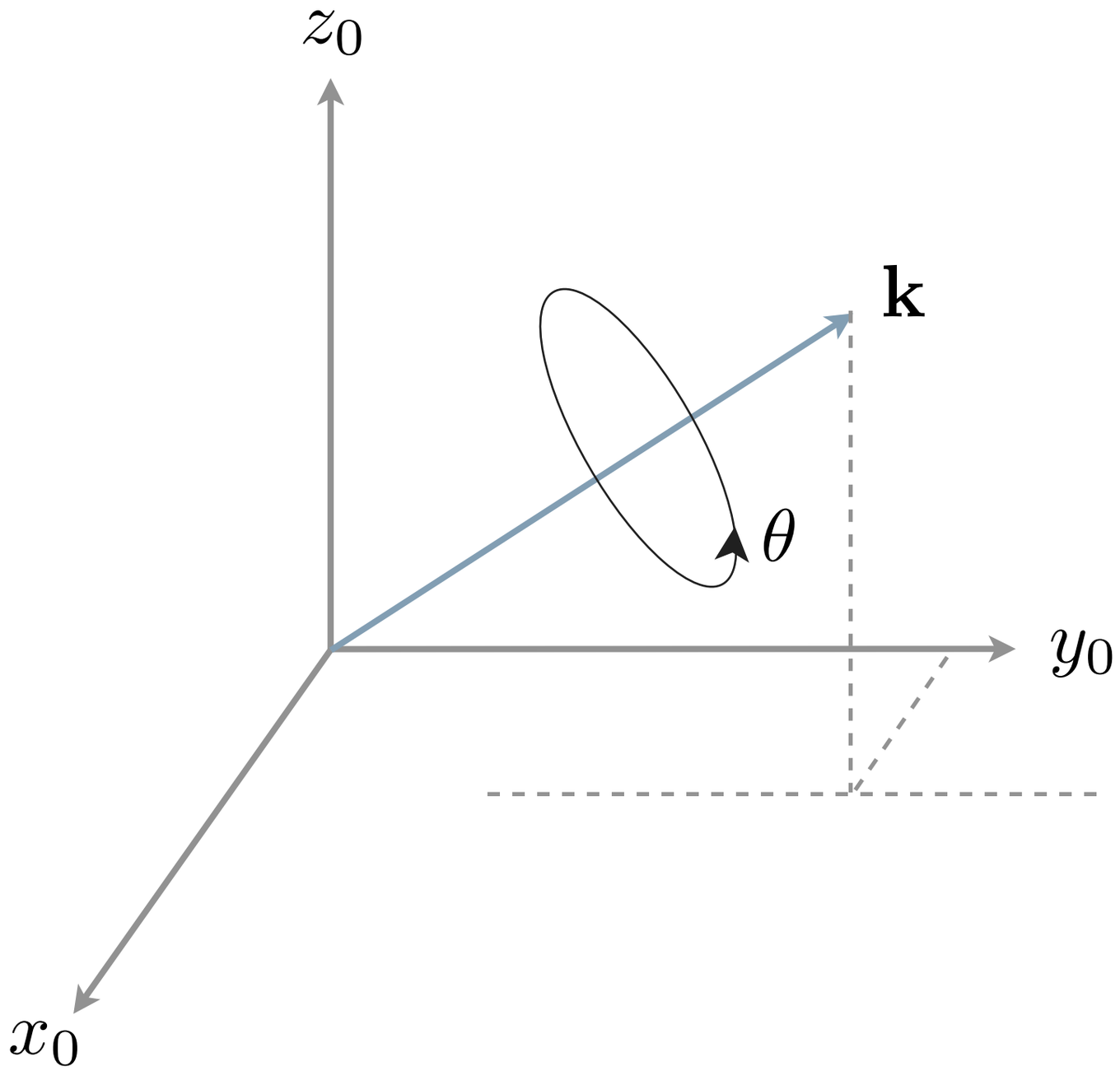
$$k_x^2 + k_y^2 + k_z^2 = 1$$

where $v_\theta = \operatorname{vers}\theta = 1 - \cos\theta$

$$R_{k,\theta} = \left[\begin{array}{c|c|c} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k_z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k_z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{array}\right]$$

Axis/Angle Representation

any rotation matrix can be represented this way!



$$R = R_{k,\theta}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Is axis/angle solution unique?

No. $R_{k,\theta} = R_{-k,-\theta}$

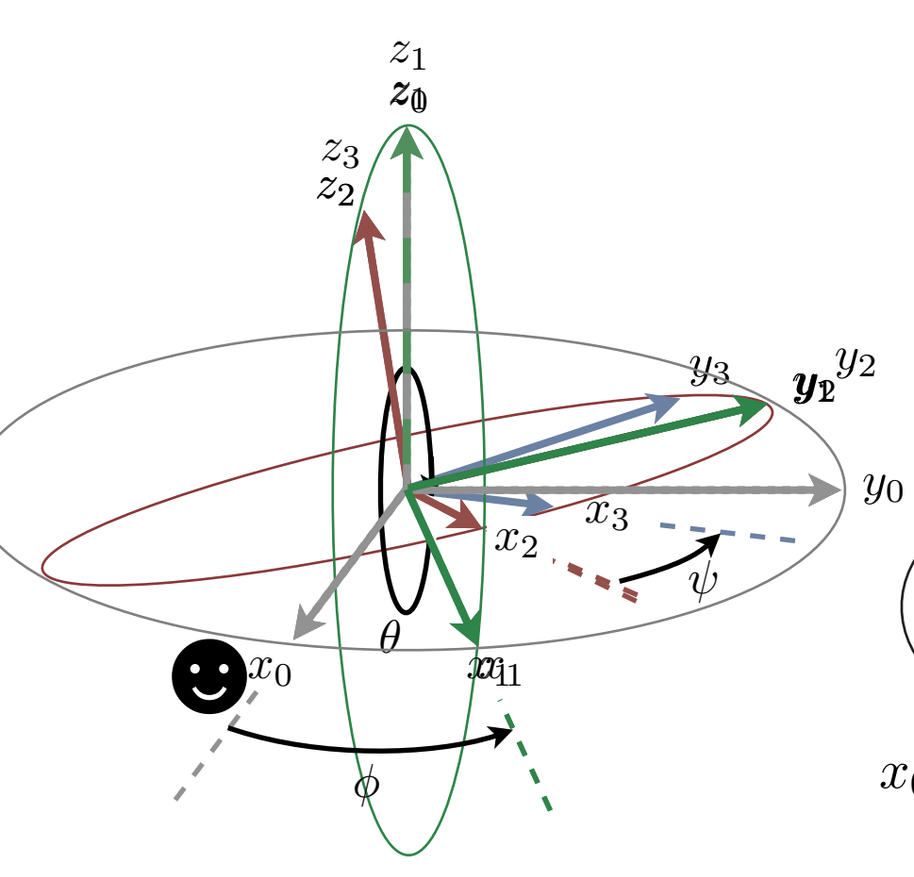
$$\theta = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

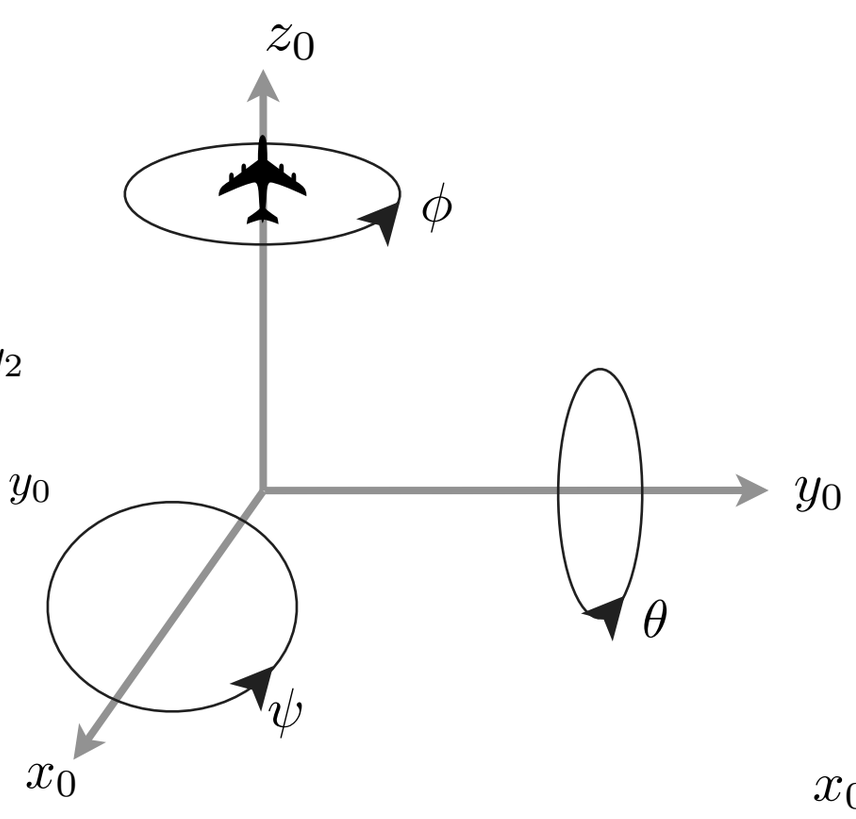
$$k = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

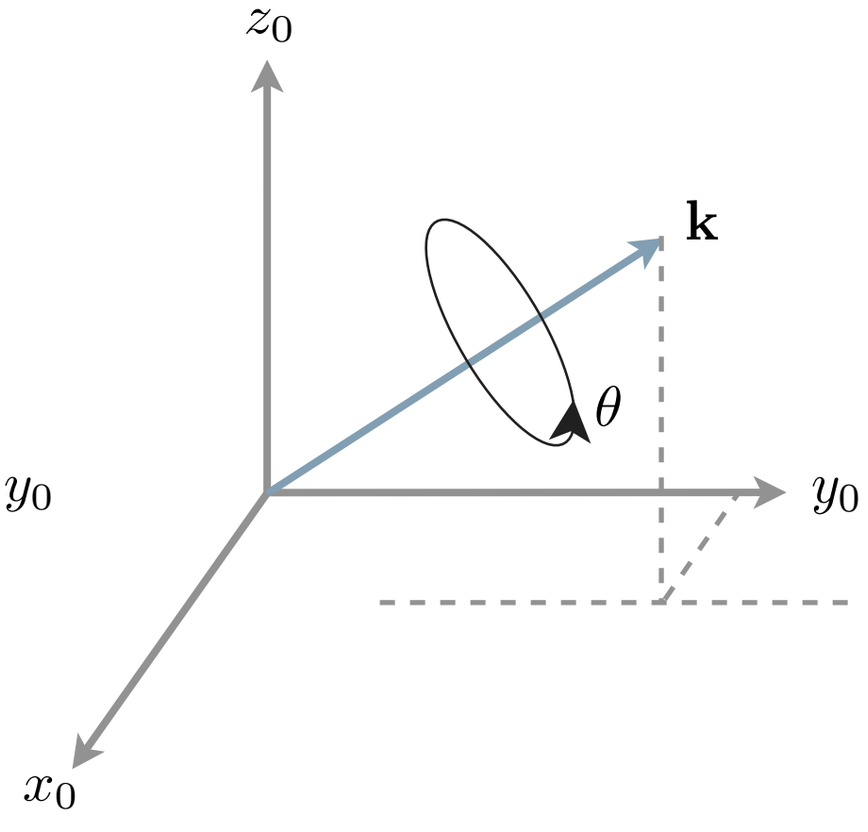
Talk to the person next to you.

Explain one of the three parameterization approaches to your partner, then switch.

Talk about the third one together.







MEAM 520

Homogeneous Transformations

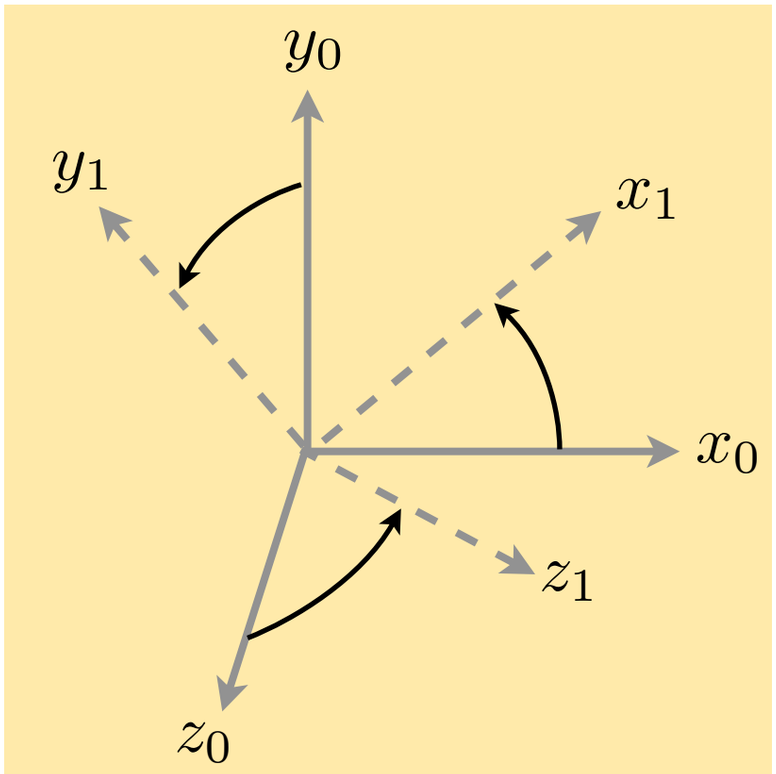
Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

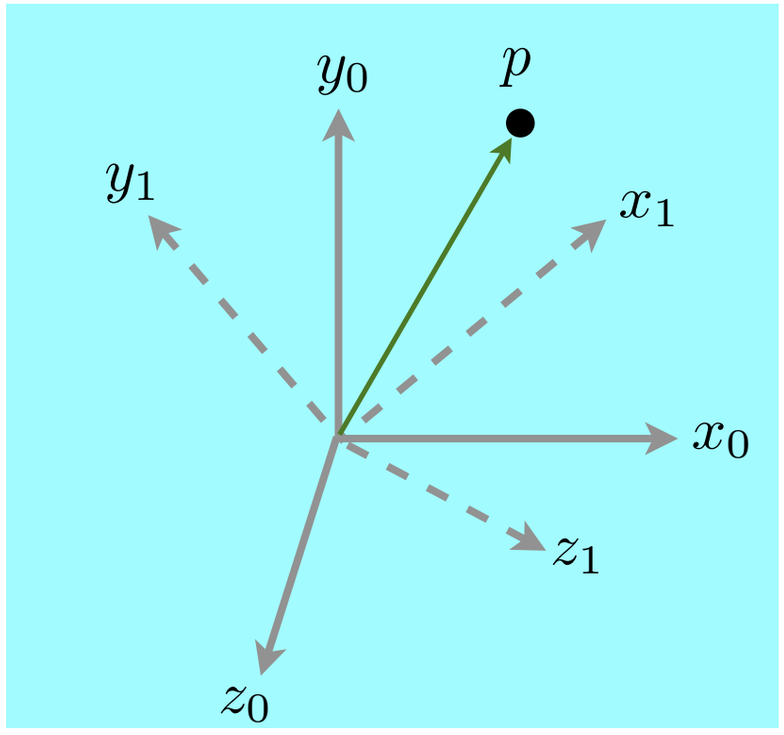
Lecture 5: September 12, 2013

Interpretations of Rotation Matrices

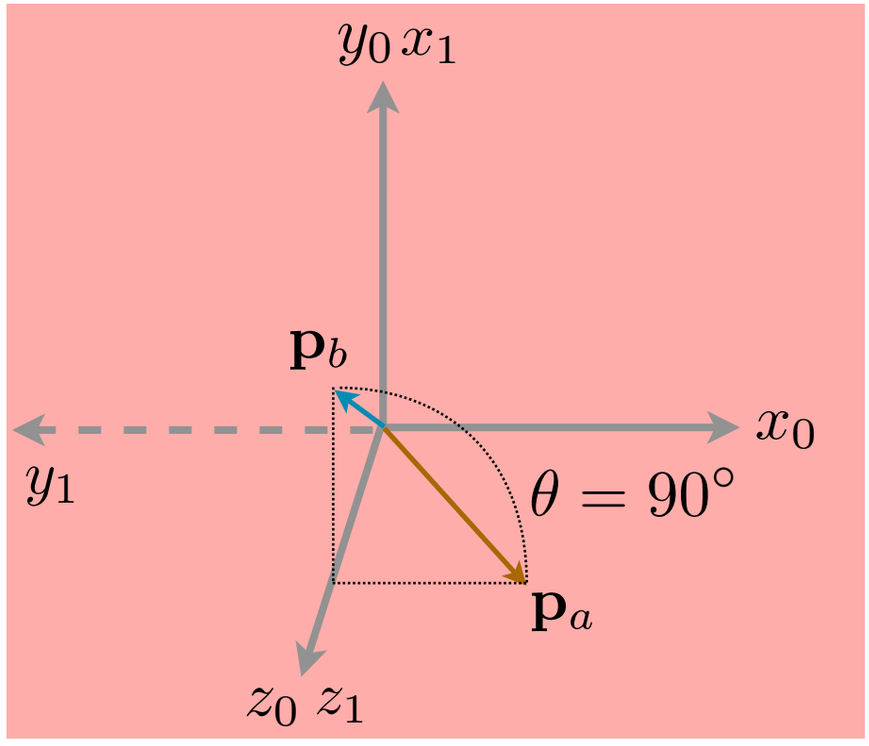
$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$



Orientation of one coordinate frame with respect to another frame



Coordinate transformation relating the coordinates of a point p in two different frames



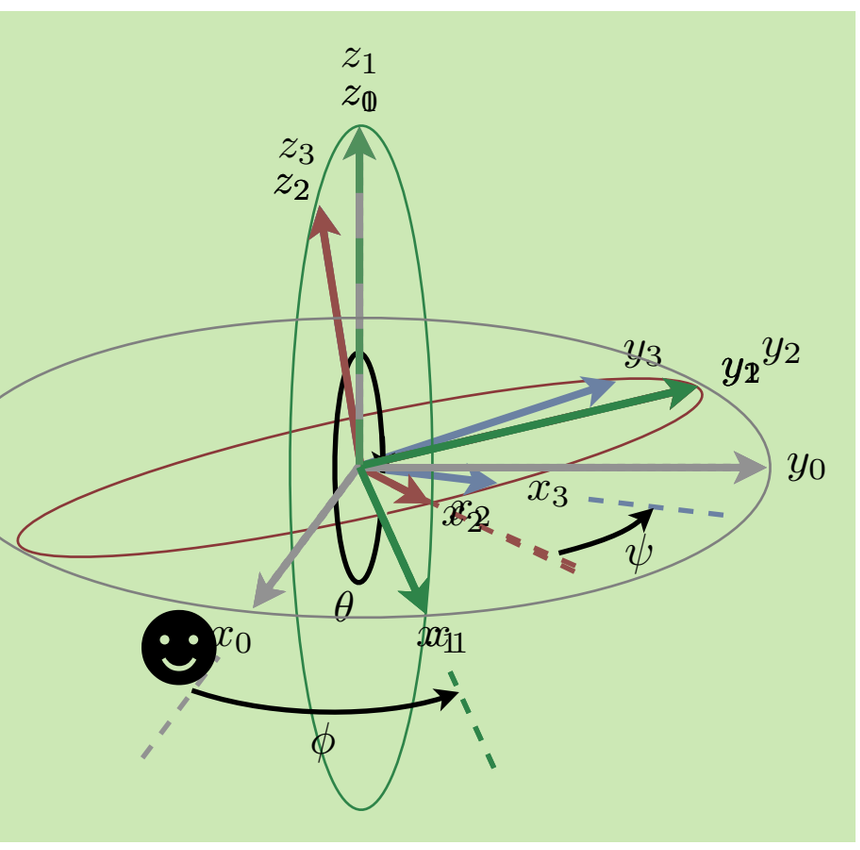
Operator taking a vector and rotating it to yield a new vector in the same coordinate frame

Parameterizations of Rotation Matrices

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Euler Angles

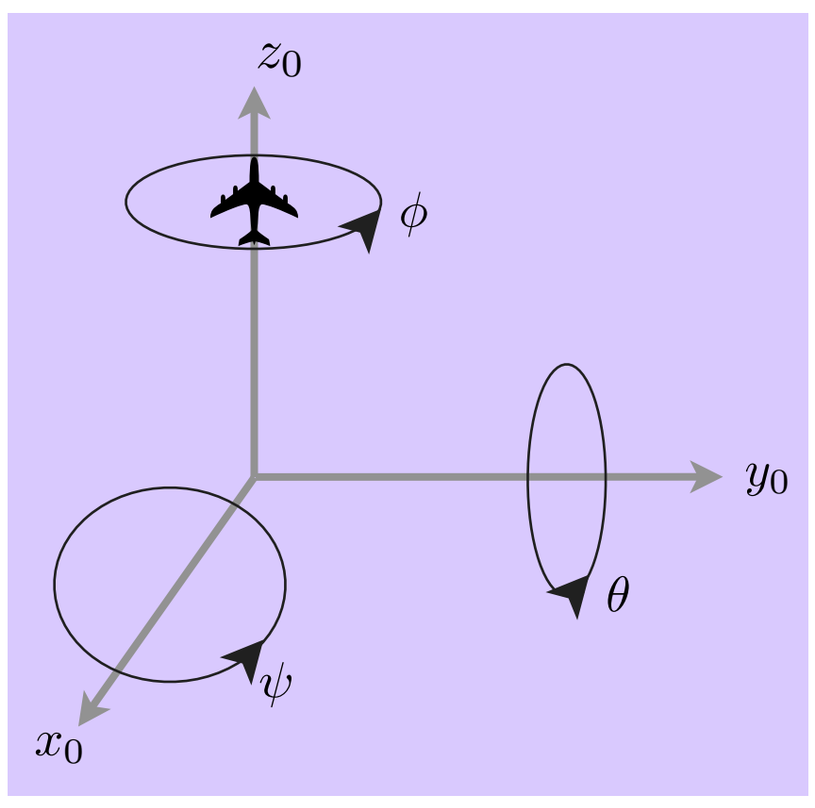
three rotations about intermediate frames



Our book uses ZYZ

Roll, Pitch, Yaw Angles

three rotations about the fixed frame



Our book uses XYZ

Axis/Angle

a unit vector (axis) and one angle

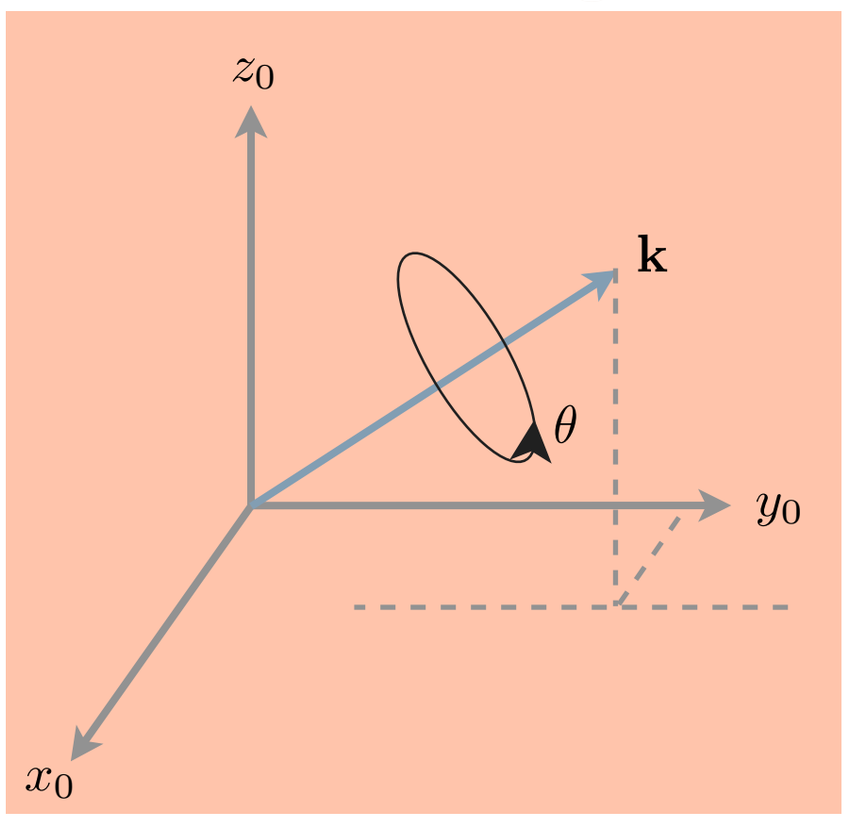


Figure 1

File Edit View Insert Tools Desktop Window Help

Press any key to continue.

$$R_1^0 = \begin{matrix} -0.1620 & -0.9324 & -0.3231 \\ 0.7479 & -0.3296 & 0.5762 \\ -0.6437 & -0.1483 & 0.7507 \end{matrix}$$

φ_a = 119.3 degrees

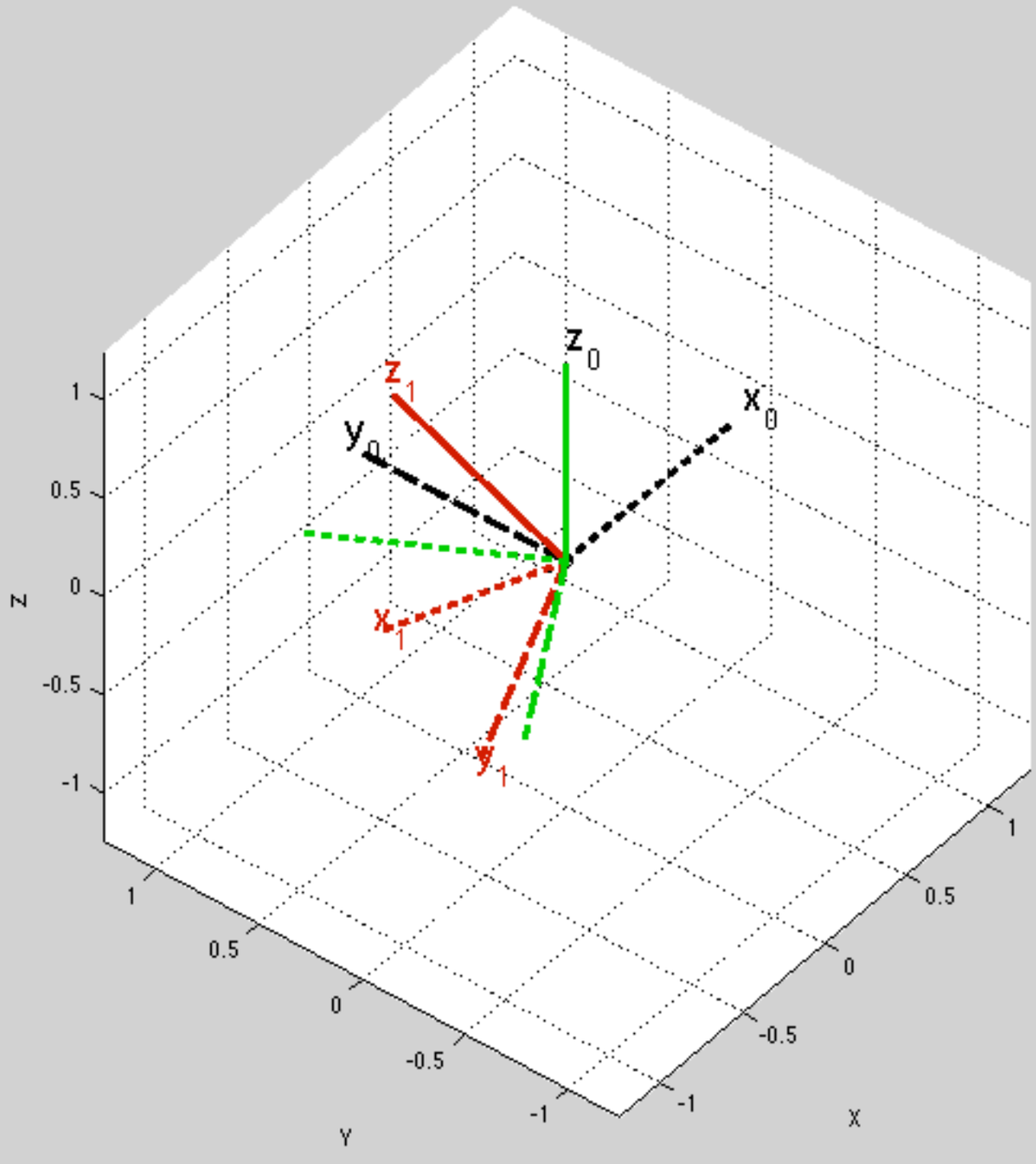
θ_a = 41.3 degrees

ψ_a = -13.0 degrees

φ_b = -60.7 degrees

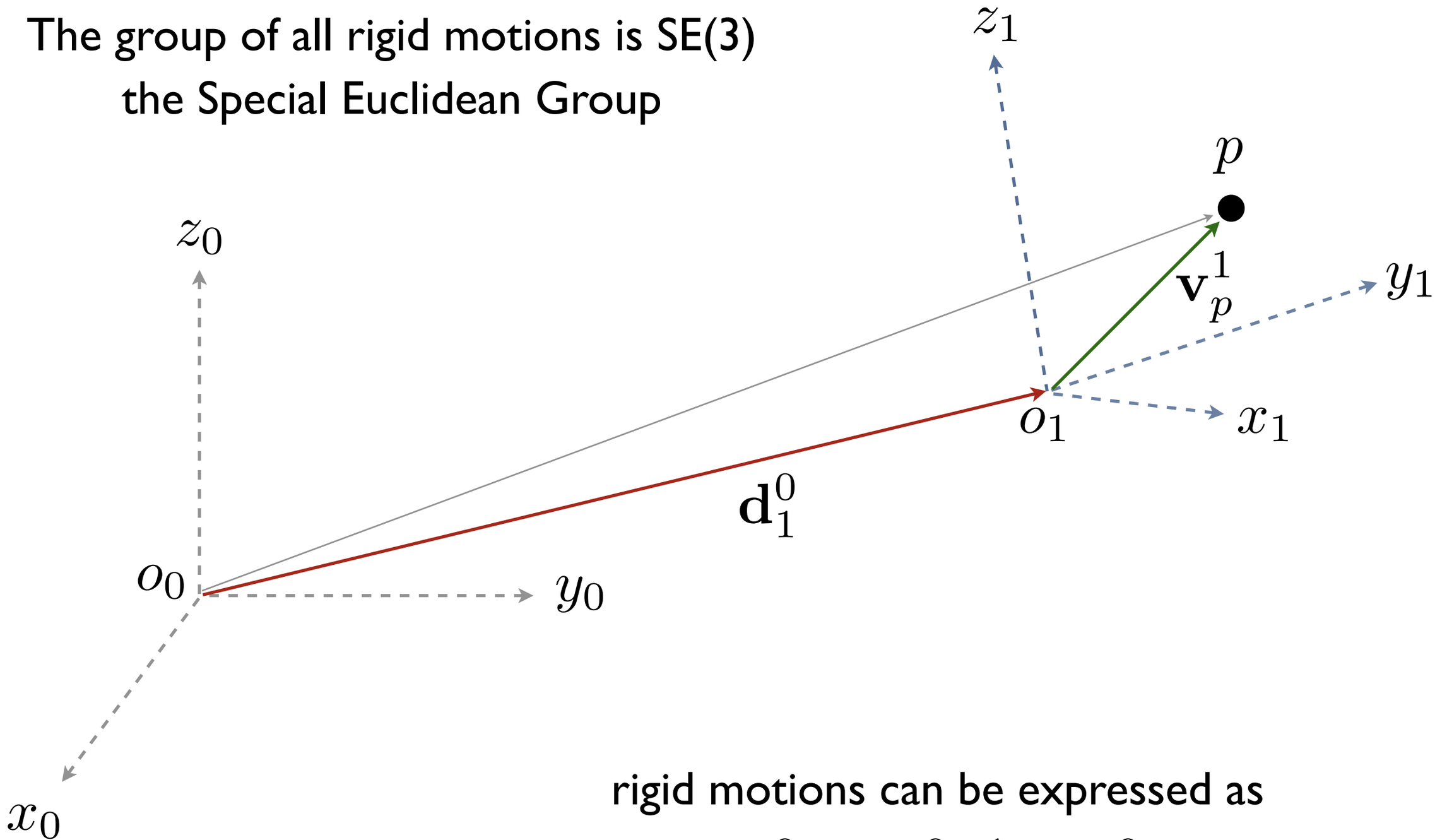
θ_b = -41.3 degrees

ψ_b = 167.0 degrees



a **rigid motion** couples pure translation with pure rotation

The group of all rigid motions is SE(3)
the Special Euclidean Group



rigid motions can be expressed as

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1 + \mathbf{d}_1^0$$

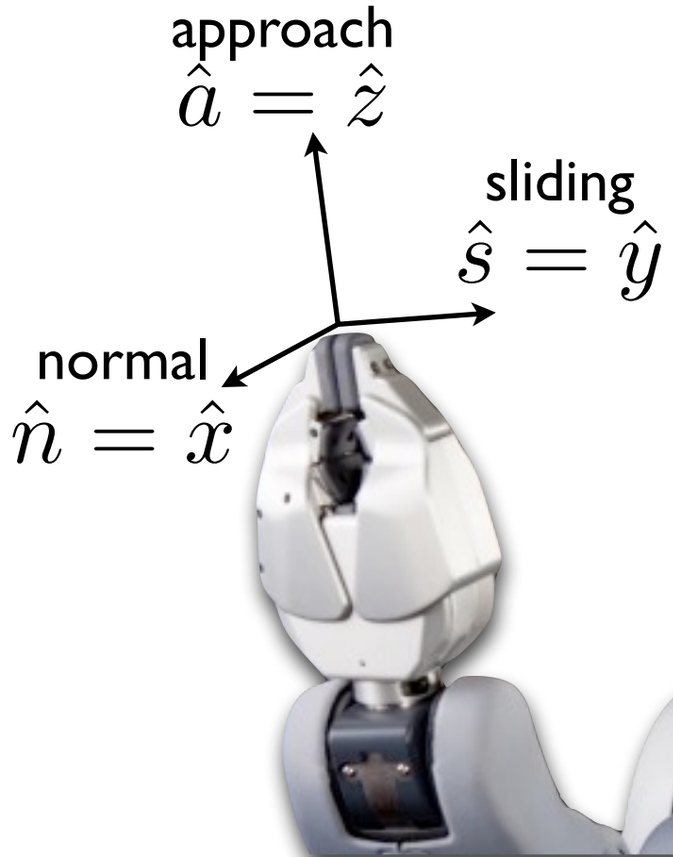
Homogeneous Transformations

a **homogeneous transformation** is a matrix representation of rigid motion, defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R}$ is the 3x3 rotation matrix, and $\mathbf{d}$ is the 3x1 translation vector

$$\mathbf{H} = \begin{bmatrix} \overbrace{\begin{matrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{matrix}}^{\mathbf{R}_1^0} & \overbrace{\begin{matrix} d_x \\ d_y \\ d_z \end{matrix}}^{\mathbf{d}_1^0} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$



the **homogeneous representation** of a vector is formed by concatenating the original vector with a unit scalar

$$\mathbf{P} = \begin{bmatrix} \mathrm{p} \\ 1 \end{bmatrix}$$

where p is the 3x1 vector

$$\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Homogeneous Transformations

rigid body transformations are accomplished by pre-multiplying by the homogenous transform

$$\mathbf{P}^0 = \mathbf{H}_1^0 \, \mathbf{P}^1$$

{example with quadrotor model}

$$\mathbf{v}_p^0 = \ ?$$

$$\mathbf{P}^0 = \mathbf{H}_1^0 \, \mathbf{P}^1$$

$$\mathbf{H}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

z_1

x_1

o_1

p

$\mathbf{v}_p^1$

$\mathbf{d}_1^0$

y_1

z_0

o_0

y_0

x_0

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_q^0 = \ ?$$

z_1

z_2

x_1

y_2

o_1

p

x_2

$\mathbf{v}_p^1$

$\mathbf{v}_q^2$

q

z_0

$\mathbf{d}_1^0$

y_1

o_0

y_0

x_0

composition of multiple transforms is the same as for rotation matrices:

post-multiply when successive rotations are relative to intermediate frames

$$\mathbf{H}_2^0 = \mathbf{H}_1^0 \, \mathbf{H}_2^1$$

pre-multiply when successive rotations are relative to the first fixed frame

$$\mathbf{H}_2^0 = \mathbf{H} \, \mathbf{H}_1^0$$

Composition (intermediate frame)

$$\mathbf{H}_2^0 = \mathbf{H}_1^0\,\mathbf{H}_2^1 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{d}_2^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Inverse Transform

$$\mathbf{H}_0^1 = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_1^0)^\top & -(\mathbf{R}_1^0)^\top\mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Basic Homogeneous Transformations

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad Rot_{z,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MEAM 520

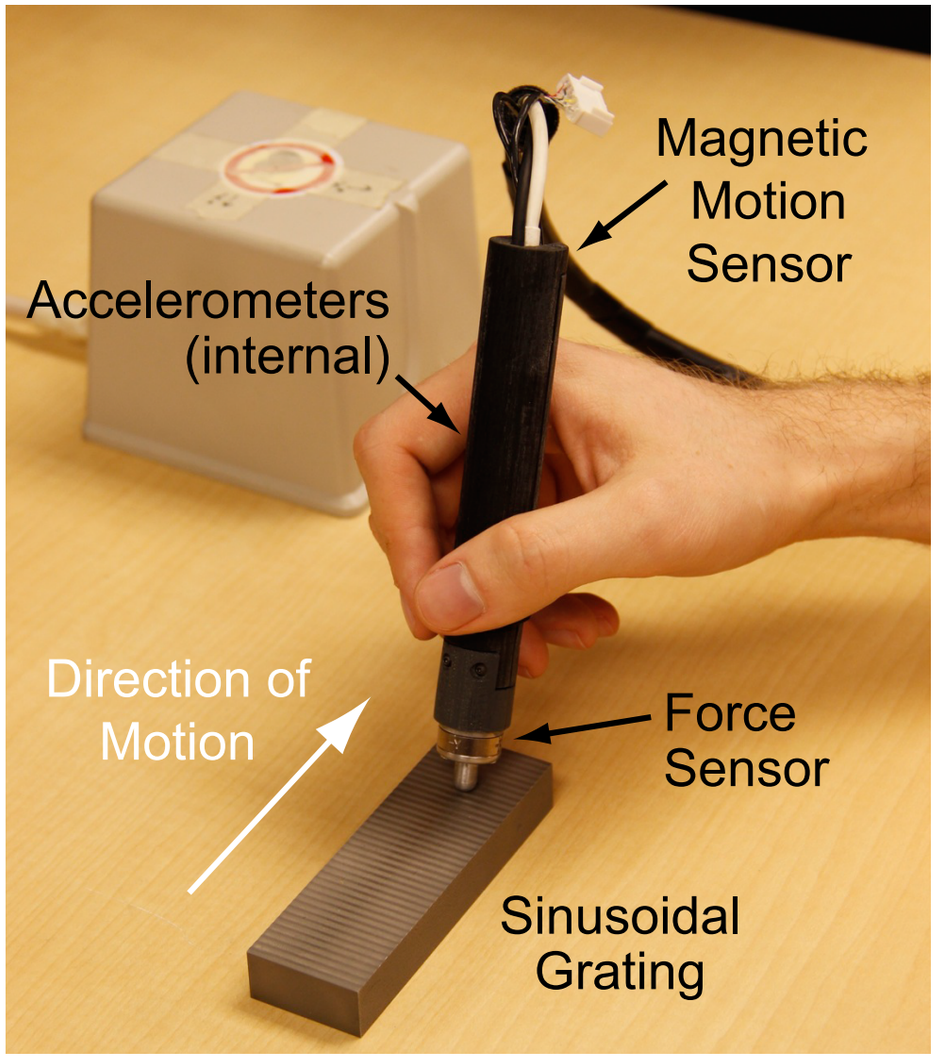
Forward Kinematics and DH Parameters

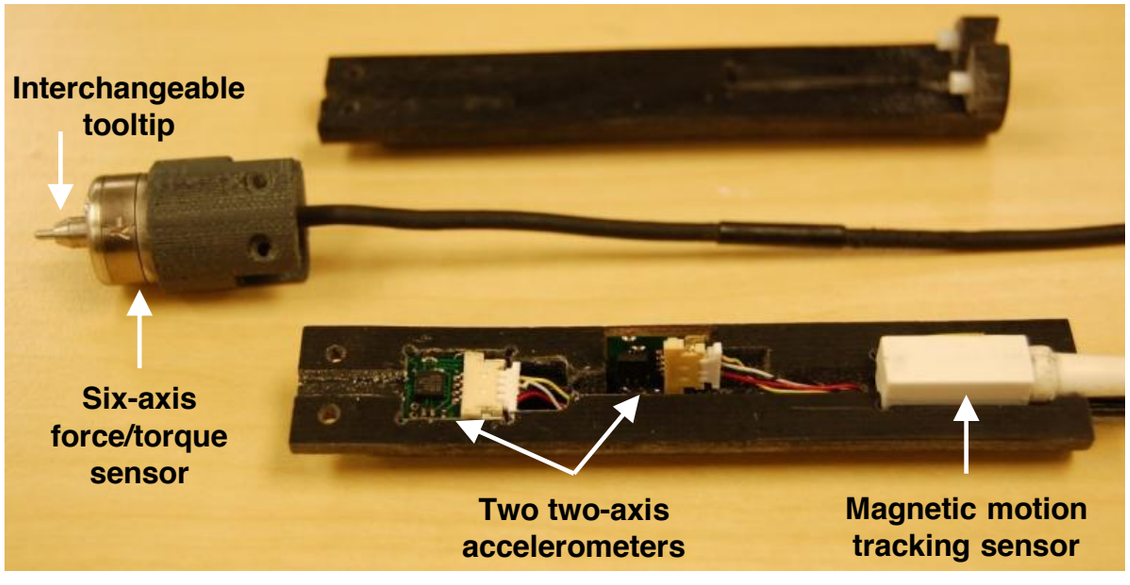
Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 6: September 17, 2013

Rigid Motions and Homogeneous Transformations





I want to calculate the position of the tip of the handheld tool in the fixed frame based on magnetic motion tracking data.

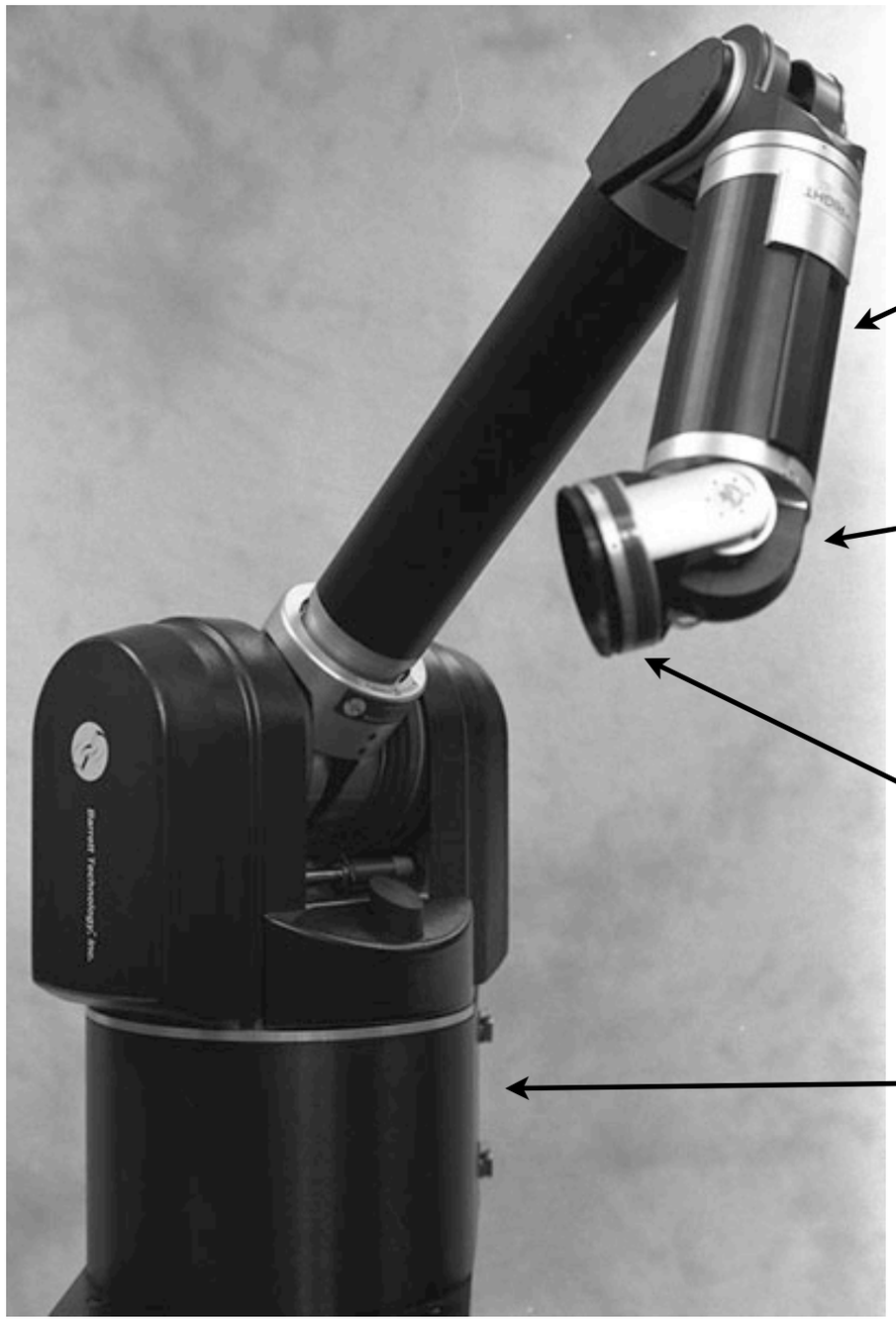
$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1 + \mathbf{d}_1^0$$

$$\mathbf{H} = \begin{bmatrix} \overbrace{\begin{matrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{matrix}}^{\mathbf{R}_1^0} & \overbrace{\begin{matrix} d_x \\ d_y \\ d_z \end{matrix}}^{\mathbf{d}_1^0} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

$$\mathbf{P} = \overbrace{\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}}^{\mathbf{v}_p^1}$$

$$\mathbf{v}_p^0 = \mathbf{H}_1^0 \, \mathbf{v}_p^1$$

General Terminology



Link : rigid body, 6 degrees of freedom

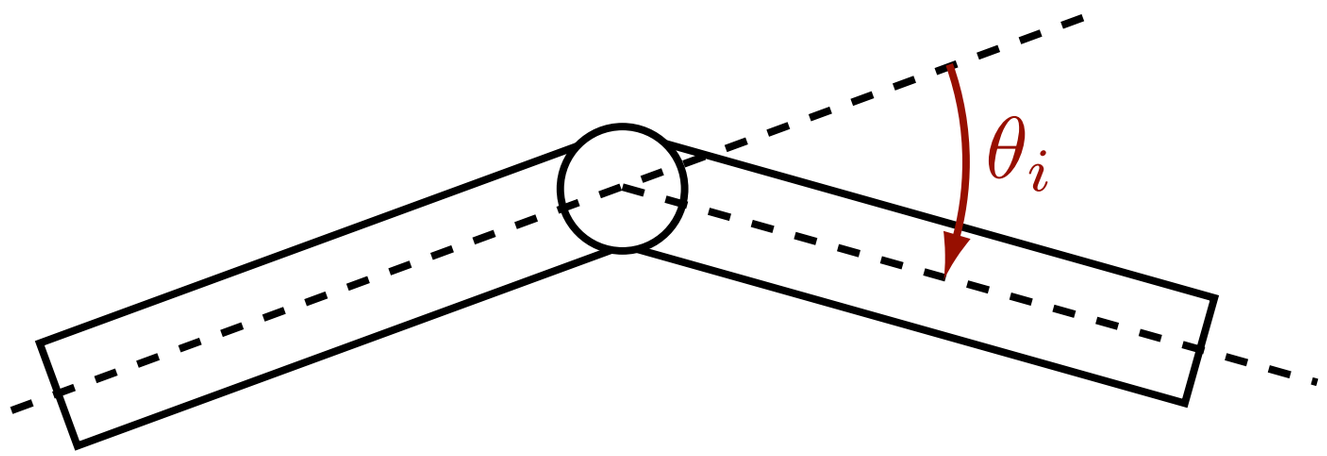
Joint : connection between two links

End-effector : interacts with the environment

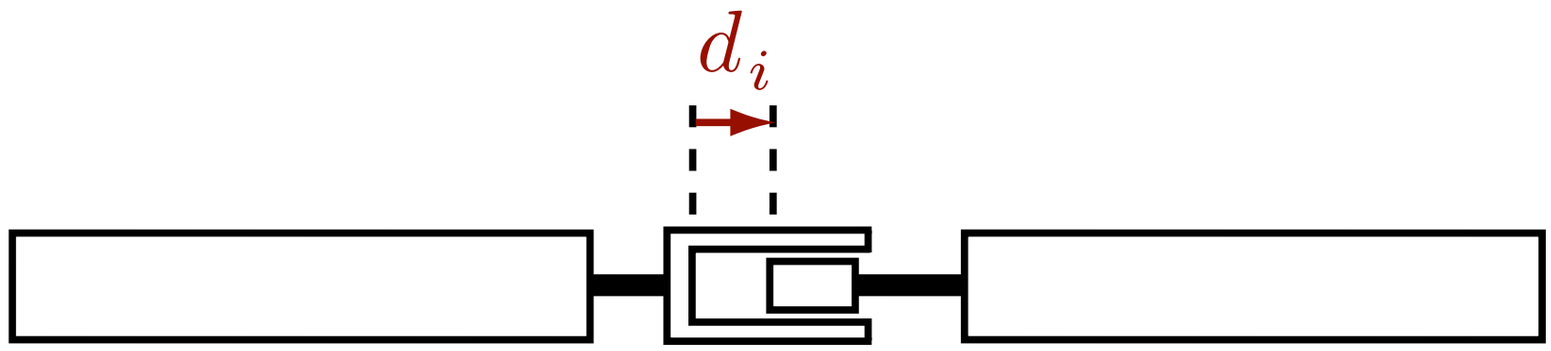
Base : connected to ground

Joint Descriptions

Revolute : angular displacement between adjacent links



Prismatic : linear displacement between adjacent links



We use z-axes to denote joint axes.

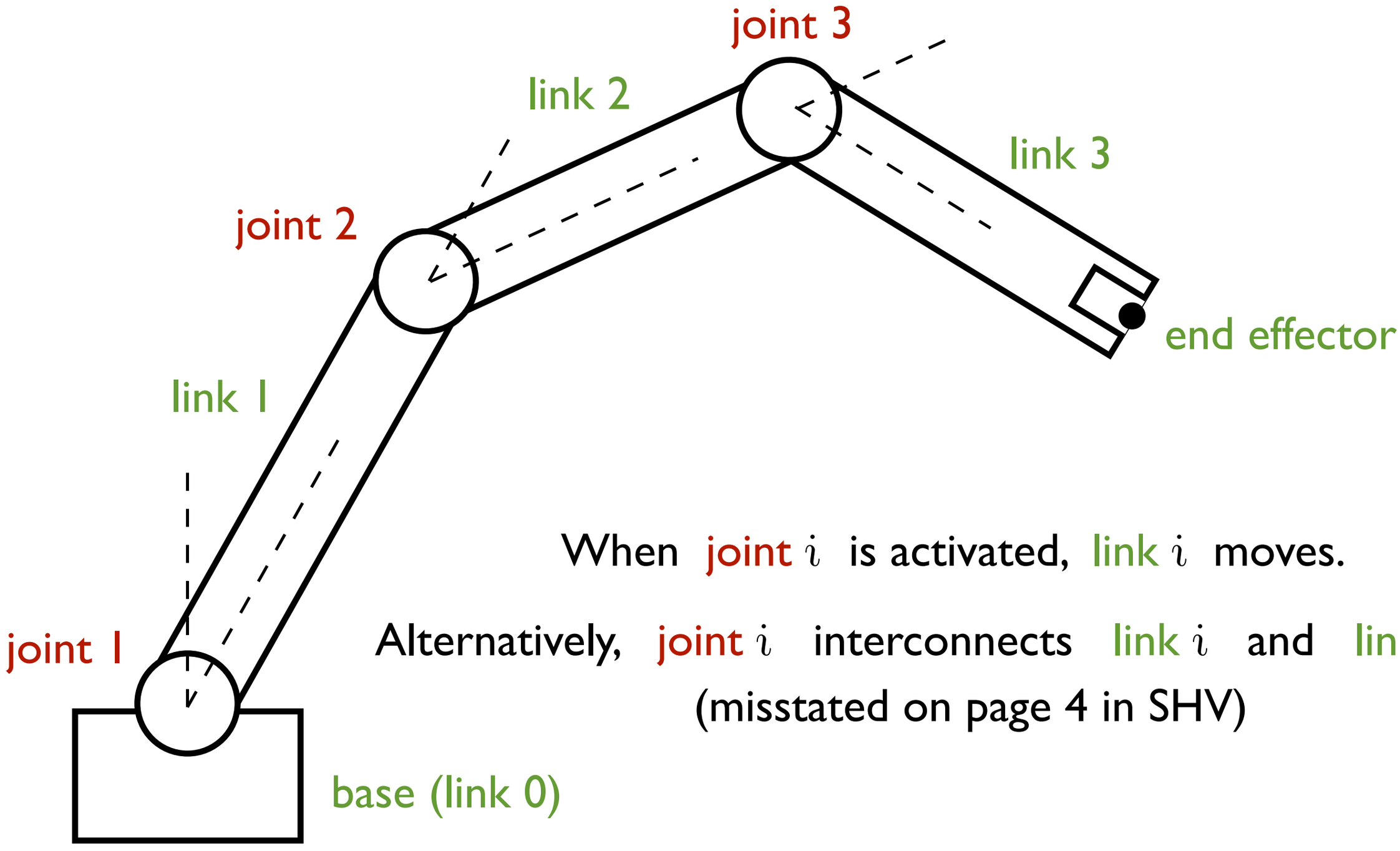
Where are the joints?

R

P

Kinematic Chains

A **kinematic chain** is a system of rigid bodies connected by joints



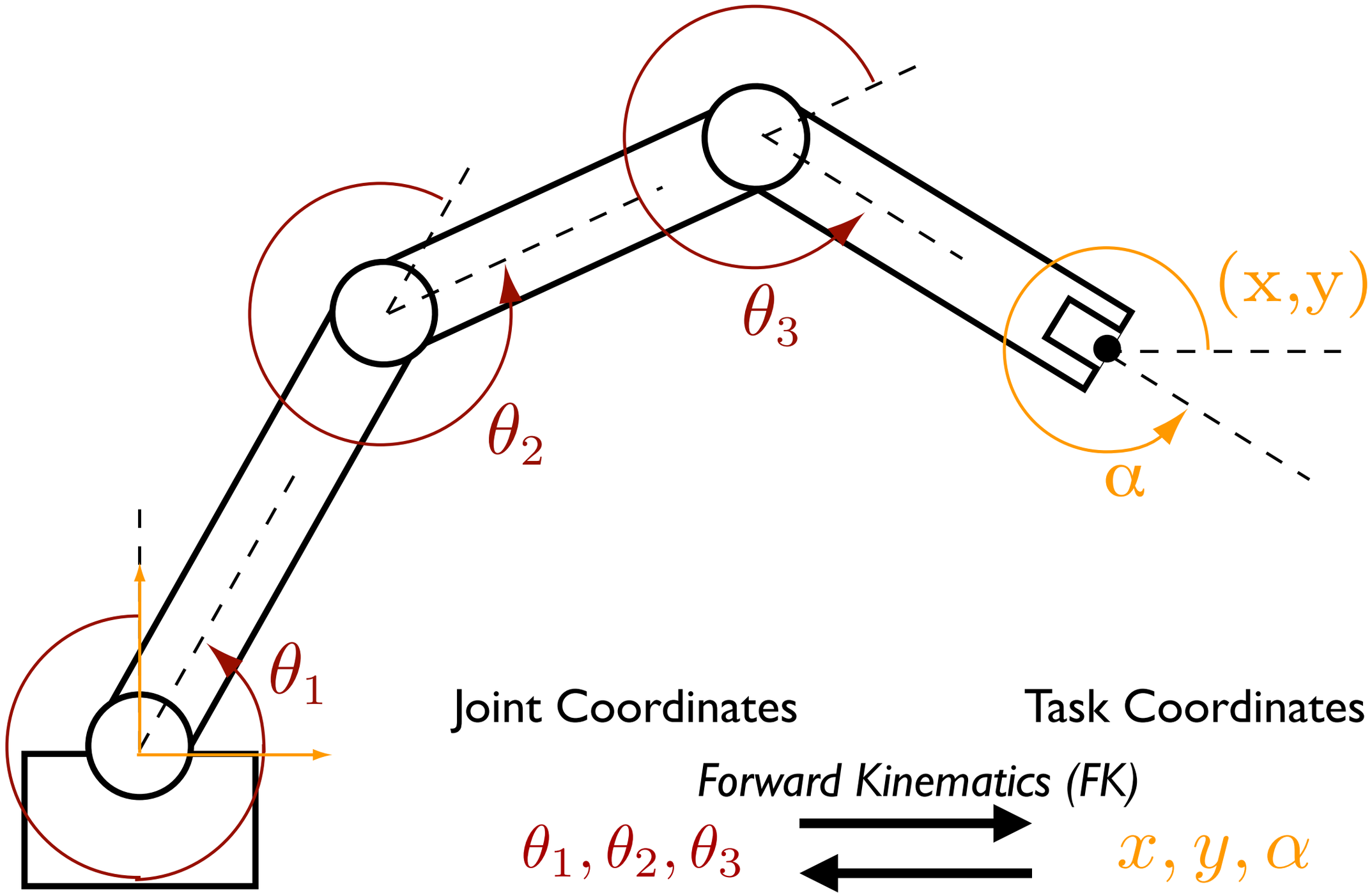
When joint i is activated, link i moves.

Alternatively, joint i interconnects link i and link i-1

(misstated on page 4 in SHV)

In a **serial** kinematic chain, each intermediate link is connected to two others

Joint and Task Coordinates



Joint Coordinates

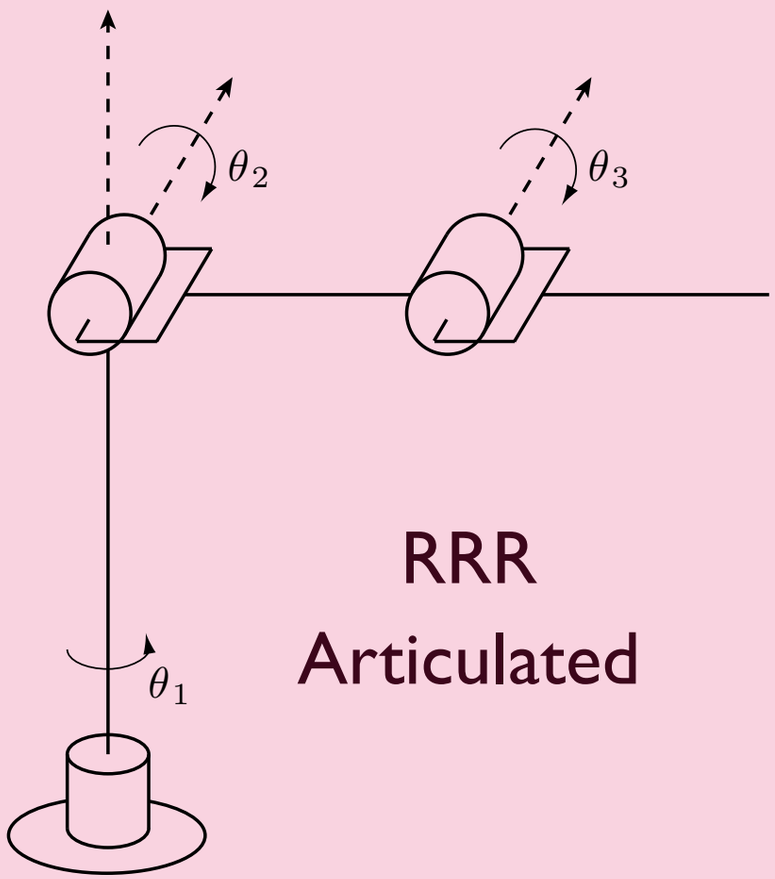
Task Coordinates

Forward Kinematics (FK)

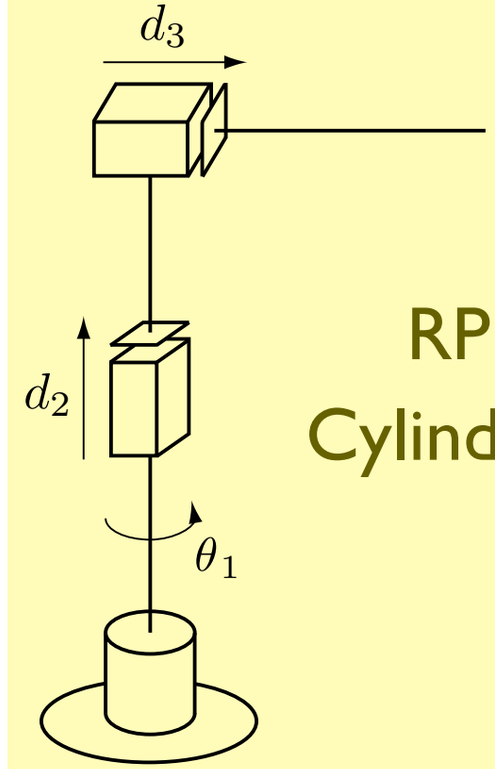
$\theta_1, \theta_2, \theta_3$ ⇄ x, y, α

Inverse Kinematics (IK)

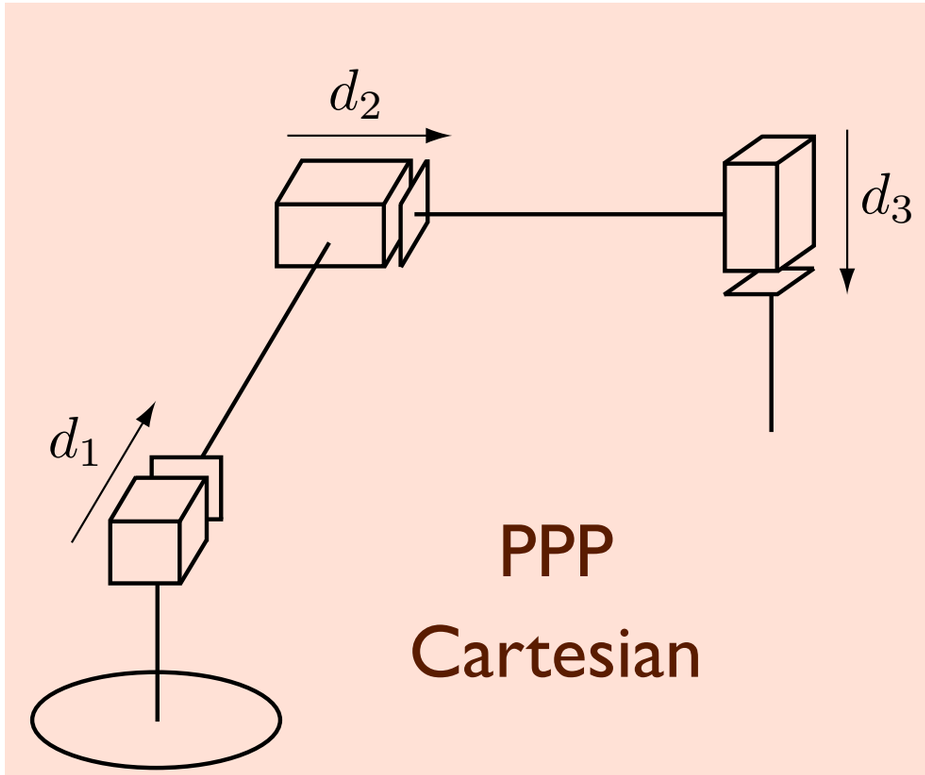
Common Manipulator Designs



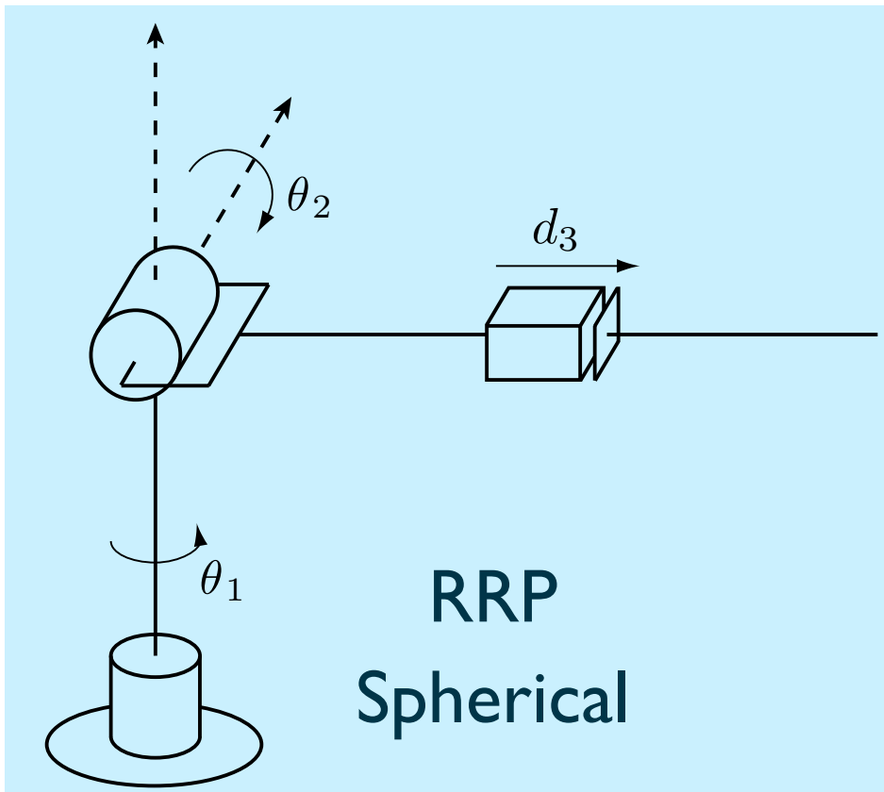
RRR
Articulated



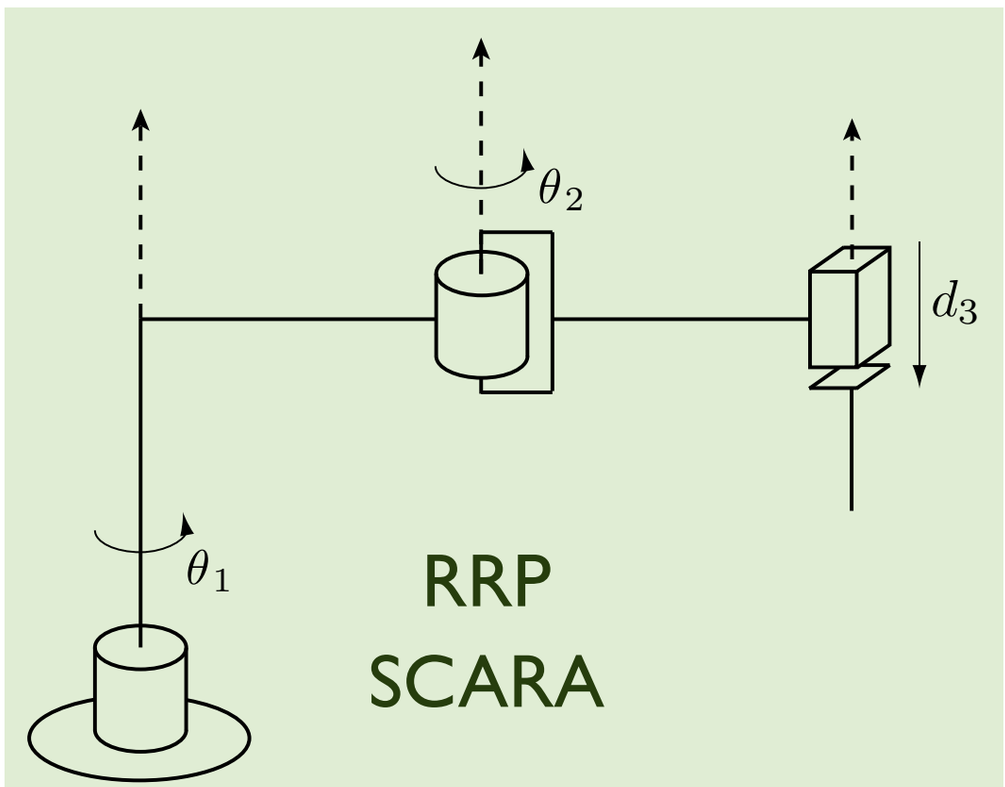
RPP
Cylindrical



PPP
Cartesian

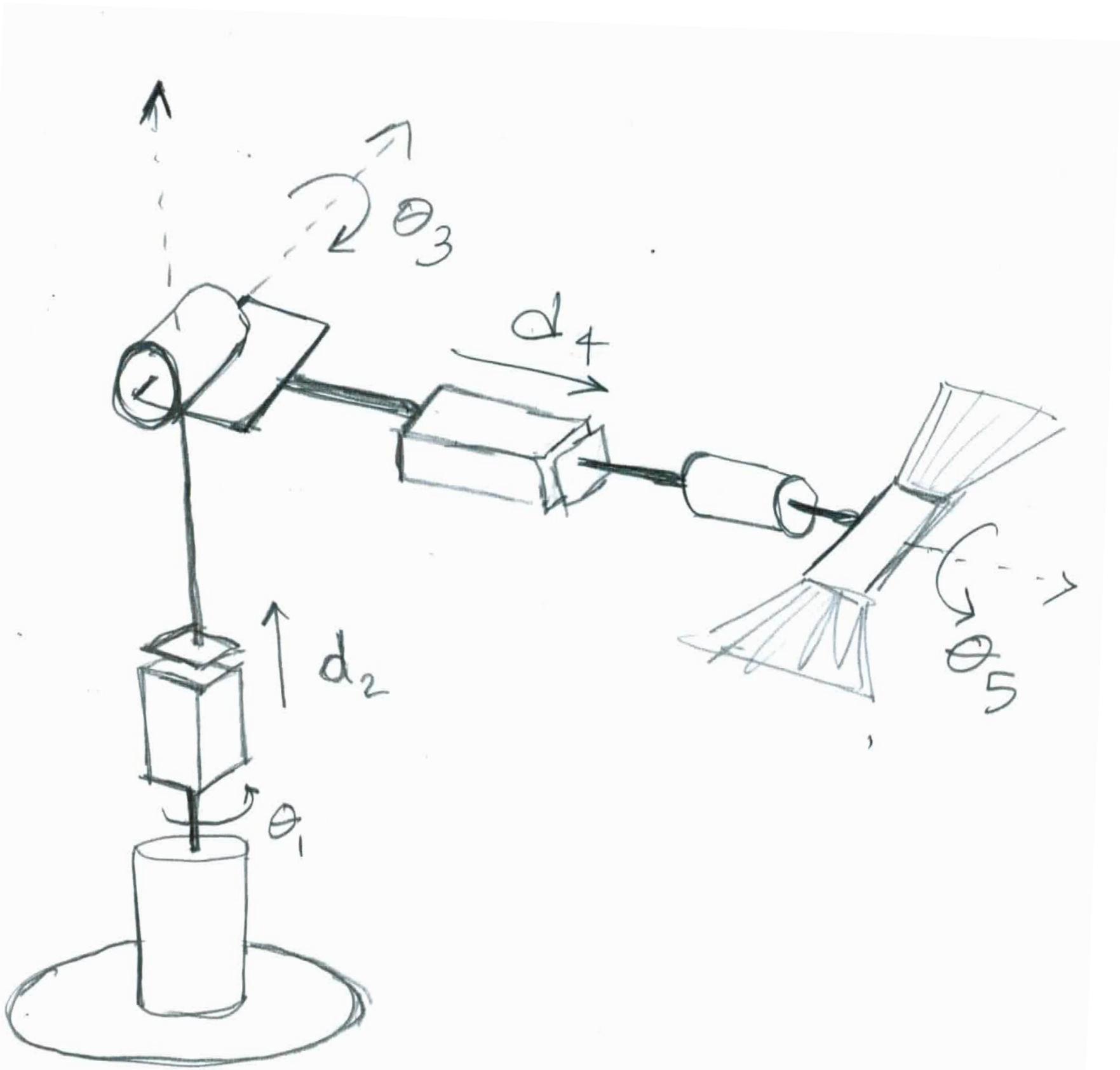


RRP
Spherical



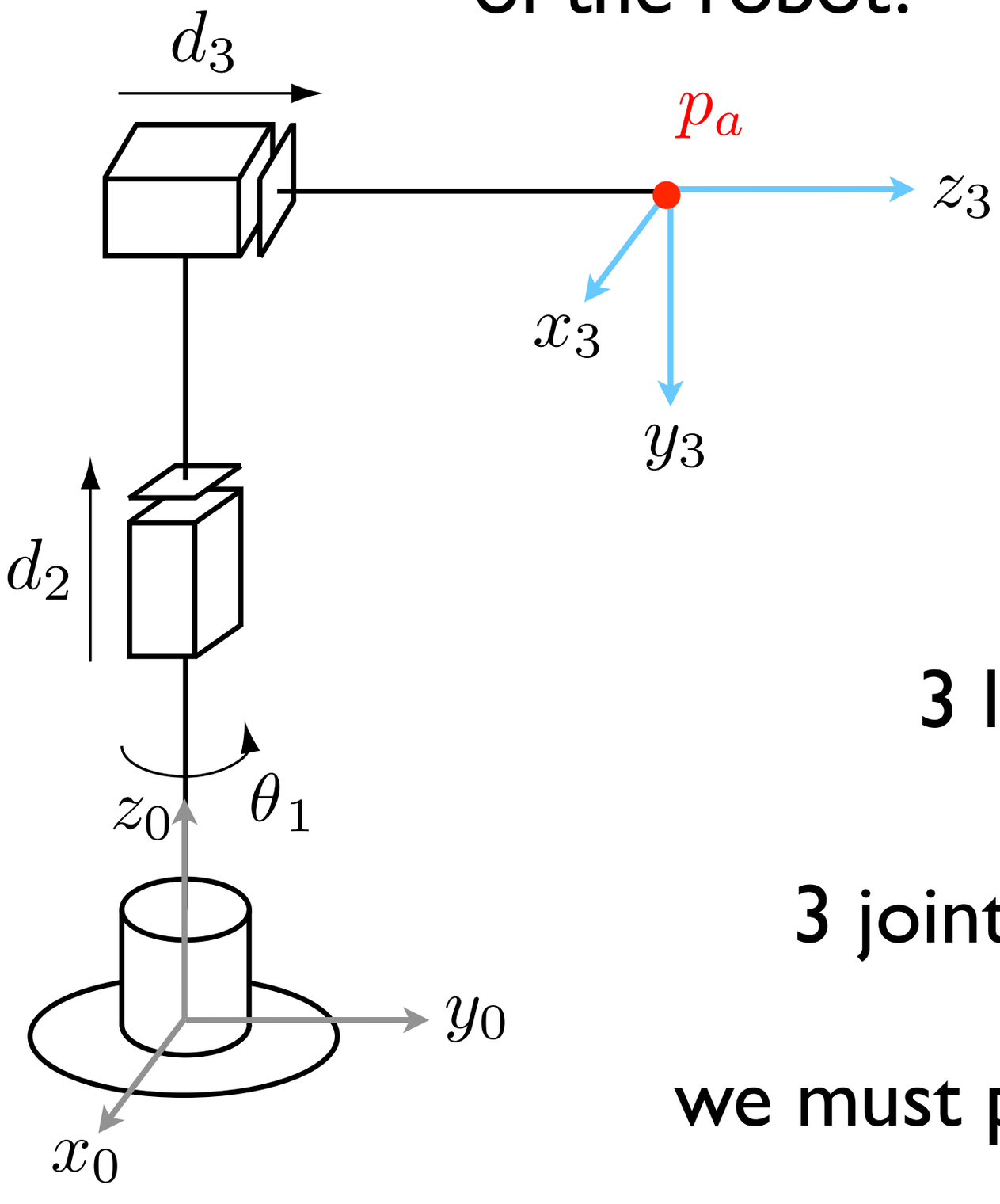
RRP
SCARA

RPRPR



Ignore final (vertical) joint.

Given (q_1, q_2, q_3), where is the tip of the robot?



3 links plus ground

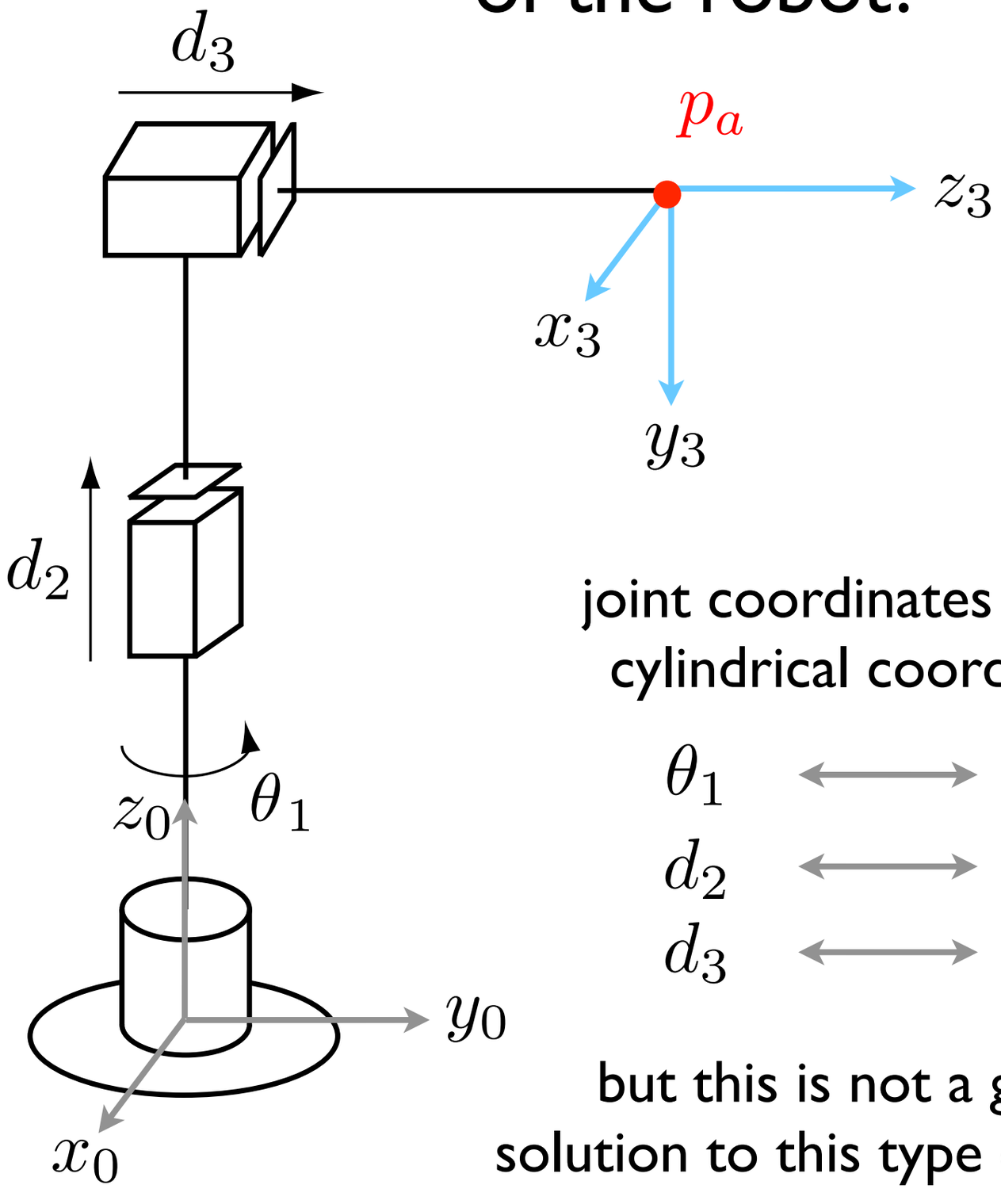
3 joints

3 joint variables (q_1, q_2, q_3)

we must place coordinate frames

Ignore final (vertical) joint.

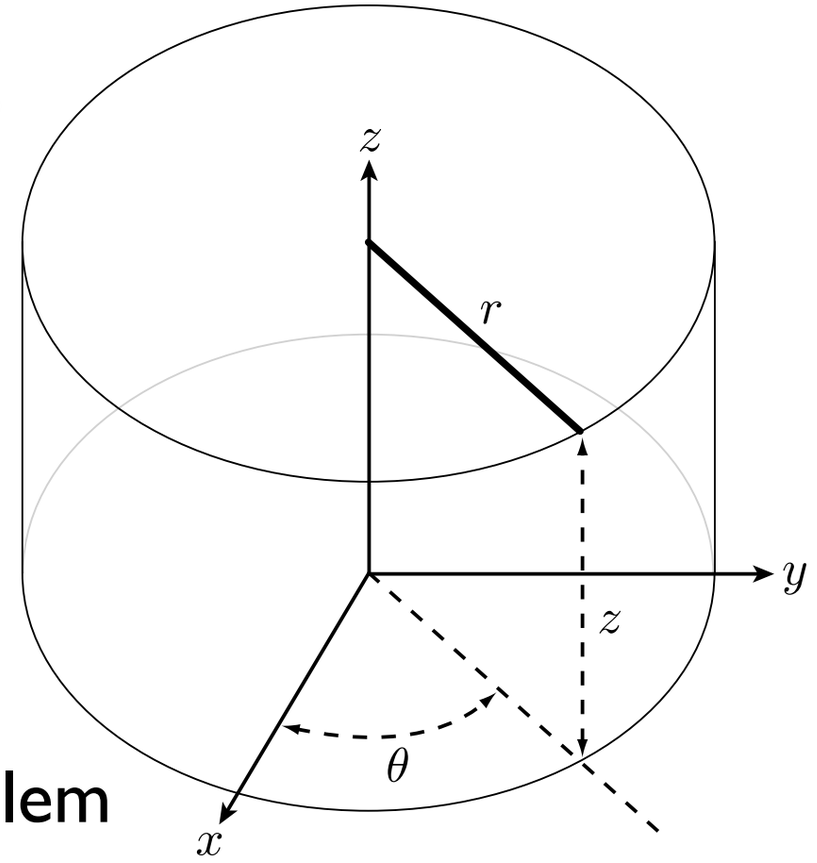
Given (q_1, q_2, q_3), where is the tip of the robot?



joint coordinates map to cylindrical coordinates

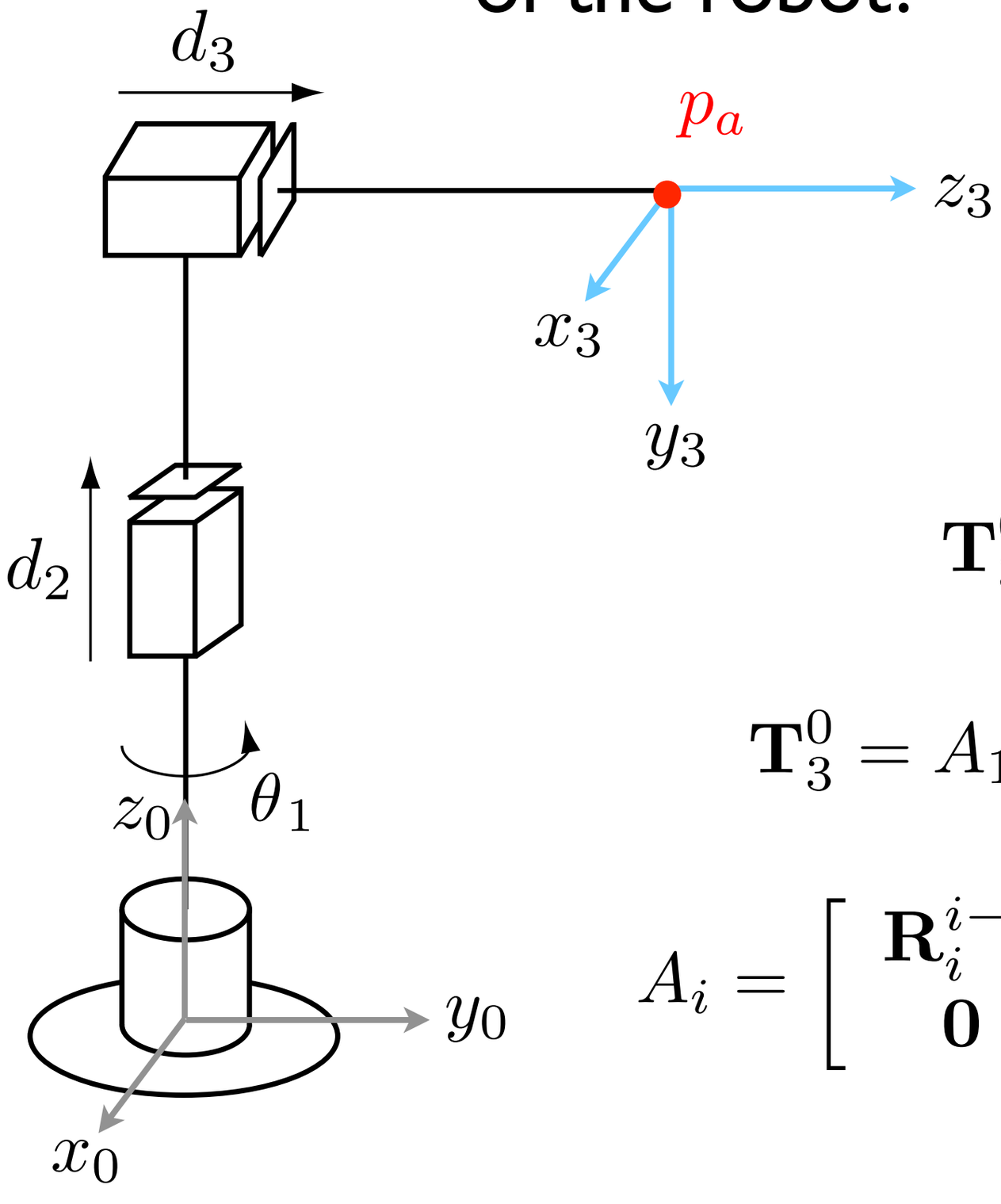
$$\theta_1 \longleftrightarrow \theta$$
$$d_2 \longleftrightarrow z$$
$$d_3 \longleftrightarrow r$$

but this is not a general solution to this type of problem



Ignore final (vertical) joint.

Given (q_1, q_2, q_3), where is the tip of the robot?



$$\mathbf{T}_3^0 = \ ?$$

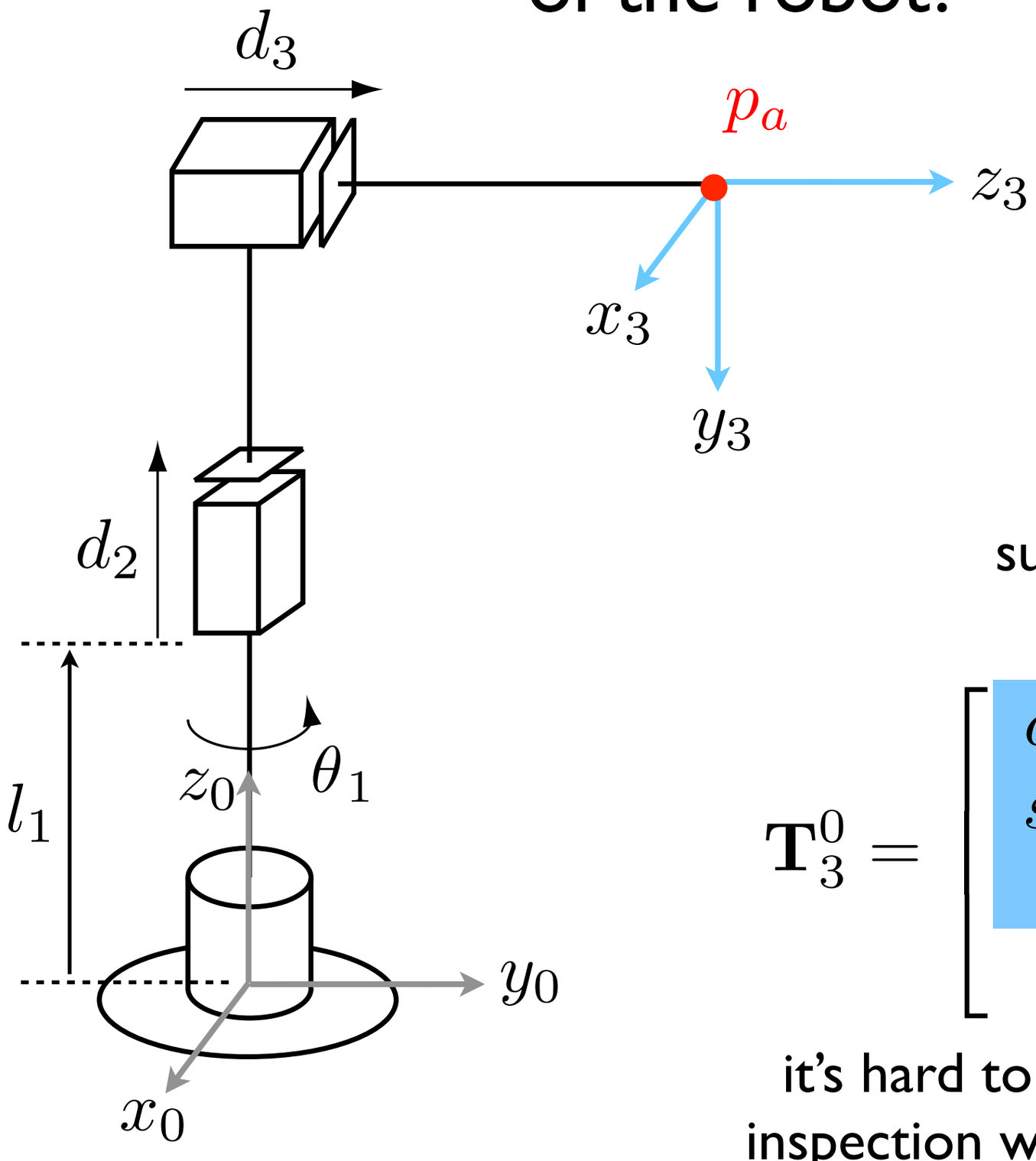
$$\mathbf{T}_3^0 = A_1(q_1)A_2(q_2)A_3(q_3)$$

$$A_i = \begin{bmatrix} \mathbf{R}_i^{i-1} & \mathbf{d}_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{P}_a^0 = \mathbf{T}_3^0 \mathbf{P}_a^3$$

Ignore final (vertical) joint.

Given (q_1, q_2, q_3), where is the tip of the robot?



superscript * marks joint angles, which vary over time

$$\mathbf{T}_3^0 = \begin{bmatrix} c_1^* & 0 & -s_1^* & -d_3^* s_1^* \\ s_1^* & 0 & c_1^* & d_3^* c_1^* \\ 0 & -1 & 0 & d_2^* + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

it's hard to write transformation matrices by inspection when the robot is more complicated

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains

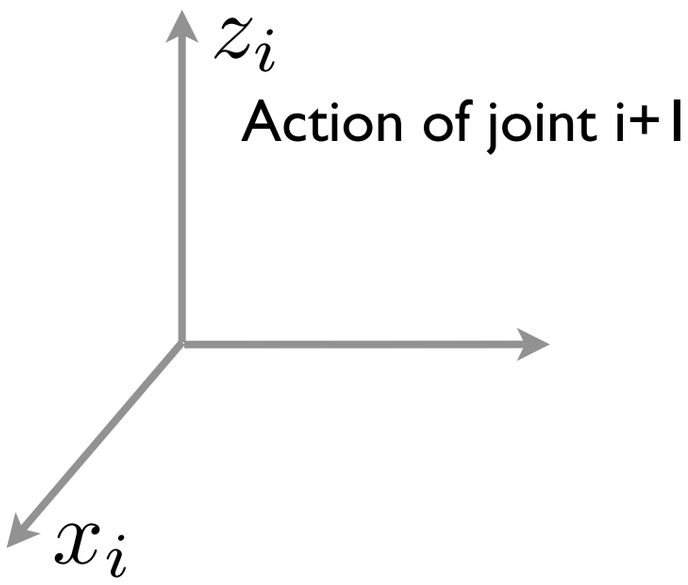
Start by drawing a schematic of the robot in its zero pose.

Then attach one frame to each link:

the joint variable for joint i+1 acts along/around z_i

the orientation of z_i determines the joint angle's positive direction

the axis x_i is perpendicular to, and intersects, z_{i-1}



Must also choose a location for the base (0) frame:

Origin must be on z_0.

x_0 and y_0 are chosen for convenience.

Takes you from previous (i-1) frame to this (i) frame

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains (see page 80 for parameter definitions)

DH parameters are usually written in this order, but I prefer the opposite order.

---

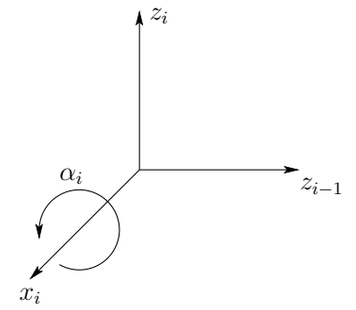
a_i

Link Length the distance between z_{i-1} and z_i, measured along x_i

To be continued....

α_i

Link Twist the angle between z_{i-1} and z_i, measured in the plane normal to x_i (right-hand rule around x_i)



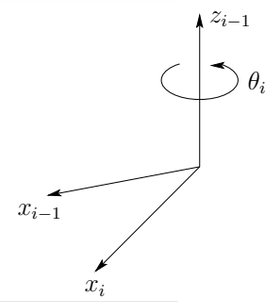
---

d_i

Link Offset the distance between x_{i-1} and x_i, measured along z_{i-1}

θ_i

Joint Angle the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1} (right-hand rule around z_{i-1})



Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

MEAM 520

More DH Parameters

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 7: September 19, 2013

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains

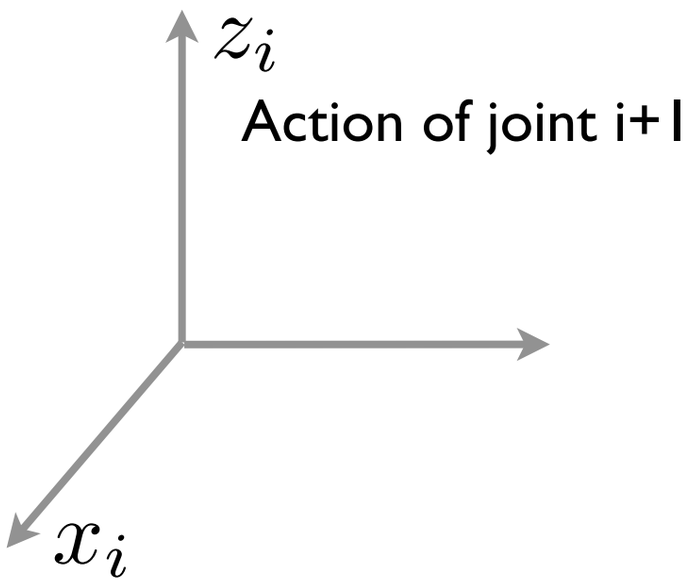
Start by drawing a schematic of the robot in its zero pose.

Then attach one frame to each link:

the joint variable for joint i+1 acts along/around z_i

the orientation of z_i determines the joint angle's positive direction

the axis x_i is perpendicular to, and intersects, z_{i-1}



Takes you from previous (i-1) frame to this (i) frame

Must also choose a location for the base (0) frame:

Origin must be on z_0.

x_0 and y_0 are chosen for convenience.

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains (see page 110-111 for parameter definitions)

DH parameters are usually written in this order, but I prefer the opposite order.

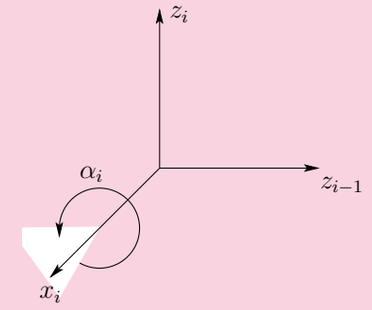
x step

a_i

Link Length the distance between z_{i-1} and z_i, measured along x_i

α_i

Link Twist the angle between z_{i-1} and z_i, measured in the plane normal to x_i (right-hand rule around x_i)



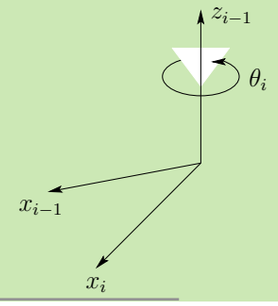
z step

d_i

Link Offset the distance between x_{i-1} and x_i, measured along z_{i-1}

θ_i

Joint Angle the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1} (right-hand rule around z_{i-1})



In what order are the transformations applied?

Intermediate frames
Post-multiply

$$A_i = Rot_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate around z by theta

Translate along z by d

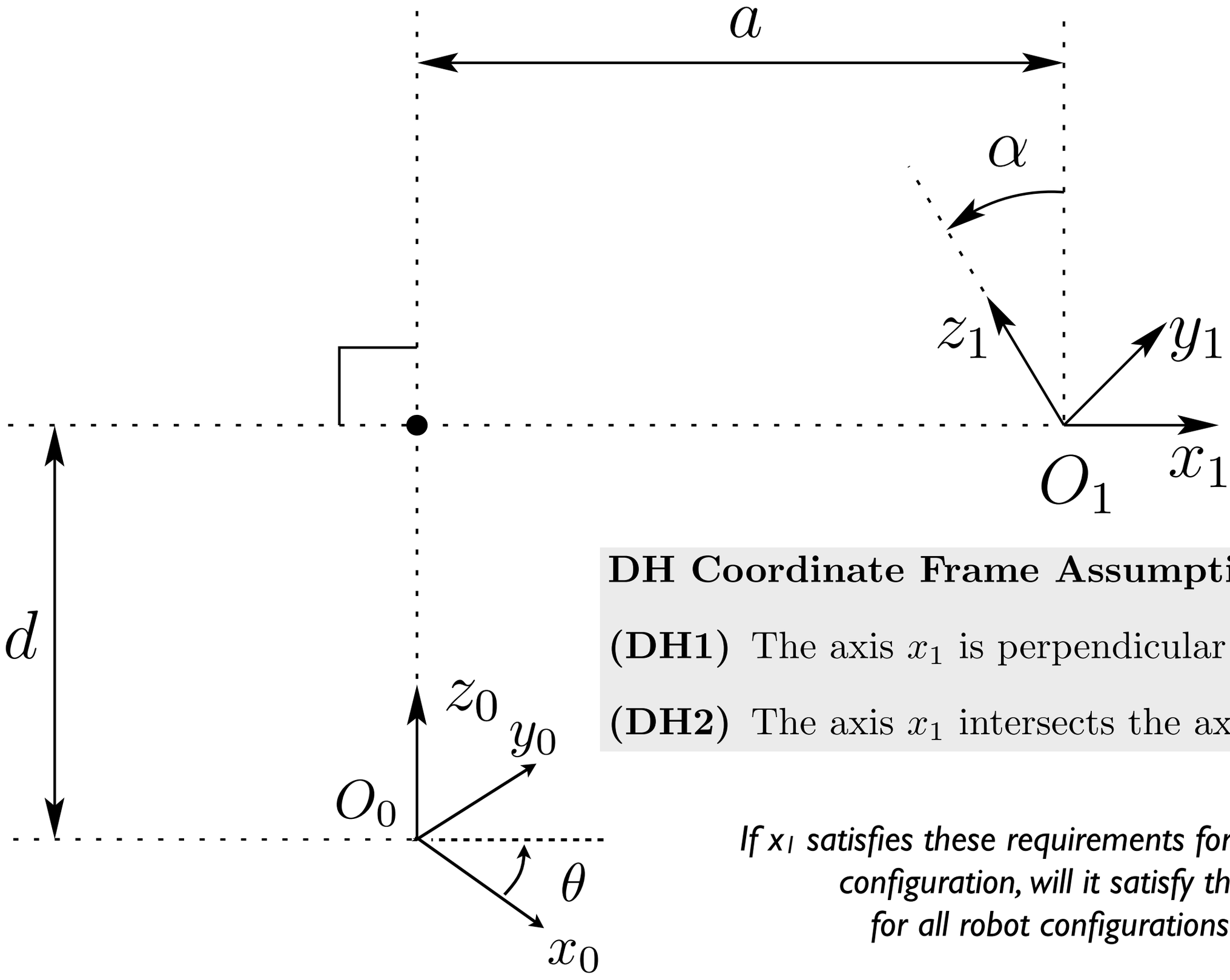
Translate along new x by a

Rotate around new x by alpha

Can you represent any transformation in this way?

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

DH parameters can be used to represent the transformation between any two rigid bodies providing the frames are chosen following the convention.



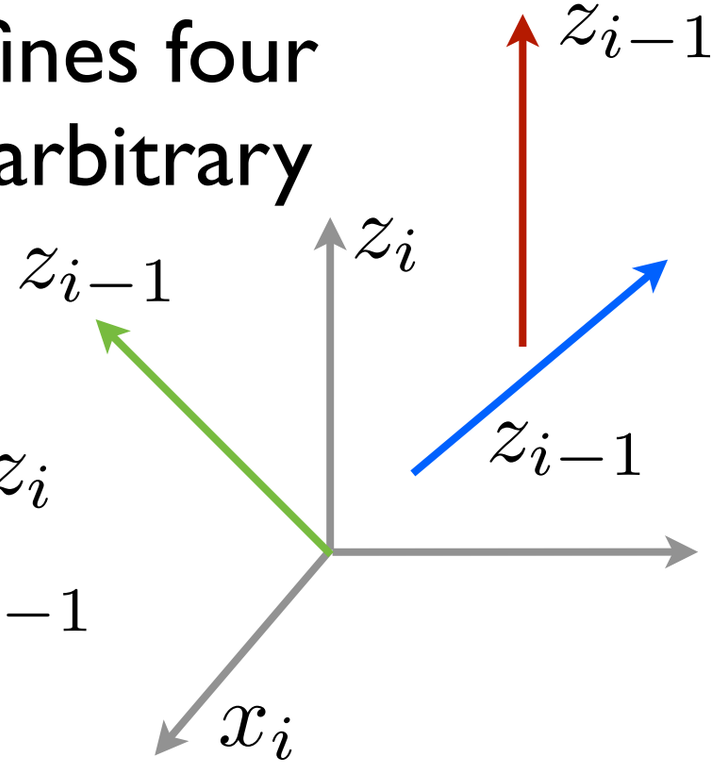
DH Coordinate Frame Assumptions

(DH1) The axis x_1 is perpendicular to the axis z_0.

(DH2) The axis x_1 intersects the axis z_0.

If x_1 satisfies these requirements for one robot configuration, will it satisfy them for all robot configurations?

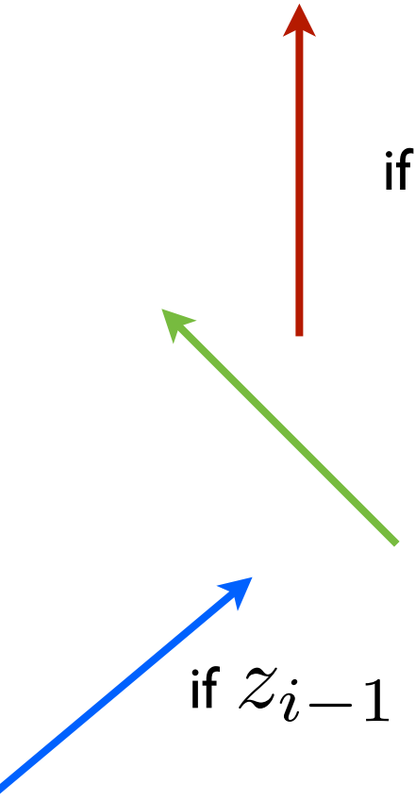
The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains



the joint variable for joint i+1 acts along/around z_i

the axis x_i is perpendicular to, and intersects, z_{i-1}

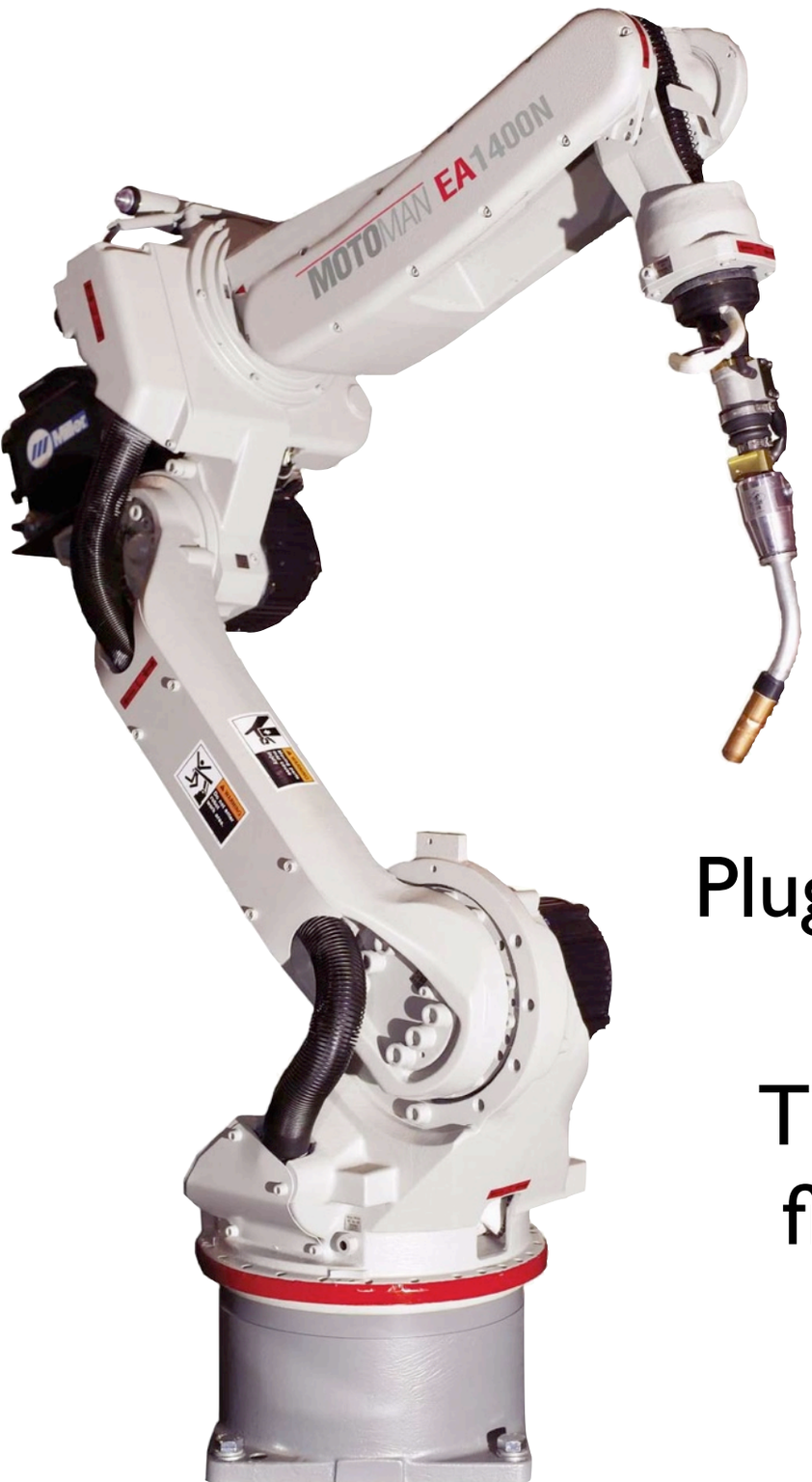
the following conventions make x placement easier (p. 82 in SHV):



if z_{i-1} is parallel to z_i	orient x_i toward z_i
if z_{i-1} intersects z_i	orient x_i normal to the plane formed by z_{i-1} and z_i
if z_{i-1} is not coplanar with z_i	orient x_i along normal with z_{i-1}, toward z_i

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

The **Denavit-Hartenberg transform** results from successive rotations and translations via the four DH parameters.



The transform from i-1 to i:

$$A_i = \text{Rot}_{z,\theta_i}\ \text{Trans}_{z,d_i}\ \text{Trans}_{x,a_i}\ \text{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Plug your DH parameters into the above formula to find each joint's transformation matrix.

Then multiply all matrices together to get the final transformation matrix from tip to base.

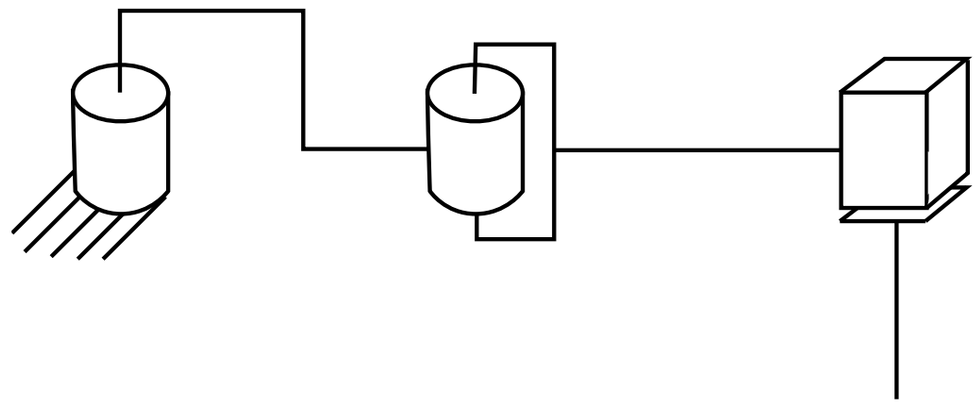
$$\mathbf{T}_n^0 = A_1(q_1)\cdots A_n(q_n)$$

$$H = \mathrm{Rot}_{x,\alpha}\,\mathrm{Trans}_{x,b}\,\mathrm{Trans}_{z,d}\,\mathrm{Rot}_{z,\theta}$$

Which pairs of matrices commute?

What does this expression remind you of?

DH Parameters for the SCARA Robot



Link	a_i	α_i	d_i	θ_i

Great List of DH Steps on page 110-111 in SHV

Procedure for Deriving the Forward Kinematics of a Manipulator
Following the Denavit-Hartenberg (DH) Convention
From "Robot Modeling and Control" by Spong, Hutchinson, and Vidyasagar

Step l: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$, perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n-th joint is revolute, set $z_n = a$ along the direction z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

Step 7: Create a table of link parameters a_i, d_i, α_i, θ_i.

(fixed!)

$a_i =$ distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i

$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ the angle between z_{i-1} and z_i measured about x_i.

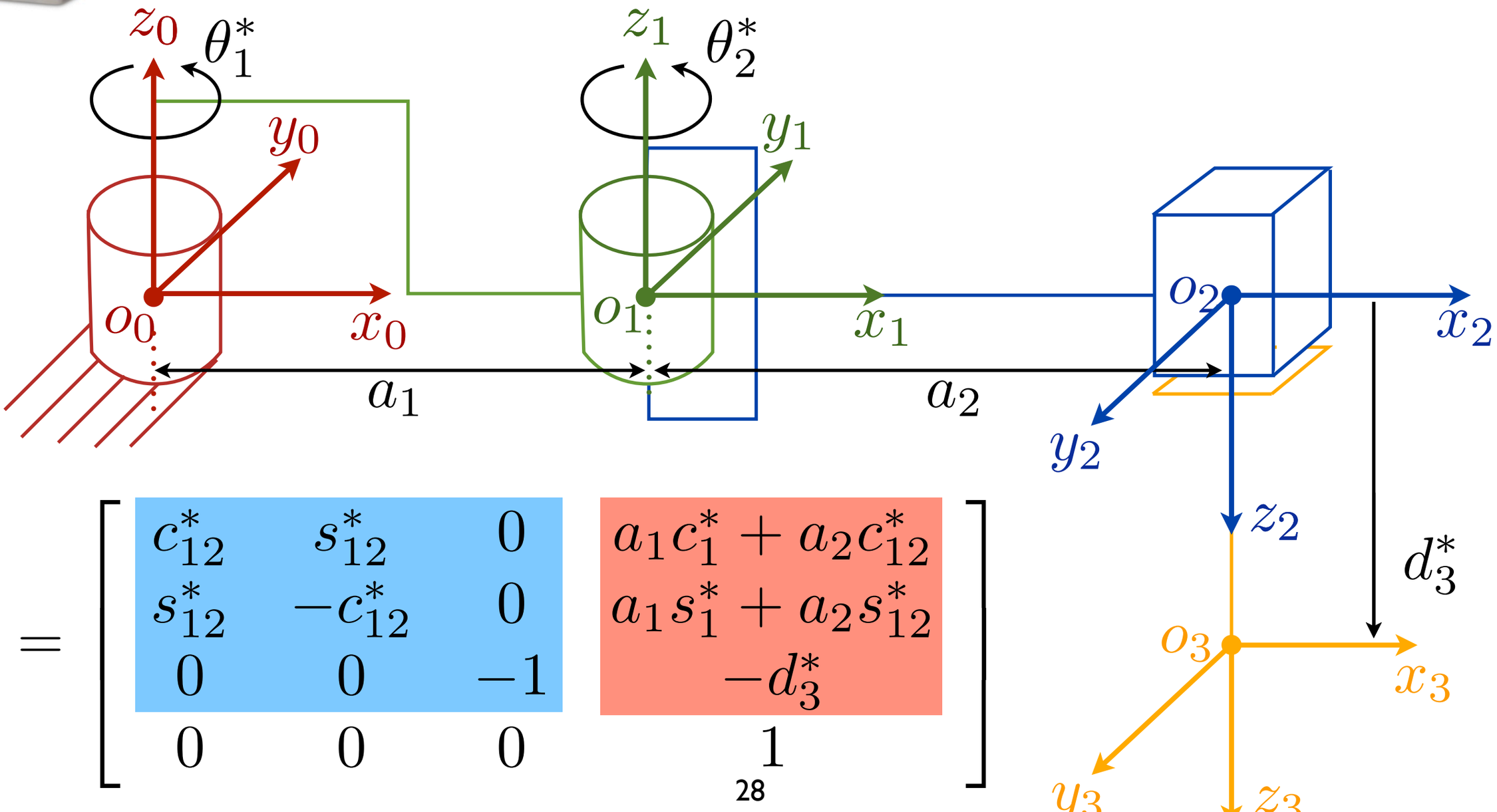
$\theta_i =$ the angle between x_{i-1} and x_i measured about z_{i-1}. θ_i is variable if joint i is revolute.

Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

$$T_3^0 = A_1\, A_2\, A_3$$



$$T_3^0 = \begin{bmatrix} c_{12}^* & s_{12}^* & 0 & a_1 c_1^* + a_2 c_{12}^* \\ s_{12}^* & -c_{12}^* & 0 & a_1 s_1^* + a_2 s_{12}^* \\ 0 & 0 & -1 & -d_3^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MEAM 520

DH Parameter Examples

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

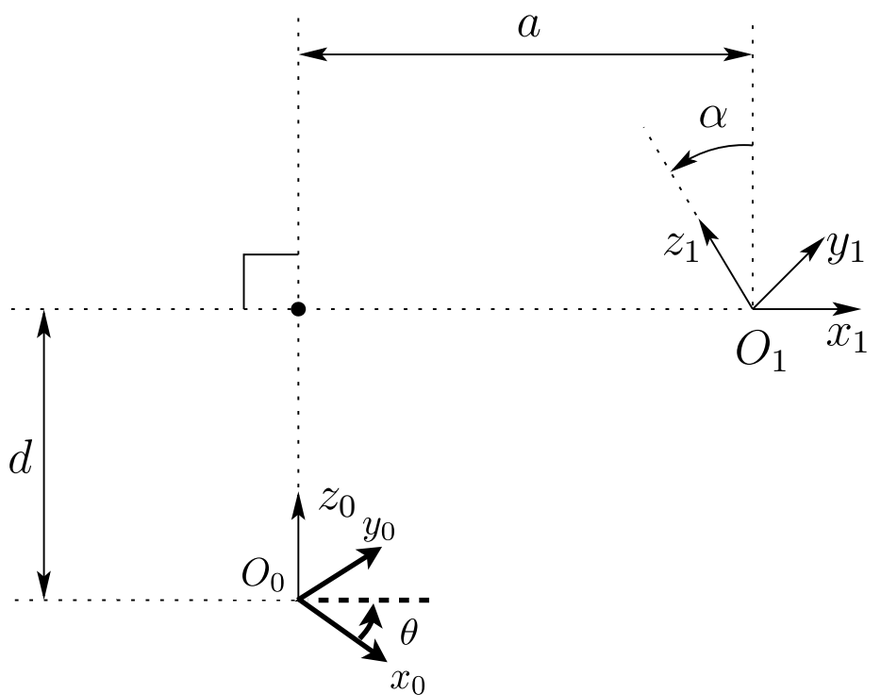
GRASP LABORATORY

Lecture 8: September 24, 2013

The Denavit-Hartenberg (DH) Convention for Manipulator Forward Kinematics

Given the design of my robot and the current values for its joint variables, where is the end-effector, and how is it oriented?

$$\mathbf{T}_n^0 = A_1(q_1) \cdots A_n(q_n)$$



$$A_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,\mathrm{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

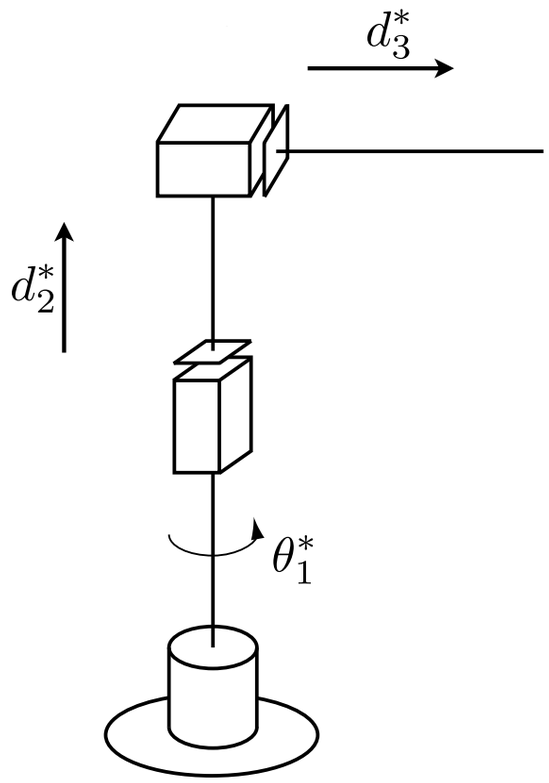
Link	a_i	α_i	d_i	θ_i

DH Coordinate Frame Assumptions

(DH1) The axis x_1 is perpendicular to the axis z_0.

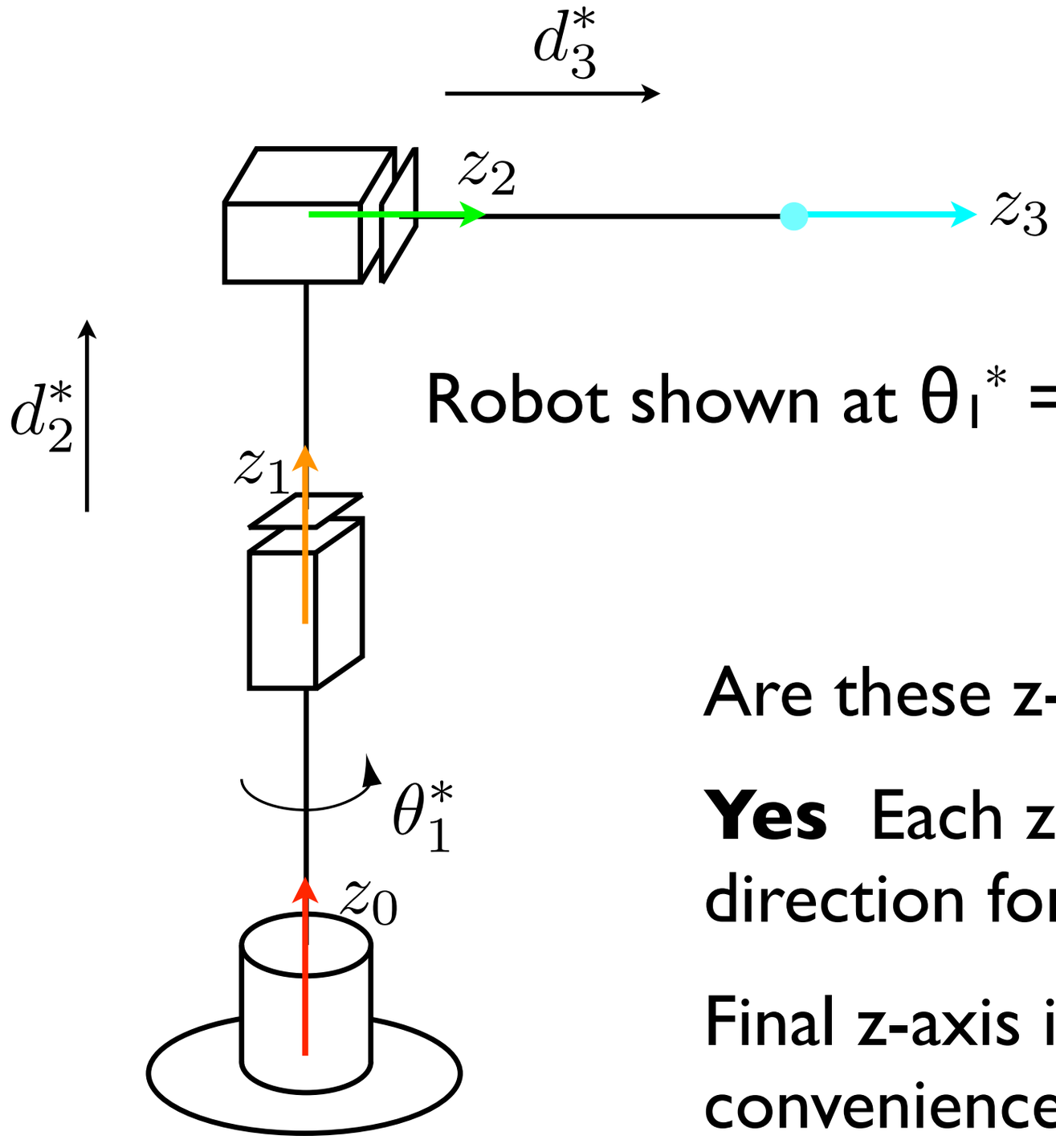
(DH2) The axis x_1 intersects the axis z_0.

DH Parameters for the Cylindrical Robot



Link	a_i	α_i	d_i	θ_i

The RPP Cylindrical Robot



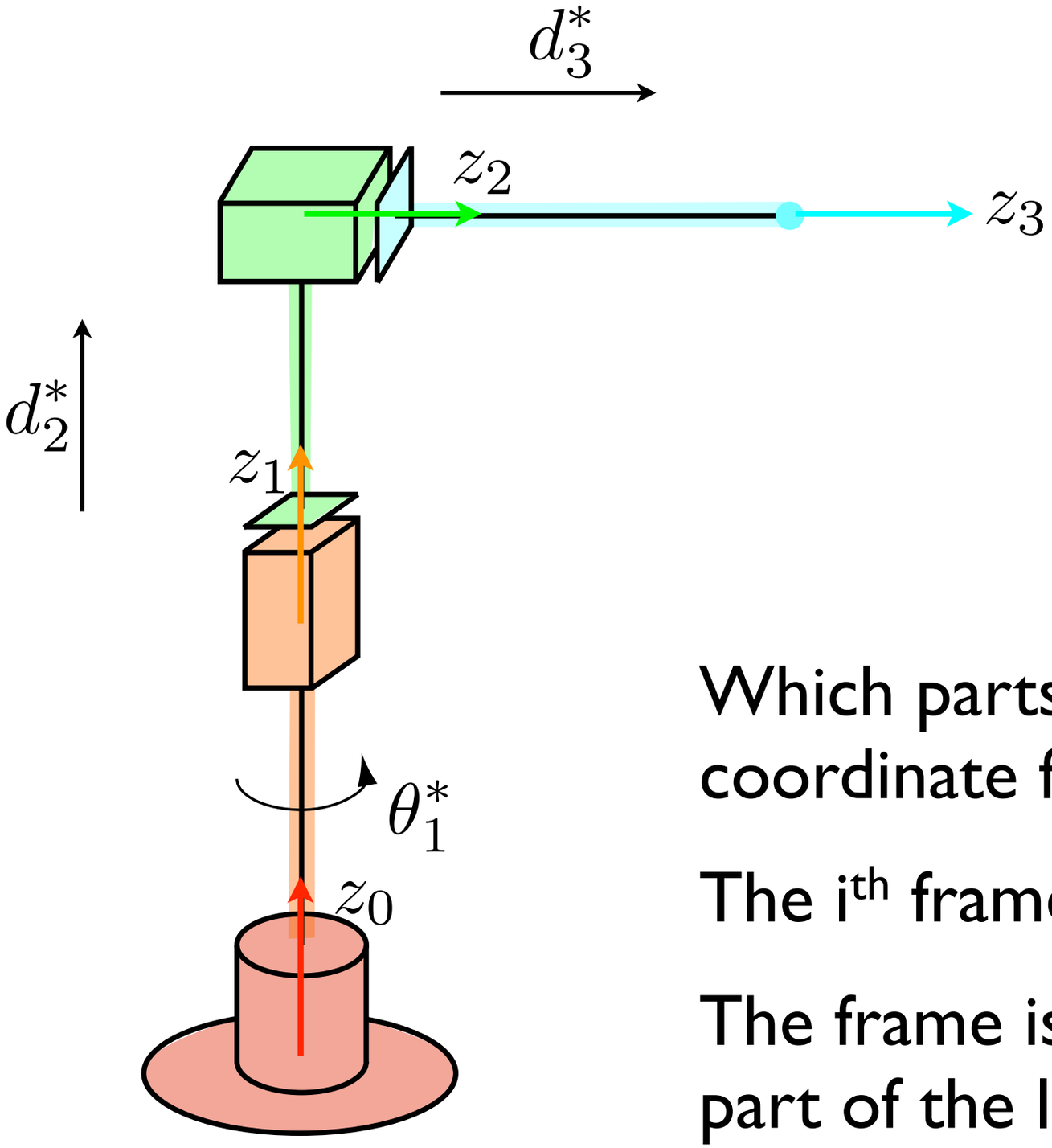
Robot shown at $\theta_1^* = 0$, $d_2^* > 0$ $d_3^* > 0$

Are these z-axis locations correct?

Yes Each z-axis defines the positive axial direction for the following joint variable.

Final z-axis is parallel to previous one for convenience. Could have chosen other.

The RPP Cylindrical Robot

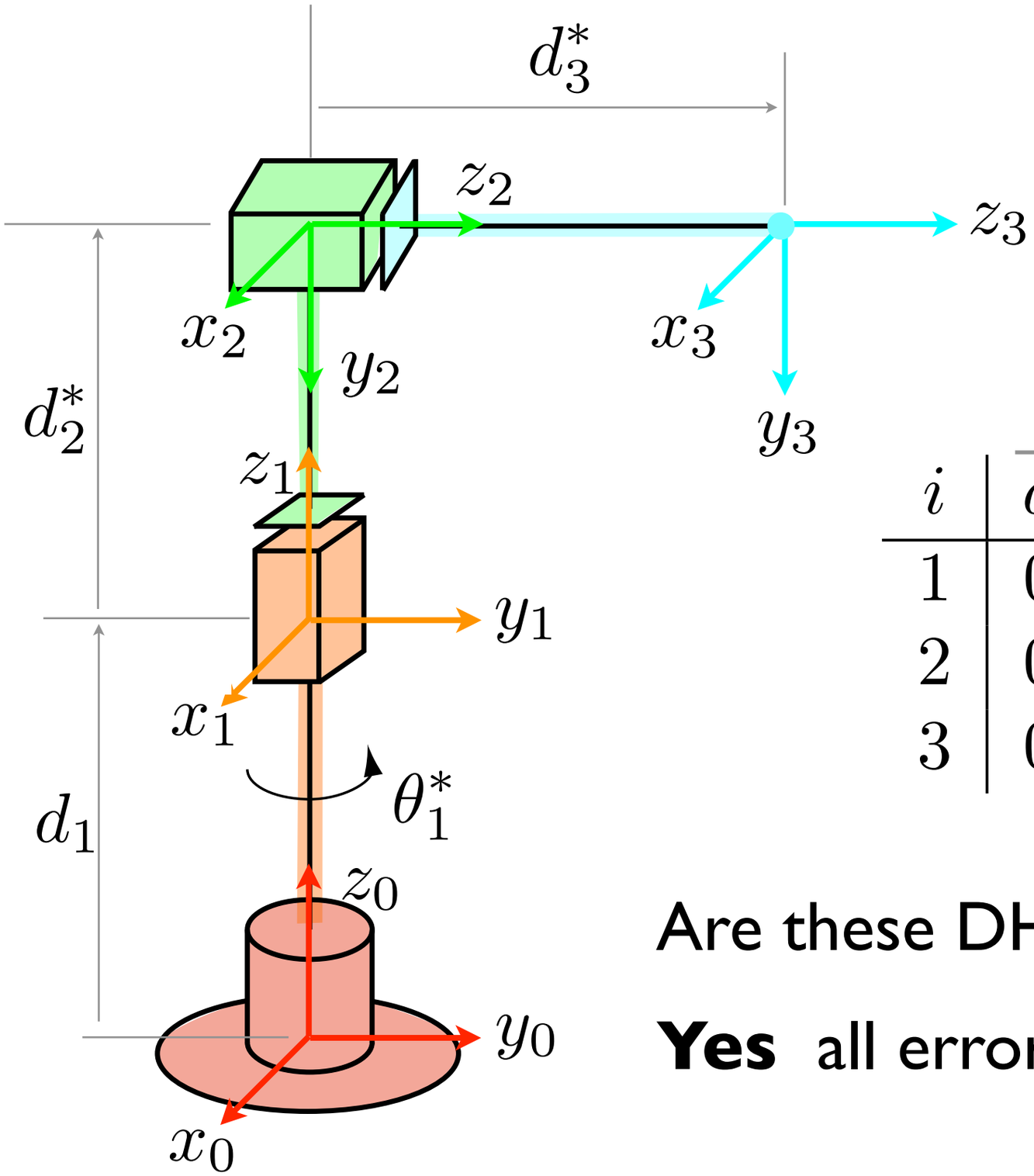


Which parts of the robot move with each coordinate frame?

The i^{th} frame moves with the i^{th} link.

The frame is typically on the most distant part of the link, defining the next axis.

The RPP Cylindrical Robot

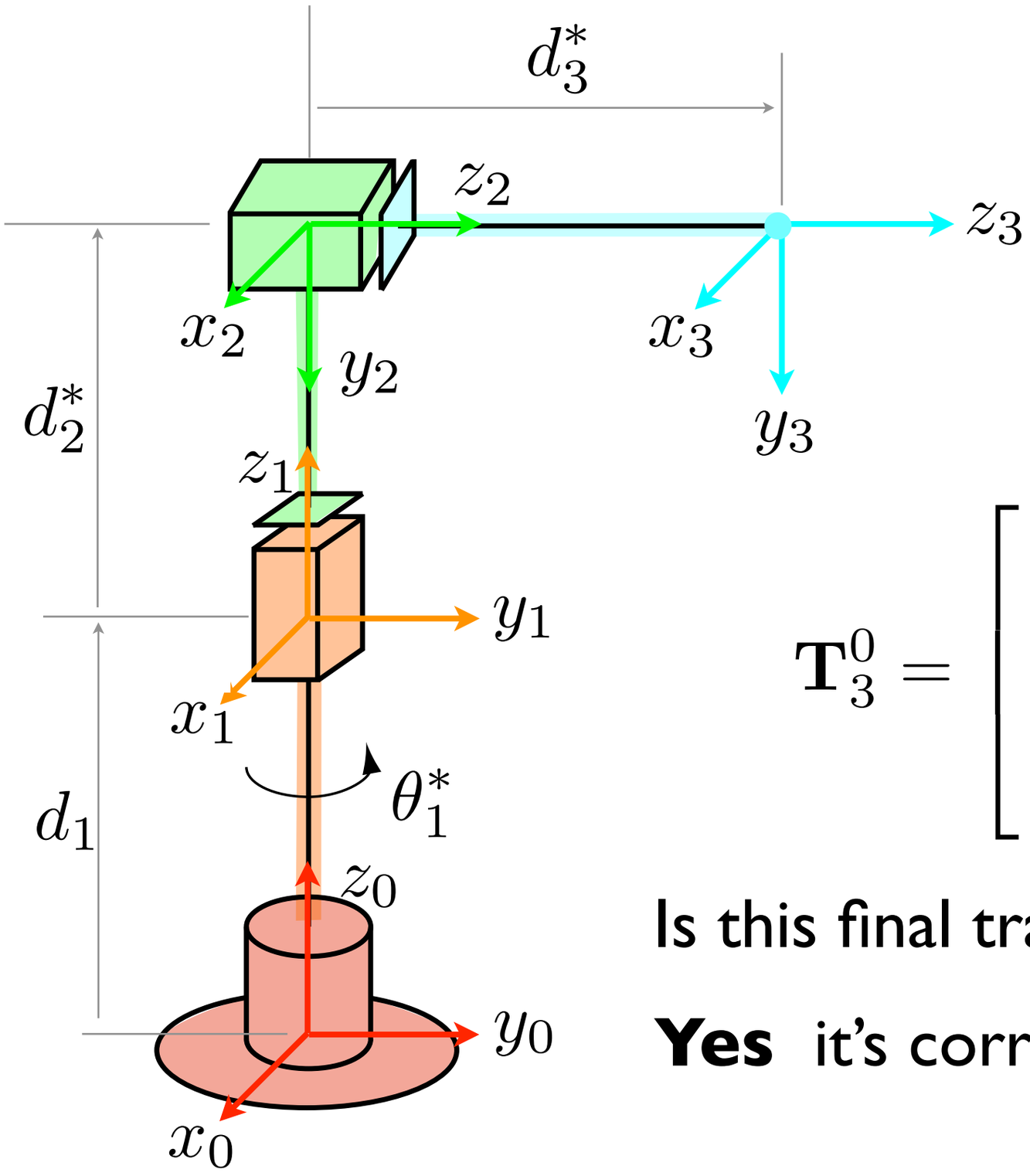


	x-step		z-step	
i	a	α	d	θ
1	0	$0°$	d_1	θ_1^*
2	0	$-90°$	d_2^*	$0°$
3	0	$0°$	d_3^*	$0°$

Are these DH parameters correct?

Yes all errors are corrected.

The RPP Cylindrical Robot

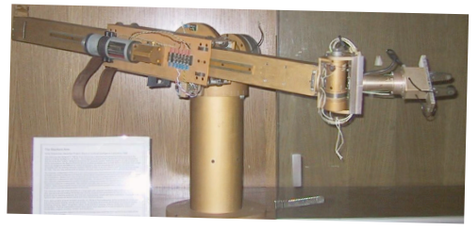


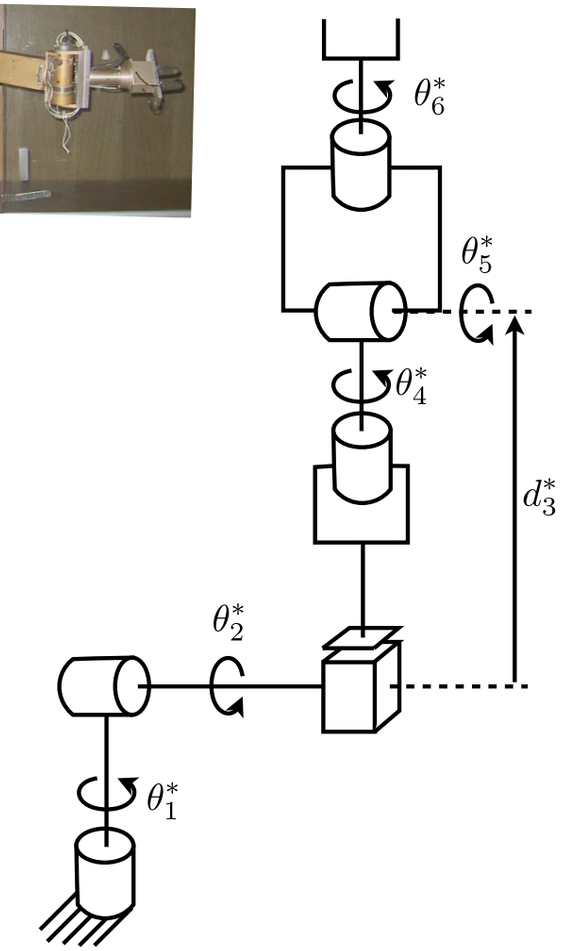
$$\mathbf{T}_3^0 = \begin{bmatrix} c_1^* & 0 & -s_1^* & -d_3^* s_1^* \\ s_1^* & 0 & c_1^* & d_3^* c_1^* \\ 0 & -1 & 0 & d_1 + d_2^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Is this final transformation correct?

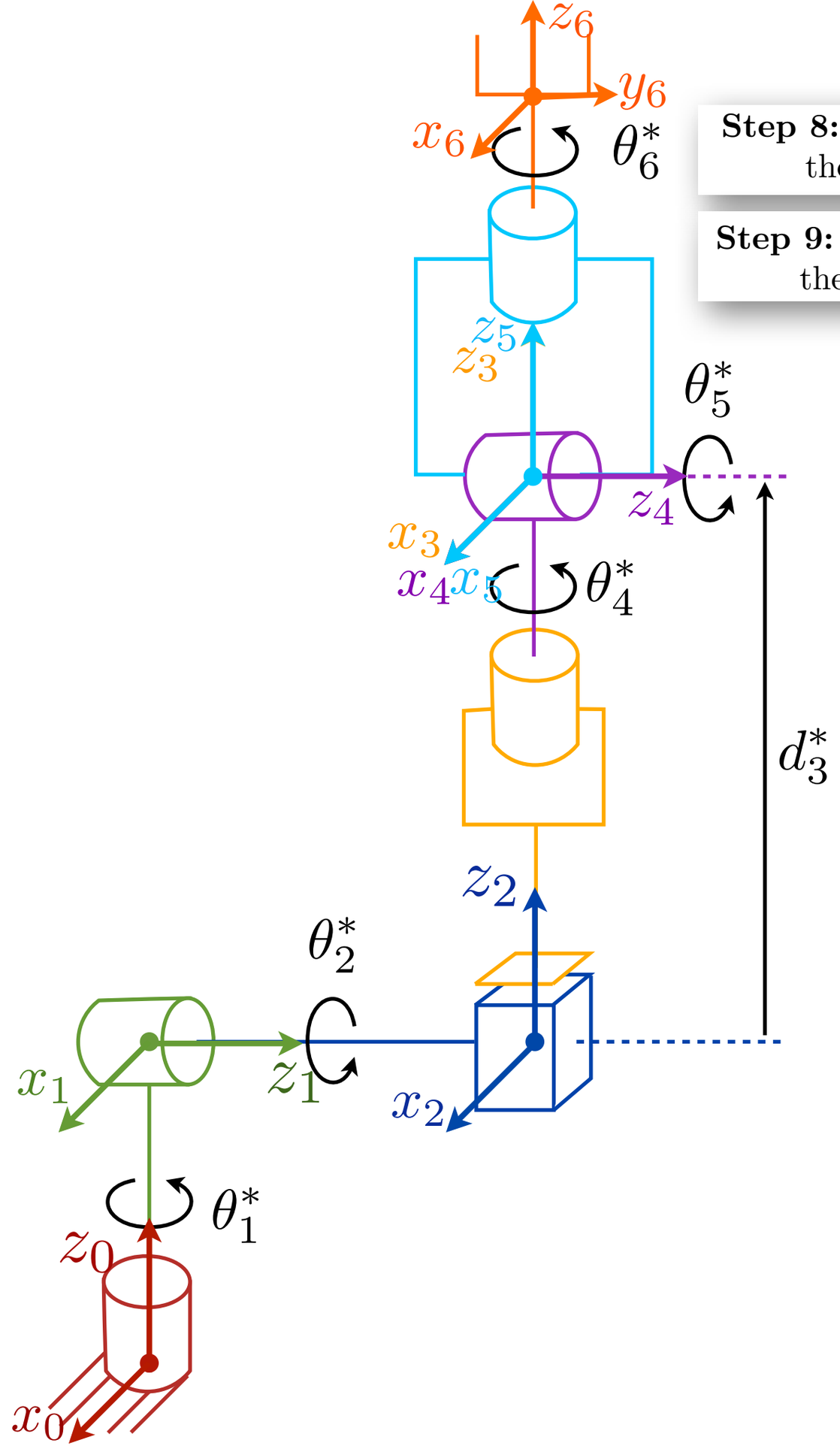
Yes it's correct!

DH Parameters for the Stanford Arm





Link	a_i	α_i	d_i	θ_i



Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

	x-step		z-step		
Link	a_i	α_i	d_i	θ_i	
1	0	$-90°$	d_1	θ_1^*	●→●
2	0	$+90°$	d_2	θ_2^*	●→●
3	0	$0°$	d_3^*	$0°$	●→●
4	0	$-90°$	0	θ_4^*	●→●
5	0	$+90°$	0	θ_5^*	●→●
6	0	$0°$	d_6	θ_6^*	●→●

MEAM 520

DH Wrap-Up, Trajectory Planning

Katherine J. Kuchenbecker, Ph.D.

General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

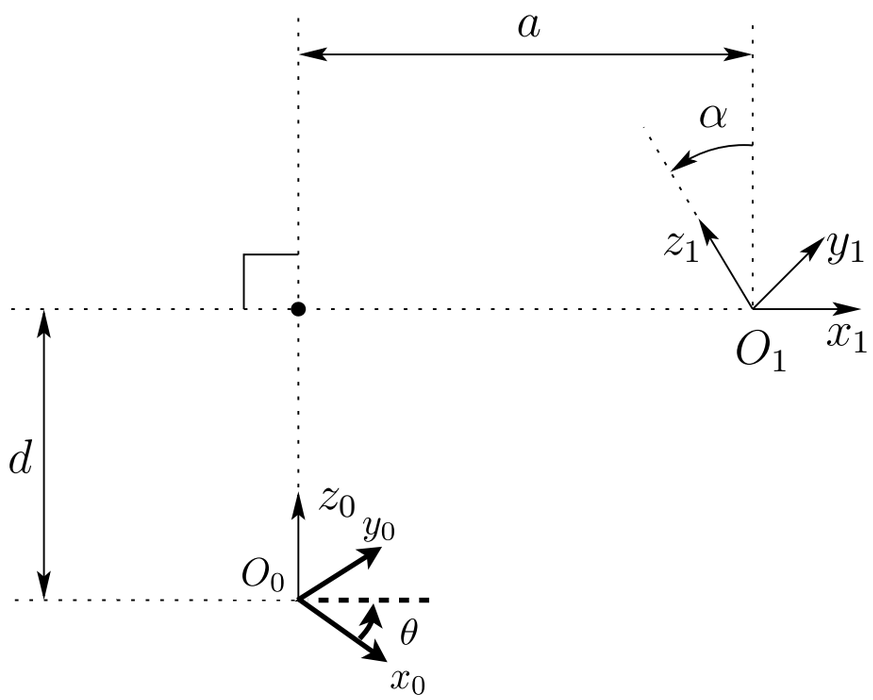
GRASP LABORATORY

Lecture 9: September 26, 2013

The Denavit-Hartenberg (DH) Convention for Manipulator Forward Kinematics

Given the design of my robot and the current values for its joint variables, where is the end-effector, and how is it oriented?

$$\mathbf{T}_n^0 = A_1(q_1) \cdots A_n(q_n)$$



$$A_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,\mathrm{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link	a_i	α_i	d_i	θ_i

DH Coordinate Frame Assumptions

(DH1) The axis x_1 is perpendicular to the axis z_0.

(DH2) The axis x_1 intersects the axis z_0.

The Denavit-Hartenberg Convention

(DH1) The axis x_i is perpendicular to the axis z_{i-1}

(DH2) The axis x_i intersects the axis z_{i-1}

---

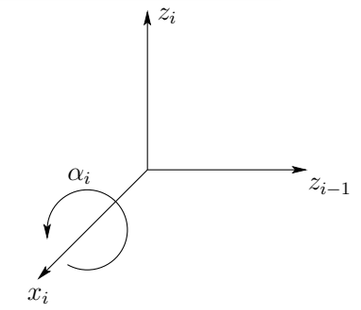
a_i

Link Length the distance between z_{i-1} and z_i, measured along x_i

---

α_i

Link Twist the angle between z_{i-1} and z_i, measured in the plane normal to x_i (right-hand rule around x_i)



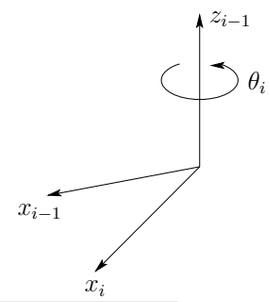
---

d_i

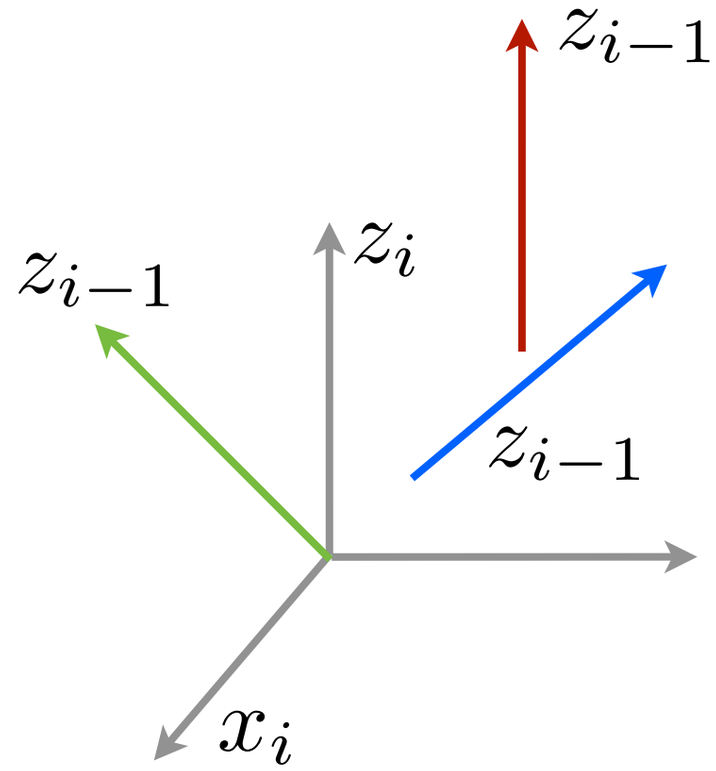
Link Offset the distance between x_{i-1} and x_i, measured along z_{i-1}

θ_i

Joint Angle the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1} (right-hand rule around z_{i-1})



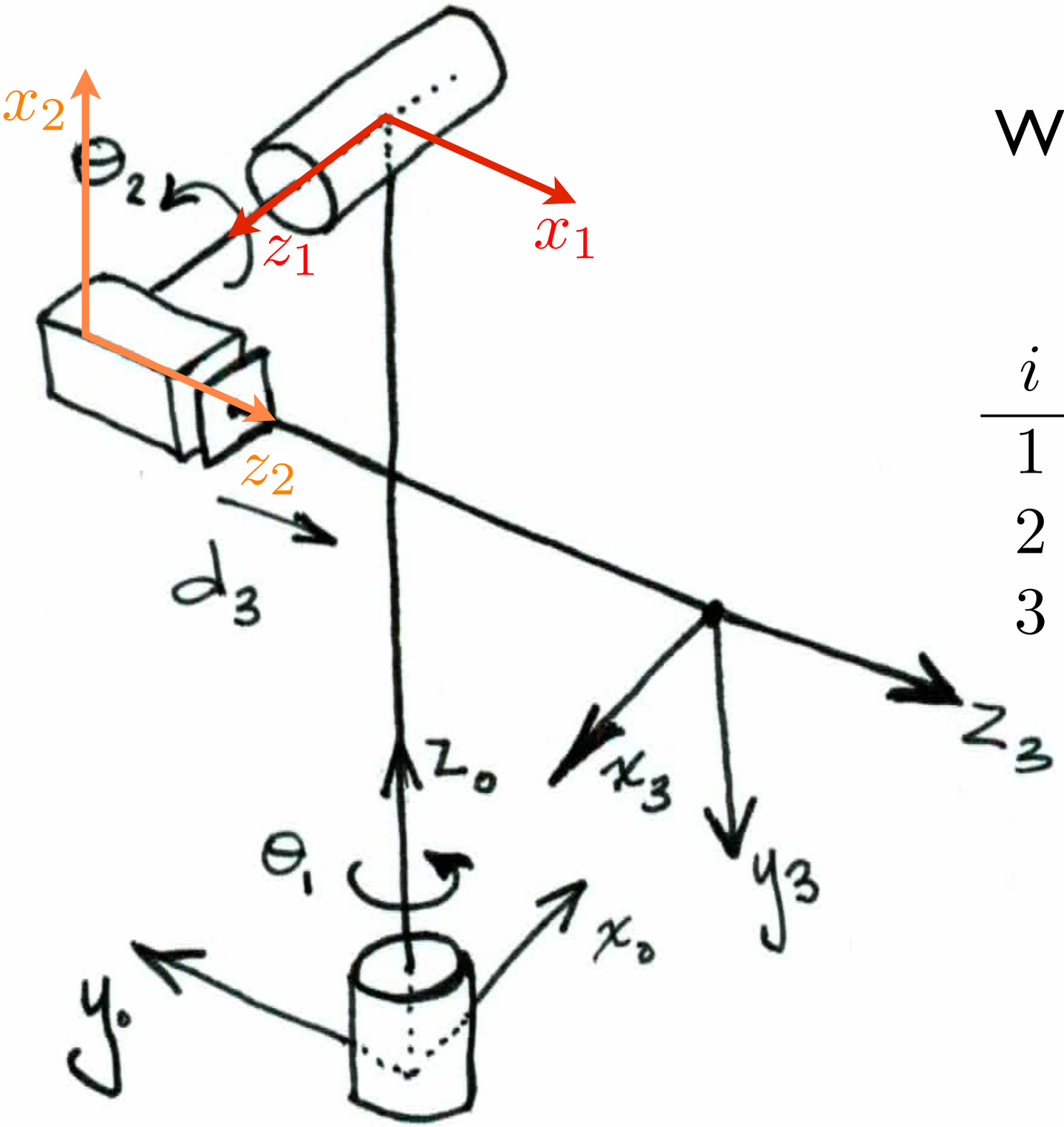
Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955



the following conventions make x placement easier (p. 82 in SHV):

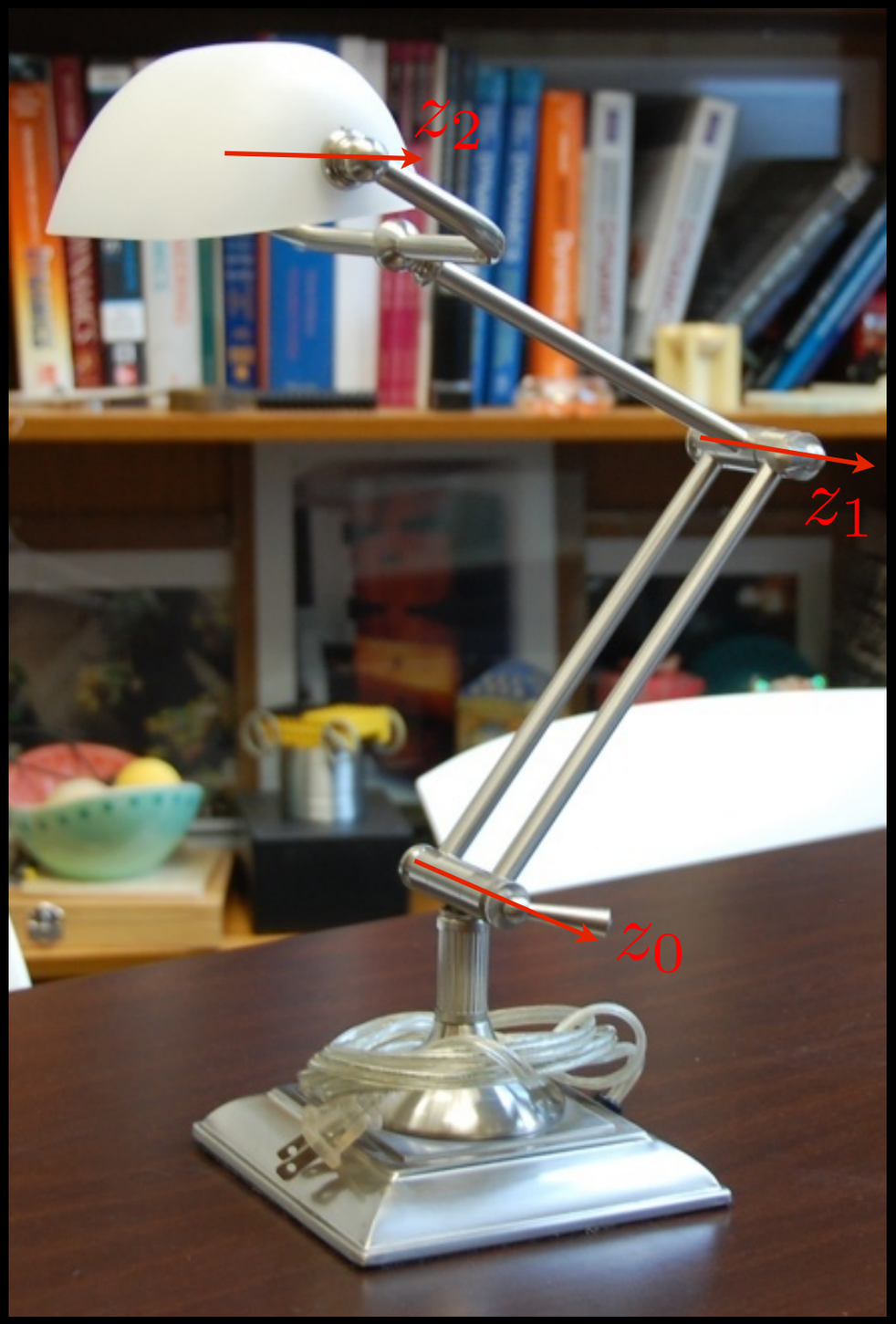
if z_{i-1} is parallel to z_i	put O_i anywhere along z_i orient x_i toward z_i
if z_{i-1} intersects z_i	put O_i at the intersection between z_{i-1} and z_i orient x_i normal to the plane formed by z_{i-1} and z_i
if z_{i-1} is not coplanar with z_i	put O_i where unique shortest line between z's hits z_i orient x_i along unique normal with z_{i-1}, toward z_i

Shown at $\theta_1^* = 0, \theta_2^* = 0,$ and $d_3* > 0$



What are the DH parameters?

i	a	α	d	θ
1	0	$+90°$	d_1	$\theta_1^* - 90°$
2	0	$+90°$	d_2	$\theta_2^* + 90°$
3	0	$0°$	d_3^*	$+90°$



A manipulator's **configuration** is a complete specification of the location of every point on the manipulator.

$$\vec{q} = \begin{bmatrix} q_i \\ \vdots \\ q_n \end{bmatrix}$$

- We use a **joint variable** to denote each joint's position.
- Value defines the joint's displacement from a **zero configuration**.
- Use **θ** for revolute joints.
- Use **d** for prismatic joints.
- Axis orientation defines the **positive direction**.
- Knowing all joint variable values defines the configuration.

A **trajectory** is a function of time $\vec{q}(t)$

Such that $\vec{q}(t_0) = \vec{q}_s$

and $\vec{q}(t_f) = \vec{q}_f$

Parameterized by time, so we can compute velocities and accelerations along the trajectory by differentiation.

A **via point** is a configuration that the robot needs to achieve.

Via points could be obtained in many ways:

Hand programming by picking joint angles.

Specified as a T06 pose, using IK to calculate the corresponding joint angles.

Physically leading the robot through the desired motion with a teach pendant.

A **trajectory** is a function of time $\vec{q}(t)$

Parameterized by time, so we can compute velocities and accelerations along the trajectory by differentiation.

Such that $\vec{q}(t_0) = \vec{q}_s$

and $\vec{q}(t_f) = \vec{q}_f$

How many trajectories exist that satisfy these constraints?

Infinitely many.

What if I also specify starting and final velocities?

There are still infinitely many trajectories.

Roboticists typically choose trajectories from a **finitely parameterizable family**, such as polynomials of degree n.

For point-to-point motion, each joint's motion is typically planned independently, so we'll consider just a single joint angle.

$$\text{instead of } \vec{q}(t)$$

$$q(t) = \theta_i(t) \quad \text{or} \quad q(t) = d_i(t)$$

Simplest Situation: Specifying Joint Value Only

Initial Condition

$$q(t_0) = q_0$$

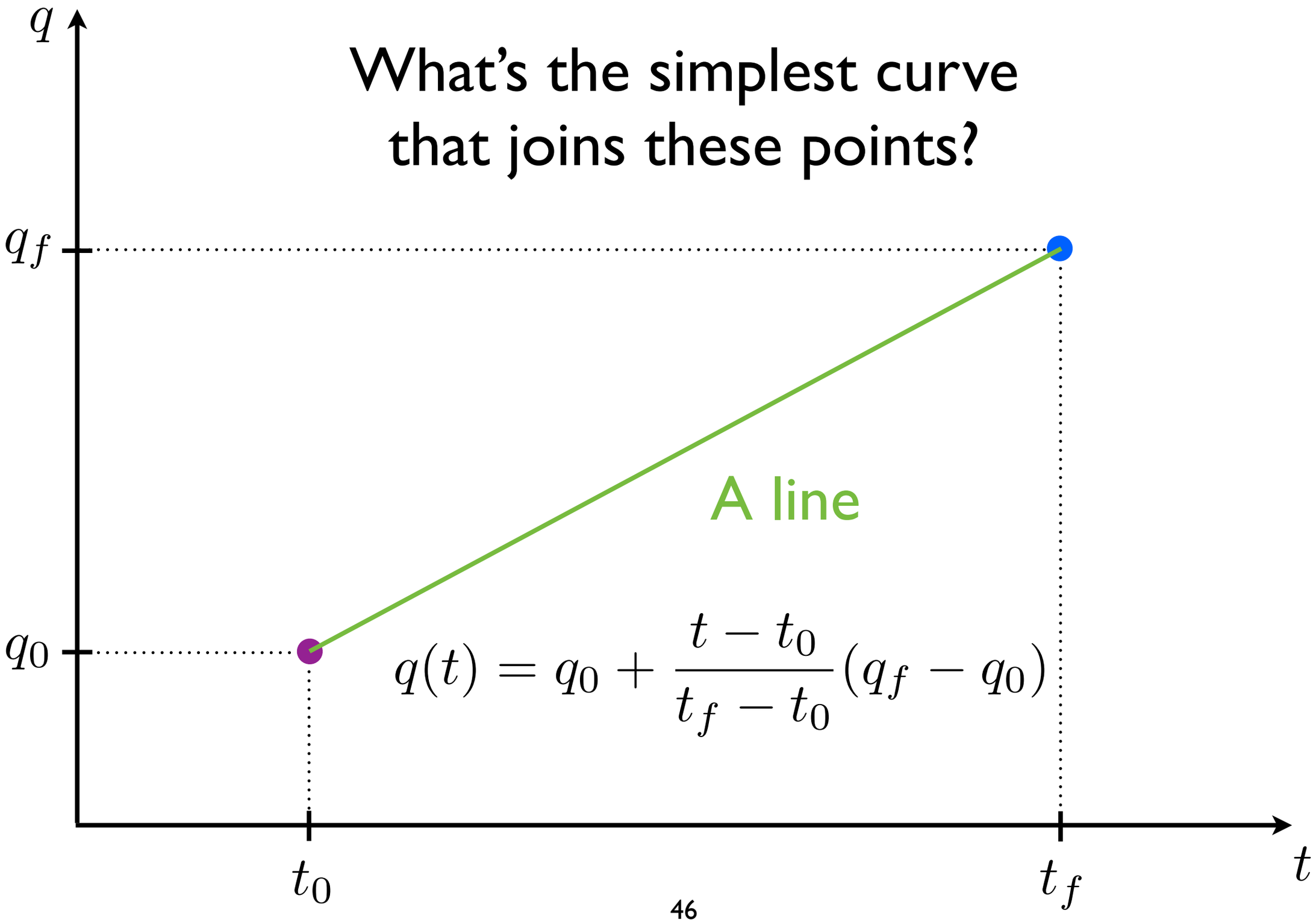
Final Condition

$$q(t_f) = q_f$$

Simplest Situation: Specifying Joint Value Only

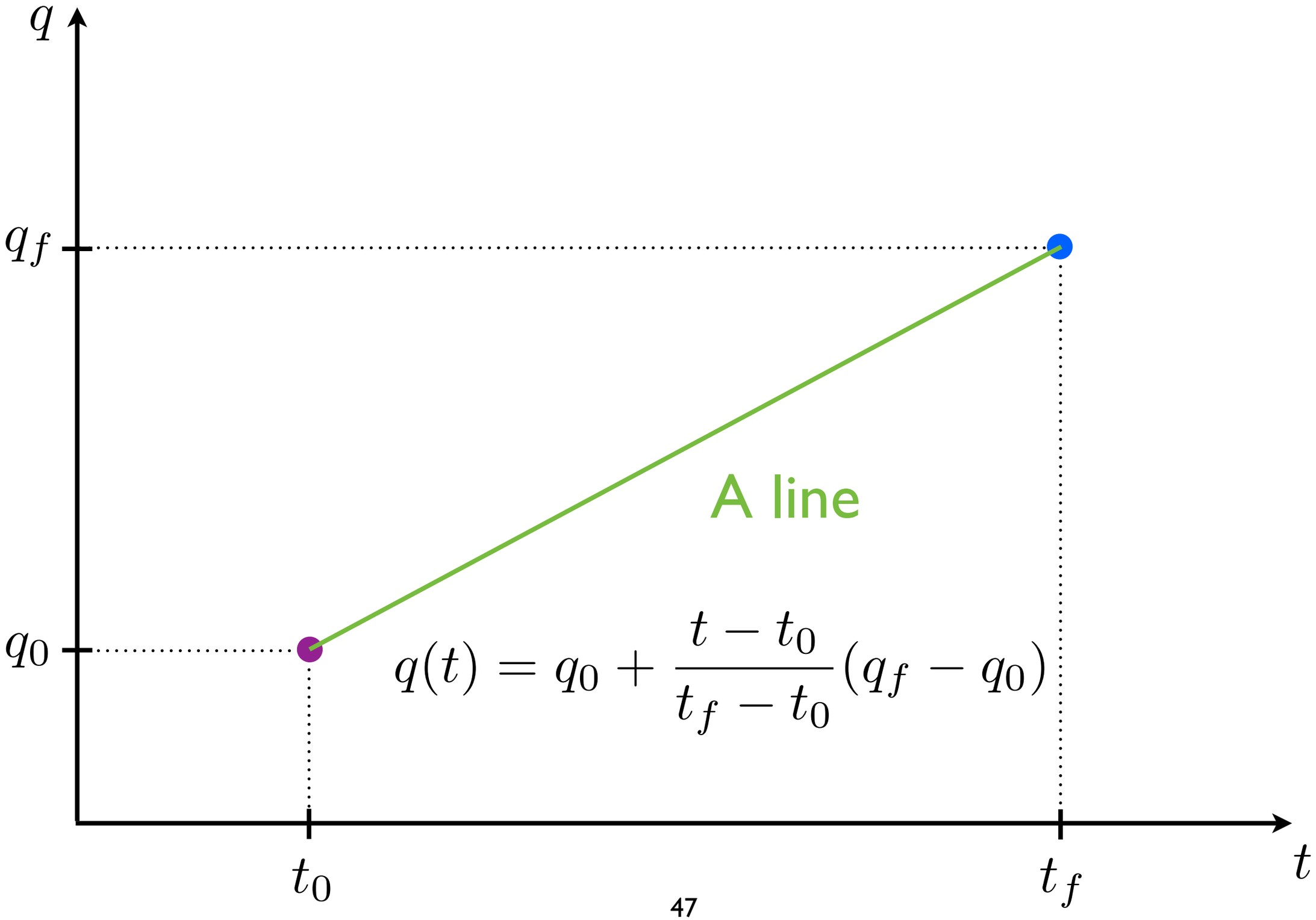
$$q(t_0) = q_0 \qquad\qquad q(t_f) = q_f$$

What's the simplest curve that joins these points?



Linear interpolation is really useful!

Why do you think SHV doesn't present lines?



MEAM 520

Trajectory Planning

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 10: October 1, 2013

A **trajectory** is a function of time $\vec{q}(t)$

Such that $\vec{q}(t_0) = \vec{q}_s$

and $\vec{q}(t_f) = \vec{q}_f$

How many trajectories exist that satisfy these constraints?

Infinitely many.

What if I also specify starting and final velocities?

There are still infinitely many trajectories.

Roboticists typically choose trajectories from a **finitely parameterizable family**, such as polynomials of degree n.

For point-to-point motion, each joint's motion is typically planned independently, so we'll consider just a single joint angle.

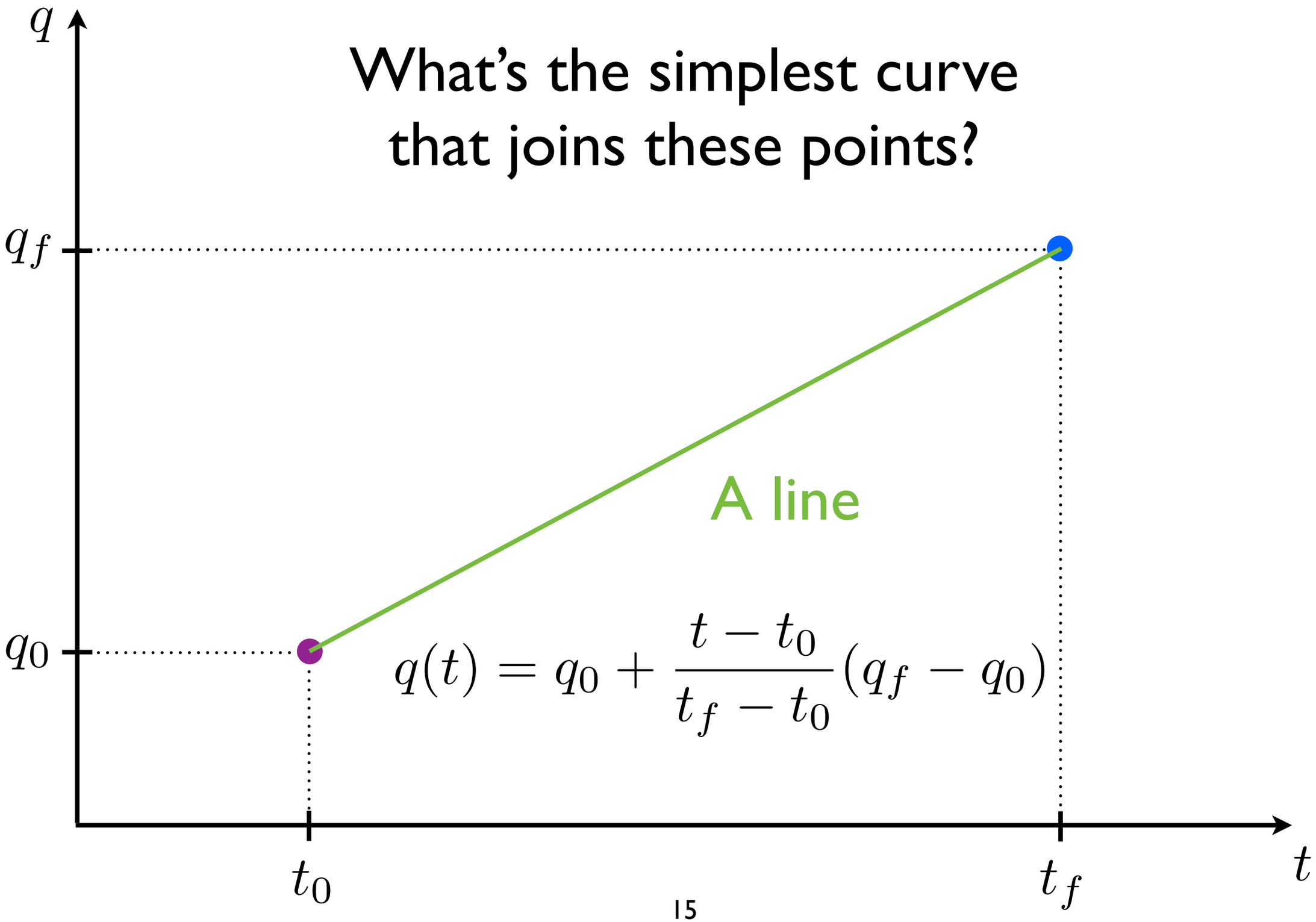
$$\text{instead of } \vec{q}(t)$$

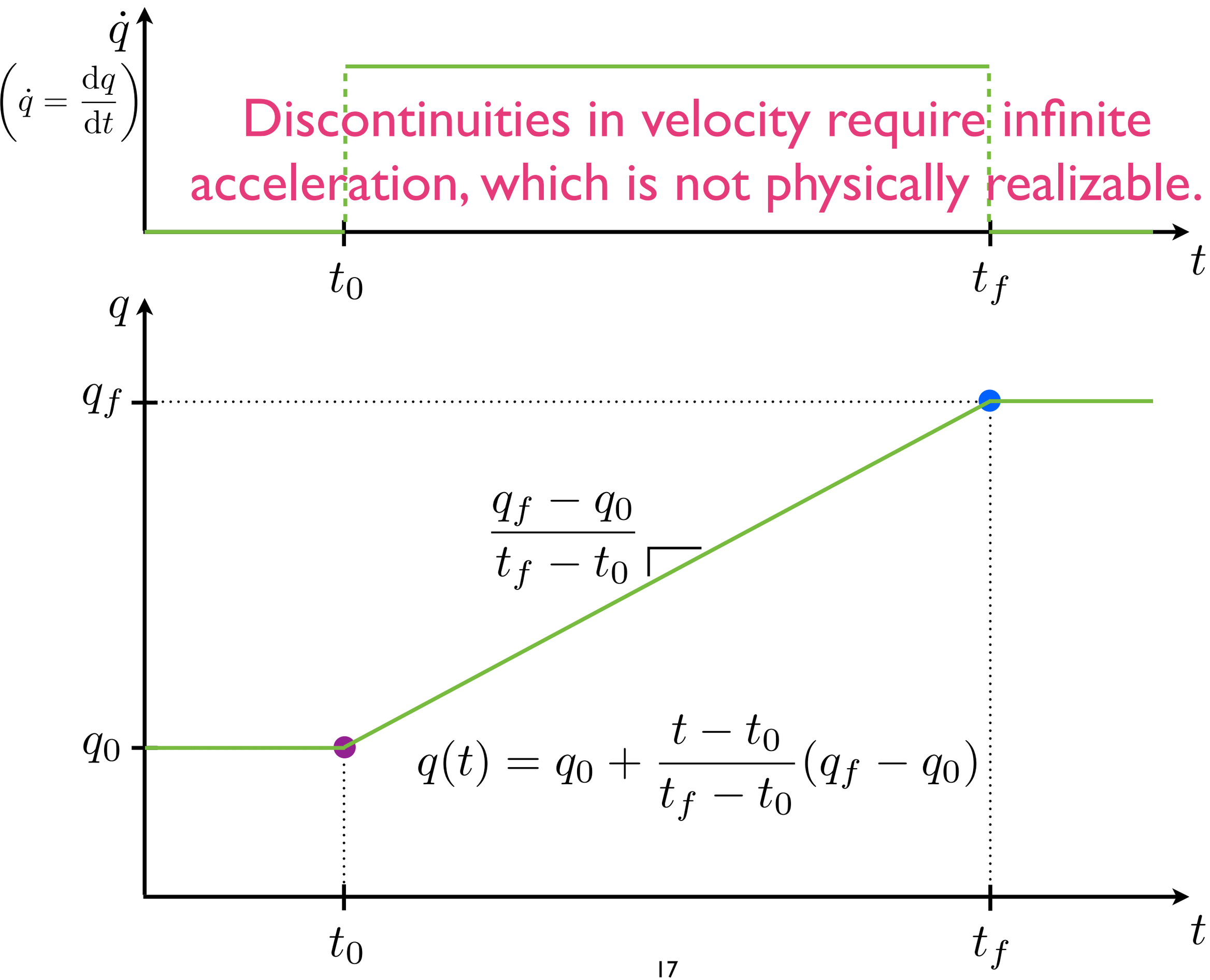
$$q(t) = \theta_i(t) \quad \text{or} \quad q(t) = d_i(t)$$

Simplest Situation: Specifying Joint Value Only

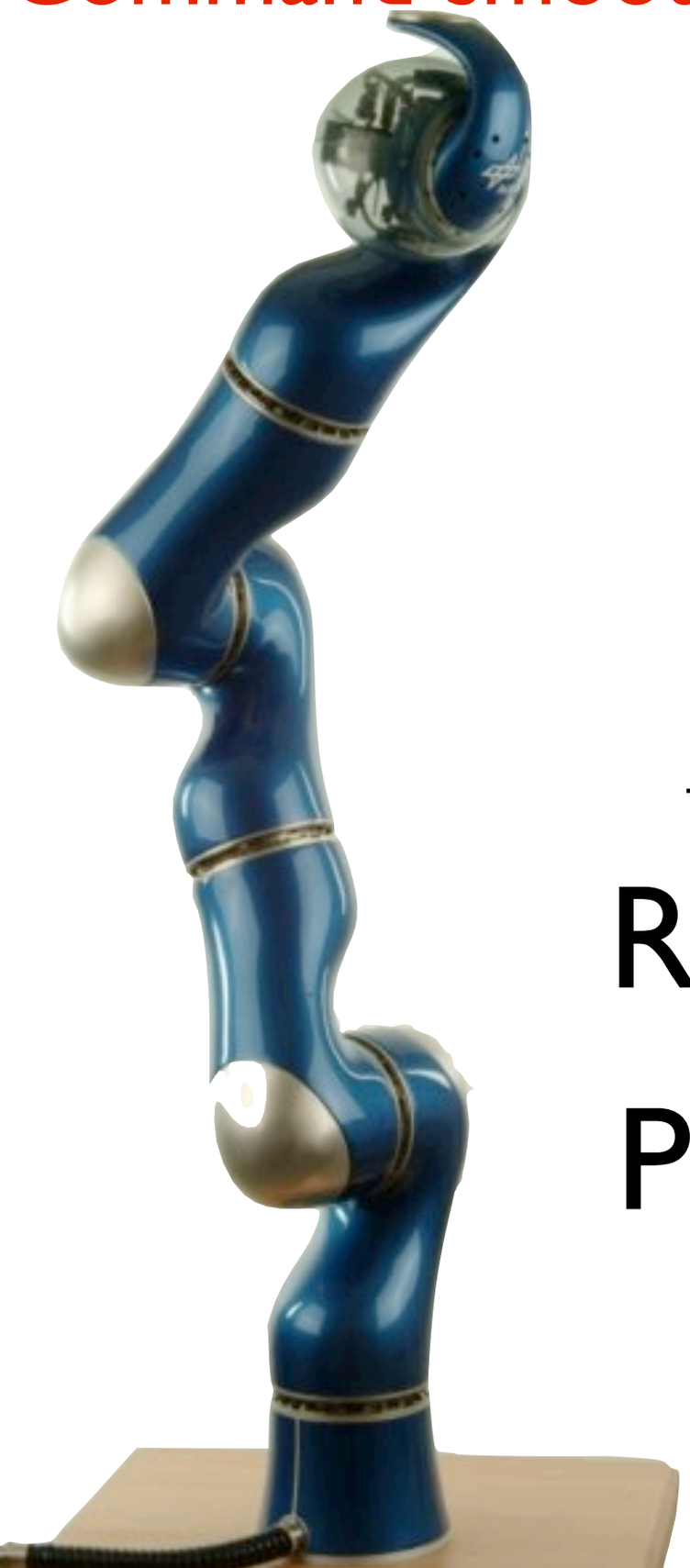
$$q(t_0) = q_0 \qquad\qquad q(t_f) = q_f$$

What's the simplest curve that joins these points?




$\dot{q}$
$\left(\dot{q} = \frac{\mathrm{d}q}{\mathrm{d}t}\right)$
Discontinuities in velocity require infinite acceleration, which is not physically realizable.
t_0
t_f
t
q
q_f
q_0
$\frac{q_f - q_0}{t_f - t_0}$
$q(t) = q_0 + \frac{t - t_0}{t_f - t_0}(q_f - q_0)$
t_0
t_f
t

Robots are actually flexible!
Command smooth trajectories to avoid exciting flexibilities.



Hanging slinky example.

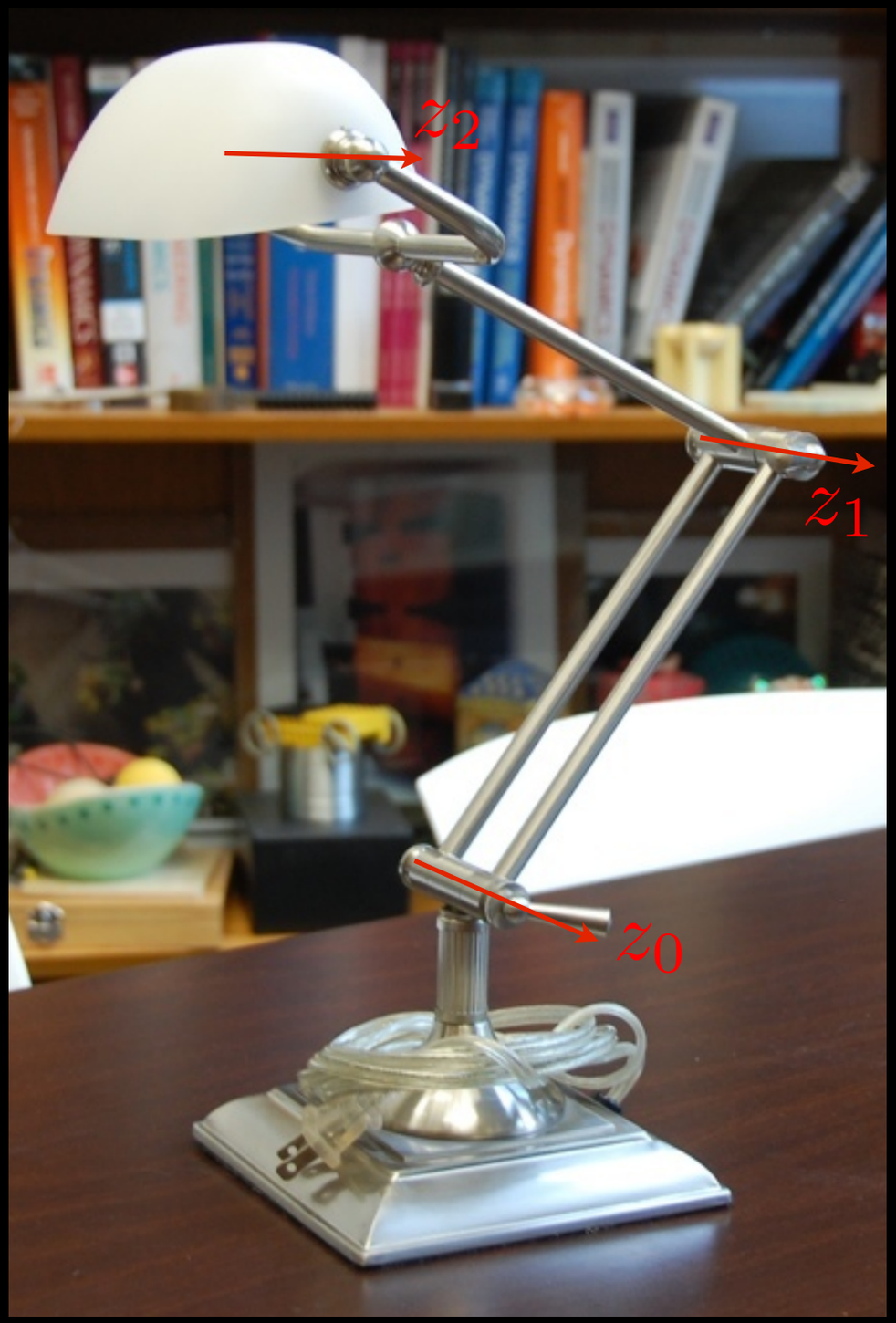
Robot manipulators are composed of:

- Rigid **links**
- Connected by **joints**
- To form a **kinematic chain**

There are two types of **joints**:

R
- **Revolute** (rotary), like a hinge, allows relative rotation between two links

P
- **Prismatic** (linear), like a slider, allows a relative linear motion (translation) between two links



Does the **configuration** of a manipulator fully define how it will move in the future?

- No. It only gives you an **instantaneous description** of the geometry.
- The manipulator's **state** is a set of variables that is sufficient to tell you its future time response when combined with dynamics and future inputs.
- State requires both the joint variables and their **derivatives**.

Specifying Joint Values and First Time Derivatives

Initial Conditions

$$q(t_0) = q_0$$ **Units** angle or distance, e.g., rad or m

$$\dot{q}(t_0) = v_0$$ **Units** angle per time or distance per time, e.g., rad/s or m/s

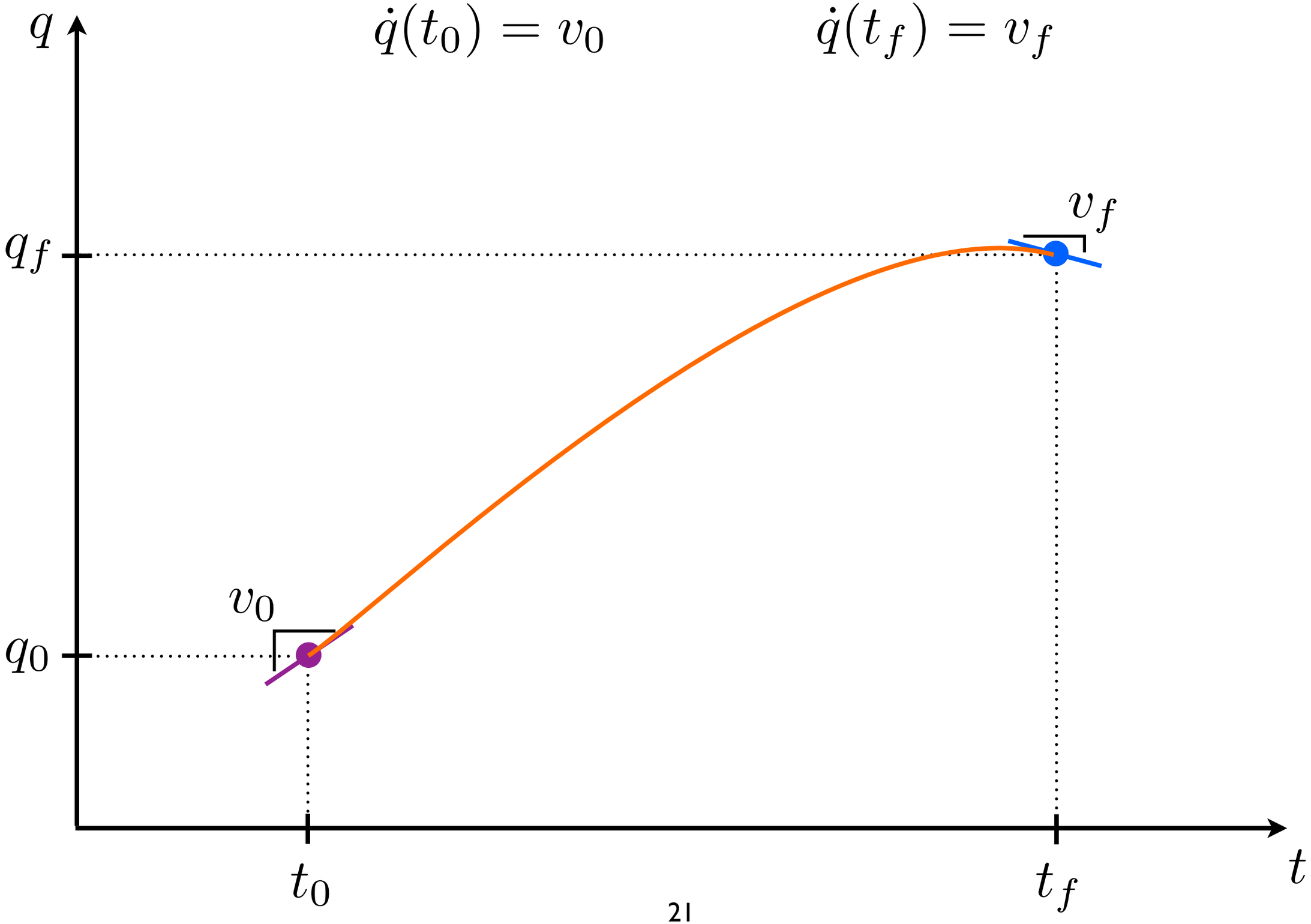
Final Conditions

$$q(t_f) = q_f$$

$$\dot{q}(t_f) = v_f$$

Specifying Joint Values and First Time Derivatives

$$q(t_0) = q_0 \qquad q(t_f) = q_f$$
$$\dot{q}(t_0) = v_0 \qquad \dot{q}(t_f) = v_f$$



Specifying Joint Values and First Time Derivatives

Cubic Polynomial Trajectories

start → end

$$q(t_0) = q_0 \longrightarrow q(t_f) = q_f$$

$$\dot{q}(t_0) = v_0 \longrightarrow \dot{q}(t_f) = v_f$$

cubic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

Specifying Joint Values and First Time Derivatives
Cubic Polynomial Trajectories

System of Four Equations

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

$$\vec{x} = A^{-1}\vec{b}$$

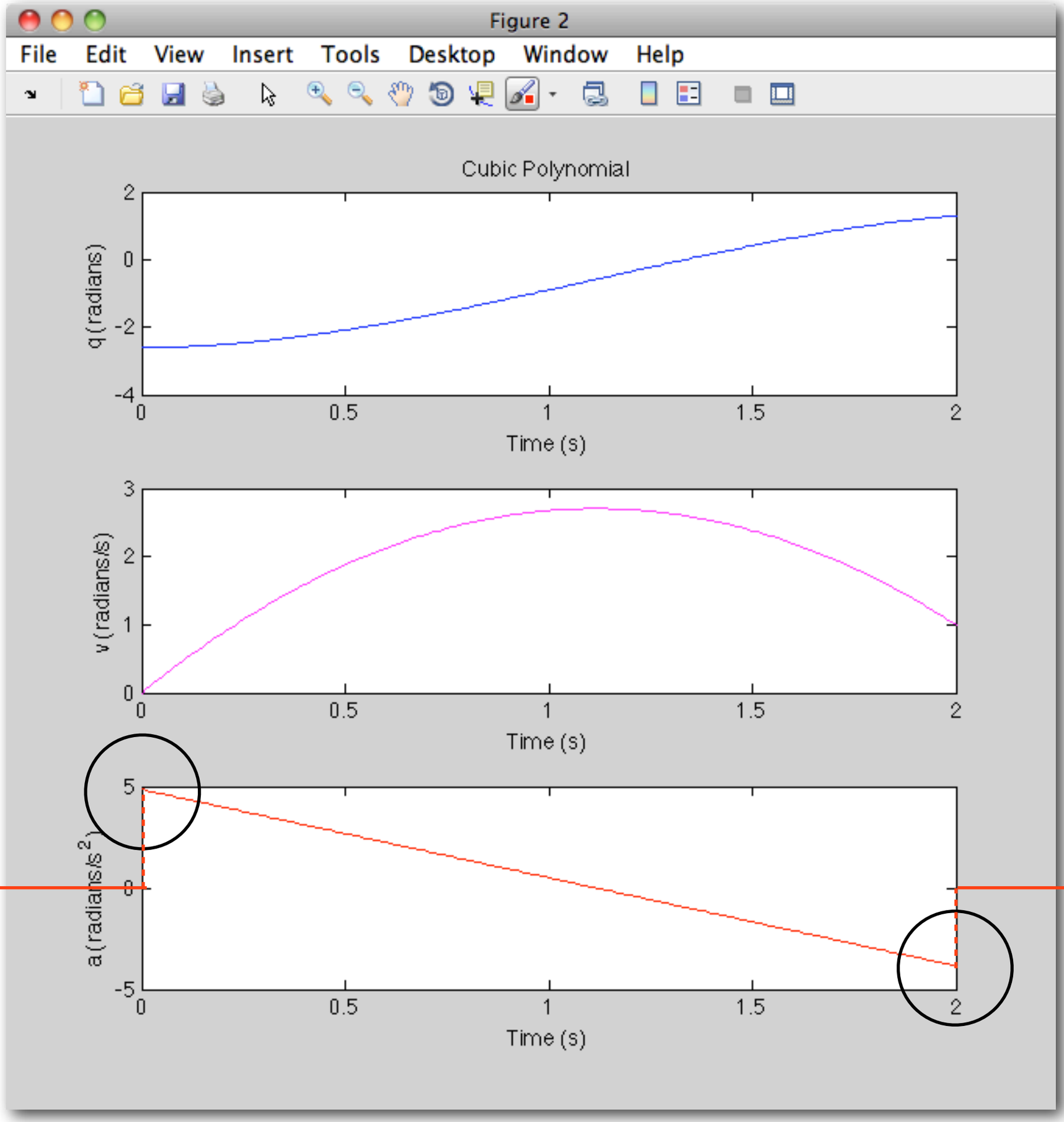
time matrix

conditions ↓ unknown coefficients

Reformulate in $\vec{b} = A\vec{x}$ form

$$\vec{x} = A \backslash \vec{b}$$

$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



Discontinuities in acceleration require **step changes in force/torque**, which excites vibrational modes in the robot.

Time derivative of acceleration is **jerk**.

We don't want **infinite jerk**.

Specifying Joint Values Plus First and Second Time Derivatives

Initial Conditions

$$q(t_0) = q_0$$ **Units** angle or distance, e.g., rad or m

$$\dot{q}(t_0) = v_0$$ **Units** angle per time or distance per time, e.g., rad/s or m/s

$$\ddot{q}(t_0) = \alpha_0$$ **Units** angle per time per time or distance per time per time, e.g., rad/s^2 or m/s^2

Not the same α as in DH!

Final Conditions

$$q(t_f) = q_f$$

$$\dot{q}(t_f) = v_f$$

$$\ddot{q}(t_f) = \alpha_f$$

What kind of curve to use?

Specifying Joint Values Plus First and Second Time Derivatives

Quintic Polynomial Trajectories

start		end
$q(t_0) = q_0$	$\longrightarrow$	$q(t_f) = q_f$
$\dot{q}(t_0) = v_0$	$\longrightarrow$	$\dot{q}(t_f) = v_f$
$\ddot{q}(t_0) = \alpha_0$	$\longrightarrow$	$\ddot{q}(t_f) = \alpha_f$

quintic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

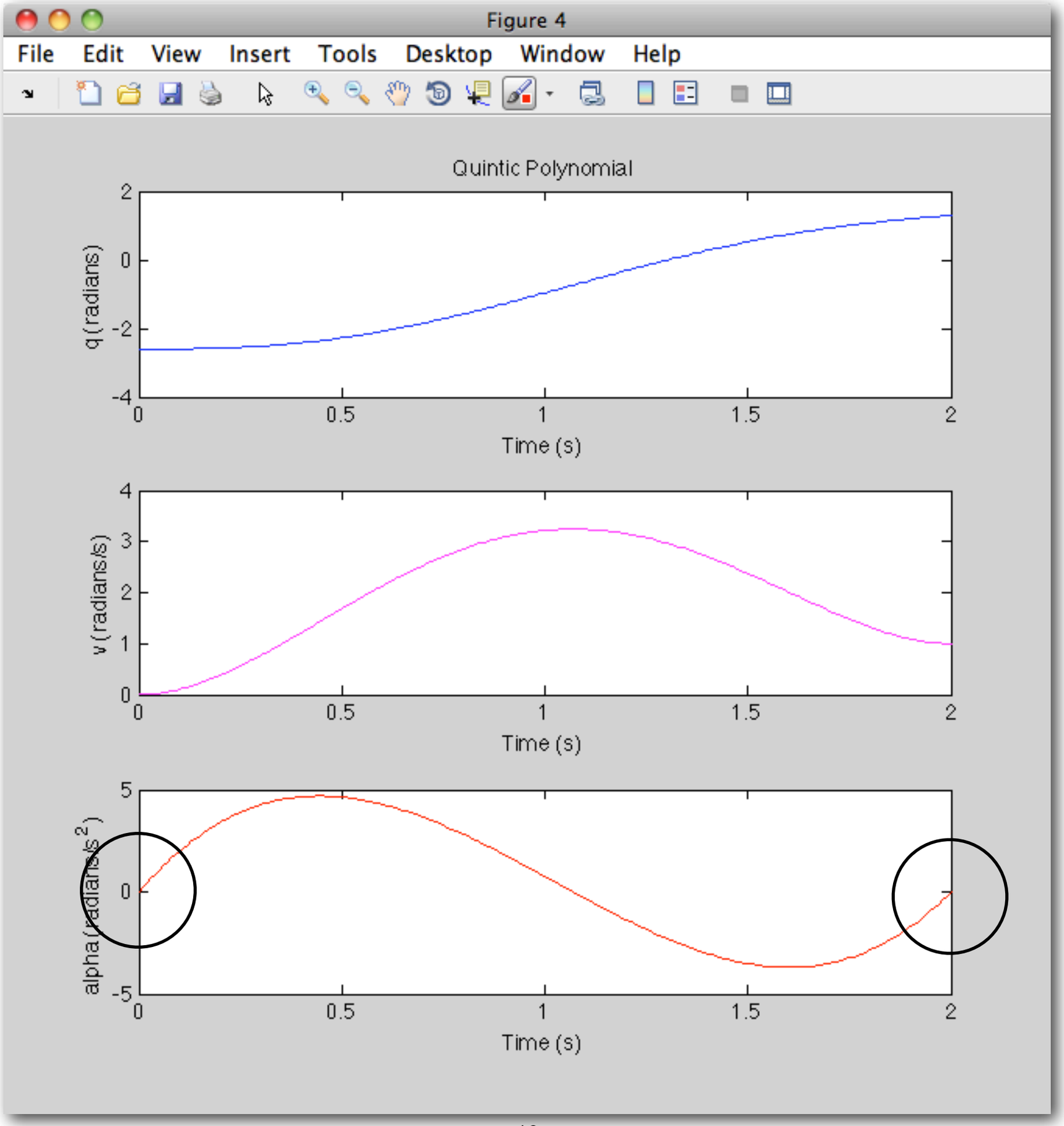


Figure 4
File Edit View Insert Tools Desktop Window Help
Quintic Polynomial
q (radians)
Time (s)
v (radians/s)
Time (s)
alpha (radians/s^2)
Time (s)

Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

Ramp up velocity to desired value for a short time at start.

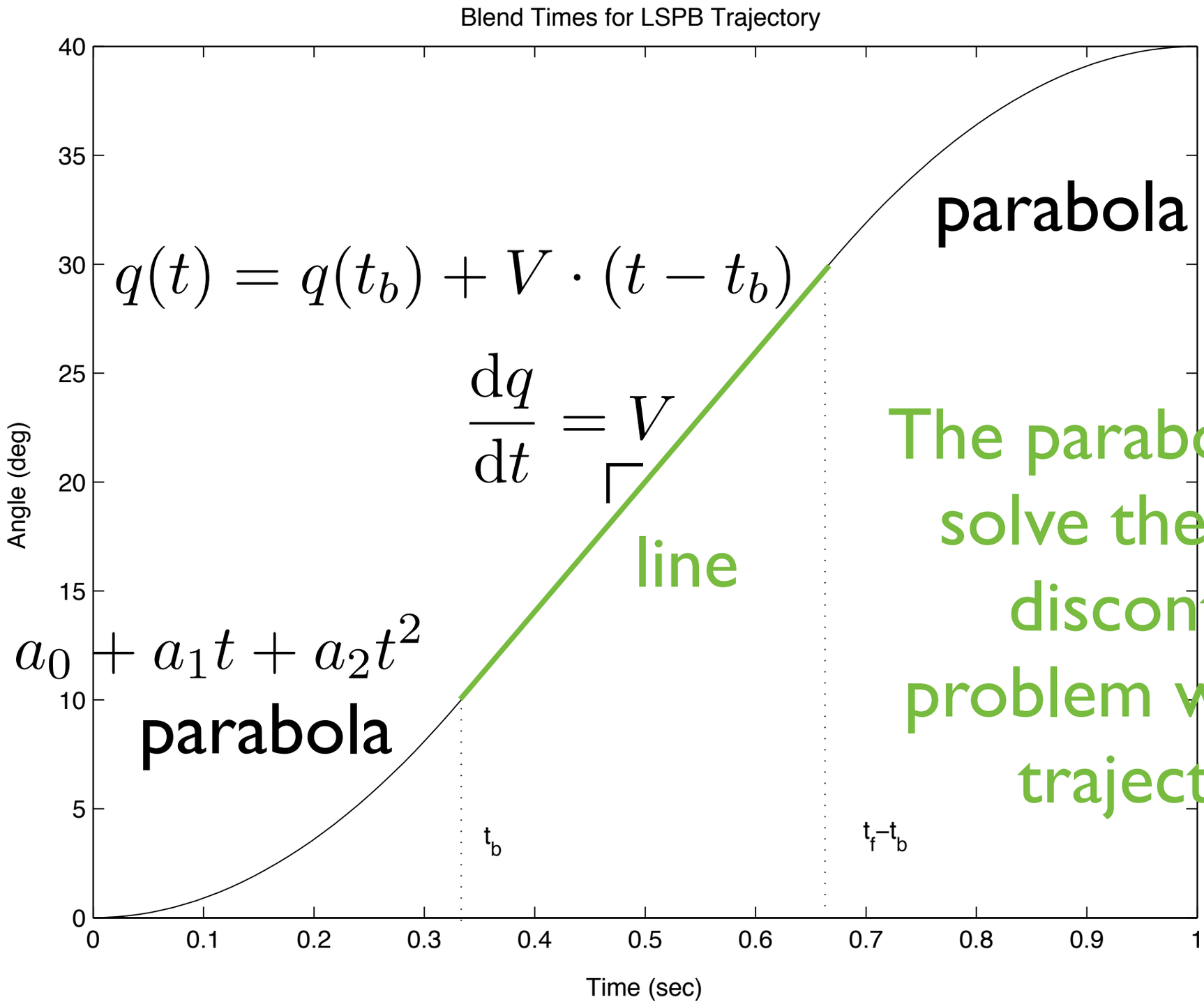
Move at constant velocity for a while.

Ramp down velocity to final value for a short time at end.

Start and end blend times are usually equal.

Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

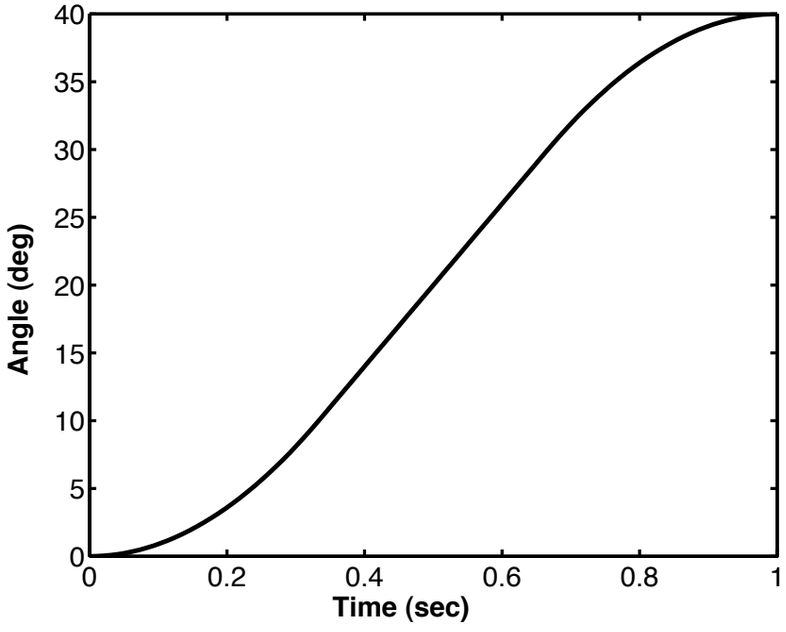


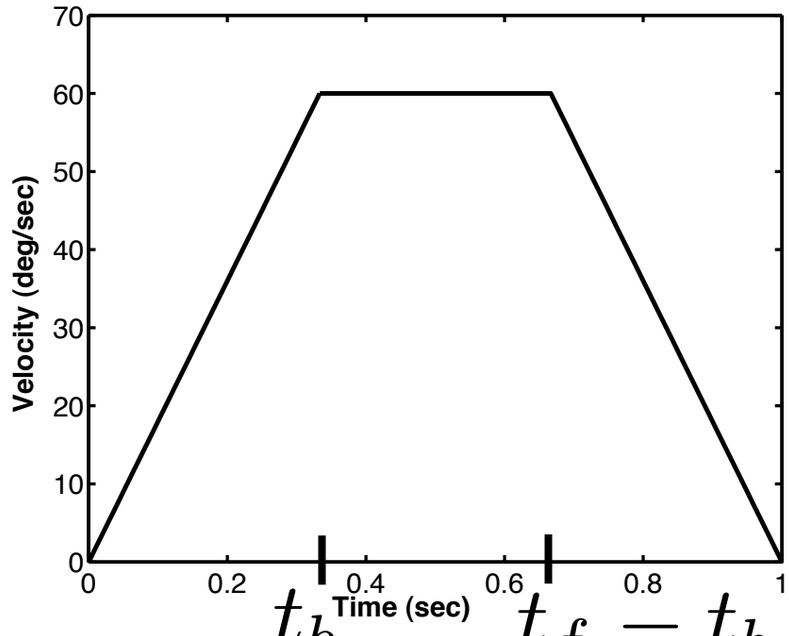
The parabolic blends solve the velocity discontinuity problem with linear trajectories.

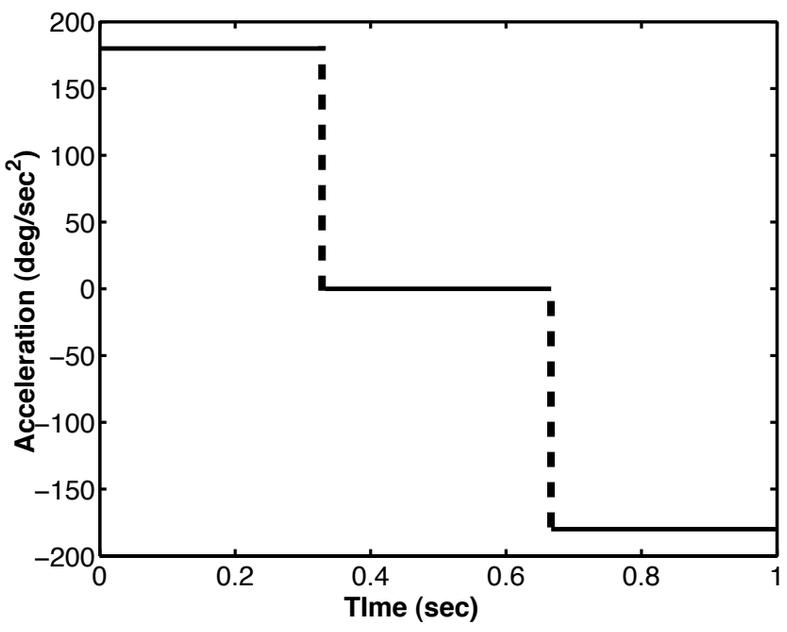
Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

Piecewise constant accelerations







Trapezoidal velocity profile

Limits

$$0 < t_b \leq \frac{t_f}{2}$$

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}$$

Not minimum jerk...

Getting There As Fast As Possible

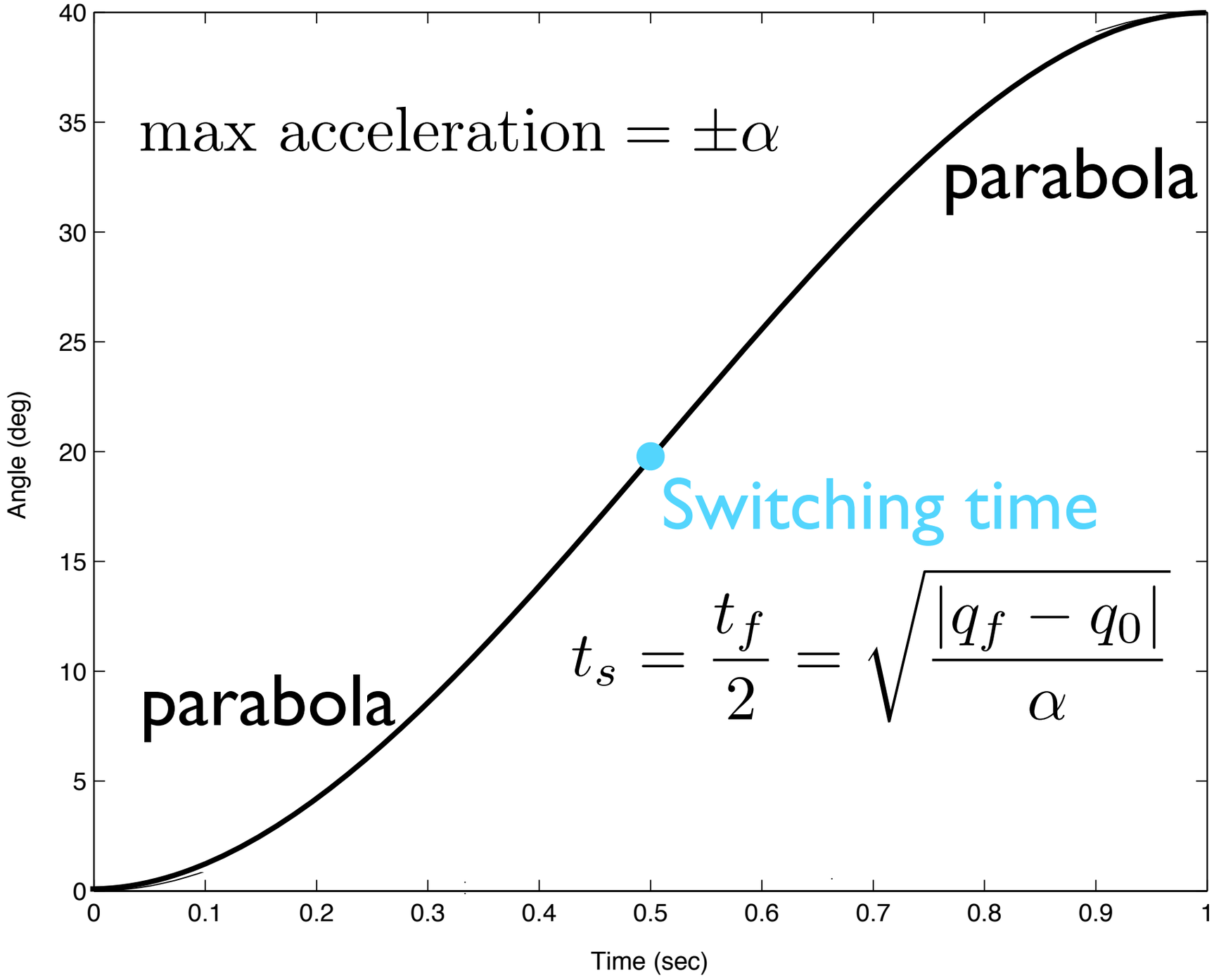
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories

Leave final time unspecified.
Specify the maximum acceleration possible,
typically set by actuator limit.
Apply maximum acceleration in one direction,
then abruptly switch to negative maximum acceleration.

Typically starting and ending at rest.
Switching time is halfway through the trajectory.

Getting There As Fast As Possible

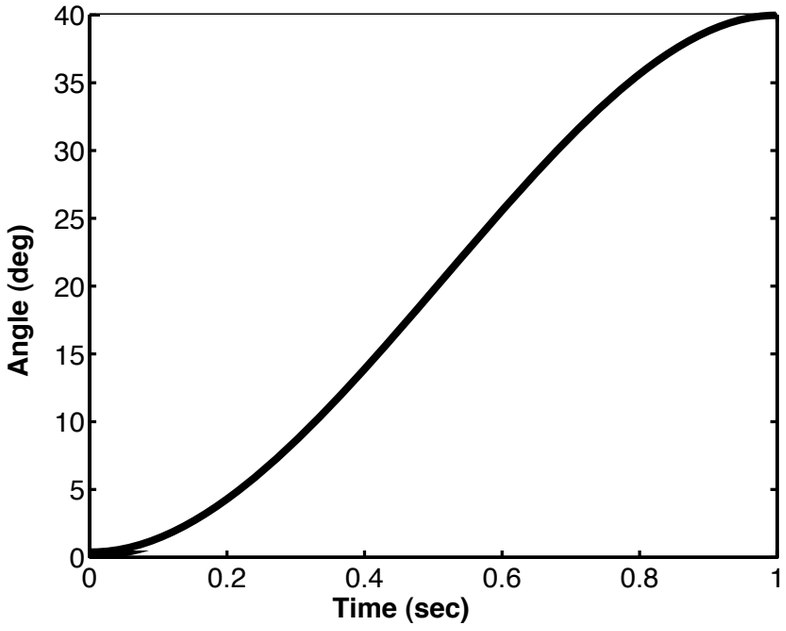
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories

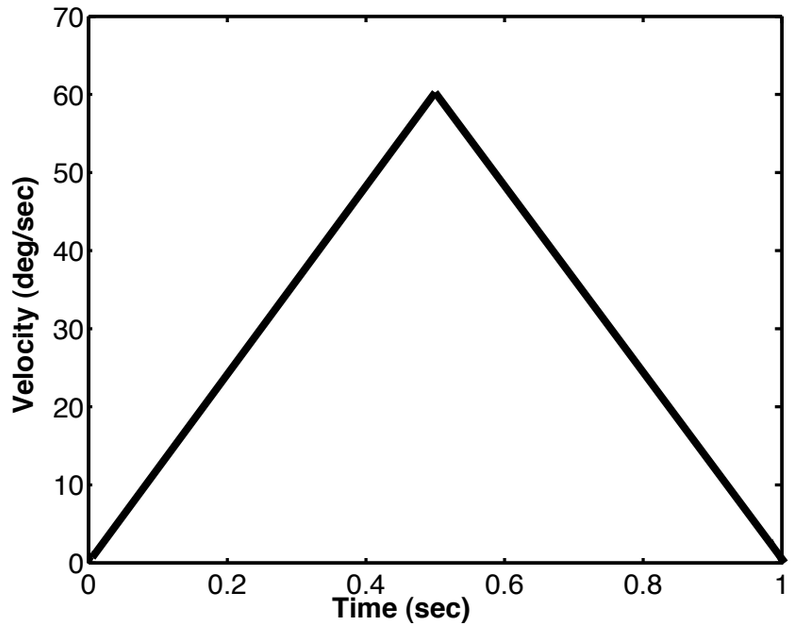


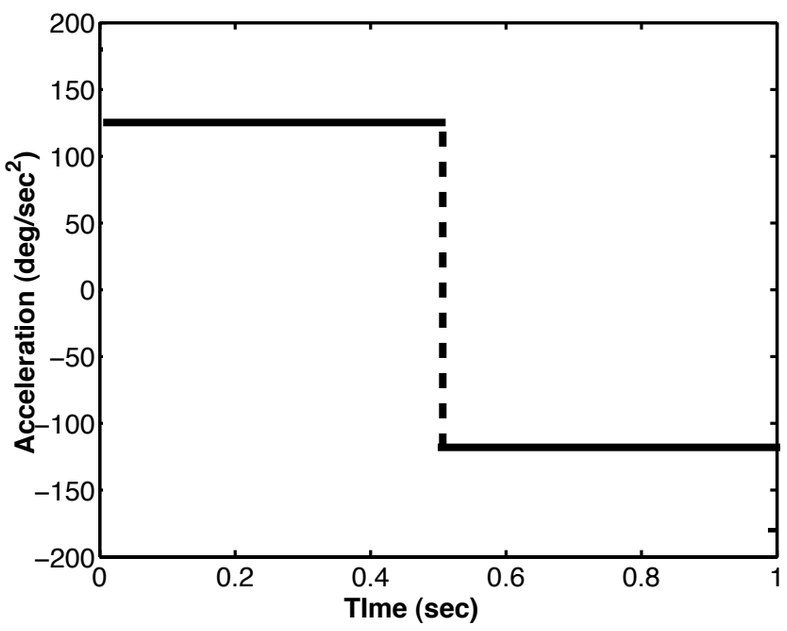
Getting There As Fast As Possible

Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories

Piecewise constant max accelerations







Triangular velocity profile

Not minimum jerk...
...but fast!

Moving Through Via Points

You could solve for a single high-order polynomial that hits all your via points.

This approach yields a nice continuously differentiable curve.

However, it is intractable when many via points are used because the linear system's dimension become very large.

Moving Through Via Points

Instead, use low-order polynomials for trajectory segments between adjacent via points.

Ensure that position, velocity, and acceleration constraints are satisfied at the via points, where we switch from one polynomial to the next.

Final conditions for one polynomial become the initial conditions for the next!

MEAM 520

More Trajectory Planning

Katherine J. Kuchenbecker, Ph.D.

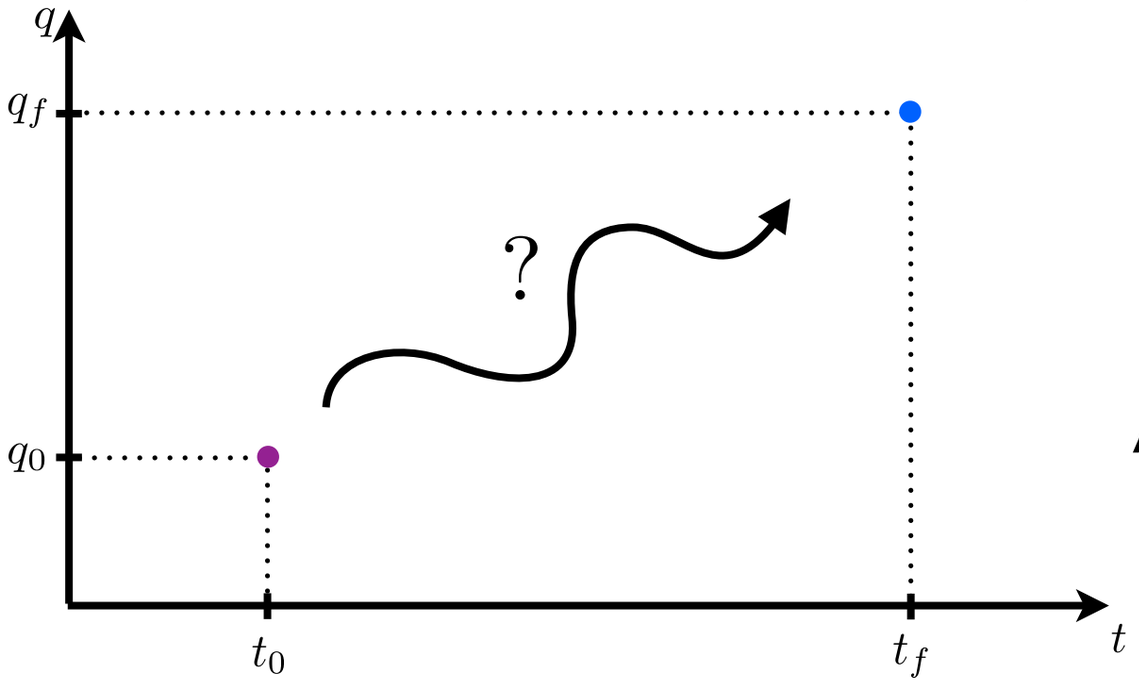
General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 11: October 3, 2013

Trajectory Planning



	Initial Conditions	Final Conditions
Position	$q(t_0) = q_0$	$q(t_f) = q_f$
Velocity	$\dot{q}(t_0) = v_0$	$\dot{q}(t_f) = v_f$
Acceleration	$\ddot{q}(t_0) = \alpha_0$	$\ddot{q}(t_f) = \alpha_f$
Jerk	$\dddot{q}(t_0) \neq \infty$	$\dddot{q}(t_f) \neq \infty$

First-Order Polynomial (Line)

$$q(t) = a_0 + a_1 t$$

Third-Order Polynomial (Cubic)

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Solving for Coefficients

$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Fifth-Order Polynomial (Quintic)

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Sequencing Successive Low-Order Polynomials Through Multiple Via Points

Linear Segment with Parabolic Blends (LSPB, 1 Line + 2 Quadratics)

$$q(t) = b_0 + b_1 t + b_2 t^2 \quad q(t) = a_0 + a_1 t \quad q(t) = c_0 + c_1 t + c_2 t^2$$

Minimum Time Trajectory (Bang-Bang, 2 Quadratics)

$$q(t) = b_0 + b_1 t + b_2 t^2 \quad q(t) = c_0 + c_1 t + c_2 t^2$$

Activity 2

MEAM 520 – October 3, 2013 – Prof. K. J. Kuchenbecker – University of Pennsylvania

Trajectory Planning Questions

1. The equation $q(t) = a_0 + a_1 t$ defines a line. Solve for the coefficients a_0 and a_1 that satisfy the initial and final position constraints of $q(t_0) = q_0$ and $q(t_f) = q_f$.

2. We discussed using linear algebra to solve for the coefficients of the cubic polynomial that satisfies the specified conditions. Will there always be a solution? If no, when does it fail?

$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

3. For which of the five trajectory types can q leave the interval between q_0 and q_f for the time span $t_0 \leq t \leq t_f$? Explain.

Work with one or two partners to answer the first three questions.

1. The equation $q(t) = a_0 + a_1 t$ defines a line. Solve for the coefficients a_0 and a_1 that satisfy the initial and final position constraints of $q(t_0) = q_0$ and $q(t_f) = q_f$.

$$q(t) = a_0 + a_1 t$$

$$q_0 = a_0 + a_1 t_0$$

$$q_f = a_0 + a_1 t_f$$

$$\boxed{a_0 = q_0 - \frac{q_f - q_0}{t_f - t_0} \cdot t_0} \qquad \boxed{a_1 = \frac{q_f - q_0}{t_f - t_0}}$$

2. We discussed using linear algebra to solve for the coefficients of the cubic polynomial that satisfies the specified conditions. Will there always be a solution? If no, when does it fail?

$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

fails when $t_f = t_0$

/Users/kuchenbe/Documents/teaching/meam 520/lectures/11 trajectories/polynomials/analyze_cubic_polynomial.m

EDITOR | PUBLISH | VIEW

```
1   %% Clear the workspace.
2 - clear
3
4
5   %% Define the variables as symbols that represent real numbers.
6 - syms t0 tf q0 qf v0 vf real
7
8
9   %% Solve for the cubic polynomial coefficients that meet these conditions.
10
11  % Put initial and final conditions into a column vector.
12 - conditions = [q0 v0 qf vf]'
13
14  % Put time elements into matrix.
15 - mat = [1    t0   t0^2     t0^3;
16           0     1   2*t0   3*t0^2;
17           1    tf   tf^2     tf^3;
18           0     1   2*tf   3*tf^2]
19
20  % Solve for coefficients.
21 - coeffs = mat \ conditions
22
23  % Pull individual coefficients out.
24 - a0 = coeffs(1);
25 - a1 = coeffs(2);
26 - a2 = coeffs(3);
27 - a3 = coeffs(4);
28
29
30  %% Figure out when there is a solution.
31
32  % There is no solution when the determinant of the matrix equals zero.
33
34  % Calculate determinant of matrix.
35 - detmat = det(mat)
36
37  %detmat = simplify(detmat)
38
```

script | Ln 12 Col 28

3. For which of the five trajectory types can q leave the interval between q_0 and q_f for the time span $t_0 \leq t \leq t_f$? Explain.

First-Order Polynomial (Line) Does not leave interval.

$$q(t) = a_0 + a_1 t$$

Third-Order Polynomial (Cubic) Could leave interval*.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

*Depends on initial and final velocities. When both are zero, does not leave interval.

Fifth-Order Polynomial (Quintic) Could leave interval**.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

**Depends on initial and final velocities and accelerations. When all are zero, does not leave interval.

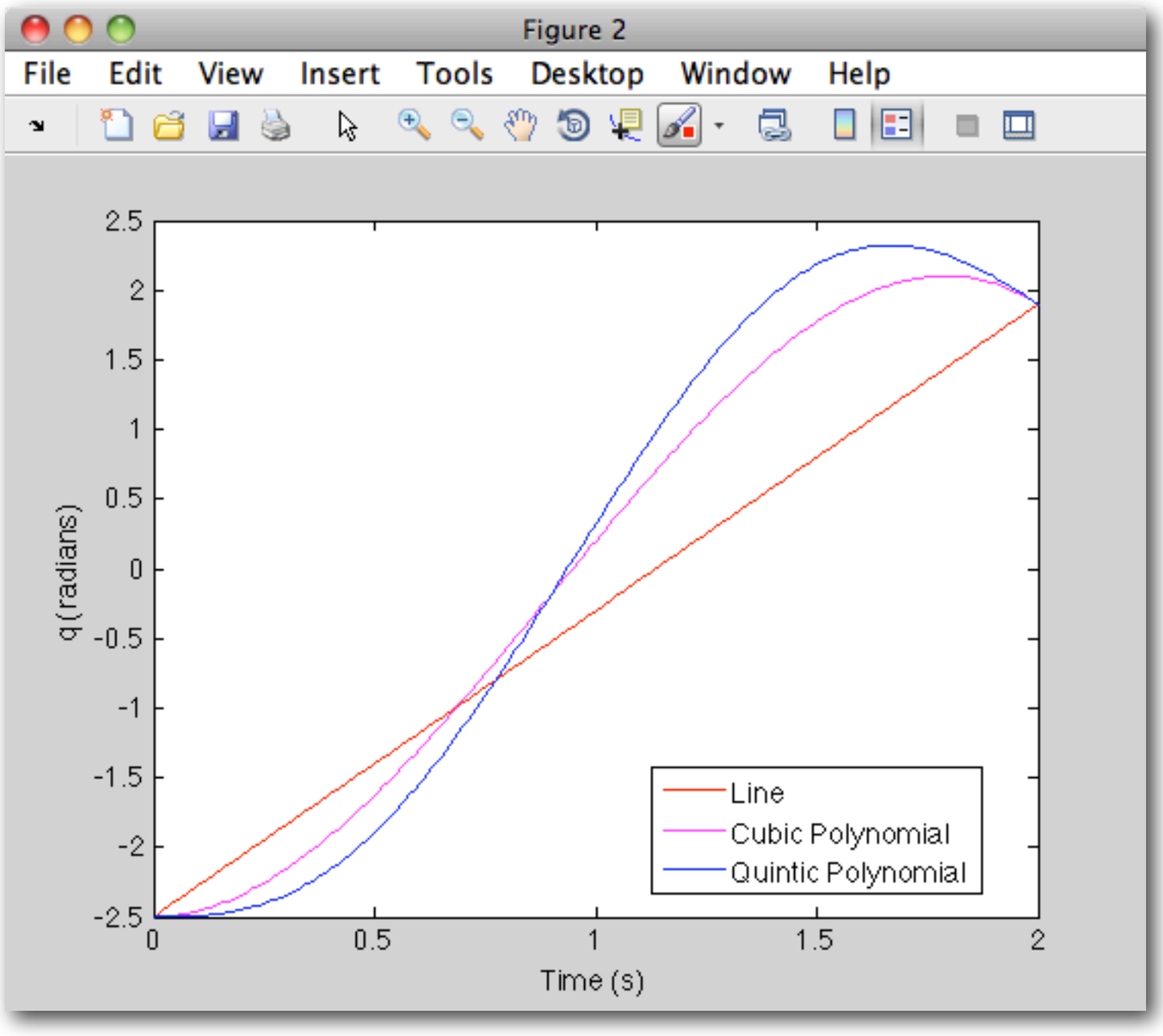
Linear Segment with Parabolic Blends (LSPB, 1 Line + 2 Quadratics) Could leave interval*.

$$q(t) = b_0 + b_1 t + b_2 t^2 \quad q(t) = a_0 + a_1 t \quad q(t) = c_0 + c_1 t + c_2 t^2$$

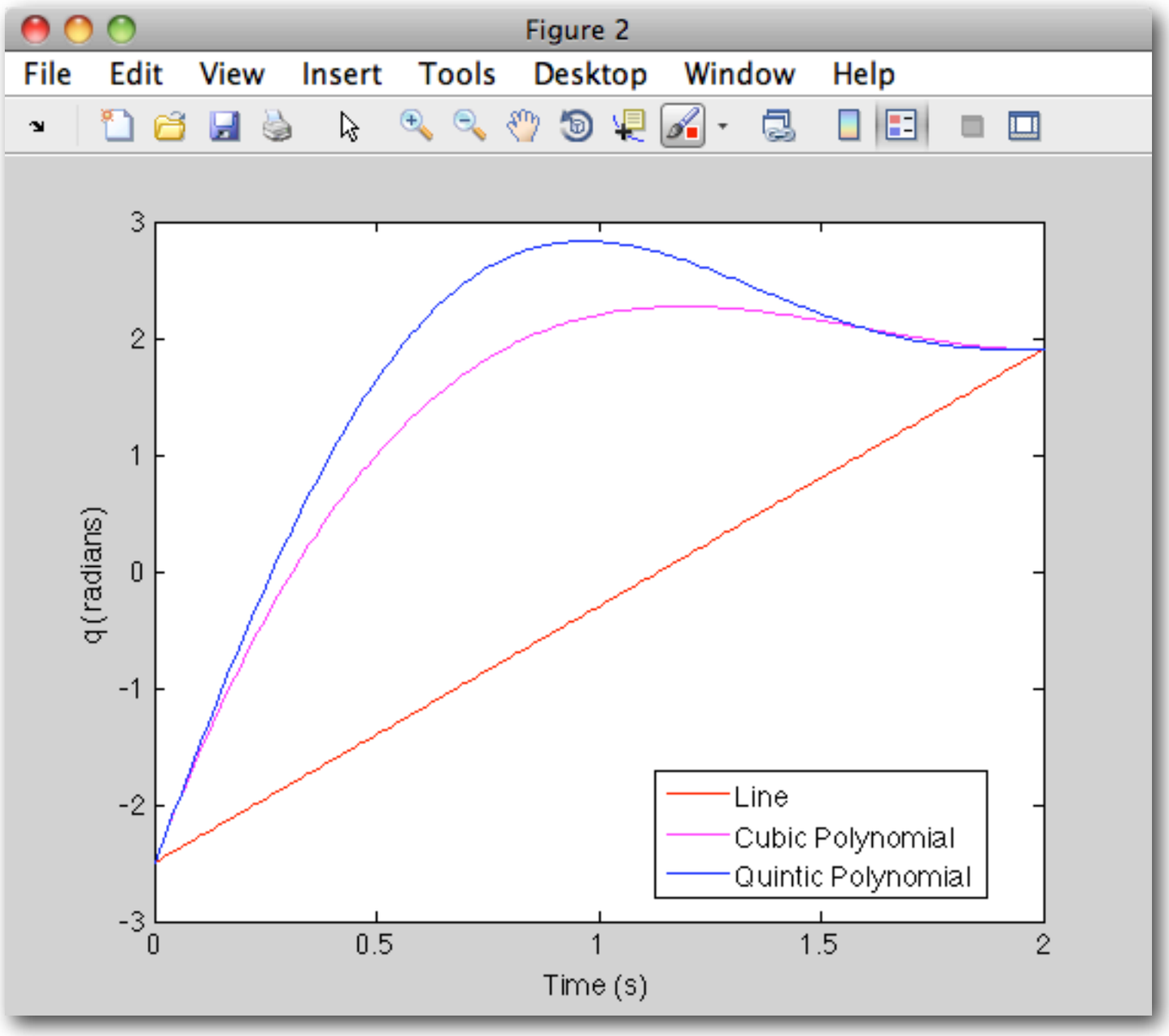
Minimum Time Trajectory (Bang-Bang, 2 Quadratics) Could leave interval*.

$$q(t) = b_0 + b_1 t + b_2 t^2 \quad q(t) = c_0 + c_1 t + c_2 t^2$$

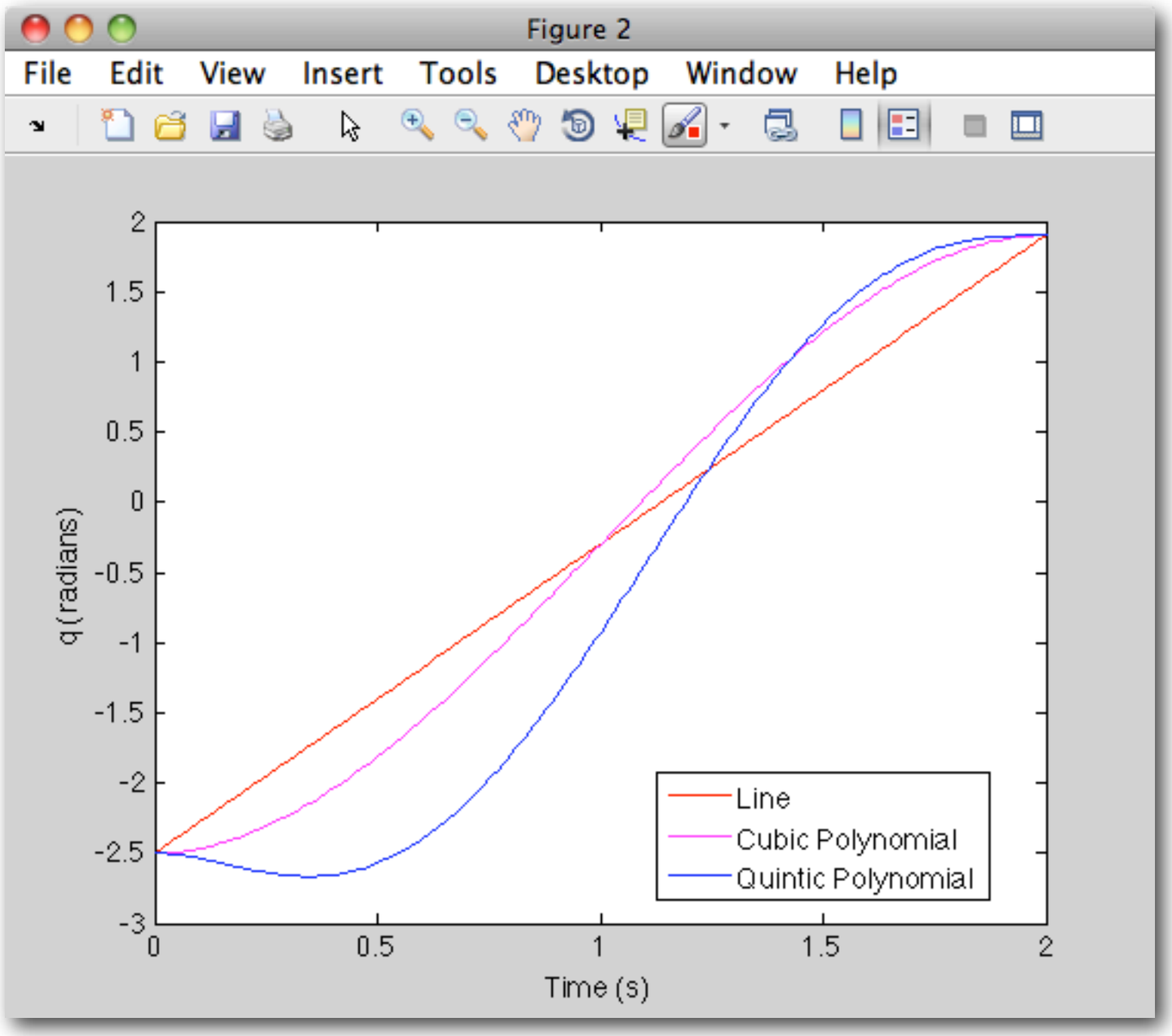
Final velocity less than zero



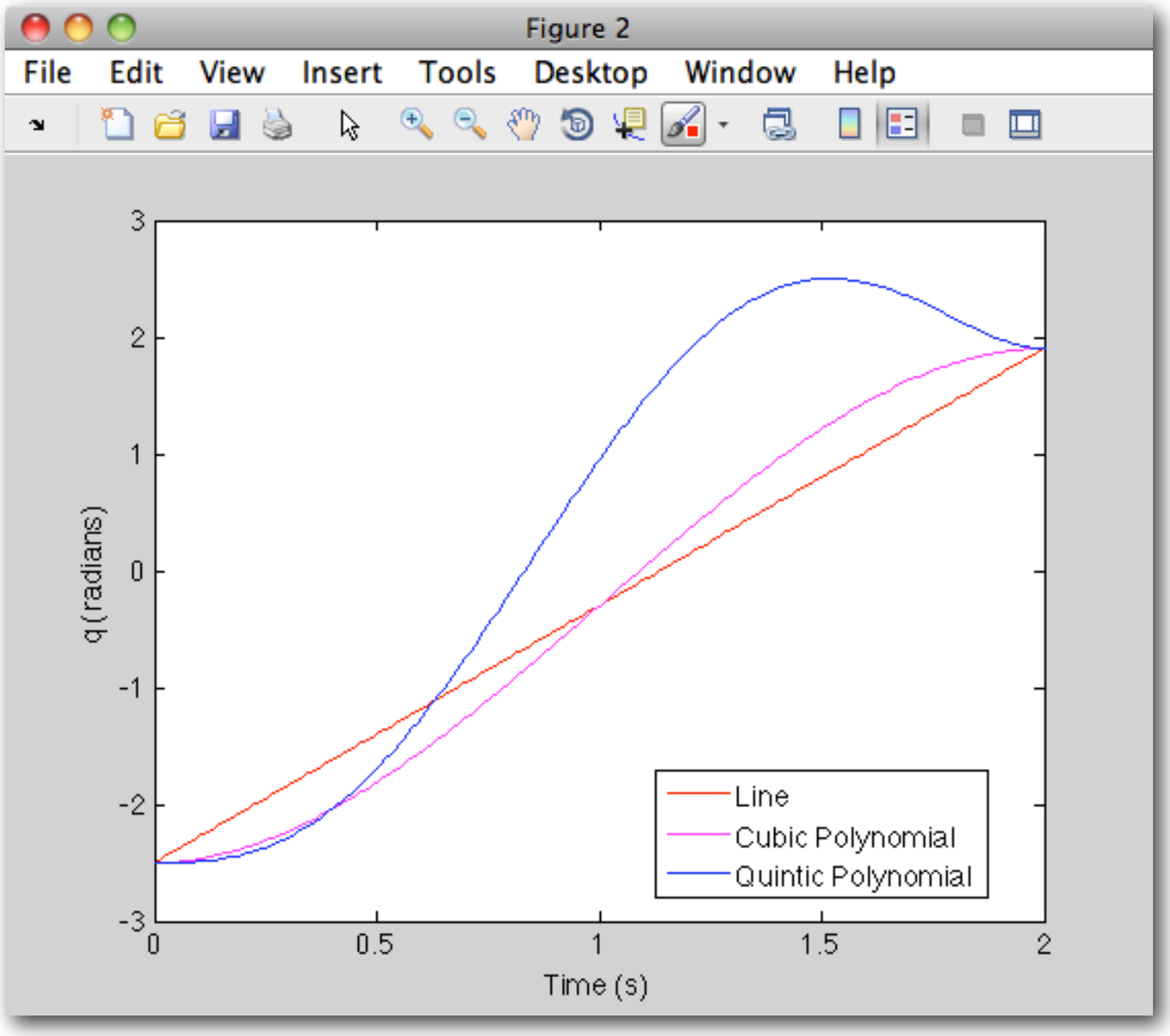
Initial velocity greater than zero and large



Initial acceleration less than zero



Final acceleration greater than zero



Activity 3

MEAM 520 – October 3, 2013 – Prof. K. J. Kuchenbecker – University of Pennsylvania

4. Why would one ever use a line or a cubic polynomial instead of a quintic polynomial?

5. How does the idea of sequencing low-order polynomials such as cubics through multiple via points relate to LSPB and Bang-Bang trajectories?

6. Set up the equations to solve for all the coefficients of a general LSPB given initial time t_0, final time t_f, initial position q_0, final position q_f, initial velocity v_0, final velocity v_f, and blend duration t_b.

Work with one or two partners to answer the last three questions.

$$q(t) = b_0 + b_1 t + b_2 t^2 \quad q(t) = a_0 + a_1 t \quad q(t) = c_0 + c_1 t + c_2 t^2$$

4. Why would one ever use a line or a cubic polynomial instead of a quintic polynomial?

- Want constant velocity (line).
- Your robot is sufficiently rigid, so you don't care about minimal jerk.
- Need lower computational complexity, e.g., real-time calculations on a microcontroller.
- Need lower memory usage, e.g., implementation on a microcontroller.
- Want to limit maximum speed.
- More ideas from class?

5. How does the idea of sequencing low-order polynomials such as cubics through multiple via points relate to LSPB and Bang-Bang trajectories?

- A linear segment with parabolic blends *is* a sequence of low-order polynomials: quadratic + line + quadratic.

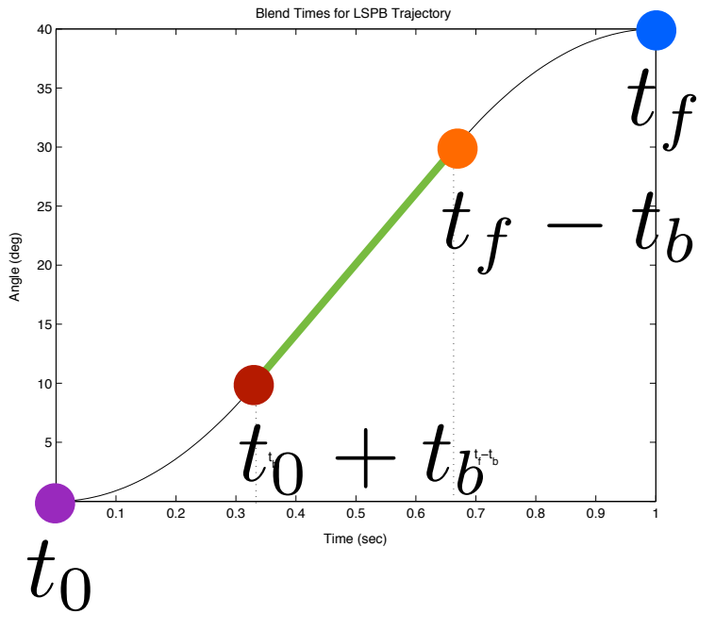
- A bang-bang trajectory *is* a sequence of low-order polynomials: quadratic + quadratic.

- But, for LSPB and BB we don't care about the particular position or velocity of the robot at the switching times. We just require position and velocity to be continuous at these points.

6. Set up the equations to solve for all the coefficients of a general LSPB given initial time t_0, final time t_f, initial position q_0, final position q_f, initial velocity v_0, final velocity v_f, and blend duration t_b.

$$q(t) = b_0 + b_1 t + b_2 t^2 \qquad q(t) = a_0 + a_1 t \qquad q(t) = c_0 + c_1 t + c_2 t^2$$
$$\dot{q}(t) = b_1 + 2b_2 t \qquad \dot{q}(t) = a_1 \qquad \dot{q}(t) = c_1 + 2c_2 t$$

8 parameters – need 8 equations



Position and velocity at four points in time

$$q_0 = b_0 + b_1 t_0 + b_2 t_0^2$$
$$v_0 = b_1 + 2b_2 t_0$$
$$b_0 + b_1(t_0 + t_b) + b_2(t_0 + t_b)^2 = a_0 + a_1(t_0 + t_b)$$
$$b_1 + 2b_2(t_0 + t_b) = a_1$$
$$\ldots$$

MEAM 520

Velocity Kinematics and Jacobians

Katherine J. Kuchenbecker, Ph.D.

General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

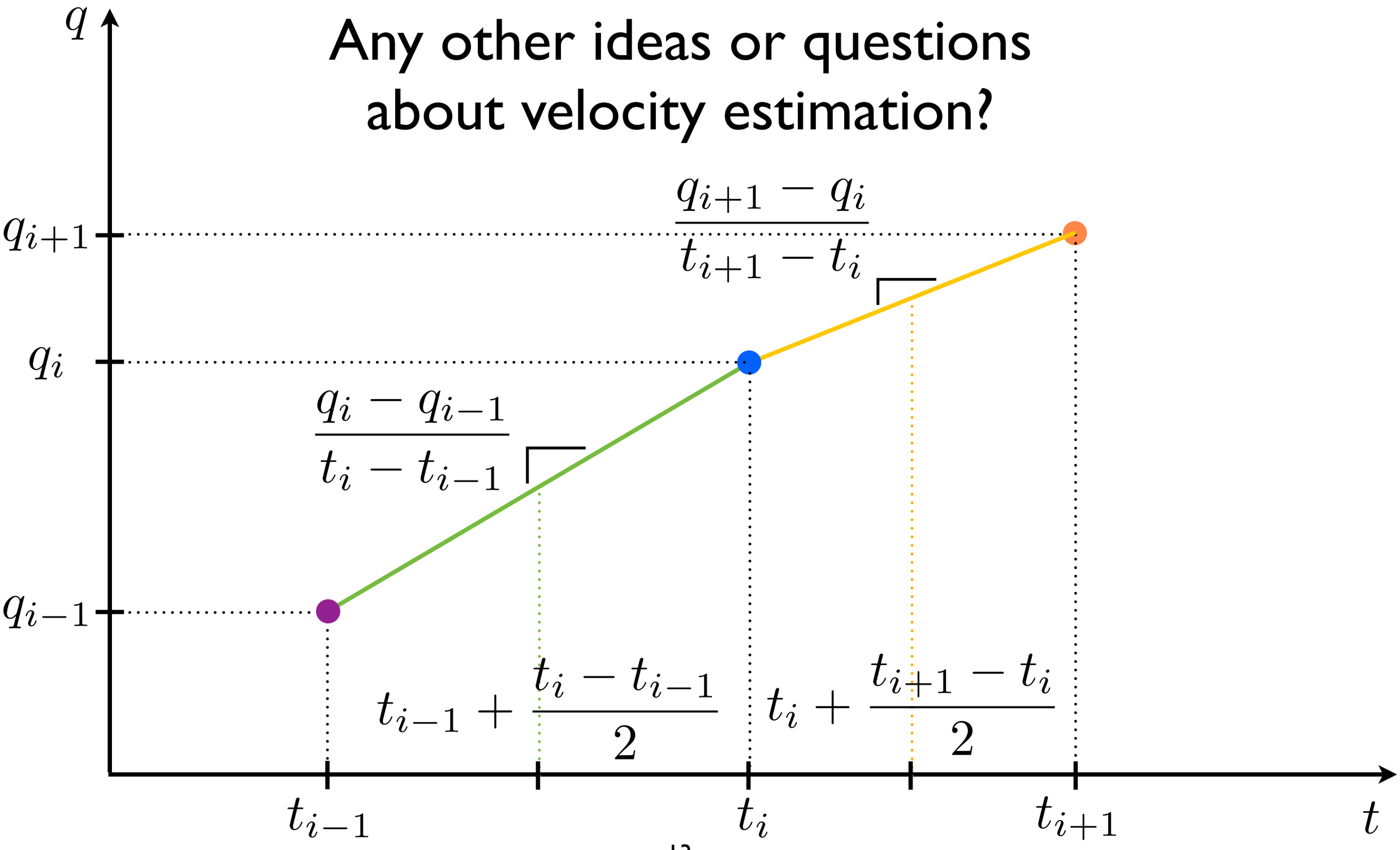
GRASP LABORATORY

Lecture 13: October 15, 2013

How many velocity estimates do you get? $n-1$

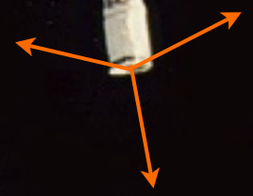
What time should you associate with each velocity?

Any other ideas or questions about velocity estimation?



Attach a coordinate frame rigidly to the object.

Linear / translational velocity quantifies how the position of the frame's origin changes over time.



Angular / rotational velocity quantifies how the frame's orientation changes over time.

Many representations for orientation

Finite rotations are not vector quantities

What is the time derivative of a rotation matrix?

$$R = R(\theta) \in SO(3)$$

$$\text{e.g., } R(\theta) = R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\dot{R} = \frac{dR}{dt} = ?$$

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta}\frac{d\theta}{dt}$$

✓

?

What is the derivative of a rotation matrix?

$$\frac{dR}{d\theta} = ?$$

$$R\,R^T = I$$

$$\frac{d}{d\theta}\left(R\,R^T\right) = \frac{d}{d\theta}(I)$$

product rule

$$\frac{dR}{d\theta}R^T + R\frac{dR^T}{d\theta} = 0$$

Interesting equation! Sum of two matrices equals **zero**.

$$\text{define } S = \frac{dR}{d\theta}R^T \qquad S^T = \left(\frac{dR}{d\theta}R^T\right)^T = R\frac{dR^T}{d\theta}$$

$$S + S^T = 0$$

Even more interesting! Sum of a matrix and its transpose equals **zero**.
What does that mean about the matrix S?

Skew-Symmetric Matrices

$$S + S^T = 0$$

$$S = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

What do you know about the elements of S?

Zeros along the diagonal.
Negative values across the diagonal.

$$S = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}$$

Skew-Symmetric Matrices

$$S + S^T = 0 \qquad S = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}$$

Define the operator S

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \qquad S(\vec{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

The operator S is linear

$$S(\alpha\vec{a} + \beta\vec{b}) = \alpha S(\vec{a}) + \beta S(\vec{b})$$

But what does S do?

$$S(\vec{a})\,\vec{p} = ?$$

Skew-Symmetric Matrices

$$S(\vec{a})\,\vec{p} = ? = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$= \begin{bmatrix} -a_z p_y + a_y p_z \\ a_z p_x - a_x p_z \\ -a_y p_x + a_x p_y \end{bmatrix}$$

$$= \begin{bmatrix} a_y p_z - a_z p_y \\ a_z p_x - a_x p_z \\ a_x p_y - a_y p_x \end{bmatrix}$$

$$S(\vec{a})\vec{p} = \vec{a} \times \vec{p}$$

Skew-symmetric matrices are a compact way to represent a cross-product between vectors.

What is the derivative of a rotation matrix?

$$\frac{dR}{d\theta} = ? \qquad \text{define } S = \frac{dR}{d\theta} R^T \qquad S + S^T = 0$$

This matrix is skew-symmetric.

It also contains the quantity we are seeking.

Multiply both sides on the right by R.

$$S\,R = \frac{dR}{d\theta} R^T R \qquad R^T R = I$$

$$\boxed{\frac{dR}{d\theta} = S\,R}$$

Computing the derivative of a rotation matrix R is equivalent to multiplying that matrix R by a skew-symmetric matrix S.

$$\boxed{\begin{aligned} S &= \frac{dR}{d\theta} R^T \\ \frac{dR}{d\theta} &= S\, R \end{aligned}}$$

Example

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad \dot{R}_{x,\theta} = ?$$

$$\dot{R}_{x,\theta} = \frac{dR_{x,\theta}}{dt} = \frac{dR_{x,\theta}}{d\theta}\frac{d\theta}{dt} = S R_{x,\theta}\frac{d\theta}{dt}$$

$$S = ? = \frac{dR_{x,\theta}}{d\theta} R_{x,\theta}^T$$

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

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = S(\hat{i}) \qquad S(\vec{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$\boxed{\begin{aligned} S &= \frac{dR}{d\theta}R^T \\ \frac{dR}{d\theta} &= S\,R \end{aligned}}$$

Example

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad \dot{R}_{x,\theta} = ?$$

$$\dot{R}_{x,\theta} = \frac{dR_{x,\theta}}{dt} = \frac{dR_{x,\theta}}{d\theta}\frac{d\theta}{dt} = SR_{x,\theta}\frac{d\theta}{dt}$$

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = S(\hat{i})$$

The skew-symmetric matrix S defines the axis about which rotation is occurring.

$$\dot{R}_{x,\theta} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\dot{\theta}\sin\theta & -\dot{\theta}\cos\theta \\ 0 & \dot{\theta}\cos\theta & -\dot{\theta}\sin\theta \end{bmatrix}$$

$$\dot{R}_{x,\theta} = S(\hat{i})R_{x,\theta}\dot{\theta}$$

$$\dot{R}_{x,\theta} = S(\dot{\theta}\hat{i})R_{x,\theta}$$

$$\vec{\omega} = \dot{\theta}\hat{i}$$

$$\dot{R}_{x,\theta} = S(\vec{\omega})R_{x,\theta}$$

In general, you simply form S from the angular velocity vector and don't need to differentiate the matrix.

What is this useful for?

Understanding how angular velocities combine on a robotic manipulator.

Calculating the linear velocity of the end-effector of a robot.

Understanding how angular velocities combine on a robotic manipulator

See SHV 4.4: Addition of Angular Velocities

$$R_2^0(t) = R_1^0(t) R_2^1(t)$$

$$\omega_2^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1$$

The angular velocity of frame 2 relative to frame 0
is equal to the angular velocity of frame 1 relative to frame 0, expressed in frame 0,
plus the angular velocity of frame 2 relative to frame 1, expressed in frame 0

You can add angular velocity vectors!

Calculating the linear velocity of the end-effector of a robot

See SHV 4.5: Linear Velocity of a Point Attached to a Moving Frame

$$p^0 = R_1^0(t)p^1 + o_1^0(t)$$

$$\dot{p}^0 = \dot{R}_1^0 p^1 + \dot{o}_1^0$$

$$\dot{p}^0 = S(\omega^0)R_1^0 p^1 + \dot{o}_1^0$$

$$\dot{p}^0 = \omega^0 \times p^0 + \dot{o}_1^0$$

You can calculate the linear velocity of the end-effector from the angular velocity of its frame, its position relative to the rotational axis, and the linear velocity of its frame's origin.

How do the velocities of the joints affect
the linear and angular velocity of the end-effector?

These quantities are related by the **Jacobian**,
a matrix that generalizes the notion of an ordinary
derivative of a scalar function.

Jacobians are useful for planning and executing smooth trajectories, determining singular configurations, executing coordinated anthropomorphic motion, deriving dynamic equations of motion, and transforming forces and torques from the end-effector to the manipulator joints.

explore how **changes** in joint values affect the end-effector movement

could have **N joints**, but only **six** end-effector velocity terms (xyzpts)

The **Jacobian** matrix lets us calculate how joint velocities translate into end-effector velocities (depends on configuration)

look at it in two parts - position and orientation

$$v_n^0 = J_v \dot{q} \qquad \omega_n^0 = J_\omega \dot{q}$$

How do we calculate the position Jacobian?

for

$$x(t) = f(q_1(t), q_2(t), \ldots, q_n(t))$$

the time derivative can be found using

$$\frac{dx}{dt} = \sum_{i=1}^{n} \frac{\delta x}{\delta q_i} \frac{dq_i}{dt}$$

For an n-dimensional joint space and a cartesian workspace, the position Jacobian is a 3xn matrix composed of the partial derivatives of the end-effector position with respect to each joint variable.

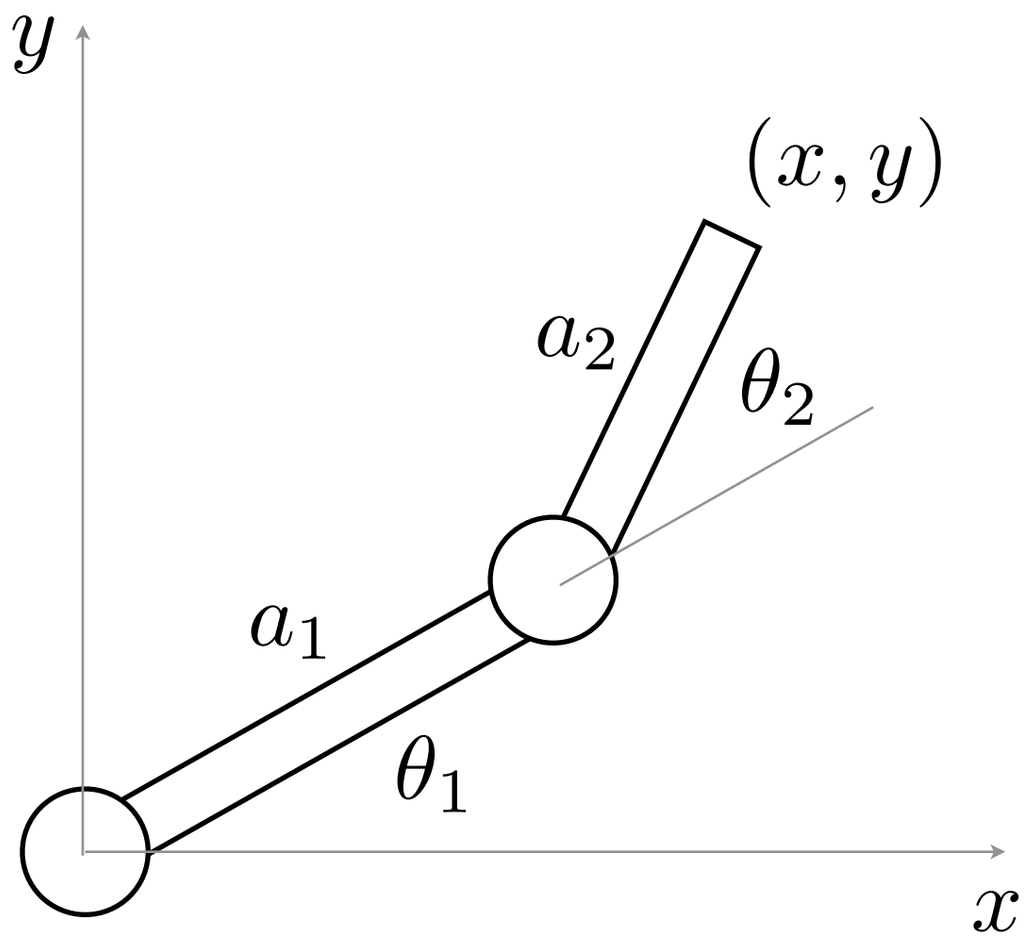
$$\mathbf{J}_p = \begin{bmatrix} \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \cdots & \frac{\delta x}{\delta q_n} \\ \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \cdots & \frac{\delta y}{\delta q_n} \\ \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \cdots & \frac{\delta z}{\delta q_n} \end{bmatrix}$$

$$\dot{\mathbf{p}} = \mathbf{J}_p(q)\ \dot{\mathbf{q}}$$

endpoint velocity

Jacobian matrix

joint velocity



From the forward kinematics, we can extract the position vector from the last column of the transform matrix:

$$\mathbf{d}_2^0 = \begin{bmatrix} a_2c_{12} + a_1c_1 \\ a_2s_{12} + a_1s_1 \\ 0 \end{bmatrix}$$

Taking the partial derivative with respect to each joint variable produces the Jacobian:

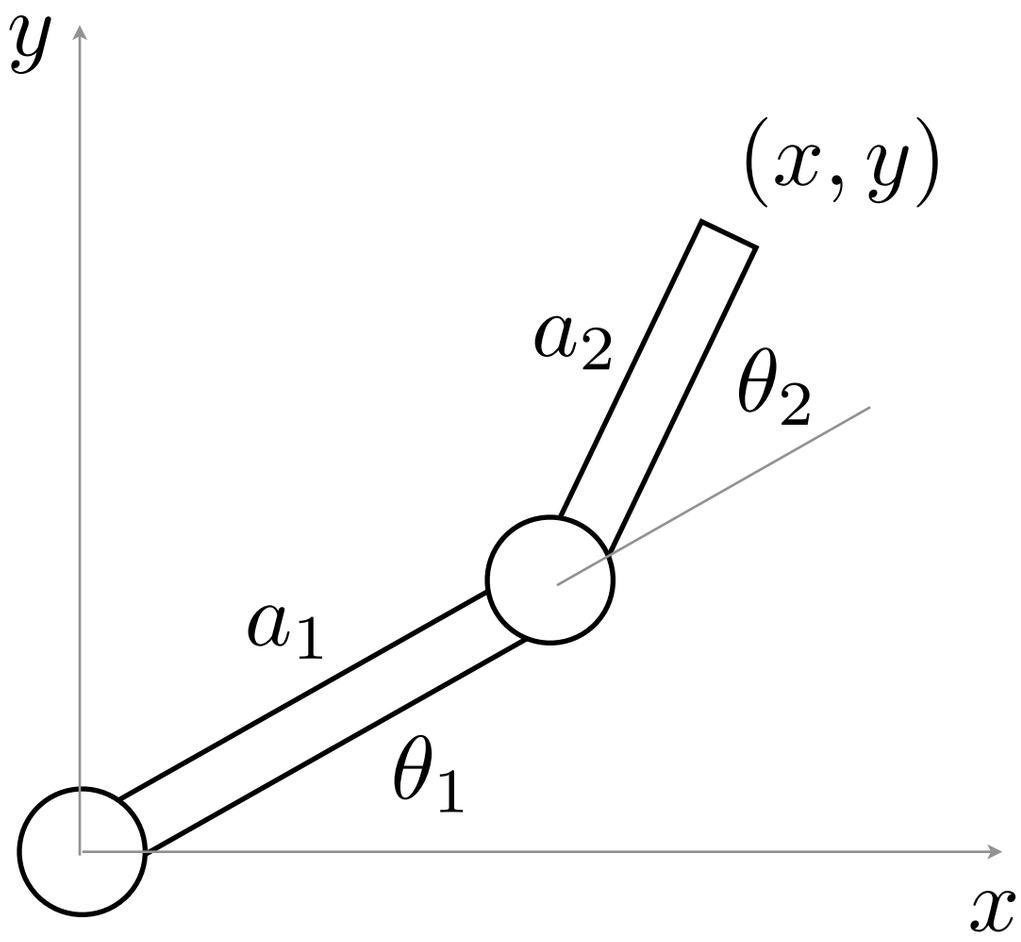
$$\mathbf{J}_p = \begin{bmatrix} \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \cdots & \frac{\delta x}{\delta q_n} \\ \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \cdots & \frac{\delta y}{\delta q_n} \\ \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \cdots & \frac{\delta z}{\delta q_n} \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix}$$

which relates instantaneous joint velocities to endpoint velocities

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

The Position Jacobian : Planar RR



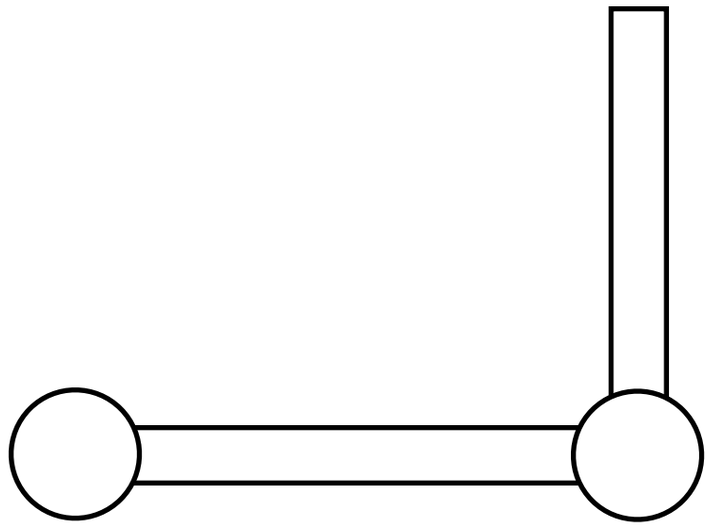
$$\mathbf{J} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix}$$

$$\theta_1 = 0, \;\; \theta_2 = \pi/2$$

$$\dot{x} = -a_2 \dot{\theta}_1 - a_2 \dot{\theta}_2$$

$$\dot{y} = a_1 \dot{\theta}_1$$

$$\dot{z} = 0$$



MEAM 520

Jacobians and Singularities

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 14: October 17, 2013

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta}\frac{d\theta}{dt}$$

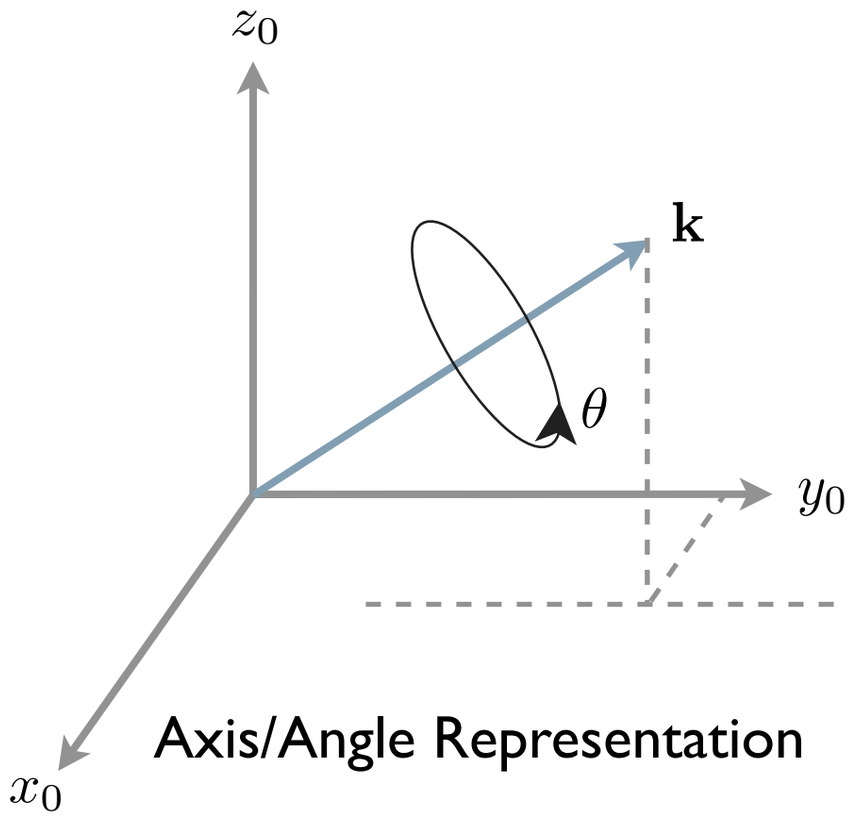
✓

?

$$S + S^T = 0$$

$$S(\vec{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$S(\vec{a})\vec{p} = \vec{a} \times \vec{p}$$



Axis/Angle Representation

$$\frac{dR}{d\theta} = S(\hat{\omega})\,R$$

The skew-symmetric matrix S defines the axis about which rotation is occurring.

$$\frac{dR}{dt} = S(\vec{\omega})\,R$$

In general, you simply form S from the angular velocity vector and don't need to differentiate the matrix.

$$\vec{\omega}_2^0 = \vec{\omega}_{0,1}^0 + R_1^0\vec{\omega}_{1,2}^1$$

$$\dot{\vec{p}}^0 = S(\vec{\omega}^0)R_1^0\vec{p}^1 + \dot{\vec{o}}_1^0$$

explore how **changes** in joint values affect the end-effector movement

could have **N joints**, but only **six** end-effector velocity terms (xyzpts)

The **Jacobian** matrix lets us calculate how joint velocities translate into end-effector velocities (depends on configuration)

look at it in two parts - position and orientation

$$v_n^0 = J_v \dot{q} \qquad \omega_n^0 = J_\omega \dot{q}$$

How do we calculate the linear velocity (position) Jacobian?

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \vec{p} = f(\vec{q})$$

endpoint position

forward kinematics

nonlinear

joint values

$$\dot{\vec{p}} = J_v(\vec{q})\,\dot{\vec{q}}$$

endpoint velocity

Jacobian matrix

linear approximation of derivative near a given point

joint values

joint velocity

For an n-dimensional joint space and a cartesian workspace, the position Jacobian is a 3 x n matrix composed of the partial derivatives of the end-effector position with respect to each joint variable.

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

$$\dot{\vec{p}} = J_v(\vec{q})\,\dot{\vec{q}}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

distance/time such as m/s

Evaluate the Jacobian at the robot's current pose

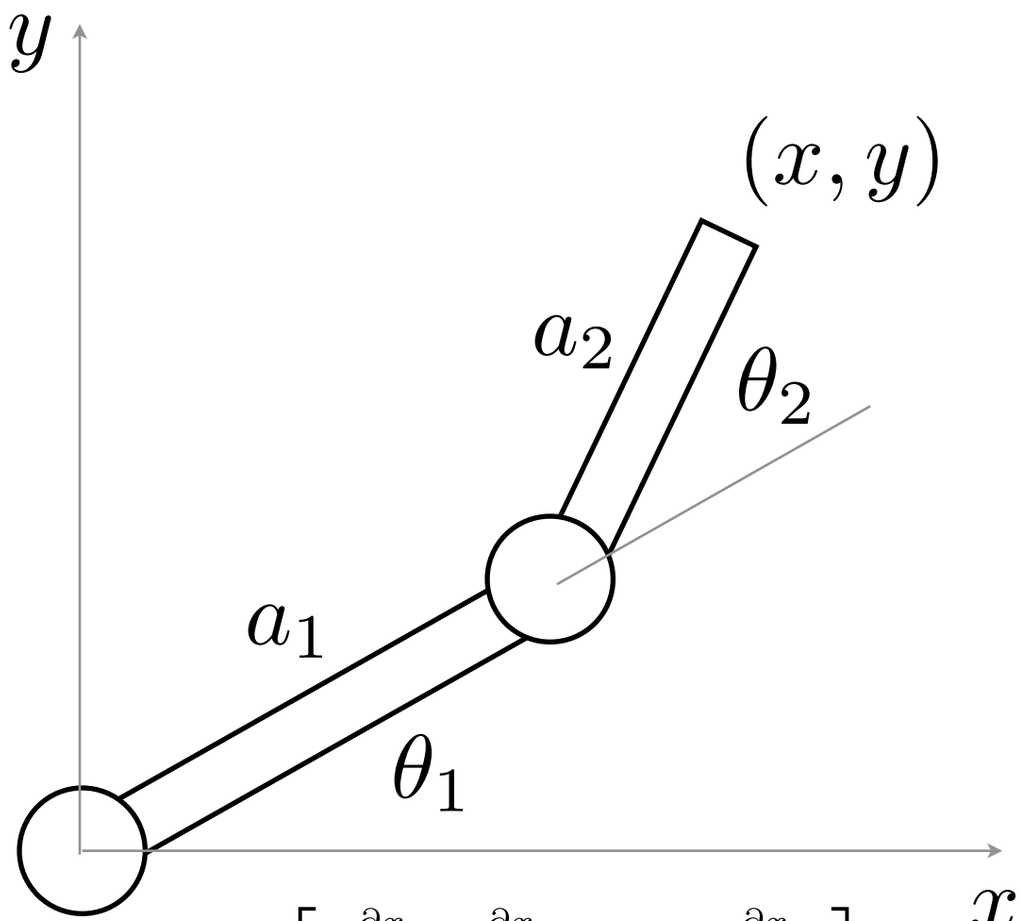
revolute: angle/time such as rad/s

prismatic: distance/time such as m/s

What units do the entries of the Jacobian have?

if joint *i* is revolute, column *i* is in distance/angle such as m/rad or m

if joint *i* is prismatic and units match, column *i* is unitless



From the forward kinematics, we can extract the position vector from the last column of the transform matrix:

$$d_2^0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_2 c_{12} + a_1 c_1 \\ a_2 s_{12} + a_1 s_1 \\ 0 \end{bmatrix}$$

Taking the partial derivative with respect to each joint variable produces the Jacobian:

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

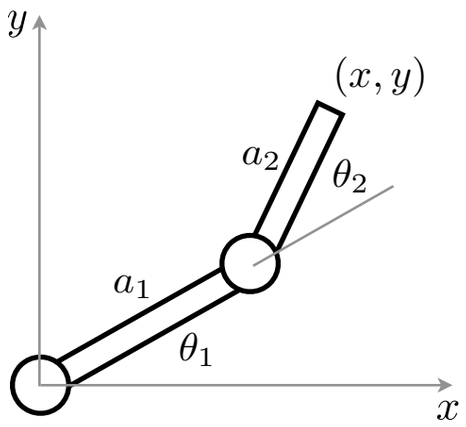
$$J_v(\vec{q}) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix}$$

which relates instantaneous joint velocities to endpoint velocities

This mapping depends on the robot's current pose!

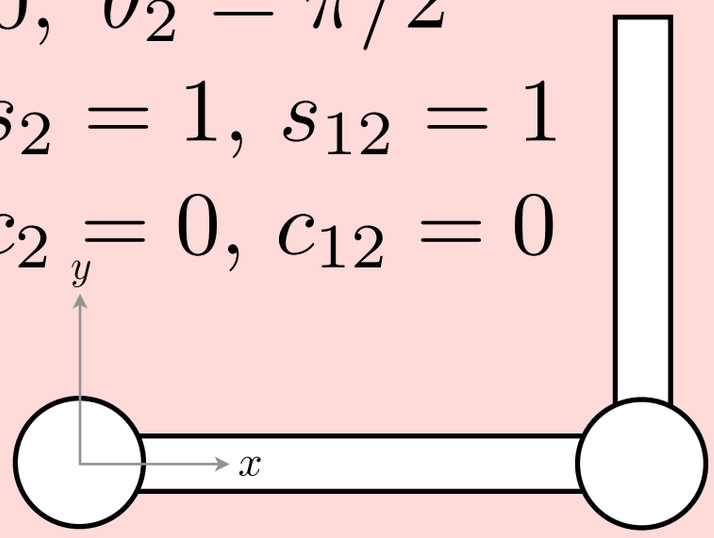
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

The Position Jacobian : Planar RR



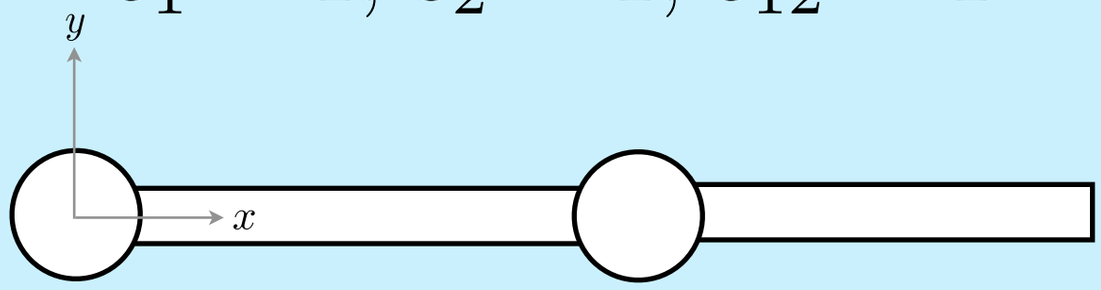
$$J_v(\vec{q}) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = \pi/2$$
$$s_1 = 0,\ s_2 = 1,\ s_{12} = 1$$
$$c_1 = 1,\ c_2 = 0,\ c_{12} = 0$$



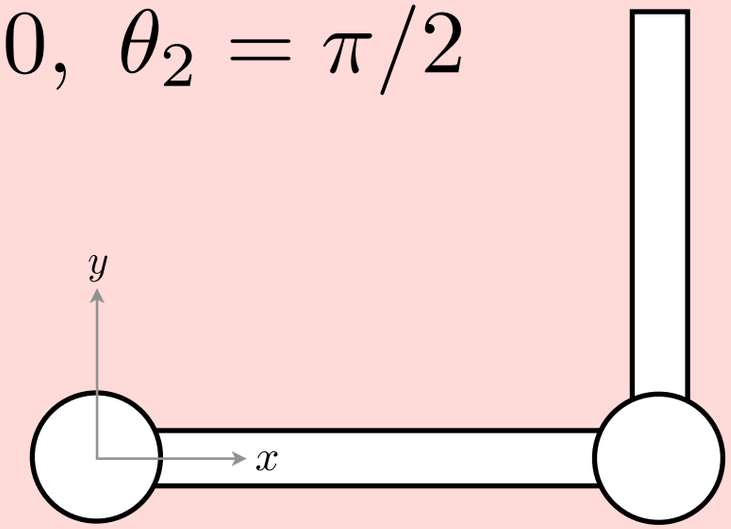
$$J_v([0 \quad \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = 0$$
$$s_1 = 0,\ s_2 = 0,\ s_{12} = 0$$
$$c_1 = 1,\ c_2 = 1,\ c_{12} = 1$$



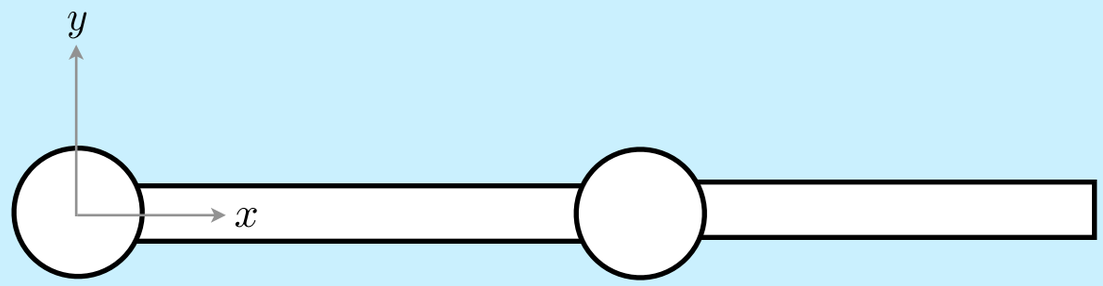
$$J_v([0 \quad 0]^T) = \begin{bmatrix} 0 & 0 \\ a_1 + a_2 & a_2 \\ 0 & 0 \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = \pi/2$$



$$J_v([0 \quad \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = 0$$



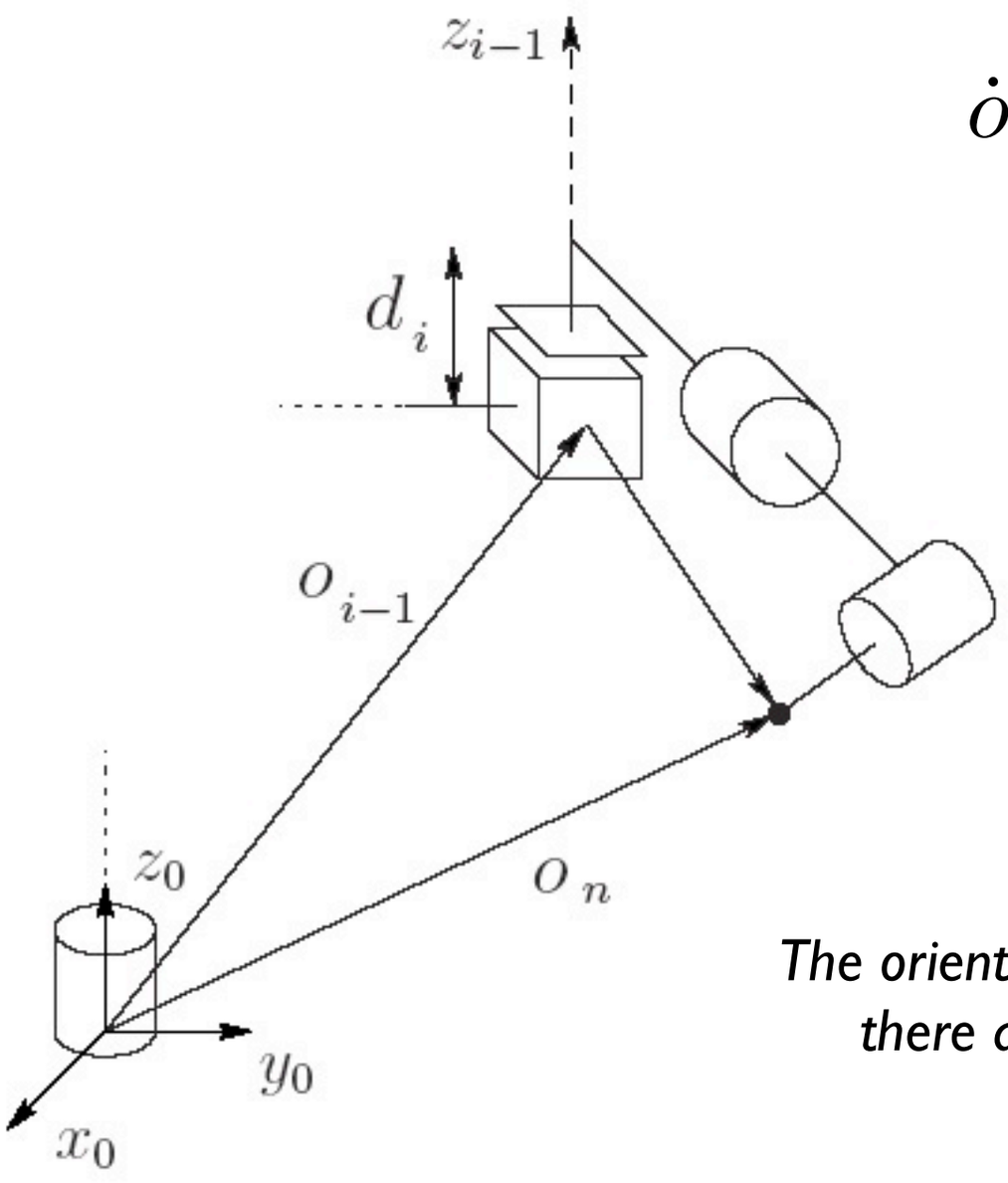
$$J_v([0 \quad 0]^T) = \begin{bmatrix} 0 & 0 \\ a_1 + a_2 & a_2 \\ 0 & 0 \end{bmatrix}$$

$$\dot{\vec{p}} = J_v(\vec{q})\,\dot{\vec{q}}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -a_2\dot{\theta}_1 - a_2\dot{\theta}_2 \\ a_1\dot{\theta}_1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ (a_1 + a_2)\dot{\theta}_1 + a_2\dot{\theta}_2 \\ 0 \end{bmatrix}$$

Prismatic Joints



$$\dot{o}_n^0 = \dot{d}_i R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \dot{d}_i z_{i-1}^0$$

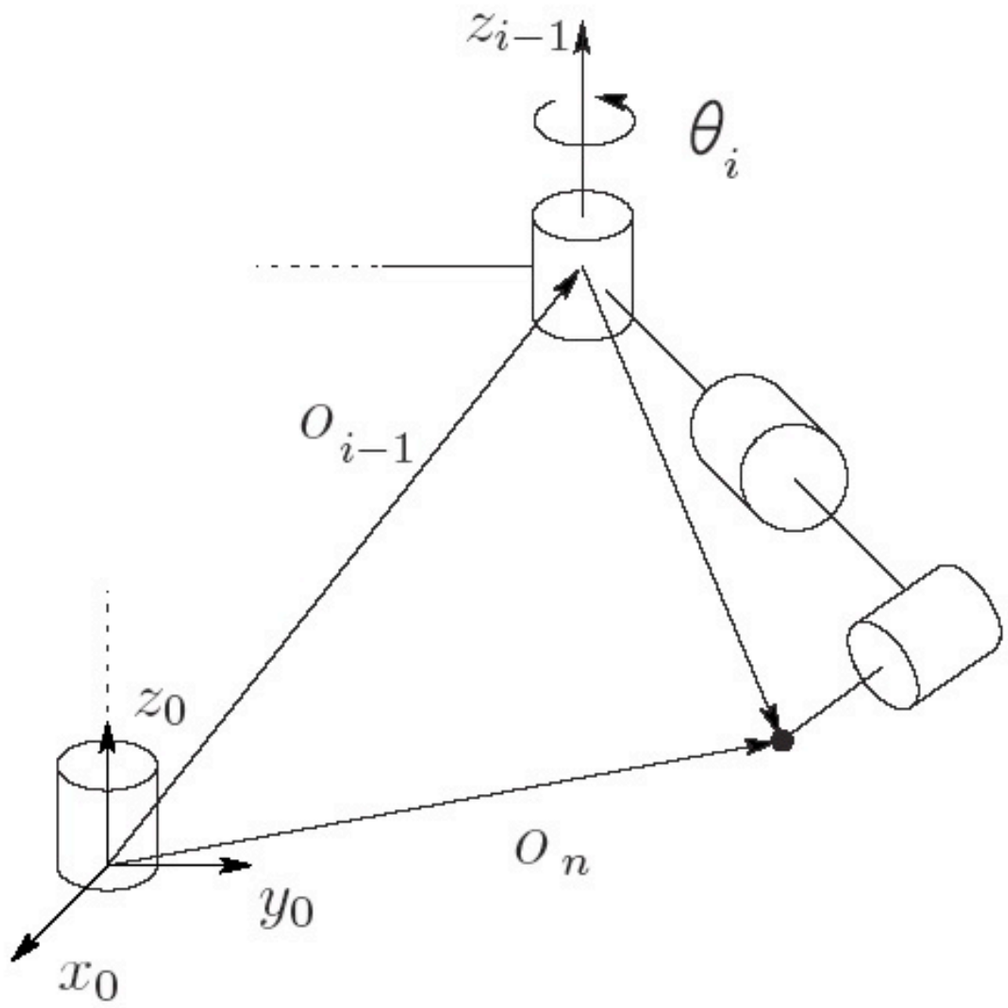
$$J_{v_i} = z_{i-1}$$

The orientation of a z-axis depends on the robot's pose if there are any revolute joints before it in the chain.

Figure 4.1: Motion of the end effector due to primsmatic joint i.

Revolute Joints

$$v = \omega \times r$$



$$\omega = \dot{\theta}_i z_{i-1}$$

$$r = o_n - o_{i-1}$$

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$$

Figure 4.2: Motion of the end effector due to revolute joint i.

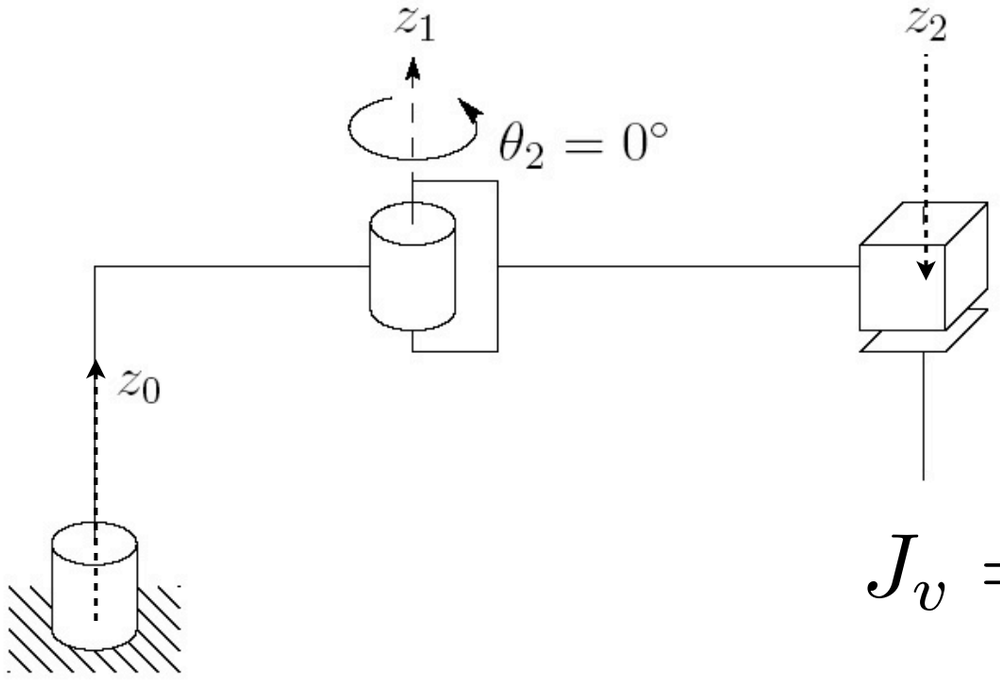
Another way to construct the position Jacobian.

Prismatic Joints

$$J_{v_i} = z_{i-1}$$

Revolute Joints

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$$



What is the SCARA's Jacobian?

$$J_v = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

I prefer to calculate position Jacobians by differentiating the end-effector position, but both approaches are valid, enabling you to check your work and increase your intuition.

$$v_n^0 = J_v \dot{q}$$

What joint velocities should I choose to cause a desired end-effector velocity?
(inverse velocity kinematics)

$$\dot{q} = J_v^{-1} v_n^0$$

Can a robot always achieve all end-effector velocities?

No. This works only when the Jacobian is square and invertible (non-singular).

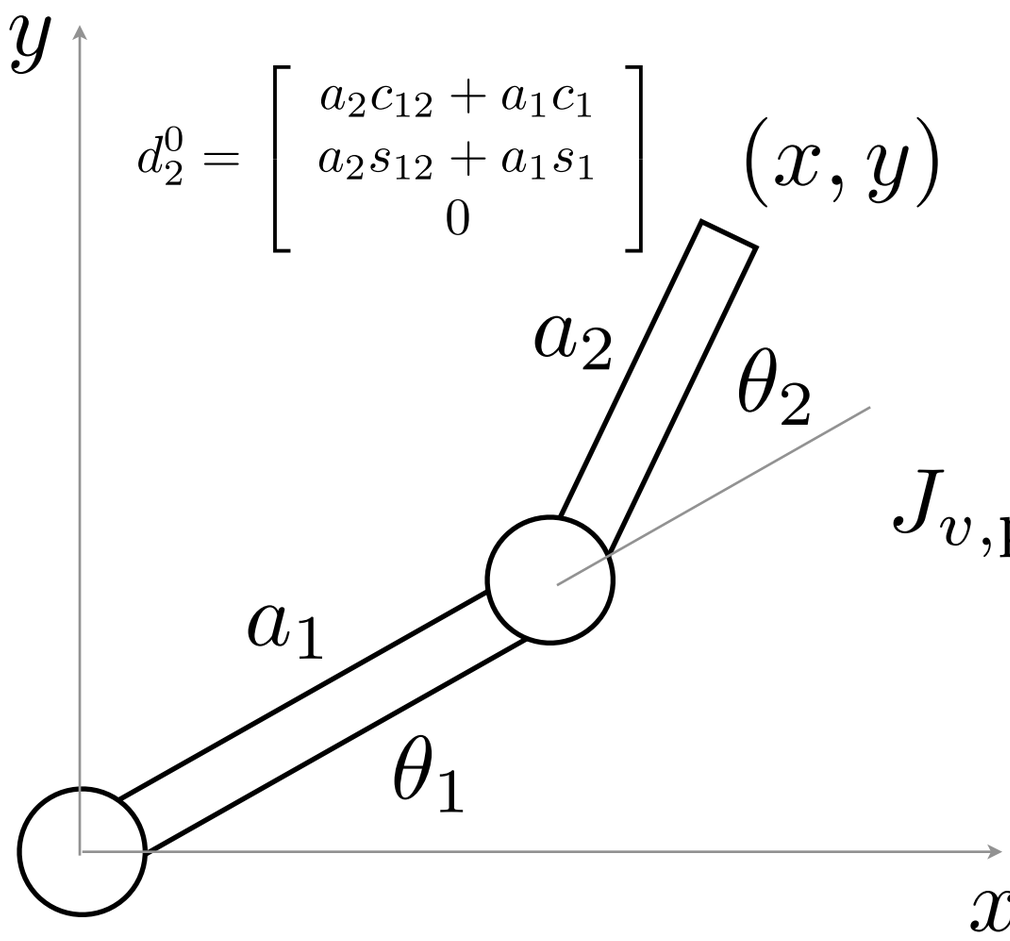
SHV 4.11 explains what to do when the Jacobian is not square:
rank test (v is in range of J)
use J^+ (right pseudoinverse of J)
when the robot has extra joints, there are many solutions

Singularities are points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom

Mathematically, singularities exist at any point in the workspace where the Jacobian matrix loses rank.

a matrix is singular if and only if its determinant is zero:

$$\det(J_v) = 0$$



$$d_2^0 = \begin{bmatrix} a_2c_{12} + a_1c_1 \\ a_2s_{12} + a_1s_1 \\ 0 \end{bmatrix}$$

$$J_v(\vec{q}) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix}$$

$$J_{v,\text{planar}}(\vec{q}) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \end{bmatrix}$$

$$\det(J_{v,\text{planar}}(\vec{q})) = ?$$

$$\det(J_{v,\text{planar}}(\vec{q})) = a_1a_2(c_1s_{12} - s_1c_{12})$$

When does $\det(\mathbf{J}) = 0$?

$\det(\mathbf{J}) = 0$ when $\theta_2 = 0$

Any other times?

$\det(\mathbf{J}) = 0$ when $a_1 = 0$ or $a_2 = 0$

if $\theta_2 = 0, c_1s_{12} - s_1c_{12} = c_1s_1 - s_1c_1 = 0$

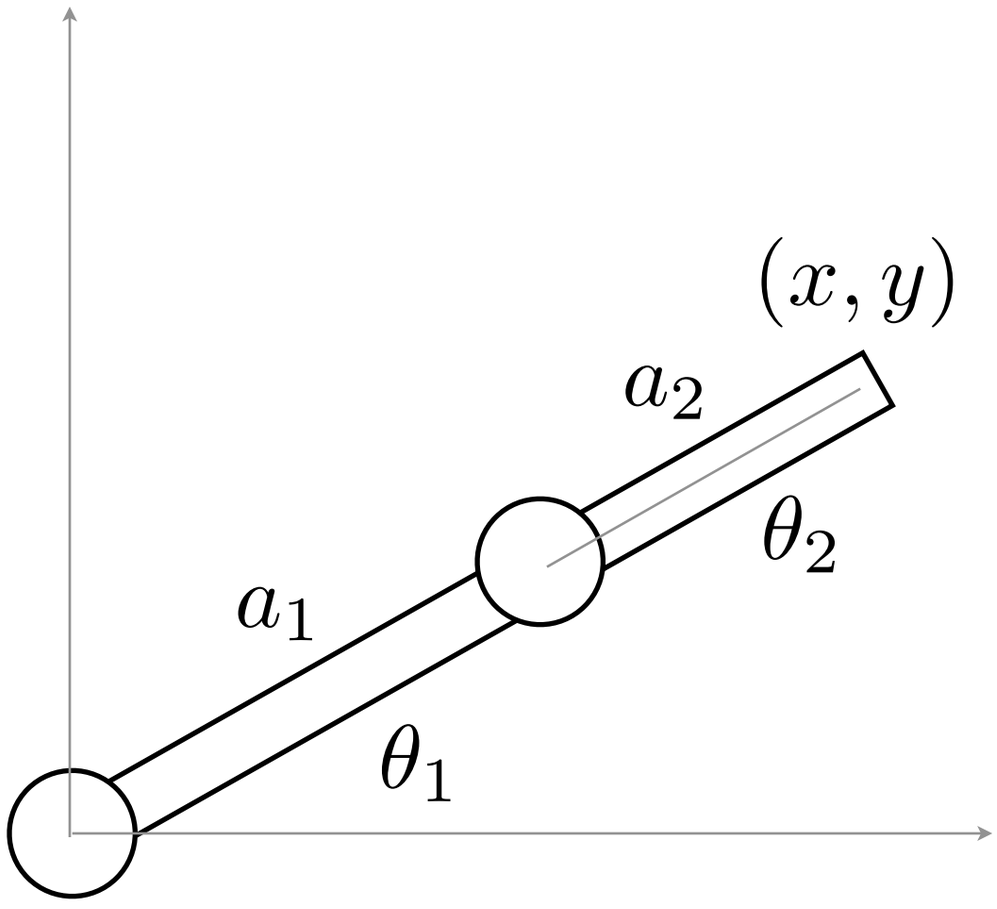
Is that the only time?

No… $\det(\mathbf{J}) = 0$ when $\theta_2 = \ldots, -2\pi, -\pi, 0, \pi, 2\pi, \ldots$

For $\theta_2 = 0$

The Jacobian collapses to have linearly dependent rows

$$\mathbf{J}_{\theta_2=0} = \begin{bmatrix} -a_1 s_1 - a_2 s_1 & -a_2 s_1 \\ a_1 c_1 + a_2 c_1 & a_2 c_1 \end{bmatrix}$$



This means that actuating either joint causes motion in the same direction

We often try to avoid singularities.

explore how **changes** in joint values affect the end-effector movement

could have **N joints**, but only **six** end-effector velocity terms (xyzpts)

The **Jacobian** matrix lets us calculate how joint velocities translate into end-effector velocities (depends on configuration)

look at it in two parts - position and orientation

$$v_n^0 = J_v \dot{q}$$

$$\omega_n^0 = J_\omega \dot{q}$$

How do we calculate the orientation Jacobian?

A Note about Notation

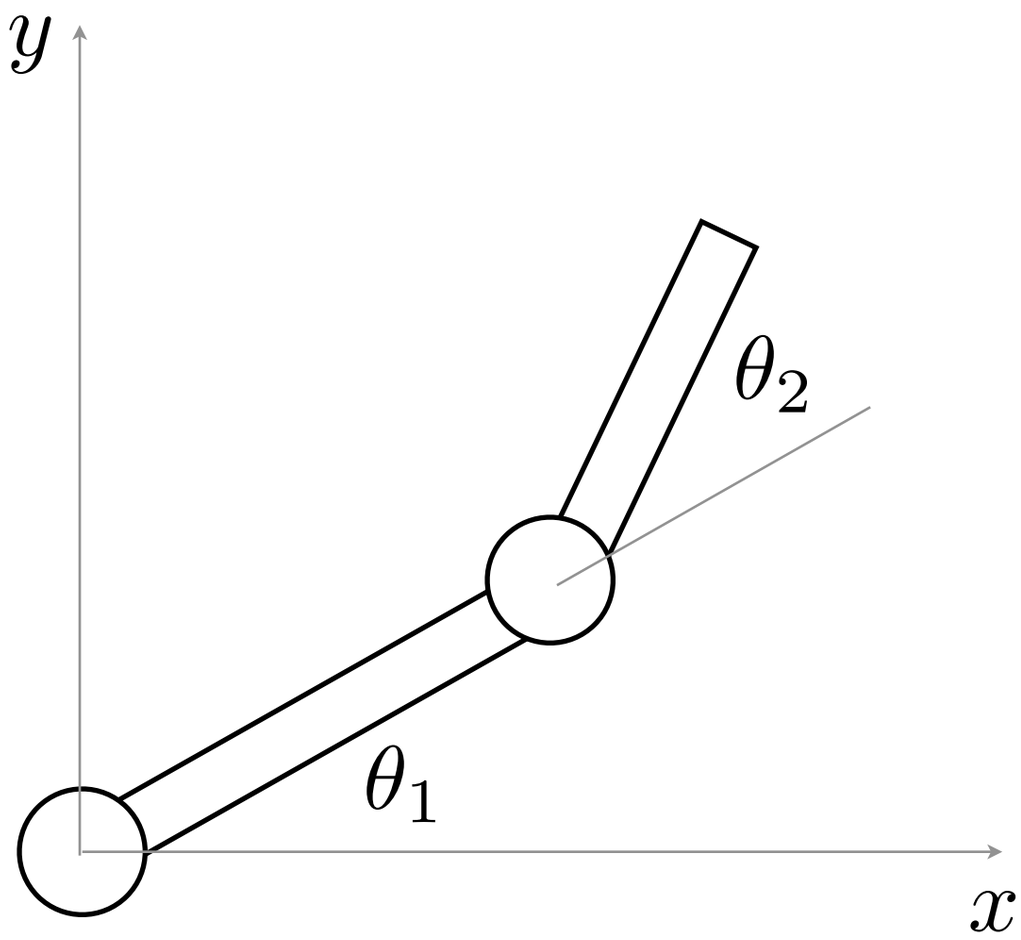
$w = J_w(q)\,\dot{q}$

w— final frame angular velocity

$$\omega_{i,j}^{k}$$

this is the angular velocity of frame **j**
with respect to frame **i**,
expressed in frame **k**

SHV 4.1 gives a good explanation of angular velocity for fixed-axis rotation. SHV 4.2-4.5 go into greater detail.



$$\omega_{0,1}^{0} = 0\,\hat{x}_0 + 0\,\hat{y}_0 + \dot{\theta}_1\,\hat{z}_0$$

$$\omega_{1,2}^{1} = 0\,\hat{x}_1 + 0\,\hat{y}_1 + \dot{\theta}_2\,\hat{z}_1$$

$$\omega_{1,2}^{0} = \mathbf{R}_1^0\,\omega_{1,2}^{1}$$

$$\begin{aligned}\omega_{0,2}^{0} &= \omega_{0,1}^{0} + \mathbf{R}_1^0\,\omega_{1,2}^{1} \\ &= 0\,\hat{x}_0 + 0\,\hat{y}_0 + (\dot{\theta}_1 + \dot{\theta}_2)\,\hat{z}_0\end{aligned}$$

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} \mathbf{R}_{i-1}^0\,\omega_{i-1,i}^{i-1}$$

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} (\mathbf{R}_{i-1}^0\,\hat{\mathbf{z}})\,\dot{\theta}_i$$

note: this holds for revolute joints only (by definition, a prismatic joint cannot create angular velocity)

And now, for that Jacobian!

$$\omega_{0,n}^0 = \sum_{i=1}^{n} \rho_i (\mathbf{R}_{i-1}^0 \hat{z})\, \dot{\theta}_i \qquad \rho_i = \begin{matrix} \text{0 for prismatic} \\ \text{1 for revolute} \end{matrix}$$

$$\omega_{0,n}^0 = \begin{bmatrix} \rho_1 \hat{\mathbf{z}} & \rho_2 \mathbf{R}_1^0 \hat{\mathbf{z}} & \rho_3 \mathbf{R}_2^0 \hat{\mathbf{z}} & \cdots & \rho_n \mathbf{R}_{n-1}^0 \hat{\mathbf{z}} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}$$

$$\omega = \mathbf{J}_\omega(q)\, \dot{\mathbf{q}}$$

(3 x n) cartesian velocity Jacobian

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_\omega \end{bmatrix}$$

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

(3 x n) angular velocity Jacobian

The Jacobian is easily constructed from the manipulator's forward kinematics.

What do you need from the forward kinematics?

4.6.3 Combining the Linear and Angular Velocity Jacobians

As we have seen in the preceding section, the upper half of the Jacobian J_v is given as

$$J_v = [J_{v_1} \cdots J_{v_n}] \tag{4.56}$$

in which the i^{th} column J_{v_i} is

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \tag{4.57}$$

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \tag{4.58}$$

in which the i^{th} column J_{ω_i} is

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolute joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \tag{4.59}$$

MEAM 520

More Jacobians and Singularities

Katherine J. Kuchenbecker, Ph.D.

General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 15: October 22, 2013

$$v_n^0 = J_v \dot{q}$$

(3 x 1) (3 x n)(n x 1)

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

Prismatic $J_{v_i} = z_{i-1}$

Revolute $J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$

$$\omega_n^0 = J_\omega \dot{q}$$

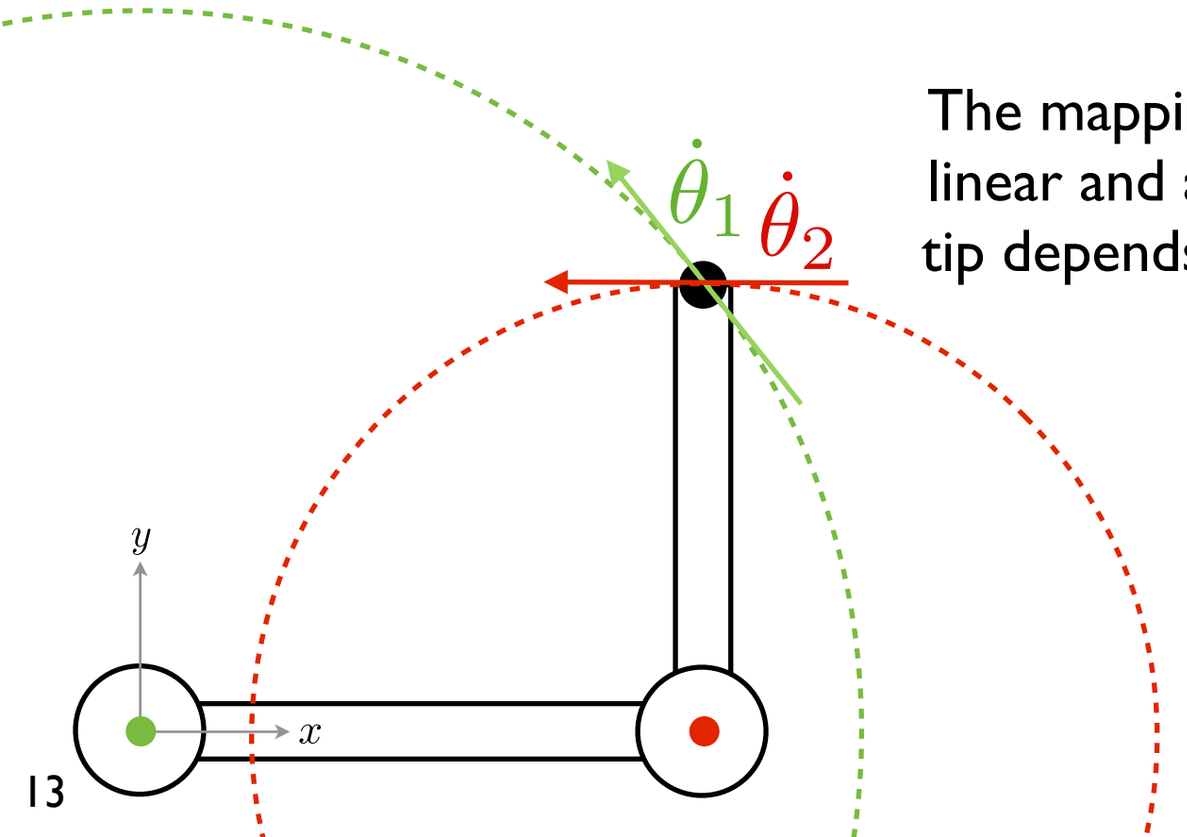
(3 x 1) (3 x n)(n x 1)

$$\omega_{0,n}^0 = \sum_{i=1}^{n} \rho_i (\mathbf{R}_{i-1}^0 \hat{z})\, \dot{\theta}_i$$

$$\rho_i = \begin{matrix} \text{0 for prismatic} \\ \text{1 for revolute} \end{matrix}$$

Prismatic $J_{\omega_i} = 0$

Revolute $J_{\omega_i} = z_{i-1}$



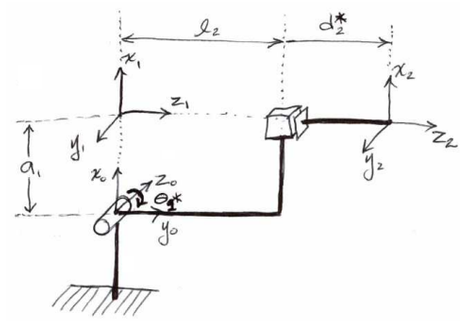
The mapping from joint velocities to the linear and angular velocity of the robot's tip depends on the robot's current pose!

Questions ?

A robot loses the ability to move its end-effector in certain directions when $\det(J_v) = 0$

Singular configurations or singularities

Two-Link Planar RP Arm With Offset (SHV 3-4)



Link	a_i	α_i	d_i	θ_i
1	a_1	-90°	0	θ_1^*
2	0	0°	$l_2 + d_2^*$	0°

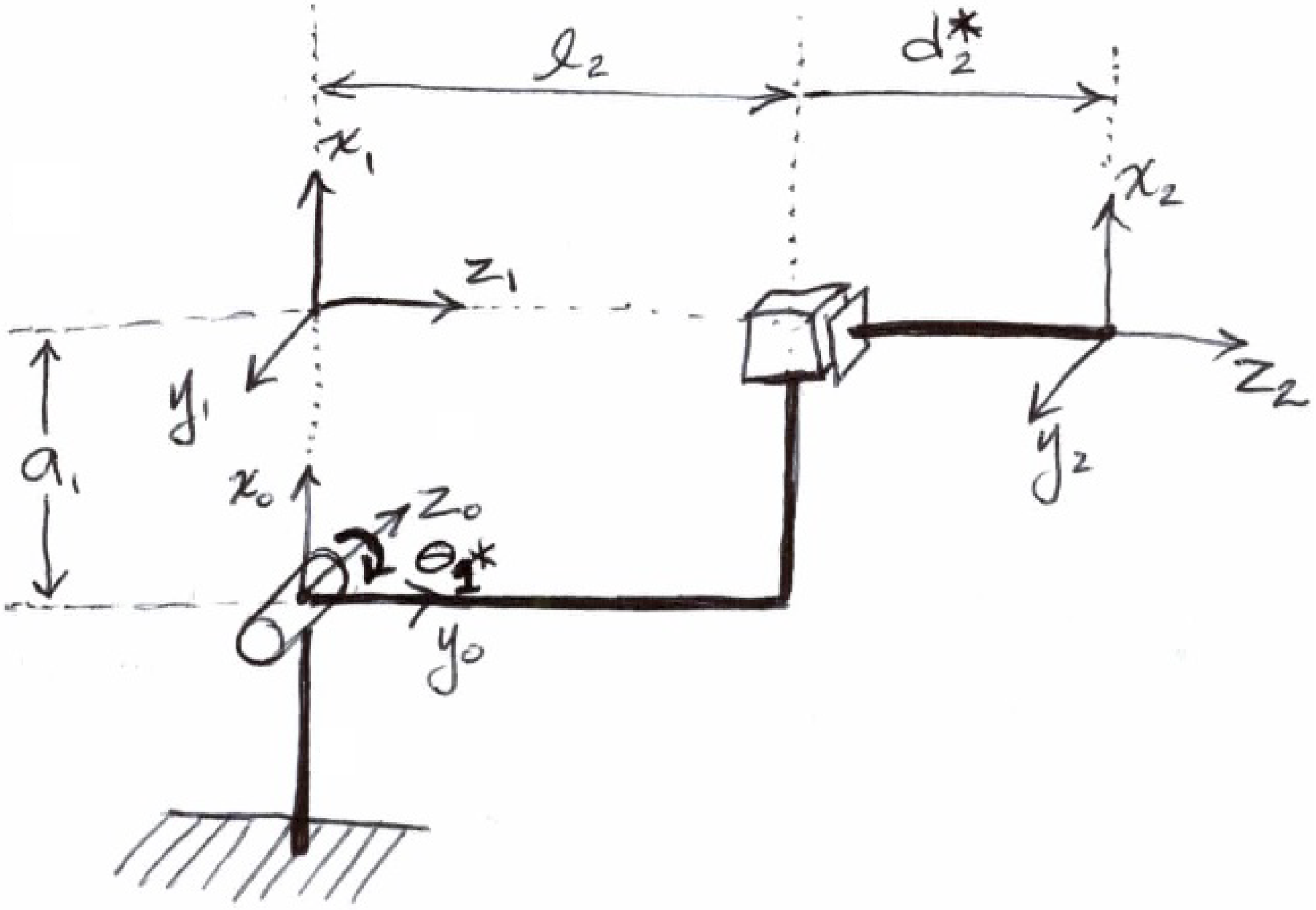
$$T_2^0 = \begin{bmatrix} c_1 & 0 & -s_1 & a_1 c_1 - (l_2 + d_2^*)\, s_1 \\ s_1 & 0 & c_1 & a_1 s_1 + (l_2 + d_2^*)\, c_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c_1 = \cos\theta_1^*$ and $s_1 = \sin\theta_1^*$

$J_v = ?$
$J_\omega = ?$

What are this robot's linear velocity singularities, if any?

Work with a partner.


l_2
d_2^*
x_1
z_1
y_1
a_1
x_0
z_0
θ_1^*
y_0
x_2
z_2
y_2

(3 x n) linear velocity Jacobian

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

(3 x n) angular velocity Jacobian

$$\xi = J(q)\dot{q}$$

(6 x 1) body velocity

(6 x n) Jacobian

(n x 1) joint velocities

Notice that the body velocity is not the time derivative of a body position vector because of the angular velocity.

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

SHV Section 4.8 – The Analytical Jacobian

The Analytical Jacobian is an alternative to the Geometric Jacobian; it uses a different representation for orientation.

Instead of enabling you to calculate the angular velocity of the end-effector's frame, it lets you calculate the time derivatives of three values that represent the orientation of the end-effector frame.

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

Euler angles are the most commonly used minimal representation.

$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{z,\psi}$$

$$\alpha = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

$$\omega = \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = B(\alpha)\dot{\alpha}$$

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B^{-1}(\alpha) \end{bmatrix} J(q)$$

We won't use the analytical Jacobian in this class, but you may encounter it elsewhere.

$$\xi = J(q)\dot{q}$$

It is mathematically challenging to find all of the singularities for a 6-DOF manipulator; the determinant of the Jacobian gets very complicated!

For a 6-DOF manipulator with a spherical wrist, we can decouple the determination of singular configurations into two simpler problems.

$$\xi = J(q)\dot{q}$$

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}]$$

(the book calls this $J = [J_P \mid J_O]$)

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\frac{J_{11}}{J_{21}} \middle| \frac{J_{12}}{J_{22}}\right]$$

$$J_{\text{wrist}} = \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix}$$

if we choose $o_4 = o_5 = o_6$

$$J_{\text{wrist}} = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix}$$

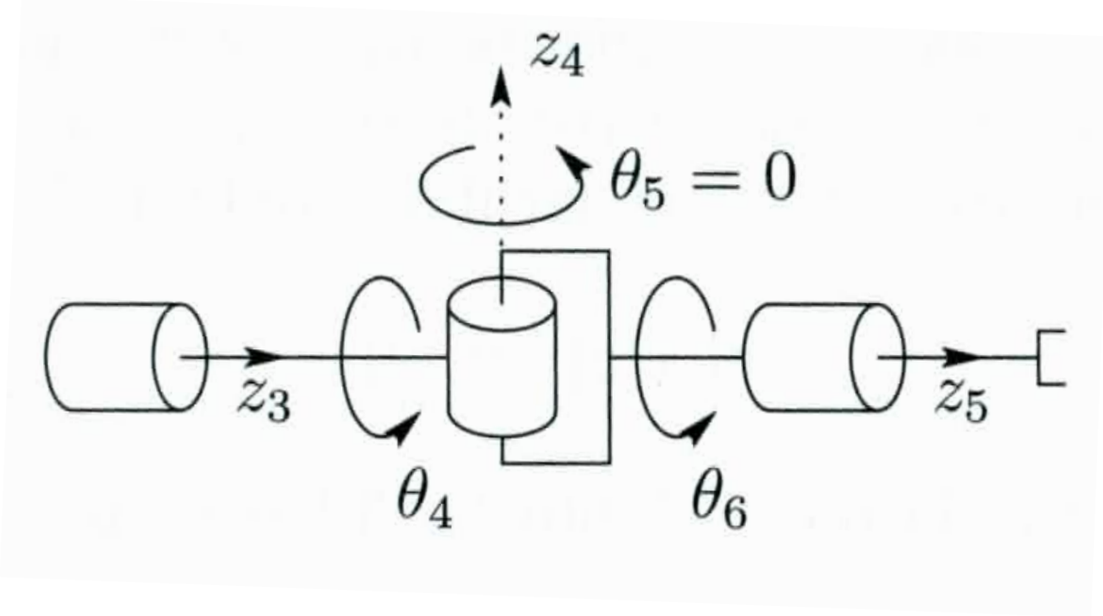
$$J = \left[\frac{J_{11}}{J_{21}} \middle| \frac{0}{J_{22}}\right] \qquad \det(J) = \det(J_{11})\det(J_{22})$$

arm wrist

$$\det(J) = \det(J_{11})\det(J_{22})$$

$$J_{22} = \begin{bmatrix} z_3 & z_4 & z_5 \end{bmatrix}$$

Singular when any two wrist axes align



$z_3 \perp z_4$ $z_4 \perp z_5$

z_3 can become $\| z_5$

$\theta_5 = 0, \pi$ are singular configurations

MEAM 520

The PUMA 260

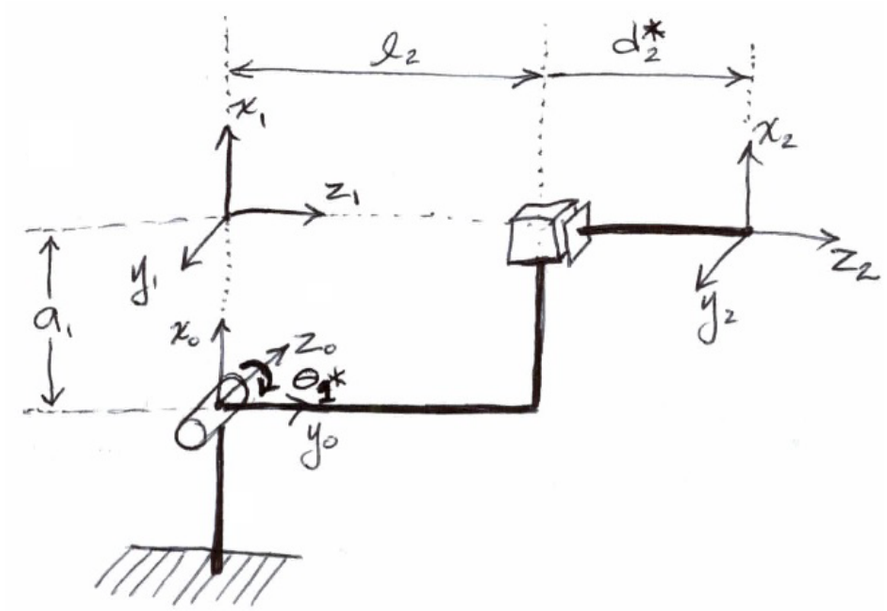
Katherine J. Kuchenbecker, Ph.D.

General Robotics, Automation, Sensing, and Perception Lab (GRASP)

MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 16: October 24, 2013



$$\xi = J(q)\dot{q}$$

(6 x 1) body velocity

but this is not the derivative of any vector

(6 x n) Jacobian

(n x 1) joint velocities

Questions ?

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

(6 x n) Analytical Jacobian

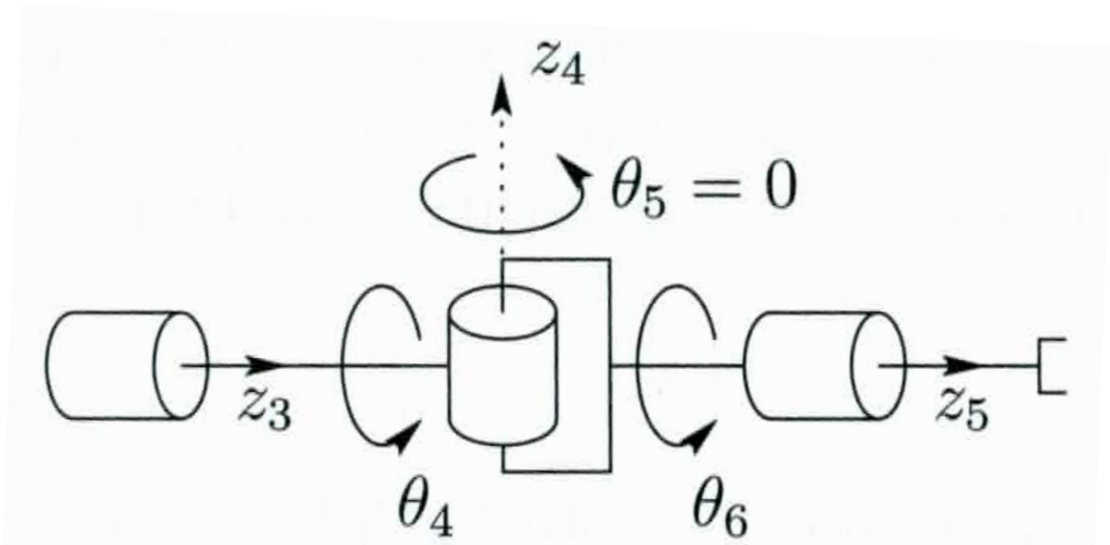
$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\frac{J_{11}}{J_{21}} \middle| \frac{J_{12}}{J_{22}}\right]$$

if we choose $o_4 = o_5 = o_6$

$$J = \left[\frac{J_{11}}{J_{21}} \middle| \frac{0}{J_{22}}\right]$$

$$\det(J) = \det(J_{11})\det(J_{22})$$

arm wrist



$\theta_5 = 0, \pi$ are singular configurations

The transpose of the Jacobian relates joint forces and torques to Cartesian end-effector forces and torques

(n × 1) (n × 6) (6 × 1)

$$\vec{\tau} = J^T(\vec{q})\vec{F}$$

joint forces and torques

Jacobian matrix transpose

endpoint forces and torques

Simplest to think about for a 3-DOF robot with all revolute joints.

We want to output a force at the tip.

(3 × 1) (3 × 3) (3 × 1)

$$\vec{\tau} = J^T(\vec{q})\vec{F}$$

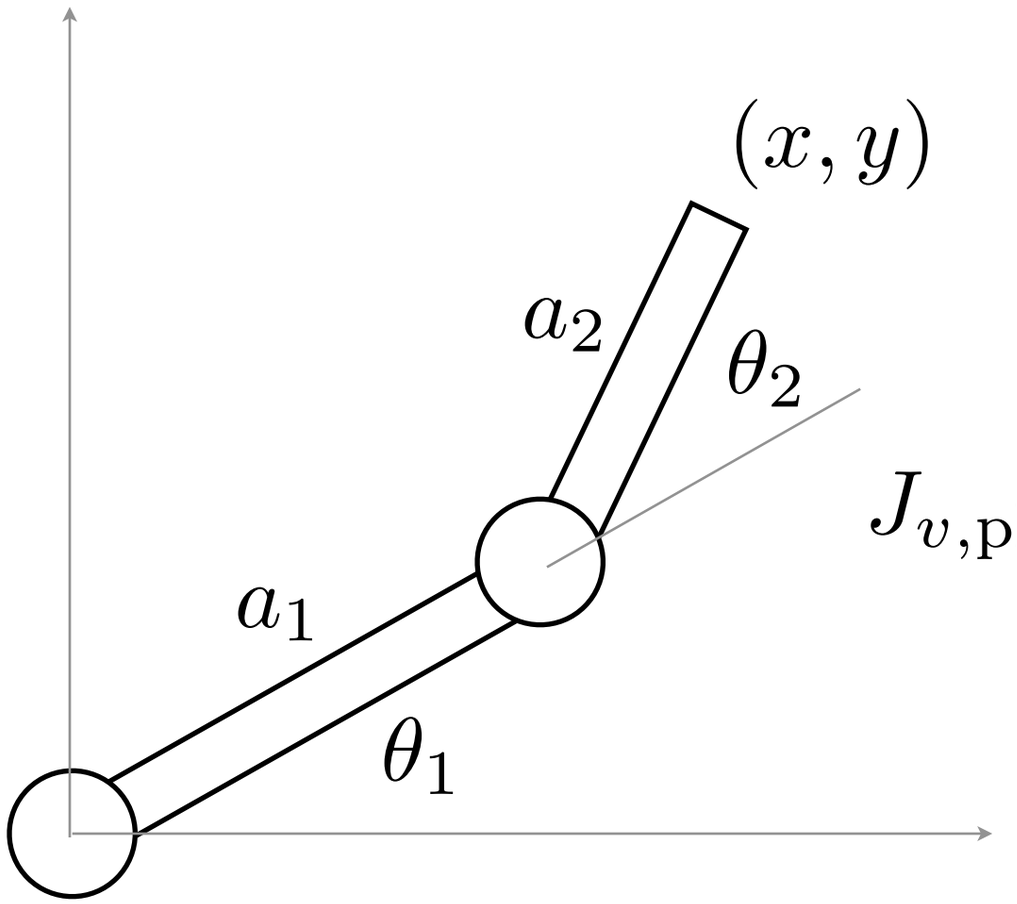
joint torques

Jacobian matrix transpose

endpoint forces

$$\vec{\tau} = J^T(\vec{q})\vec{F}$$

This relationship stems from virtual work.



Beginning with the 2 x 2 linear velocity Jacobian

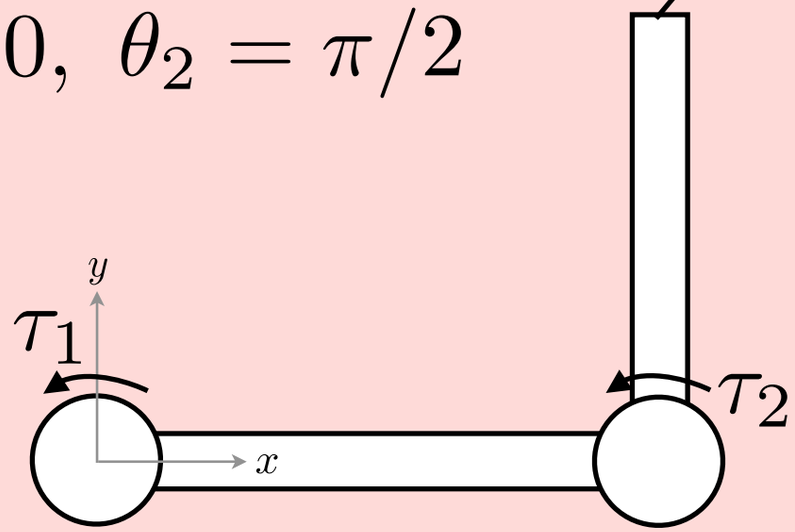
$$J_{v,\text{planar}}(\vec{q}) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$

We can solve for the joint torques necessary to exert a desired force at the end effector using the Jacobian transpose

$$\vec{\tau} = J^T(\vec{q})\vec{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} \\ -a_2 s_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}$$

$$\theta_1 = 0,\ \theta_2 = \pi/2$$



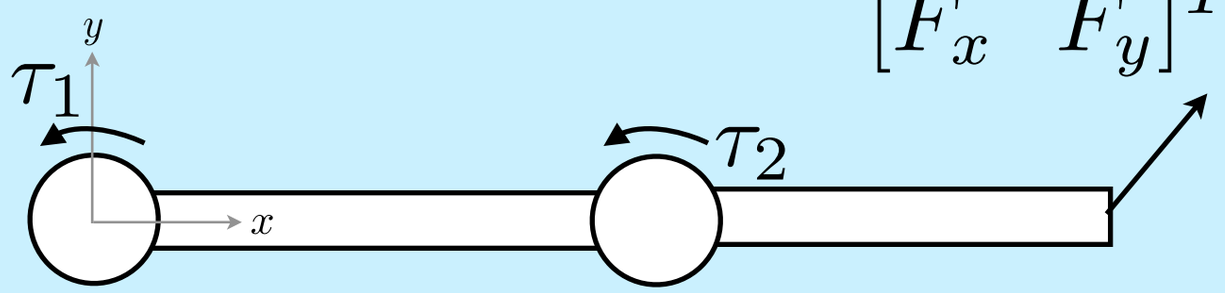
$$J_v([0 \quad \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\tau_1 = -a_2 F_x + a_1 F_y$$

$$\tau_2 = -a_2 F_x$$

Can create forces in both x and y directions.

$$\theta_1 = 0,\ \theta_2 = 0$$



$$J_v([0 \quad 0]^T) = \begin{bmatrix} 0 & 0 \\ a_1 + a_2 & a_2 \\ 0 & 0 \end{bmatrix}$$

$$\tau_1 = (a_1 + a_2) F_y$$

$$\tau_2 = a_2 F_y$$

Can't create forces in the x direction!

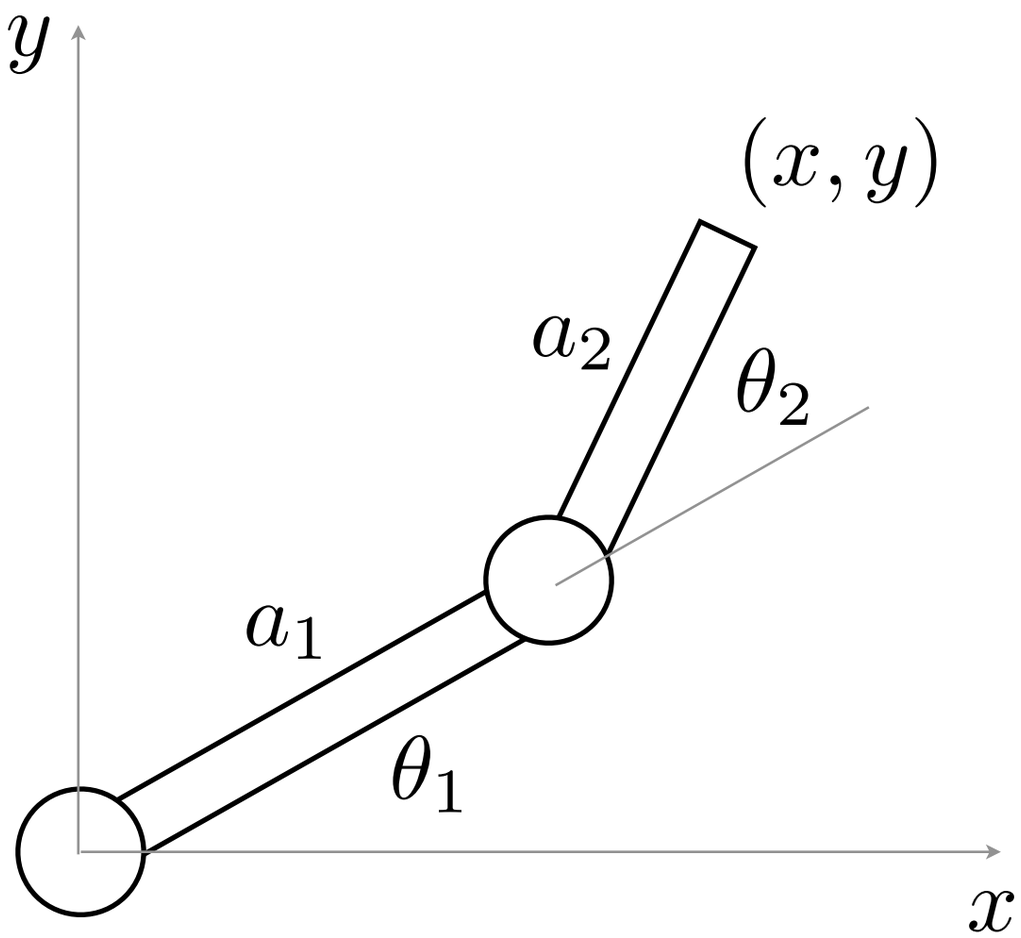
For a specific configuration, the Jacobian scales the input (joint velocities) to the output (body velocity)

$$\xi = J(q)\dot{q}$$

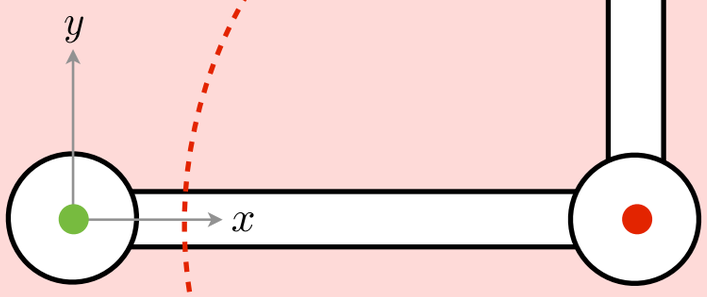
If you put in a joint velocity vector with unit norm, you can calculate in which direction and how fast the robot's end-effector will translate and rotate.

This approach allows you to calculate and plot the manipulability ellipsoid – a geometrical representation of all the possible tip velocities for a normalized joint velocity input.
A 6D ellipsoid is hard to visualize,
but 2D and 3D ellipsoids are lovely and useful.

What does the manipulability ellipsoid look like for the planar RR robot?



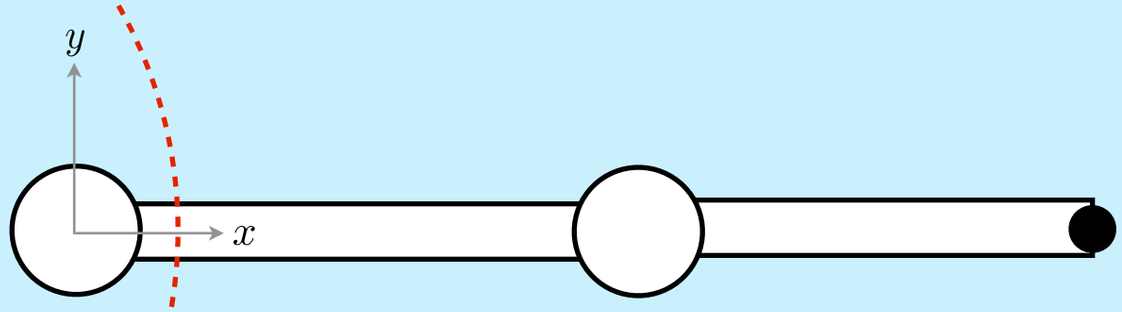
$\theta_1 = 0,\ \theta_2 = \pi/2$



$$J_v([0 \quad \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -a_2\dot{\theta}_1 & -a_2\dot{\theta}_2 \\ a_1\dot{\theta}_1 & \\ 0 & \end{bmatrix}$$

$\theta_1 = 0,\ \theta_2 = 0$

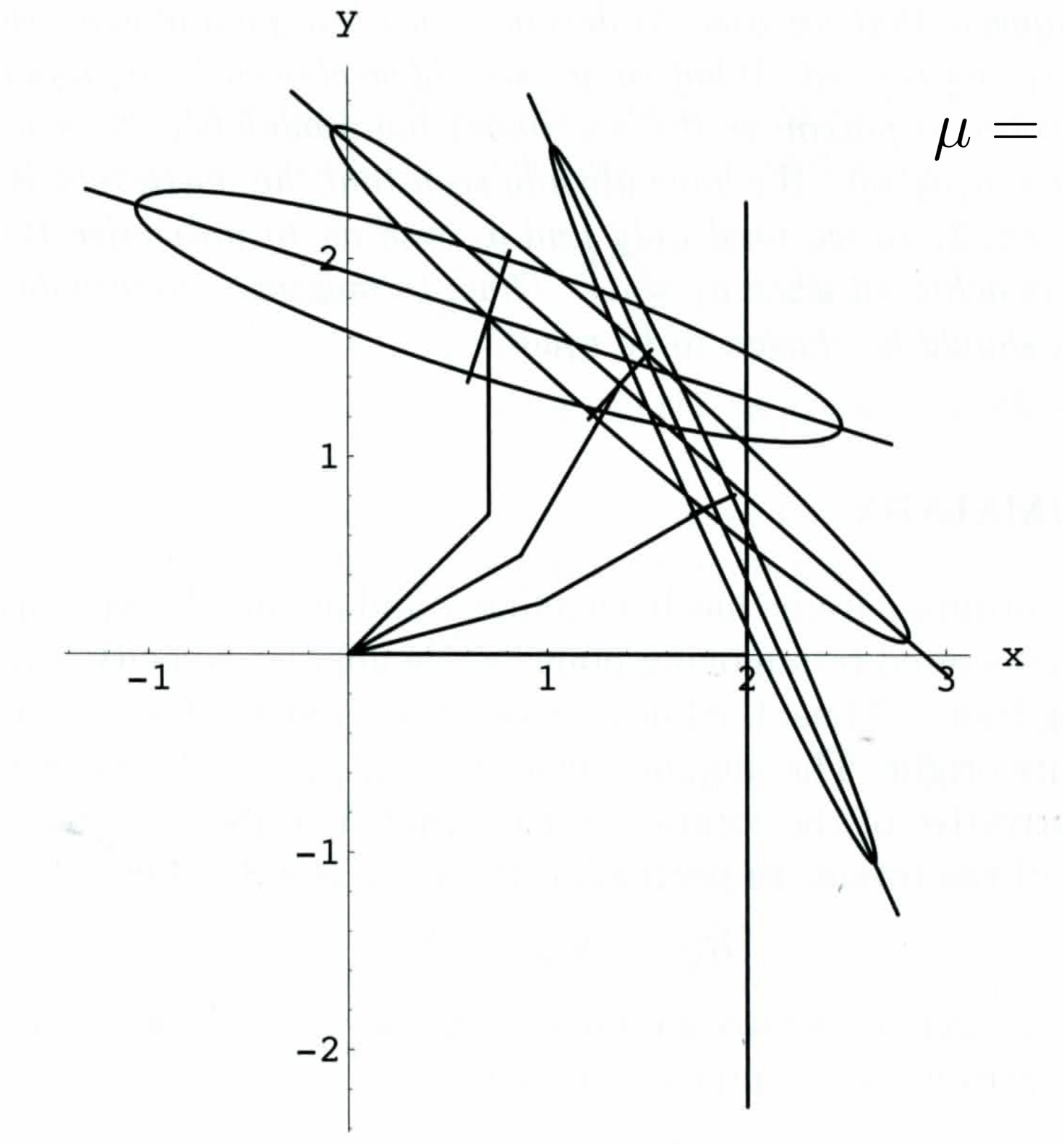


$$J_v([0 \quad 0]^T) = \begin{bmatrix} 0 & 0 \\ a_1 + a_2 & a_2 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ (a_1 + a_2)\dot{\theta}_1 + a_2\dot{\theta}_2 \\ 0 \end{bmatrix}$$

$$\dot{\vec{p}} = J_v(\vec{q})\,\dot{\vec{q}}$$

The robot's tip cannot move in the z direction, but it can move in both x and y directions...



Manipulability

$$\mu = |\det(J)| = a_1 a_2 |\sin(\theta_2)|$$

Can be used to tell you where to perform certain tasks.

Also useful for deciding how to design a manipulator.

MEAM 520

Midterm Review and More Robot Hardware

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception La ASP)
MEAM Department, SEAS, University of Pennsyl

GRASP LABORATORY

Lecture 18: October 31, 2013

What schemes did you use?

What are their main distinguishing features?

motion commands vs. *exploration with feedback*

absolute position vs. *incremental position* vs. *velocity*

Cartesian space vs. *joint space*

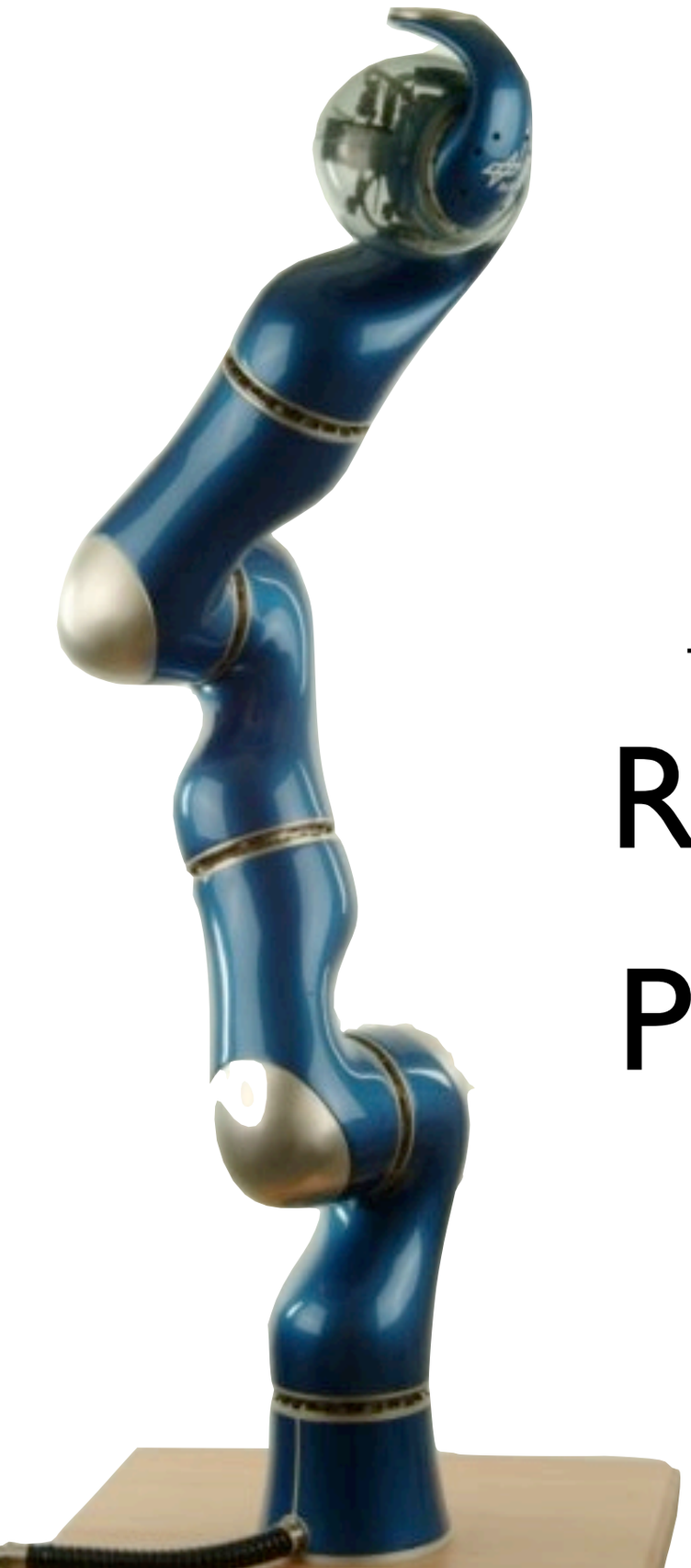
communication code and *error handling*

What is a robot?

a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks

- Robot Institute of America (RIA)

Chapter 1 of Spong, Hutchinson, and Vidyasagar



Robot manipulators are composed of:

- Rigid **links**
- Connected by **joints**
- To form a **kinematic chain**

There are two types of **joints**:

R
- **Revolute** (rotary), like a hinge, allows relative rotation between two links

P
- **Prismatic** (linear), like a slider, allows a relative linear motion (translation) between two links

Articulated Manipulator (RRR)

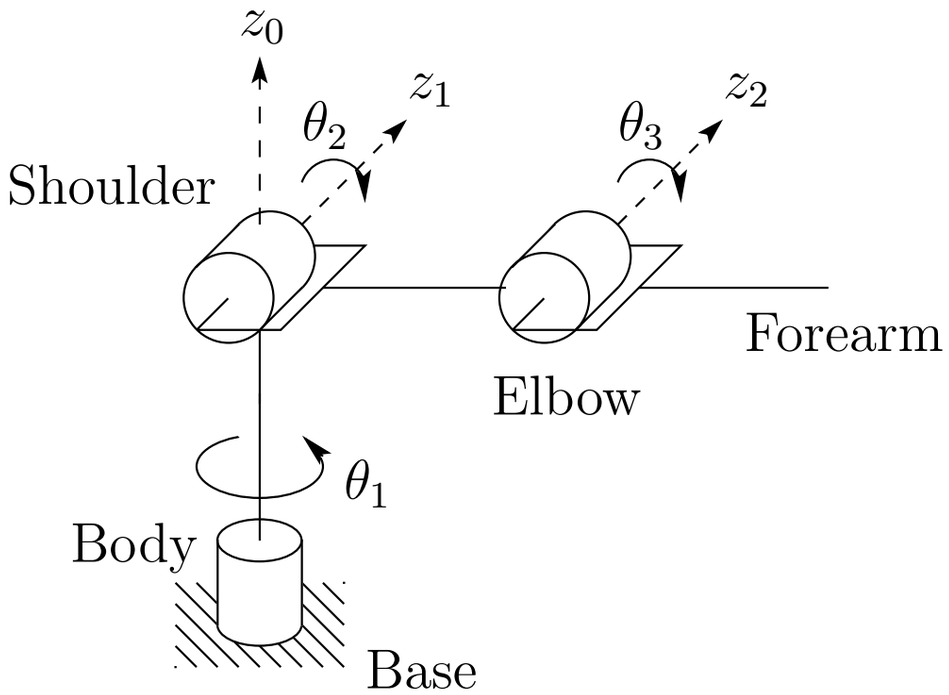


Fig. 1.8 Structure of the elbow manipulator.

Fig. 1.6 The ABB IRB1400 Robot. Photo courtesy of ABB.

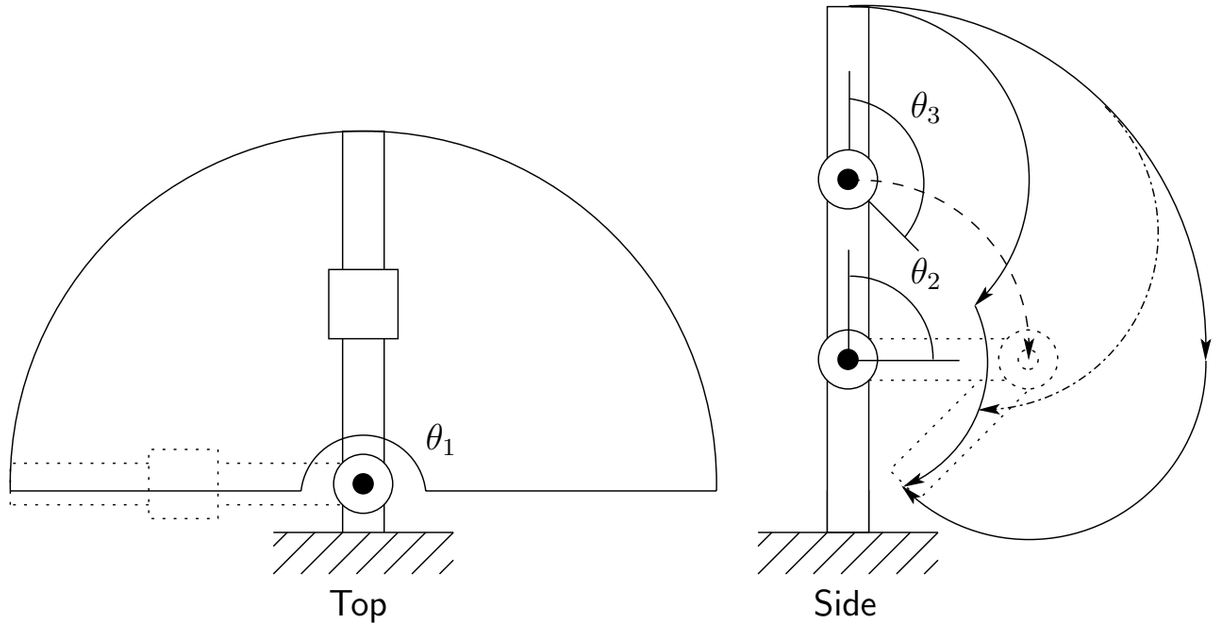


Fig. 1.9 Workspace of the elbow manipulator.

a.k.a. Revolute, Elbow, or Anthropomorphic

Spherical Manipulator (RRP)

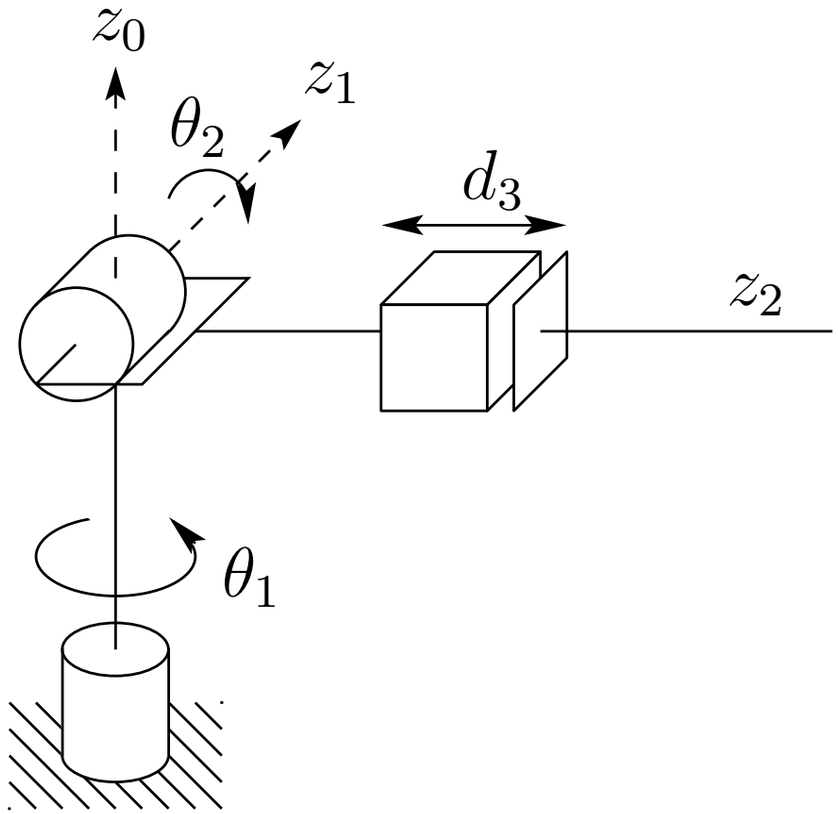


Fig. 1.10 The spherical manipulator.

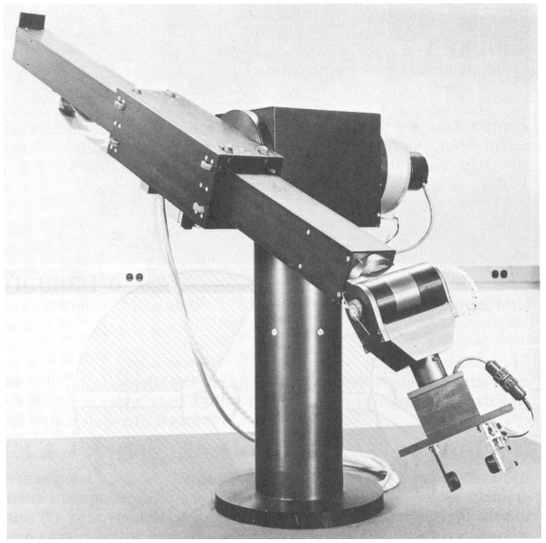


Fig. 1.11 The Stanford Arm. Photo courtesy of the Coordinated Science Lab, University of Illinois at Urbana-Champaign.

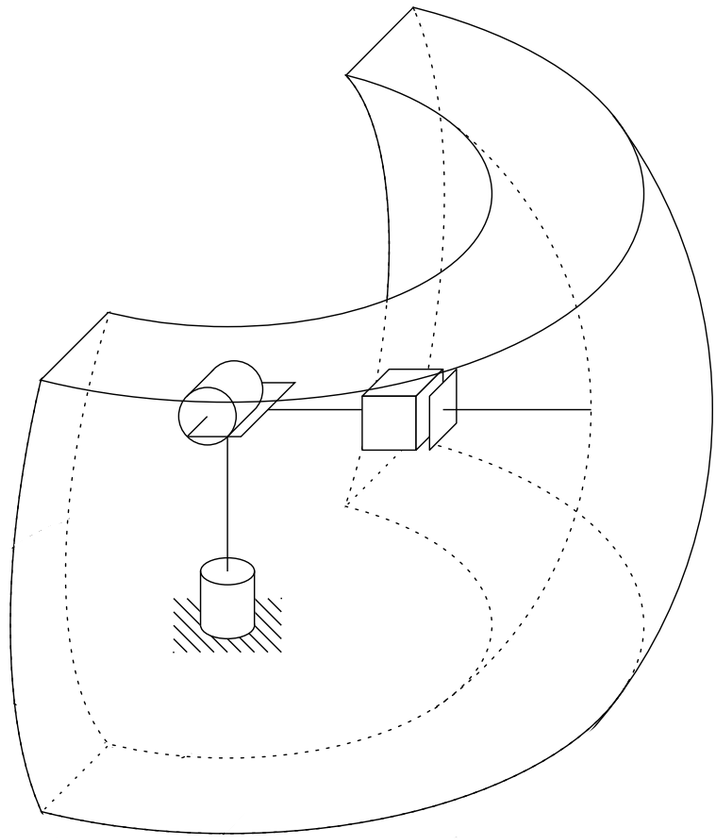


Fig. 1.12 Workspace of the spherical manipulator.

SCARA Manipulator (RRP)

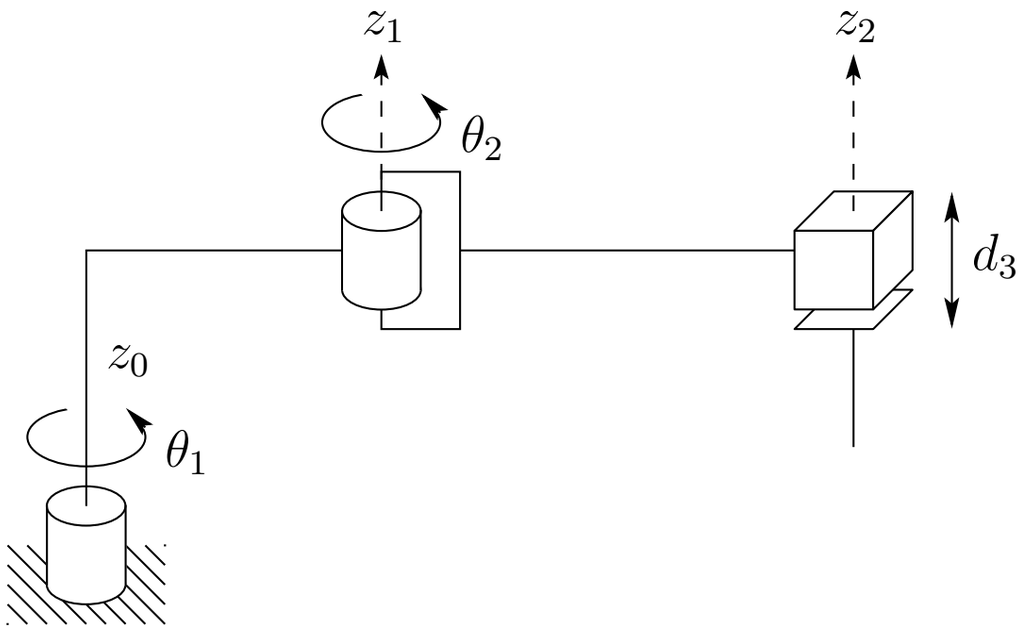


Fig. 1.13 The SCARA (Selective Compliant Articulated Robot for Assembly).

Fig. 1.14 The Epson E2L653S SCARA Robot. Photo Courtesy of Epson.

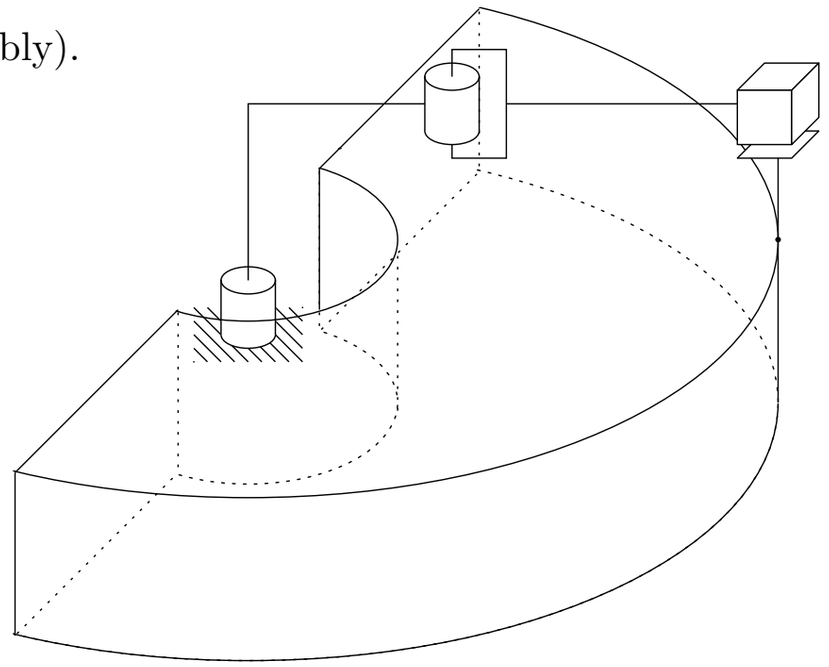


Fig. 1.15 Workspace of the SCARA manipulator.

Cylindrical Manipulator (RPP)

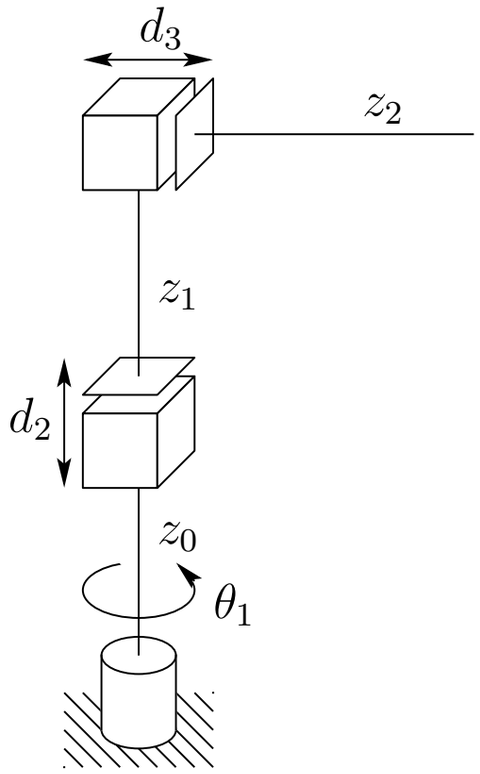


Fig. 1.16 The cylindrical manipulator.

Fig. 1.17 The Seiko RT3300 Robot. Photo courtesy of Seiko.

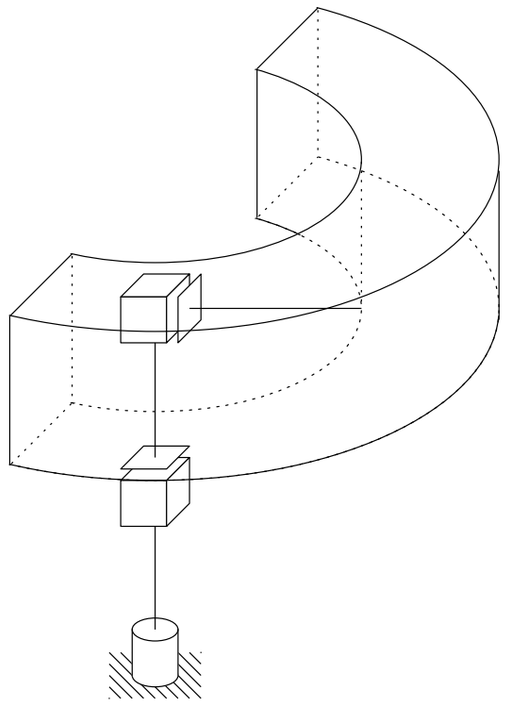


Fig. 1.18 Workspace of the cylindrical manipulator.

Cartesian Manipulator (PPP)

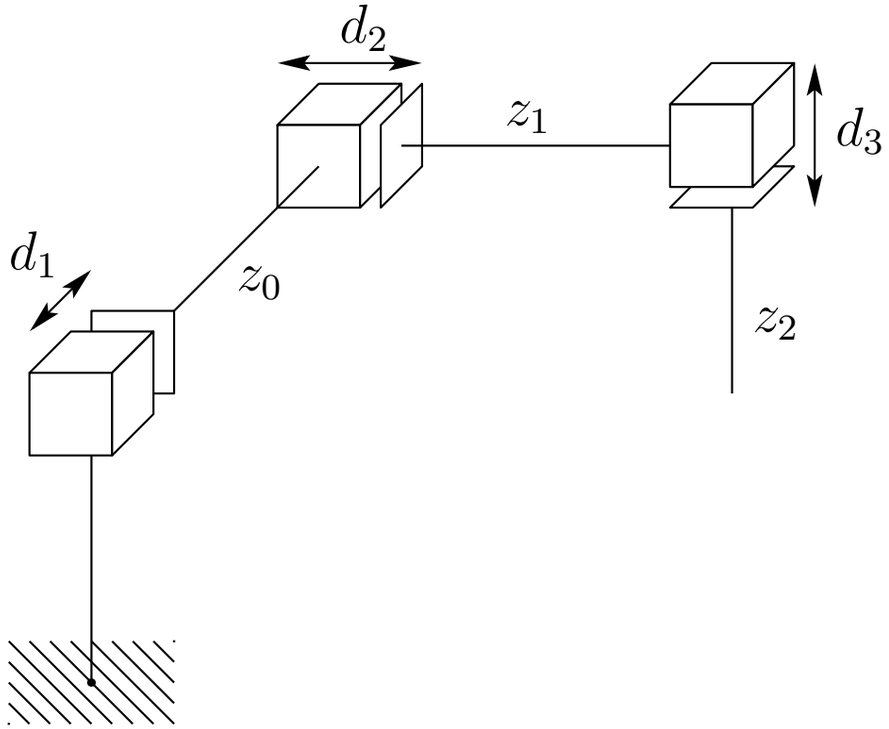


Fig. 1.19 The Cartesian manipulator.

Fig. 1.20 The Epson Cartesian Robot. Photo courtesy of Epson.

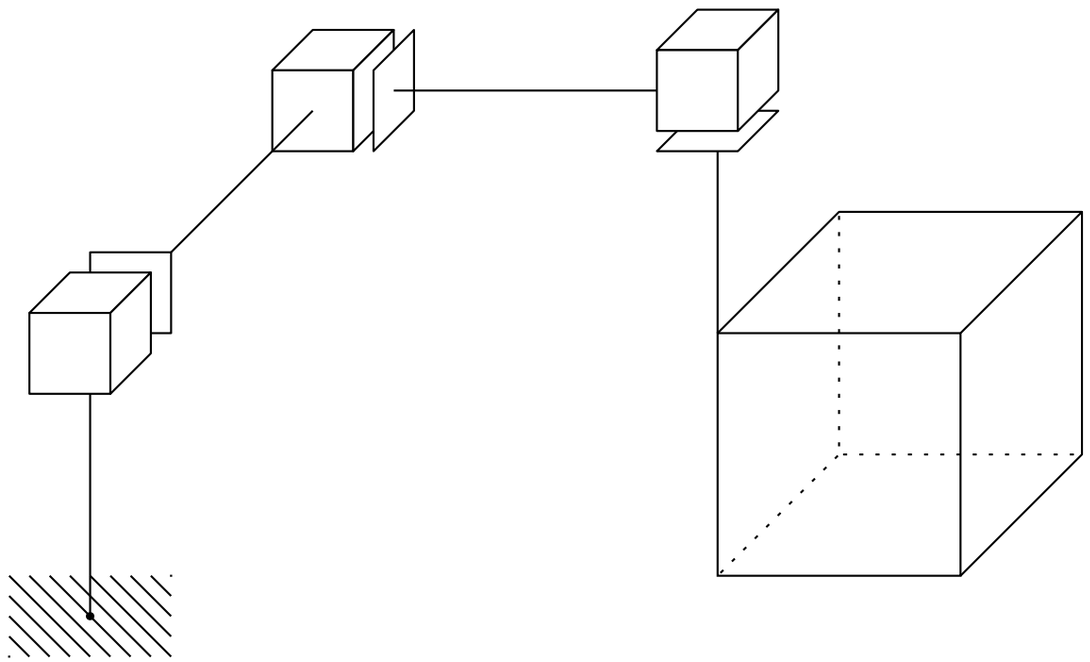
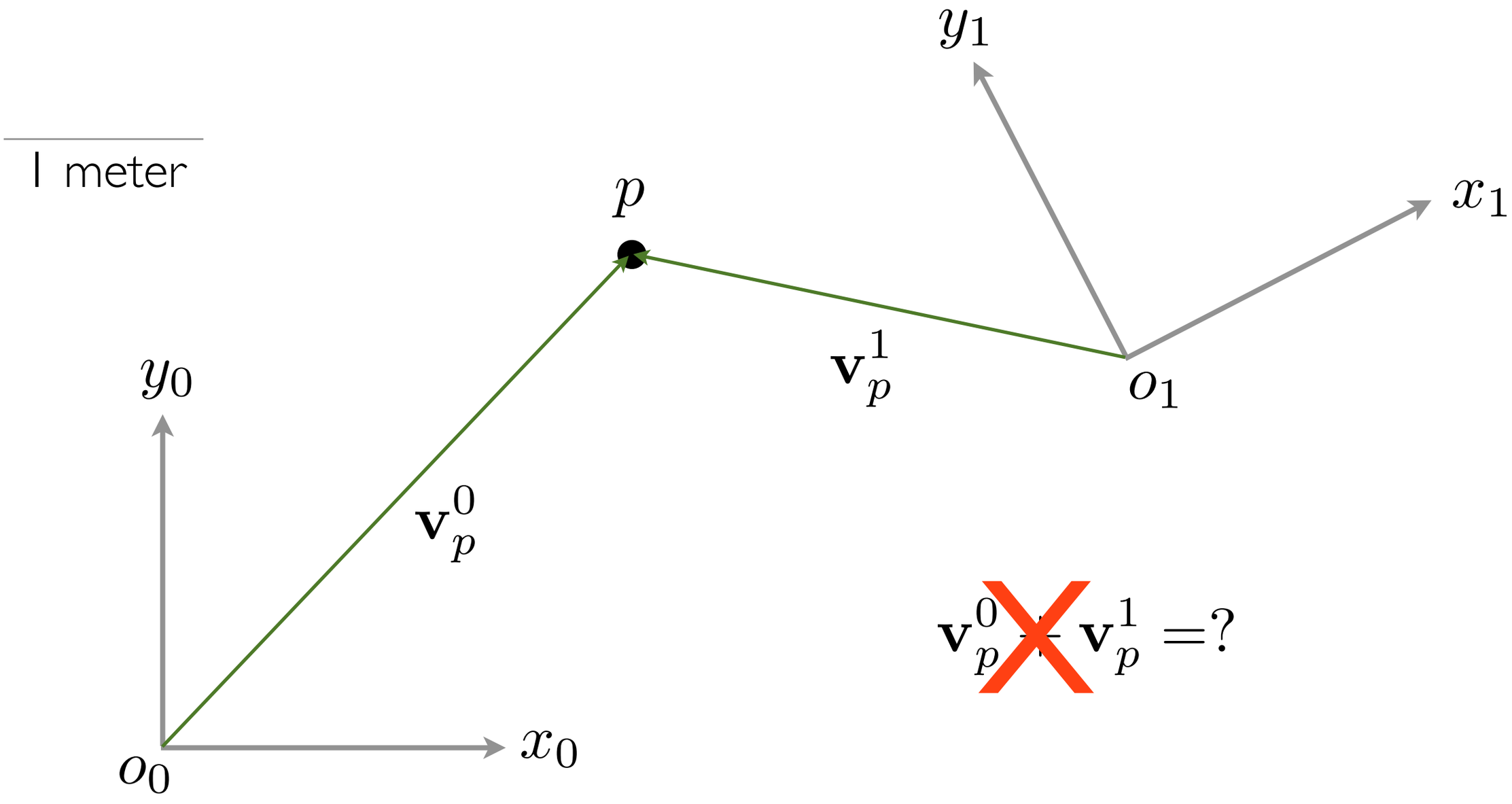


Fig. 1.21 Workspace of the Cartesian manipulator.

Multiple coordinate frames

The coordinate frame being used is designated using **superscript** notation



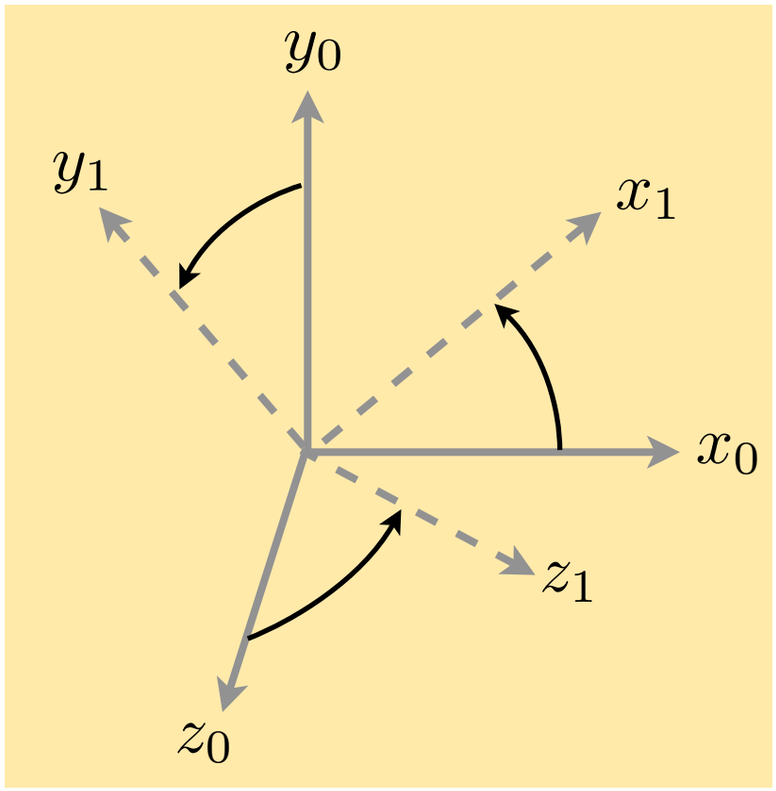
$$\mathbf{v}_p^0 \neq \mathbf{v}_p^1 \; =?$$

To perform algebraic manipulation, you must express vectors in the **same frame** or in **parallel frames**

Rotation Matrices

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Rotation Matrices - Interpretation 1 of 3



Represents the orientation of one coordinate frame with respect to another frame

$$\mathbf{R}_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

Orientation of frame 1 w.r.t. frame 0

columns are of unit length
columns are mutually orthogonal
$\det \mathbf{R}_1^0 = +1$

$\mathbf{R}_0^1 = ?$ $\quad \mathbf{R}_0^1 = \left(\mathbf{R}_1^0\right)^T$

$\left(\mathbf{R}_1^0\right)^T = \left(\mathbf{R}_1^0\right)^{-1}$

Three-Dimensional Coordinate Rotations

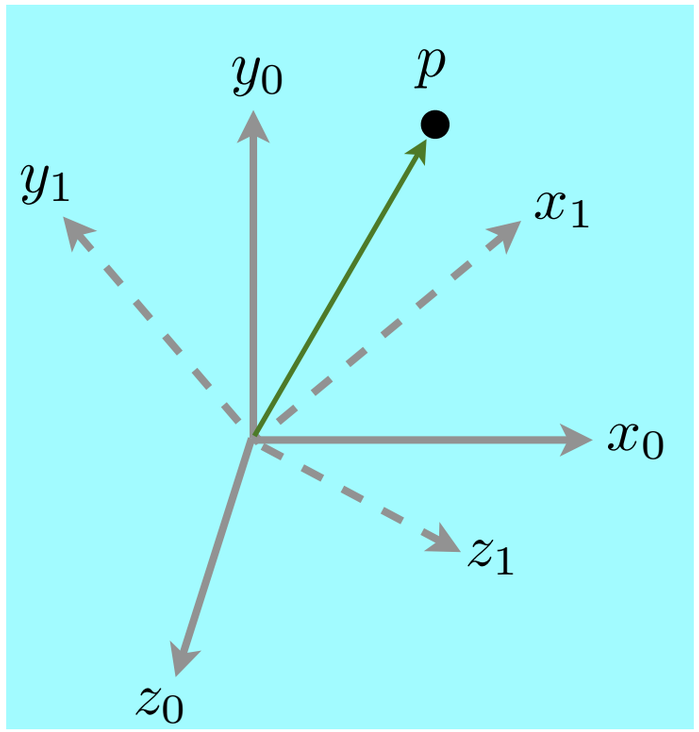
The **basic rotation matrices**
define rotations about the three
coordinate axes

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrices - Interpretation 2 of 3



Coordinate transformation relating the coordinates of a point p in two different frames

$\mathbf{R}_1^0 \qquad \mathbf{v}_p^1$

$\mathbf{v}_p^0 = ?$

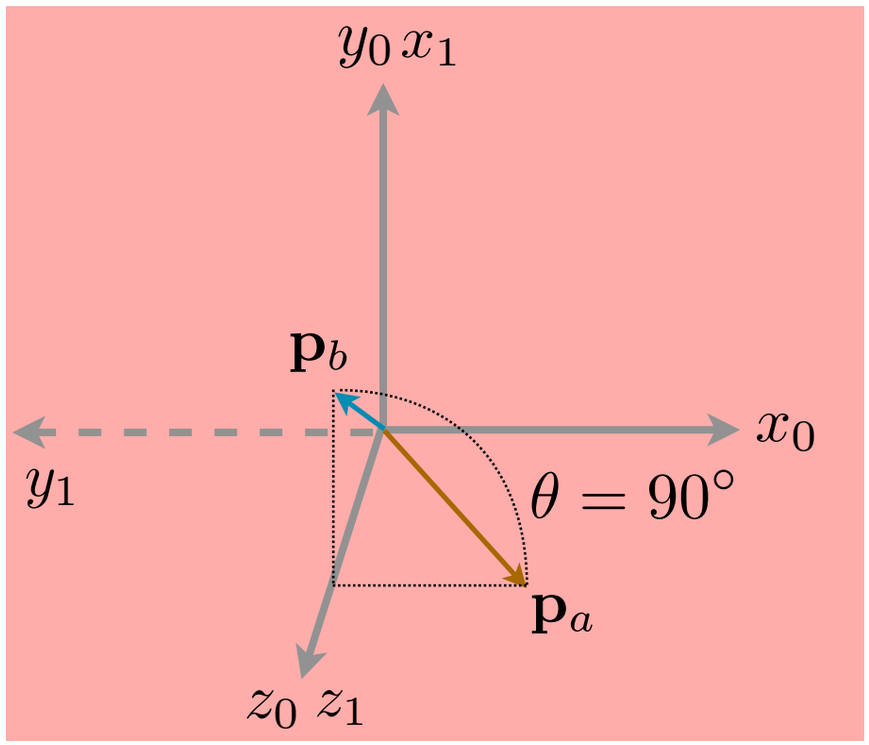
$$\boxed{\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1}$$

Subscript and superscript cancel

$\mathbf{v}_p^1 = ?$ $\qquad \left(\mathbf{R}_1^0\right)^{-1} \mathbf{v}_p^0 = \cancel{\left(\mathbf{R}_1^0\right)^{-1} \mathbf{R}_1^0} \mathbf{v}_p^1$ $\qquad \mathbf{v}_p^1 = \left(\mathbf{R}_1^0\right)^T \mathbf{v}_p^0$

$$\boxed{\mathbf{v}_p^1 = \mathbf{R}_0^1 \mathbf{v}_p^0}$$

Rotation Matrices - Interpretation 3 of 3



Operator taking a vector and rotating it to yield a new vector in the same coordinate frame

$$\mathbf{p}_a^0 \qquad\qquad \mathbf{R}_1^0$$

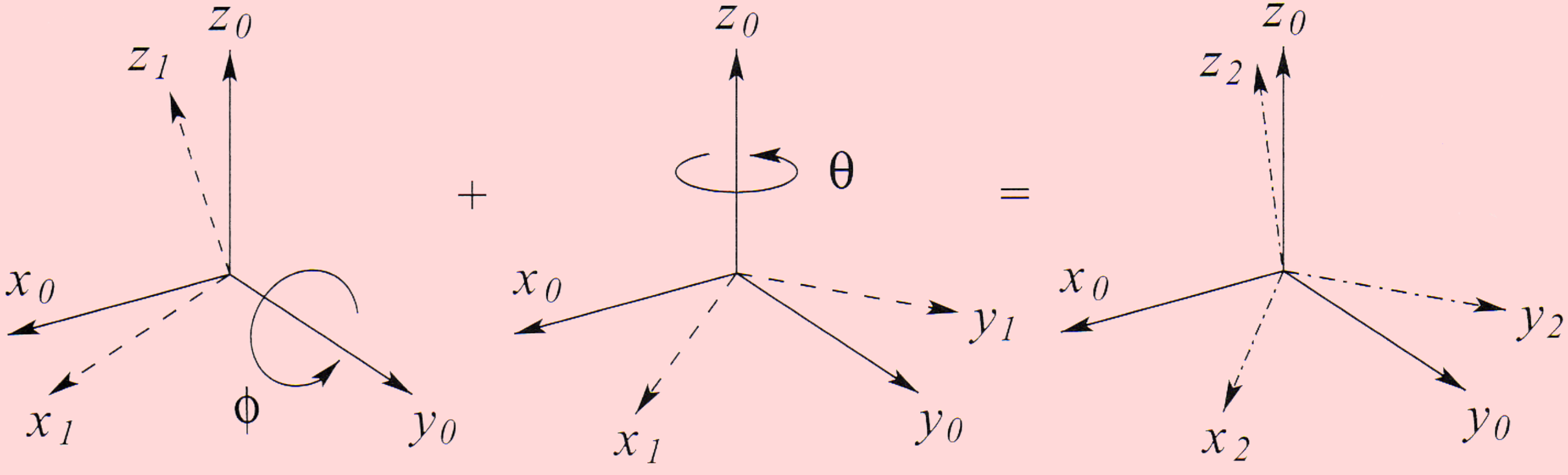
$$\mathbf{p}_b^0 = ?$$

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_b^1$$

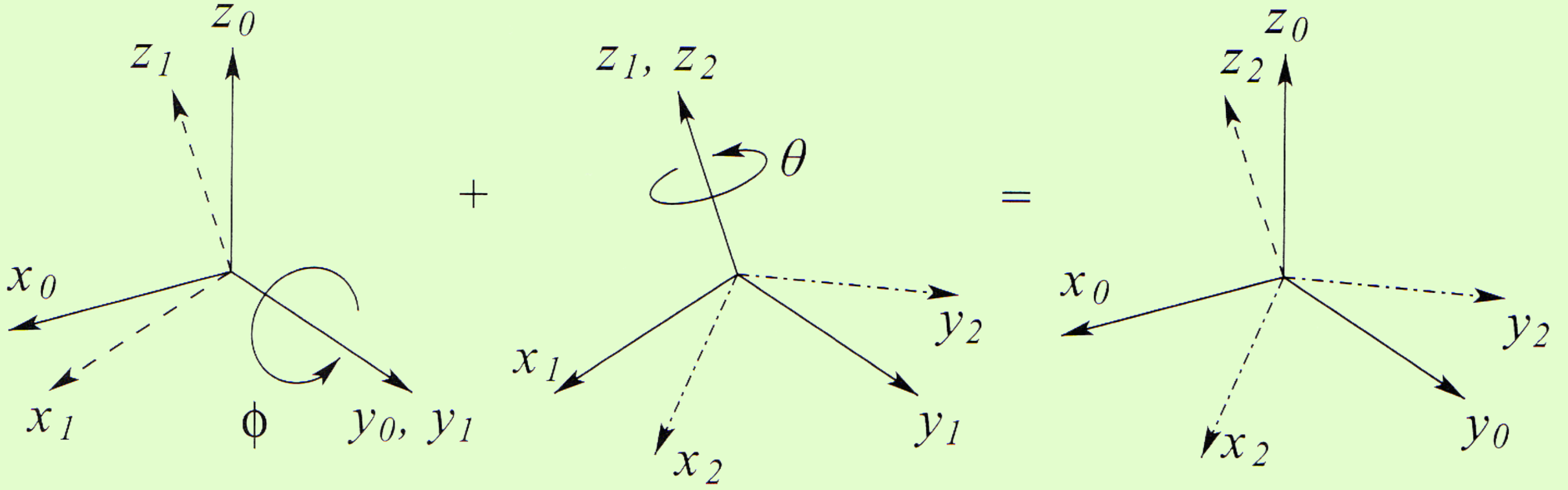
$$\mathbf{p}_b^1 = \mathbf{p}_a^0$$

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_a^0 \qquad \boxed{\mathbf{p}_b^0 = \mathbf{R}\, \mathbf{p}_a^0}$$

successive rotation about fixed frame? **pre-multiply** $\mathbf{R}_2^0 = \mathbf{R}\,\mathbf{R}_1^0$



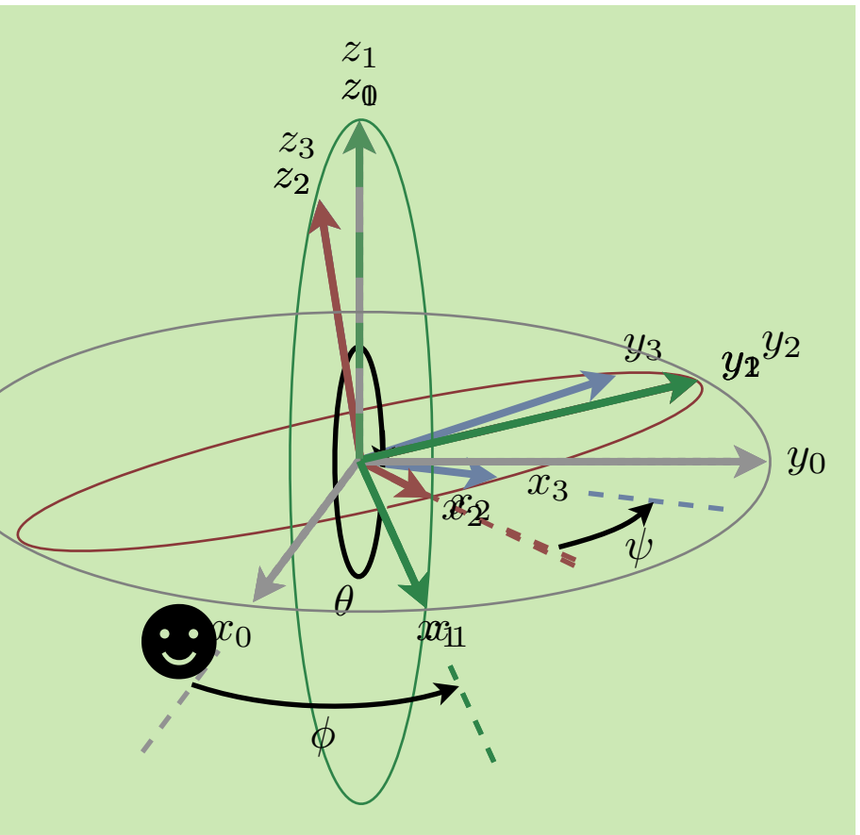
successive rotation about intermediate frame? **post-multiply** $\mathbf{R}_2^0 = \mathbf{R}_1^0\,\mathbf{R}_2^1$



Parameterizations of Rotation Matrices

Euler Angles

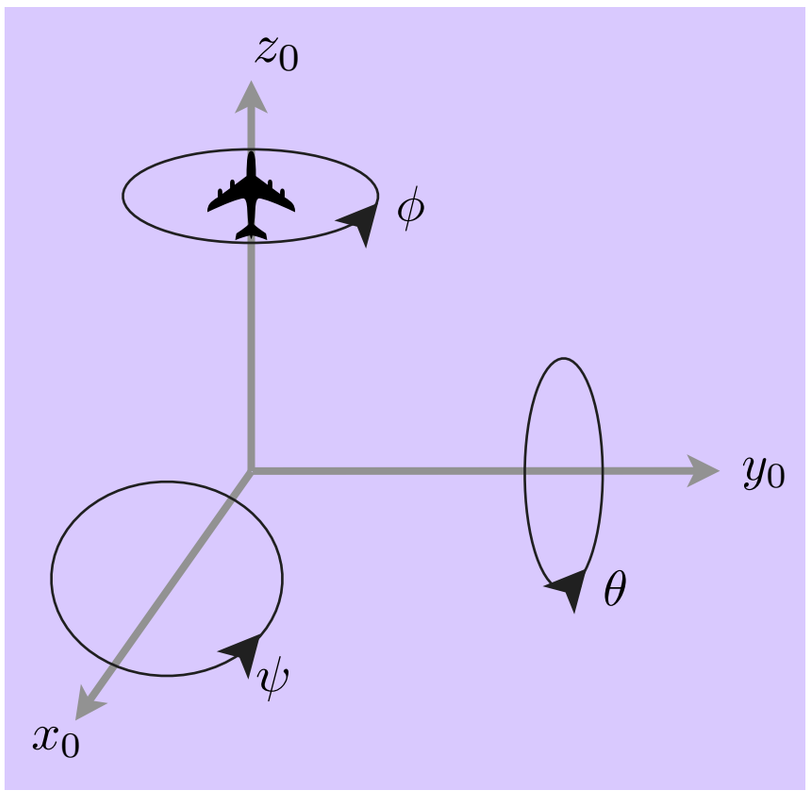
three rotations
about intermediate
frames



Our book uses ZYZ

Roll, Pitch, Yaw Angles

three rotations
about the fixed
frame



Our book uses XYZ

Axis/Angle

a unit vector
(axis)
and one angle

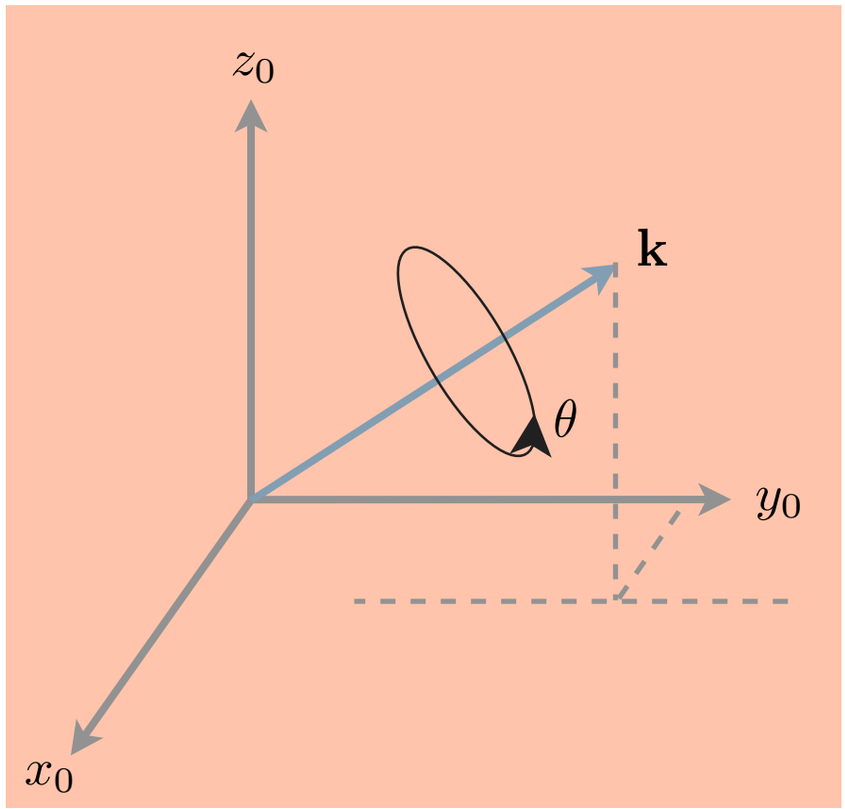


Figure 1

File Edit View Insert Tools Desktop Window Help

Press any key to continue.

$$R_1^0 = \begin{matrix} -0.1620 & -0.9324 & -0.3231 \\ 0.7479 & -0.3296 & 0.5762 \\ -0.6437 & -0.1483 & 0.7507 \end{matrix}$$

φ_a = 119.3 degrees

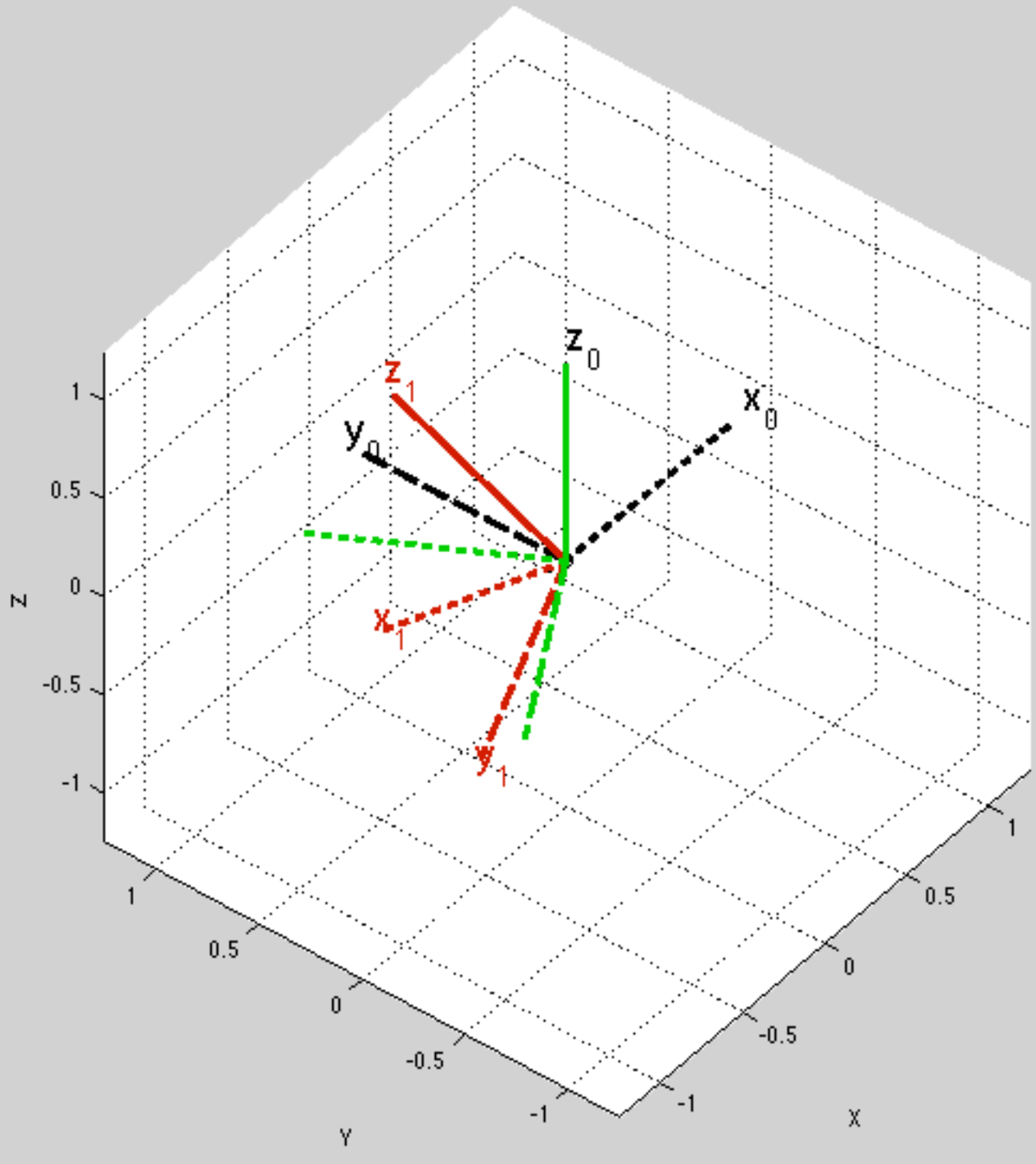
θ_a = 41.3 degrees

ψ_a = -13.0 degrees

φ_b = -60.7 degrees

θ_b = -41.3 degrees

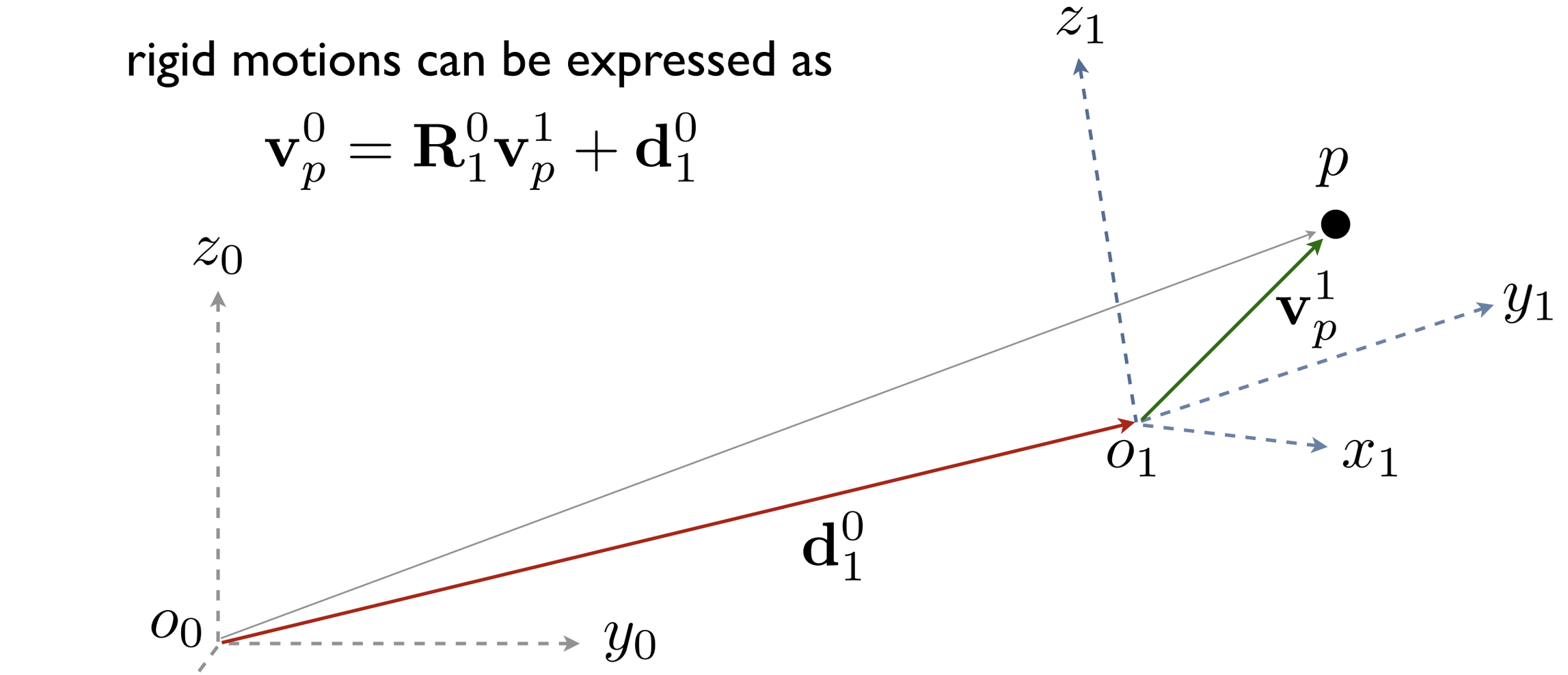
ψ_b = 167.0 degrees



a **rigid motion** couples pure translation with pure rotation

rigid motions can be expressed as

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1 + \mathbf{d}_1^0$$



$$\mathbf{H} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \quad \mathbf{v}_p^0 = \mathbf{H}_1^0 \, \mathbf{v}_p^1$$

$\mathbf{R}_1^0$ $\mathbf{d}_1^0$ $\mathbf{v}_p^1$

composition of multiple transforms is the same as for rotation matrices:

post-multiply when successive rotations are relative to intermediate frames

$$\mathbf{H}_2^0 = \mathbf{H}_1^0 \, \mathbf{H}_2^1$$

pre-multiply when successive rotations are relative to the first fixed frame

$$\mathbf{H}_2^0 = \mathbf{H} \, \mathbf{H}_1^0$$

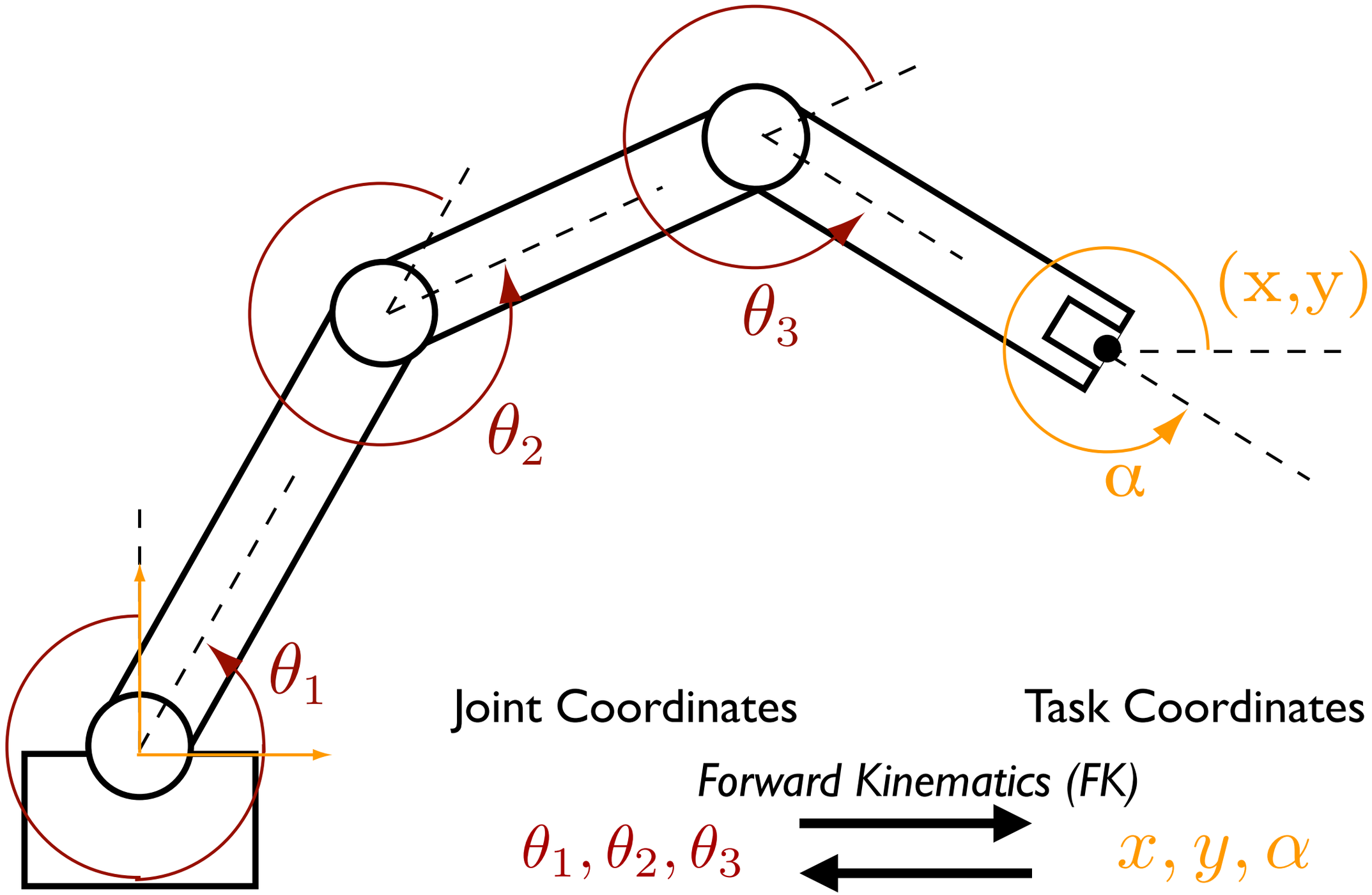
Composition (intermediate frame)

$$\mathbf{H}_2^0 = \mathbf{H}_1^0\,\mathbf{H}_2^1 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{d}_2^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Inverse Transform

$$\mathbf{H}_0^1 = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_1^0)^\top & -(\mathbf{R}_1^0)^\top\mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Joint and Task Coordinates



Joint Coordinates

Task Coordinates

Forward Kinematics (FK)

$\theta_1, \theta_2, \theta_3$ ⇄ x, y, α

Inverse Kinematics (IK)

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains

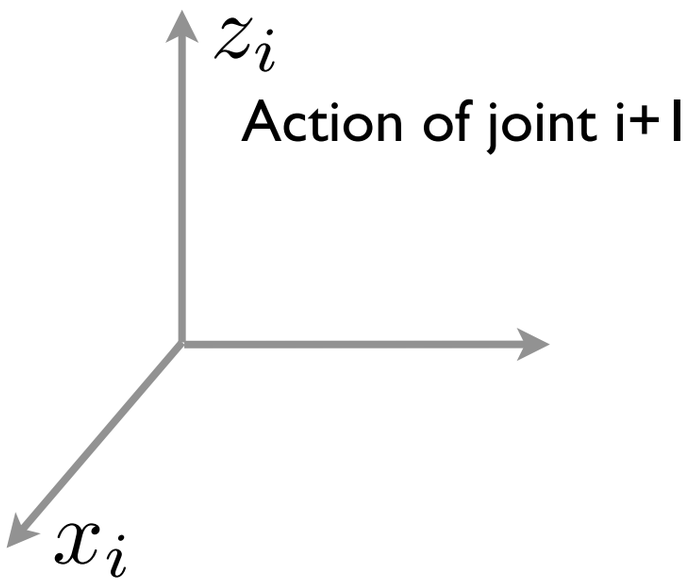
Start by drawing a schematic of the robot in its zero pose.

Then attach one frame to each link:

the joint variable for joint i+1 acts along/around z_i

the orientation of z_i determines the joint angle's positive direction

the axis x_i is perpendicular to, and intersects, z_{i-1}



Must also choose a location for the base (0) frame:

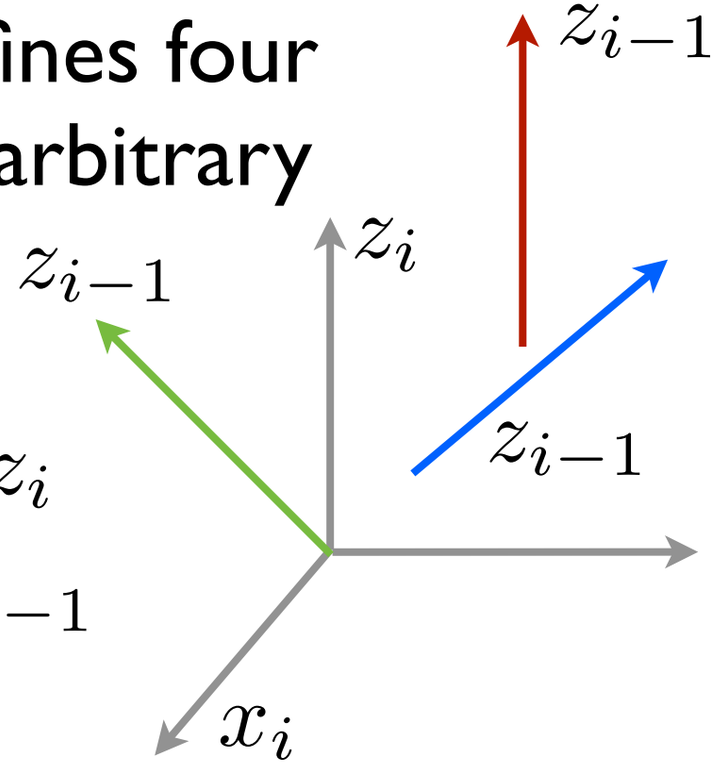
Origin must be on z_0.

x_0 and y_0 are chosen for convenience.

Takes you from previous (i-1) frame to this (i) frame

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

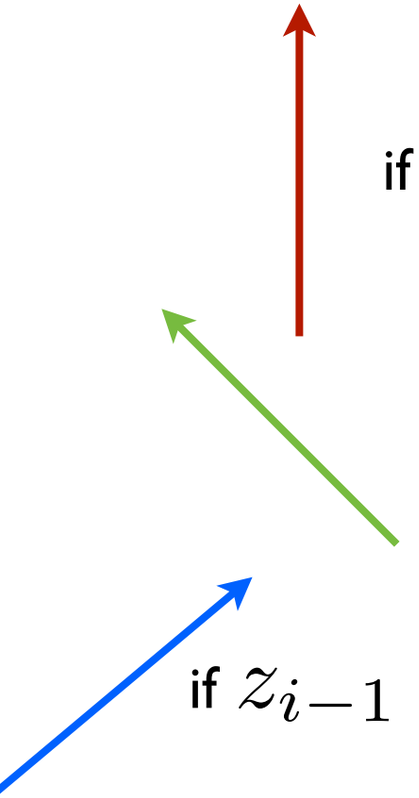
The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains



the joint variable for joint i+1 acts along/around z_i

the axis x_i is perpendicular to, and intersects, z_{i-1}

the following conventions make x placement easier (p. 82 in SHV):



if z_{i-1} is parallel to z_i	orient x_i toward z_i
if z_{i-1} intersects z_i	orient x_i normal to the plane formed by z_{i-1} and z_i
if z_{i-1} is not coplanar with z_i	orient x_i along normal with z_{i-1}, toward z_i

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains (see page 110-111 for parameter definitions)

DH parameters are usually written in this order, but I prefer the opposite order.

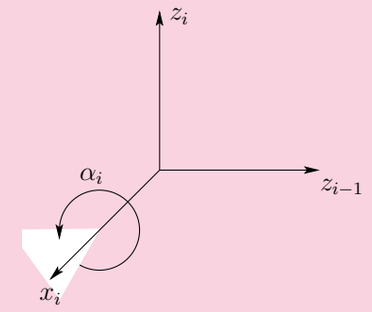
x step

a_i

Link Length the distance between z_{i-1} and z_i, measured along x_i

α_i

Link Twist the angle between z_{i-1} and z_i, measured in the plane normal to x_i (right-hand rule around x_i)



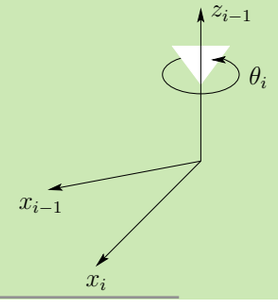
z step

d_i

Link Offset the distance between x_{i-1} and x_i, measured along z_{i-1}

θ_i

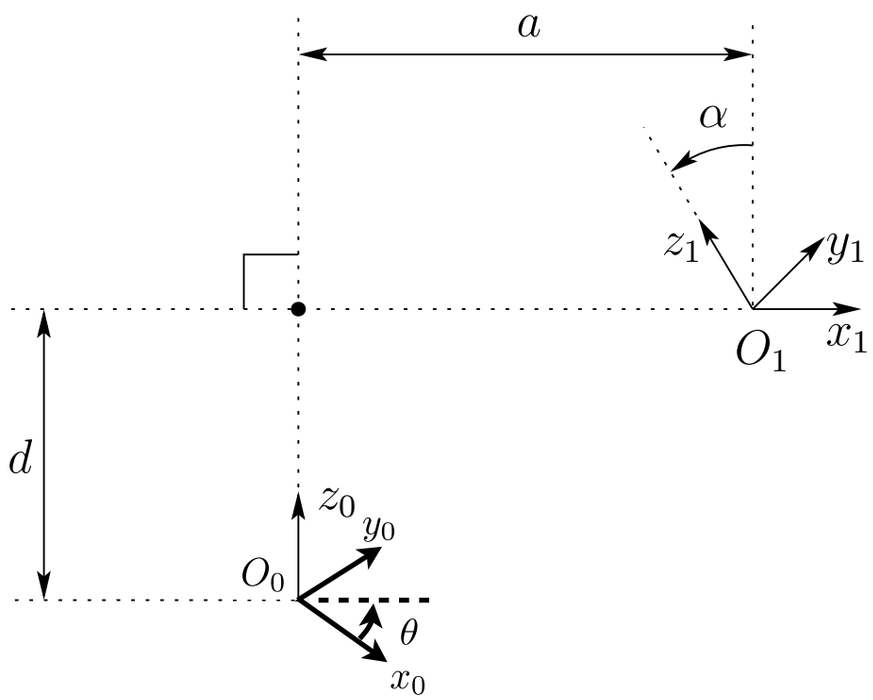
Joint Angle the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1} (right-hand rule around z_{i-1})



The Denavit-Hartenberg (DH) Convention for Manipulator Forward Kinematics

Given the design of my robot and the current values for its joint variables, where is the end-effector, and how is it oriented?

$$\mathbf{T}_n^0 = A_1(q_1) \cdots A_n(q_n)$$



$$A_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,\mathrm{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link	a_i	α_i	d_i	θ_i

DH Coordinate Frame Assumptions

(DH1) The axis x_1 is perpendicular to the axis z_0.

(DH2) The axis x_1 intersects the axis z_0.

In what order are the transformations applied?

Intermediate frames
Post-multiply

$$A_i = Rot_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate around z by theta

Translate along z by d

Translate along new x by a

Rotate around new x by alpha

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

Great List of DH Steps on page 110-111 in SHV

Procedure for Deriving the Forward Kinematics of a Manipulator
Following the Denavit-Hartenberg (DH) Convention
From "Robot Modeling and Control" by Spong, Hutchinson, and Vidyasagar

Step l: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$, perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n-th joint is revolute, set $z_n = a$ along the direction z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

Step 7: Create a table of link parameters a_i, d_i, α_i, θ_i.

(fixed!)

$a_i =$ distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i

$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

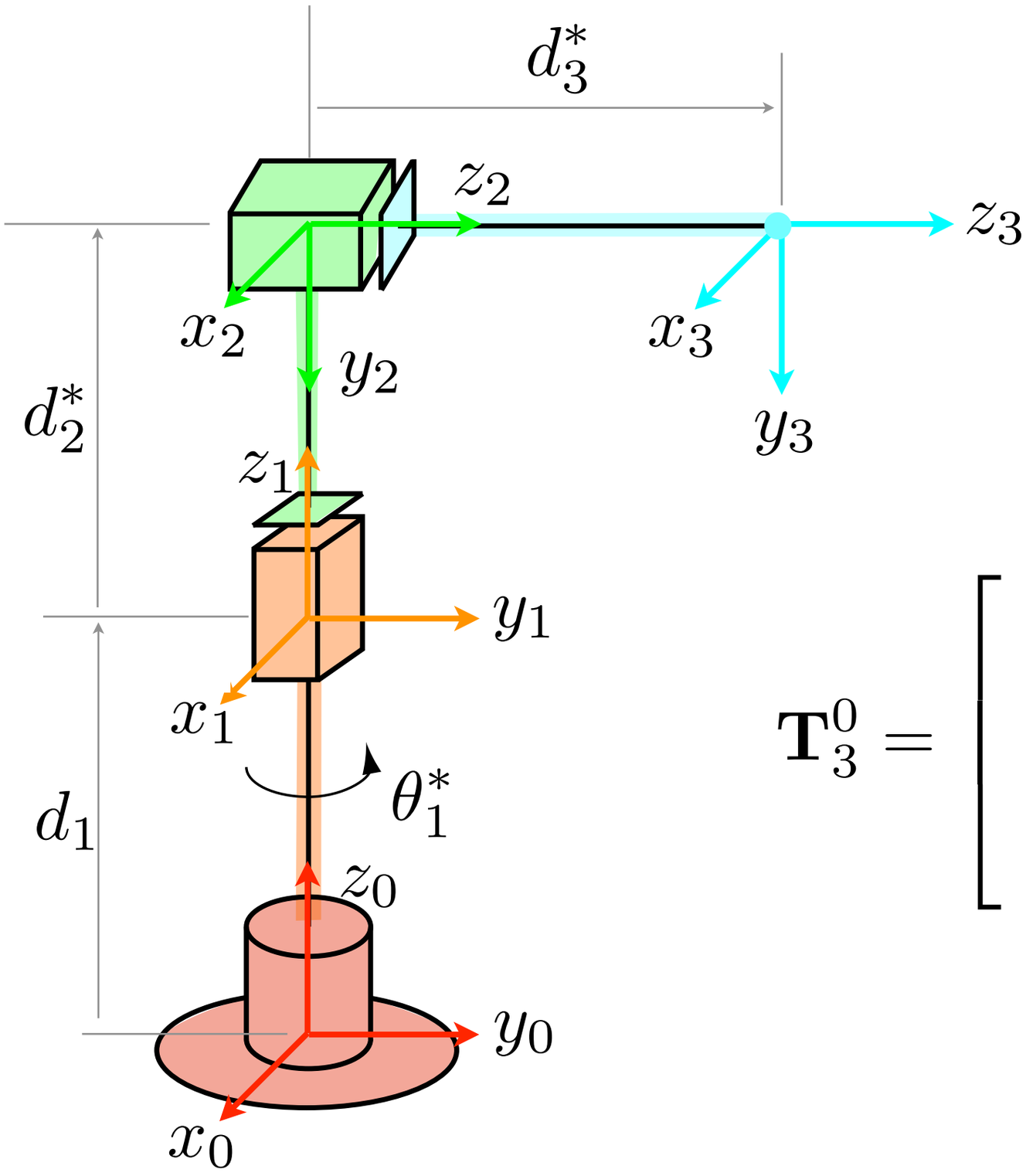
$\alpha_i =$ the angle between z_{i-1} and z_i measured about x_i.

$\theta_i =$ the angle between x_{i-1} and x_i measured about z_{i-1}. θ_i is variable if joint i is revolute.

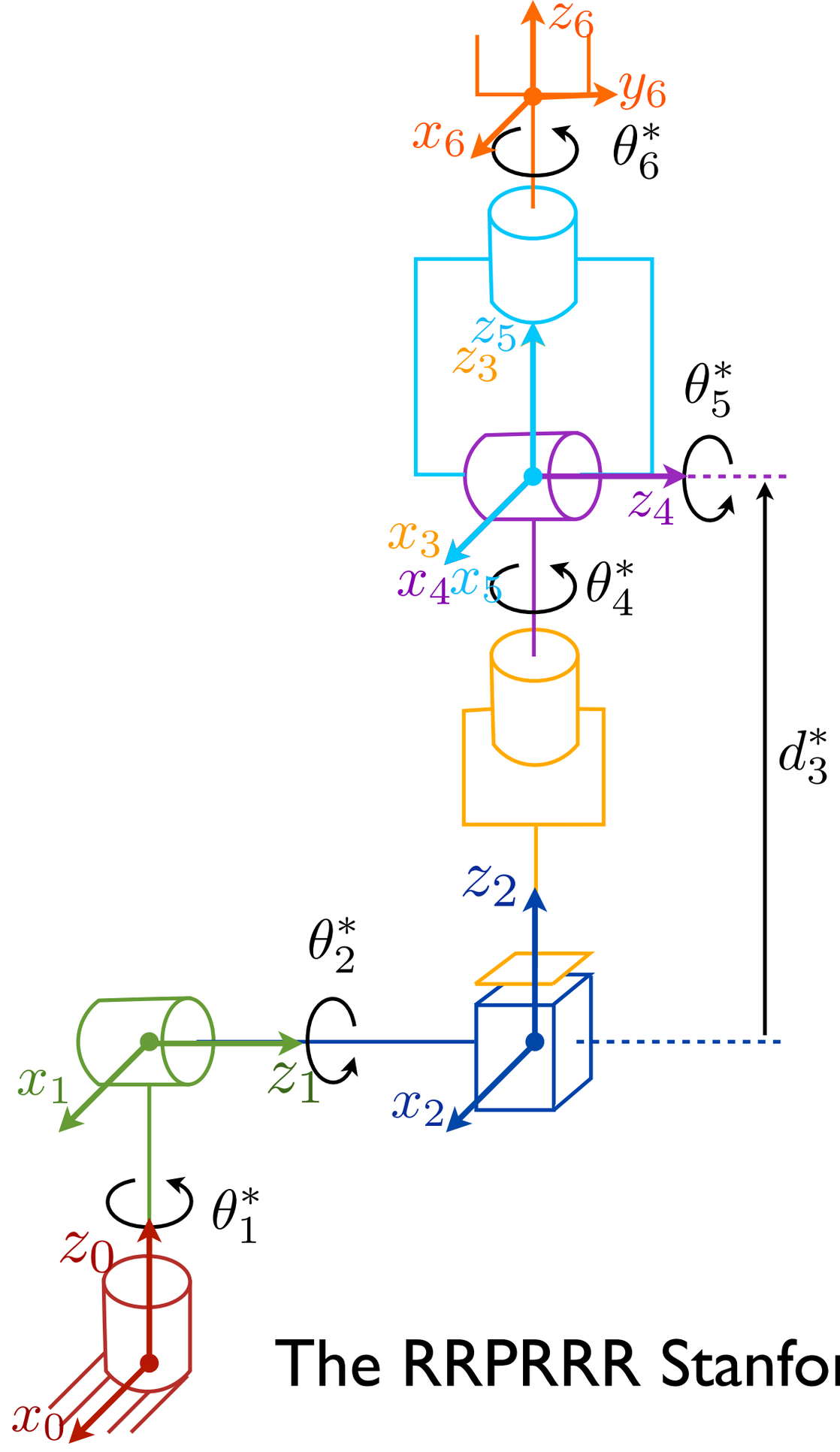
Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

The RPP Cylindrical Robot



$$\mathbf{T}_3^0 = \begin{bmatrix} c_1^* & 0 & -s_1^* & -d_3^* s_1^* \\ s_1^* & 0 & c_1^* & d_3^* c_1^* \\ 0 & -1 & 0 & d_1 + d_2^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



	x-step		z-step		
Link	a_i	α_i	d_i	θ_i	
1	0	$-90°$	d_1	θ_1^*	●→●
2	0	$+90°$	d_2	θ_2^*	●→●
3	0	$0°$	d_3^*	$0°$	●→●
4	0	$-90°$	0	θ_4^*	●→●
5	0	$+90°$	0	θ_5^*	●→●
6	0	$0°$	d_6	θ_6^*	●→●

The RRPRRR Stanford Manipulator with Spherical Wrist

Specifying Joint Values - Line

$\dot{q}$

$\left(\dot{q} = \frac{\mathrm{d}q}{\mathrm{d}t}\right)$

Discontinuities in velocity require infinite acceleration, which is not physically realizable.

t_0 t_f t

q

q_f

q_0

$$\frac{q_f - q_0}{t_f - t_0}$$

$$q(t) = q_0 + \frac{t - t_0}{t_f - t_0}(q_f - q_0)$$

t_0 t_f t

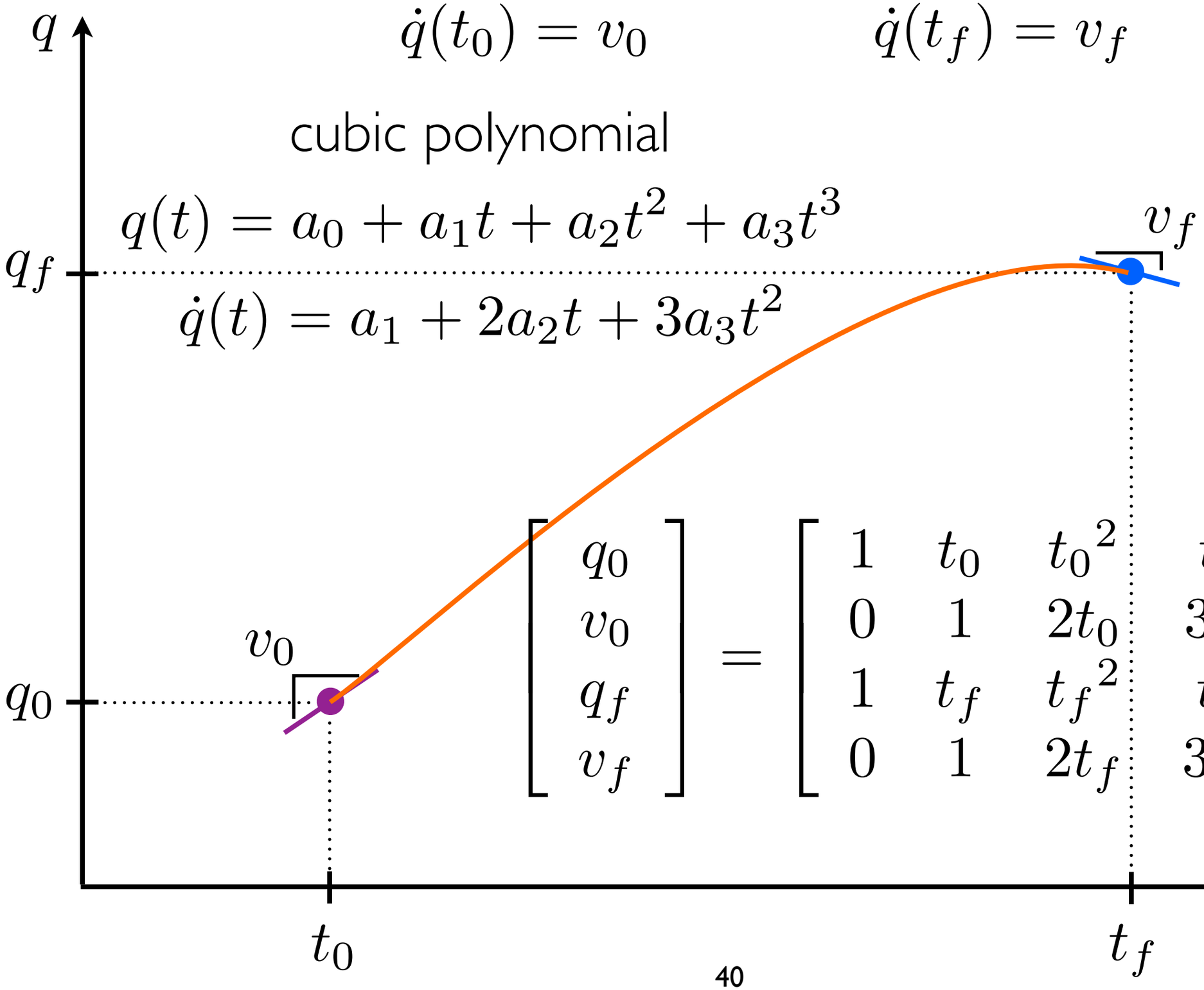
Specifying Joint Values and First Time Derivatives - Cubic

$$q(t_0) = q_0 \qquad q(t_f) = q_f$$
$$\dot{q}(t_0) = v_0 \qquad \dot{q}(t_f) = v_f$$

cubic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$



$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Specifying Joint Values Plus First and Second Time Derivatives

Quintic Polynomial Trajectories

start		end
$q(t_0) = q_0$	$\longrightarrow$	$q(t_f) = q_f$
$\dot{q}(t_0) = v_0$	$\longrightarrow$	$\dot{q}(t_f) = v_f$
$\ddot{q}(t_0) = \alpha_0$	$\longrightarrow$	$\ddot{q}(t_f) = \alpha_f$

quintic polynomial

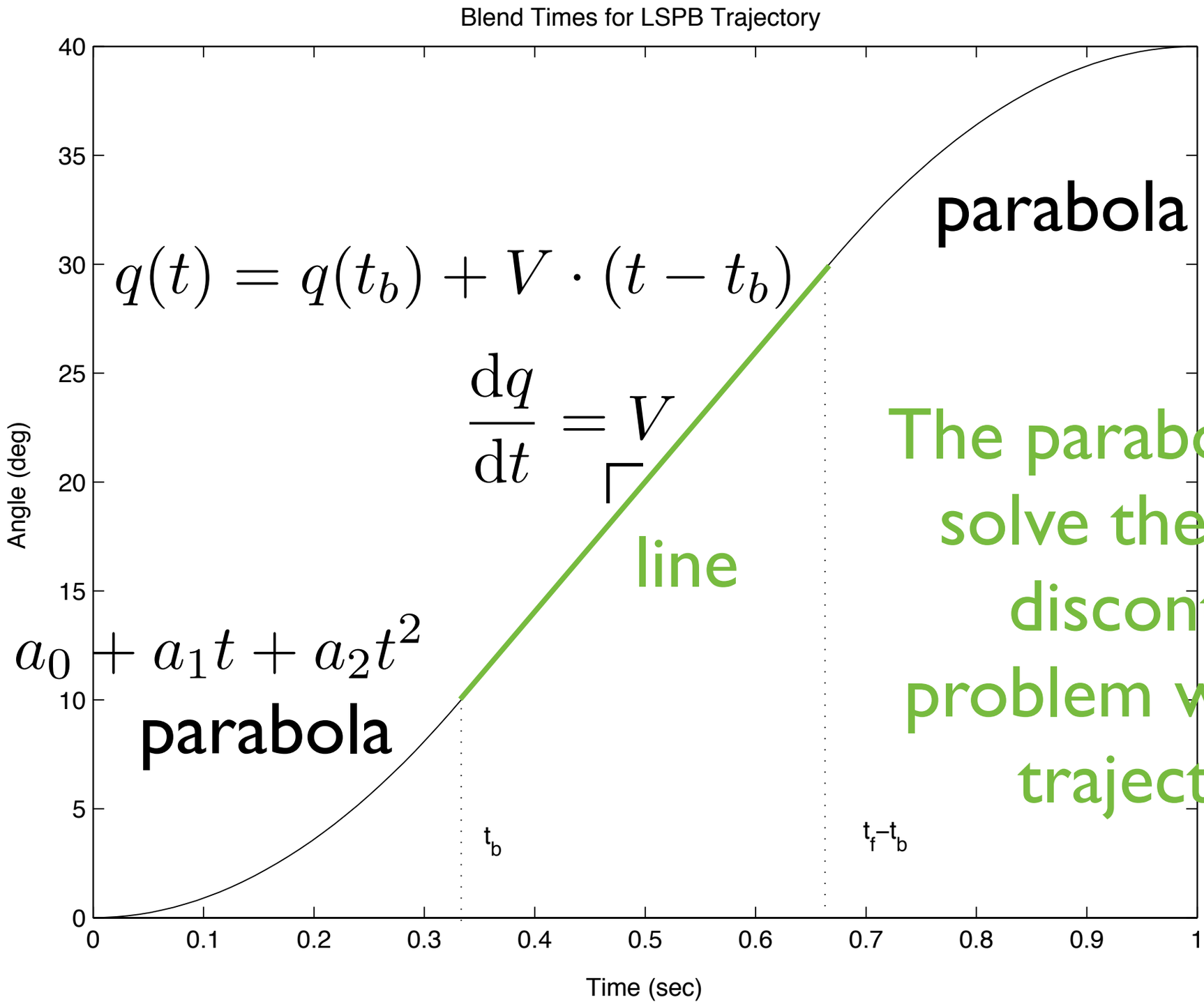
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

Specifying Constant Velocity for Central Portion

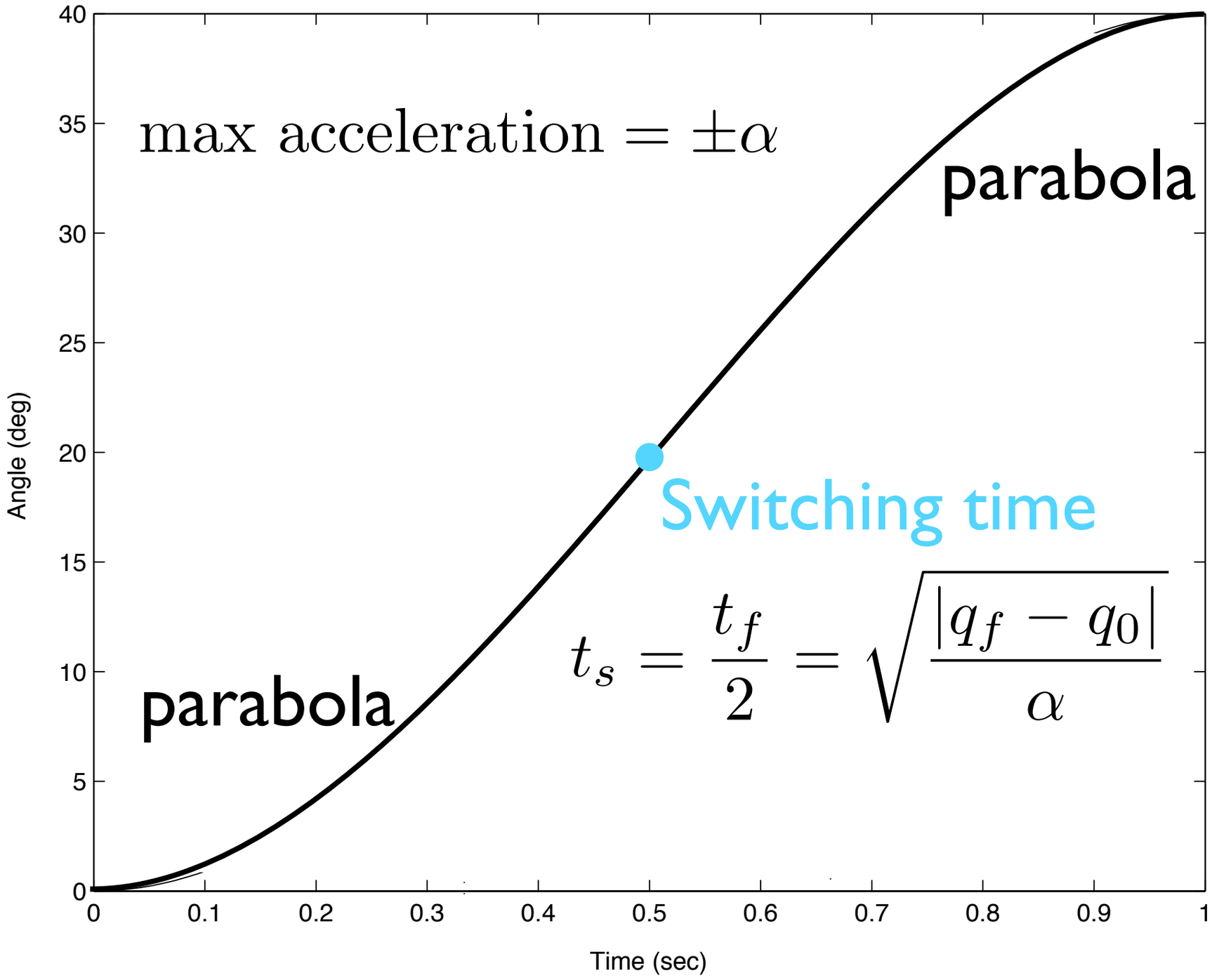
Linear Segments with Parabolic Blends (LSPB)



The parabolic blends solve the velocity discontinuity problem with linear trajectories.

Getting There As Fast As Possible

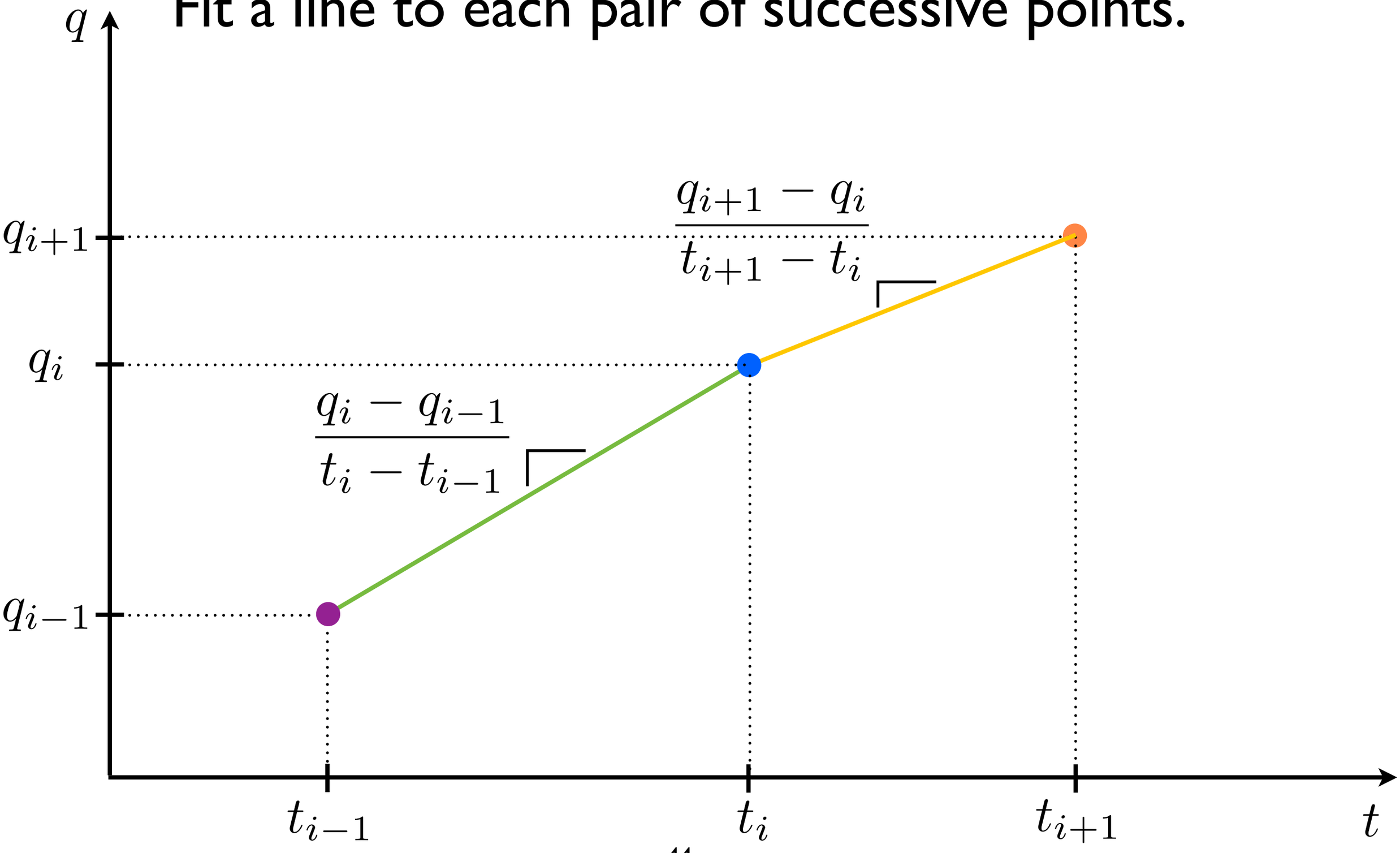
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories



You have a list of times and associated joint angles.

How to estimate joint velocity?

Fit a line to each pair of successive points.



Derivative of a Rotation Matrix

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta}\frac{d\theta}{dt}$$

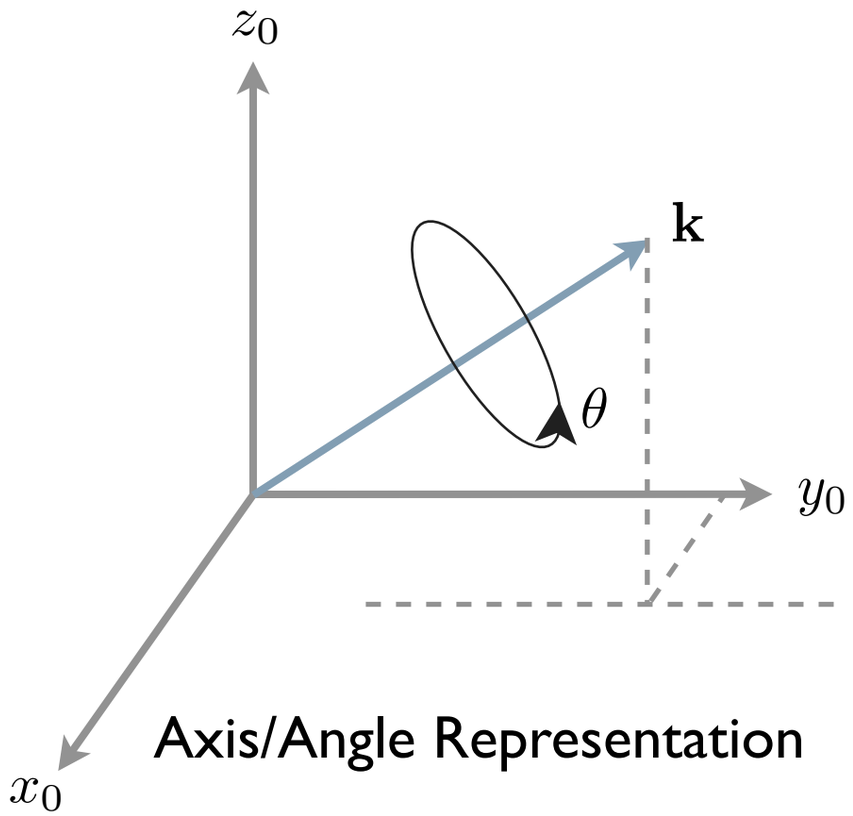
✓

?

$$S + S^T = 0$$

$$S(\vec{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$S(\vec{a})\vec{p} = \vec{a} \times \vec{p}$$



Axis/Angle Representation

$$\frac{dR}{d\theta} = S(\hat{\omega})\,R$$

The skew-symmetric matrix S defines the axis about which rotation is occurring.

$$\frac{dR}{dt} = S(\vec{\omega})\,R$$

In general, you simply form S from the angular velocity vector and don't need to differentiate the matrix.

$$\vec{\omega}_2^0 = \vec{\omega}_{0,1}^0 + R_1^0\vec{\omega}_{1,2}^1$$

$$\dot{\vec{p}}^0 = S(\vec{\omega}^0)R_1^0\vec{p}^1 + \dot{\vec{o}}_1^0$$

Jacobians

$$v_n^0 = J_v \dot{q}$$

(3 x 1) (3 x n)(n x 1)

$$\omega_n^0 = J_\omega \dot{q}$$

(3 x 1) (3 x n)(n x 1)

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

$$\omega_{0,n}^0 = \sum_{i=1}^{n} \rho_i (\mathbf{R}_{i-1}^0 \hat{z})\, \dot{\theta}_i$$

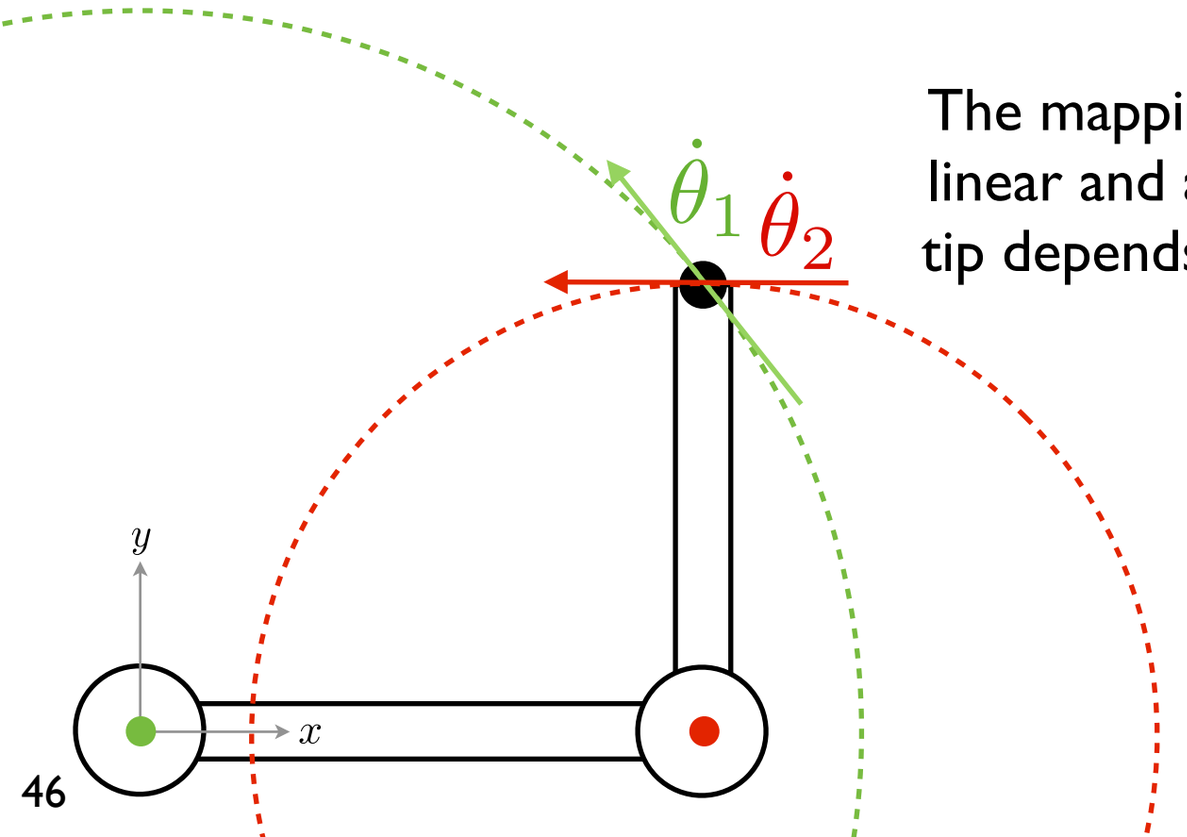
$$\rho_i = \begin{matrix} \text{0 for prismatic} \\ \text{1 for revolute} \end{matrix}$$

Prismatic $J_{v_i} = z_{i-1}$

Revolute $J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$

Prismatic $J_{\omega_i} = 0$

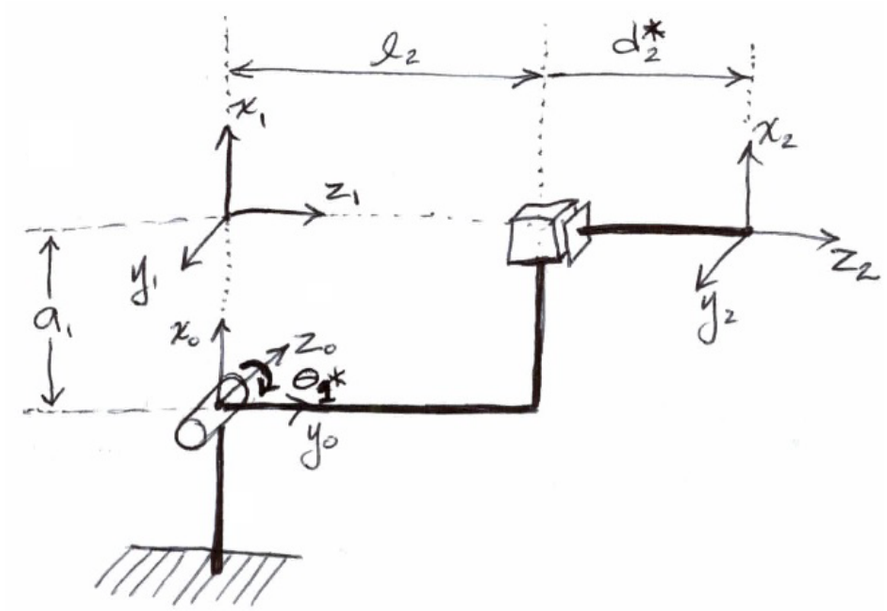
Revolute $J_{\omega_i} = z_{i-1}$



The mapping from joint velocities to the linear and angular velocity of the robot's tip depends on the robot's current pose!

A robot loses the ability to move its end-effector in certain directions when $\det(J_v) = 0$

Singular configurations or singularities



$$\xi = J(q)\dot{q}$$

(6 x 1) body velocity

but this is not the derivative of any vector

(6 x n) Jacobian

(n x 1) joint velocities

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

(6 x n) Analytical Jacobian

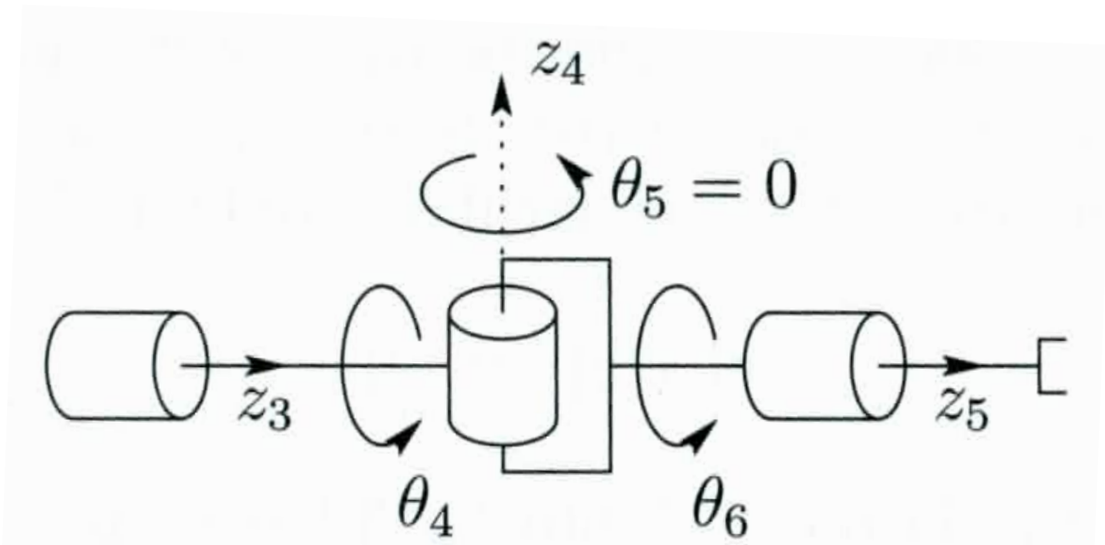
$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\frac{J_{11}}{J_{21}} \middle| \frac{J_{12}}{J_{22}}\right]$$

if we choose $o_4 = o_5 = o_6$

$$J = \left[\frac{J_{11}}{J_{21}} \middle| \frac{0}{J_{22}}\right]$$

$$\det(J) = \det(J_{11})\det(J_{22})$$

arm wrist



$\theta_5 = 0, \pi$ are singular configurations

The transpose of the Jacobian relates joint forces and torques to Cartesian end-effector forces and torques

$$\underset{(n \times 1)}{\vec{\tau}} = \underset{(n \times 6)}{J^T(\vec{q})} \underset{(6 \times 1)}{\vec{F}}$$

joint forces and torques

Jacobian matrix transpose

endpoint forces and torques

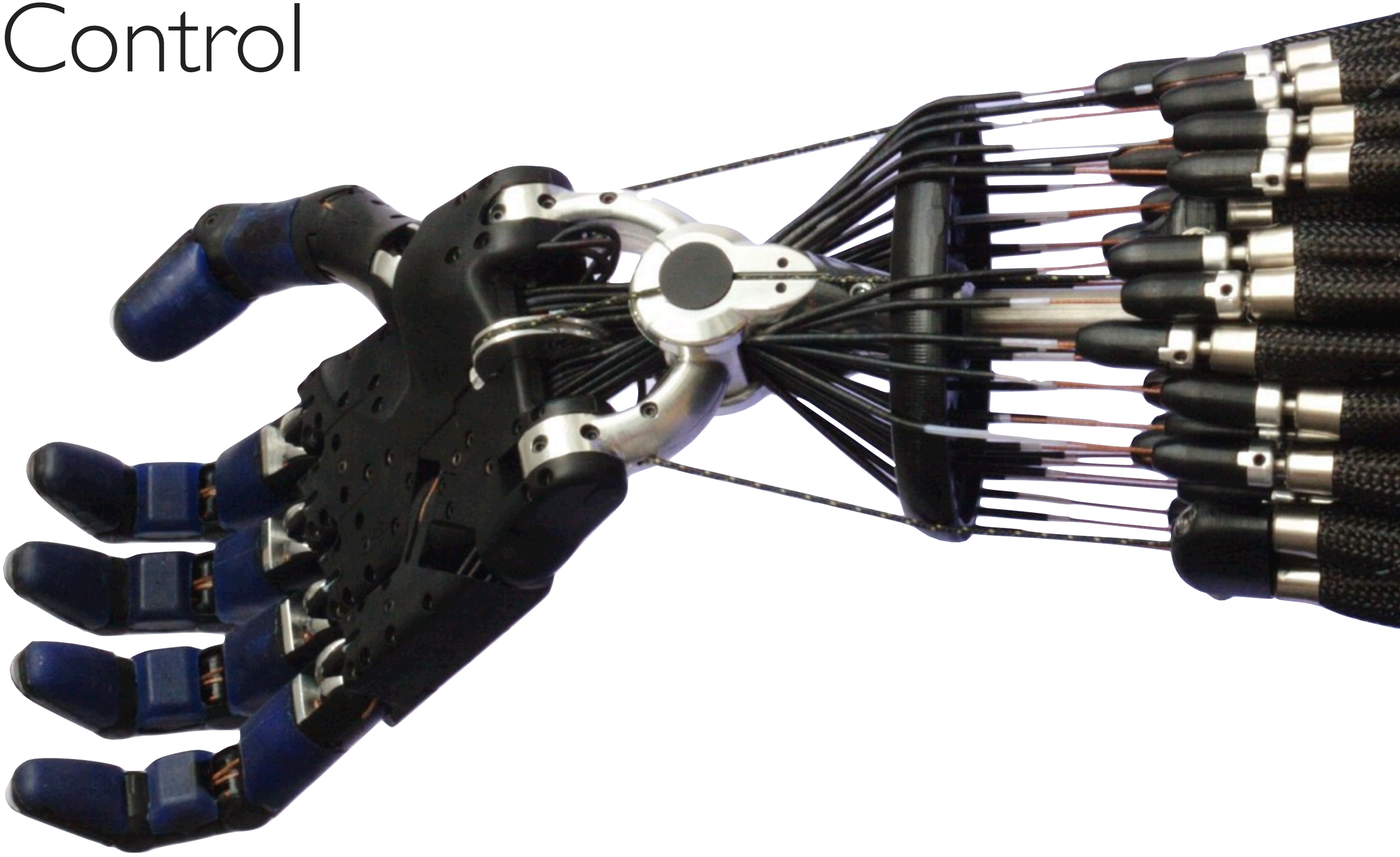
For a specific configuration, the Jacobian scales the input (joint velocities) to the output (body velocity)

$$\xi = J(q)\dot{q}$$

If you put in a joint velocity vector with unit norm, you can calculate in which direction and how fast the robot's end-effector will translate and rotate.

This approach allows you to calculate and plot the manipulability ellipsoid – a geometrical representation of all the possible tip velocities for a normalized joint velocity input.
A 6D ellipsoid is hard to visualize,
but 2D and 3D ellipsoids are lovely and useful.

Manipulator Hardware and Control



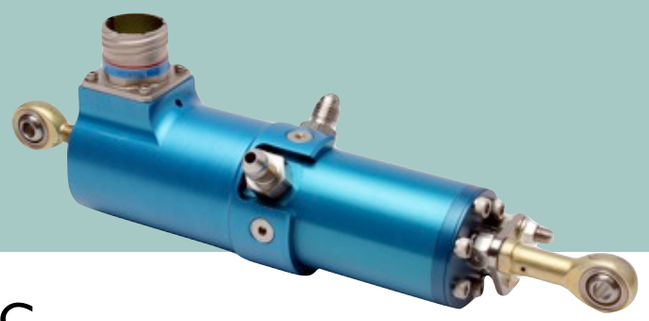
A Biological Inspiration

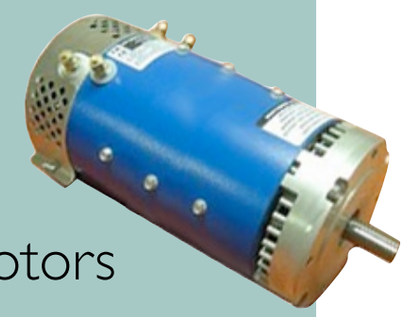
Mechanical Structure

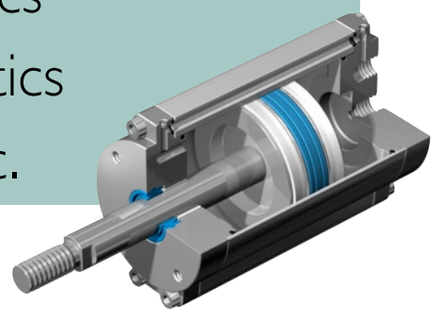
Bones	Frame / Links
Joints	Joints

Actuators

Muscles	Electric Motors
	Hydraulics
	Pneumatics
	SMA, etc.





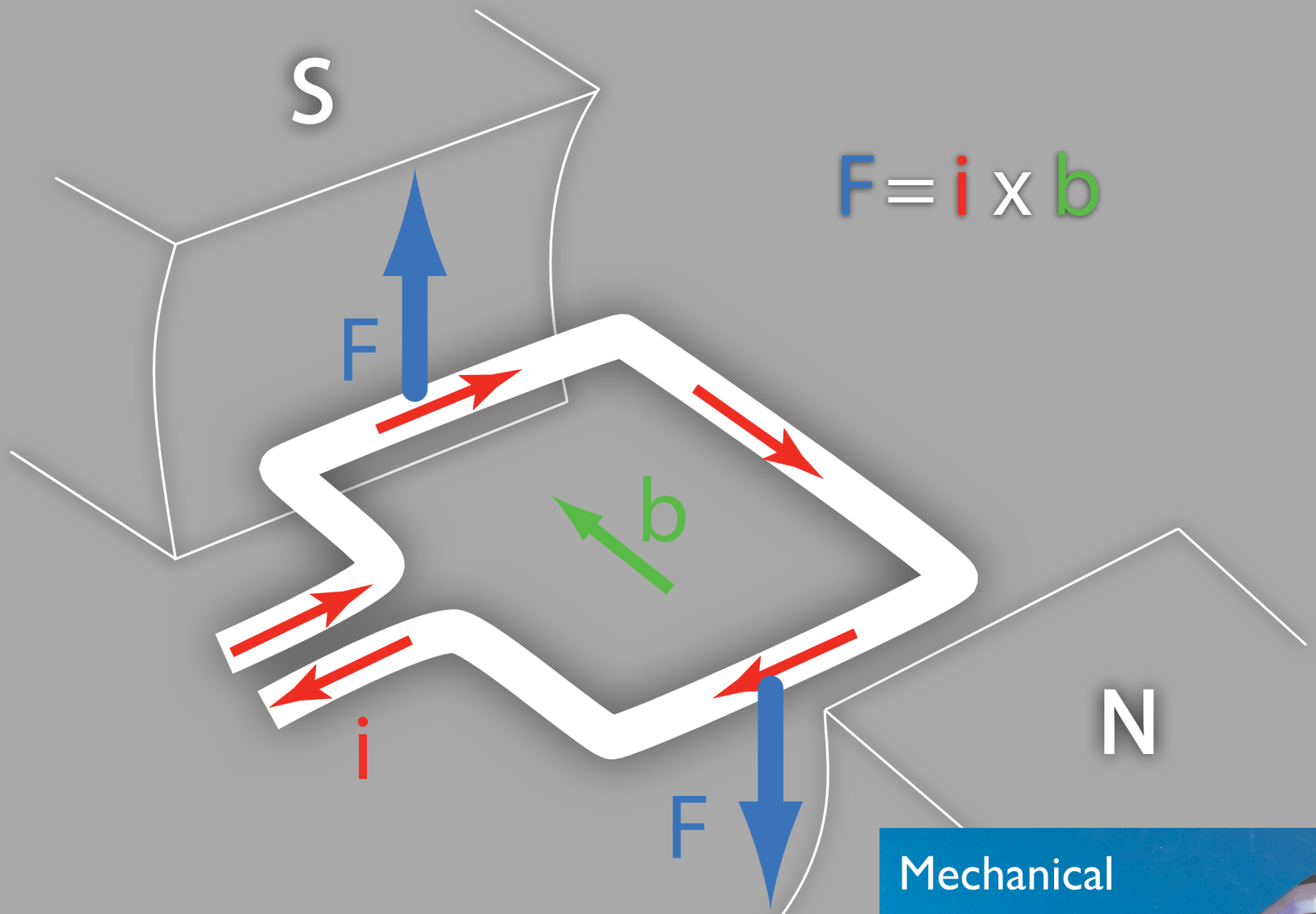


Sensors

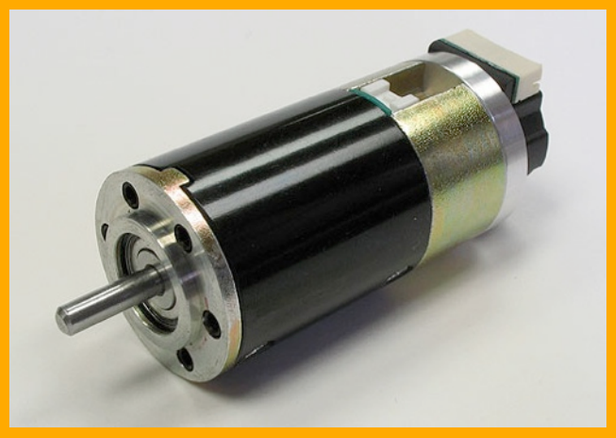
Kinesthetic	Encoders
Tactile	Load Cells
Vision	Vision
Vestibular	Accelerometers

Controller

Brain	Computer
Spinal Cord Reflex	Local feedback



$F = i \times b$



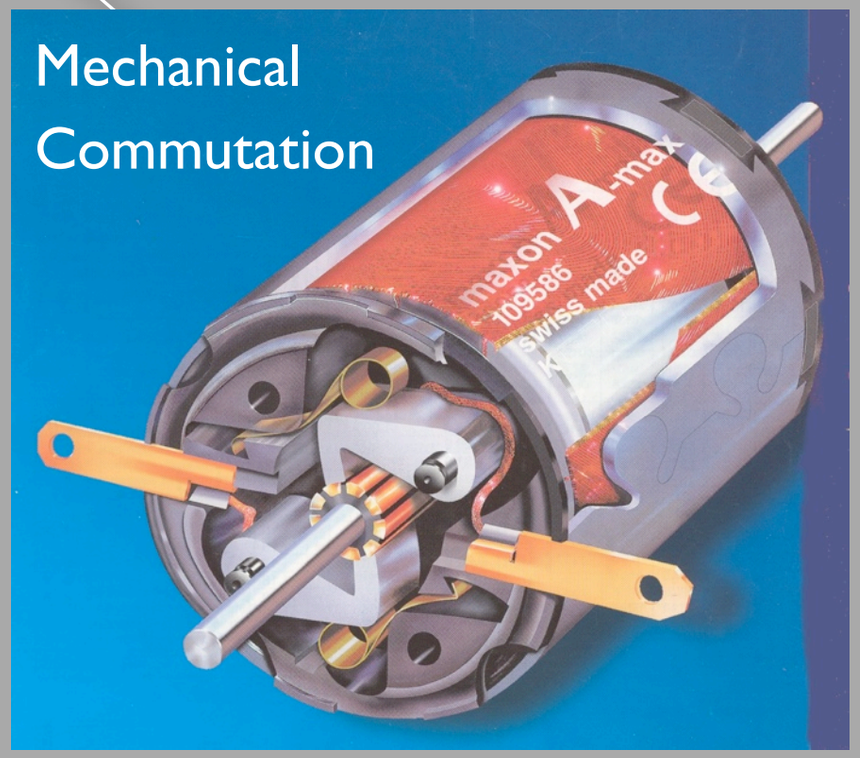
DC Brushed

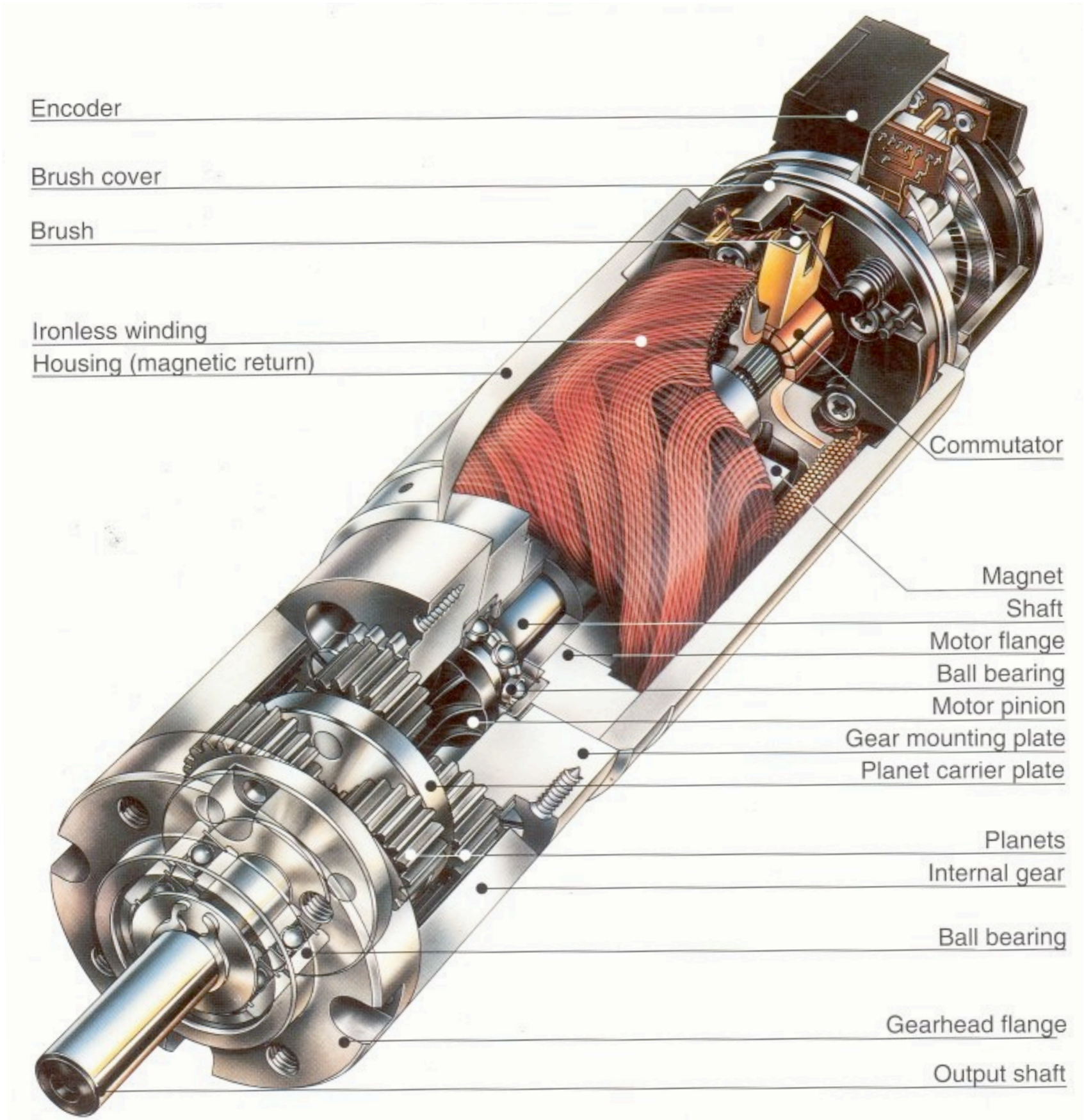
Most common!

Coil Rotor

Magnetic Stator

Brushes carry current to the rotor





torque constant (N•m/A)

$$\tau_m = k_t \, i_a$$

generated torque (N•m)

armature current (A)

back emf (V)

back-emf constant (V•s)

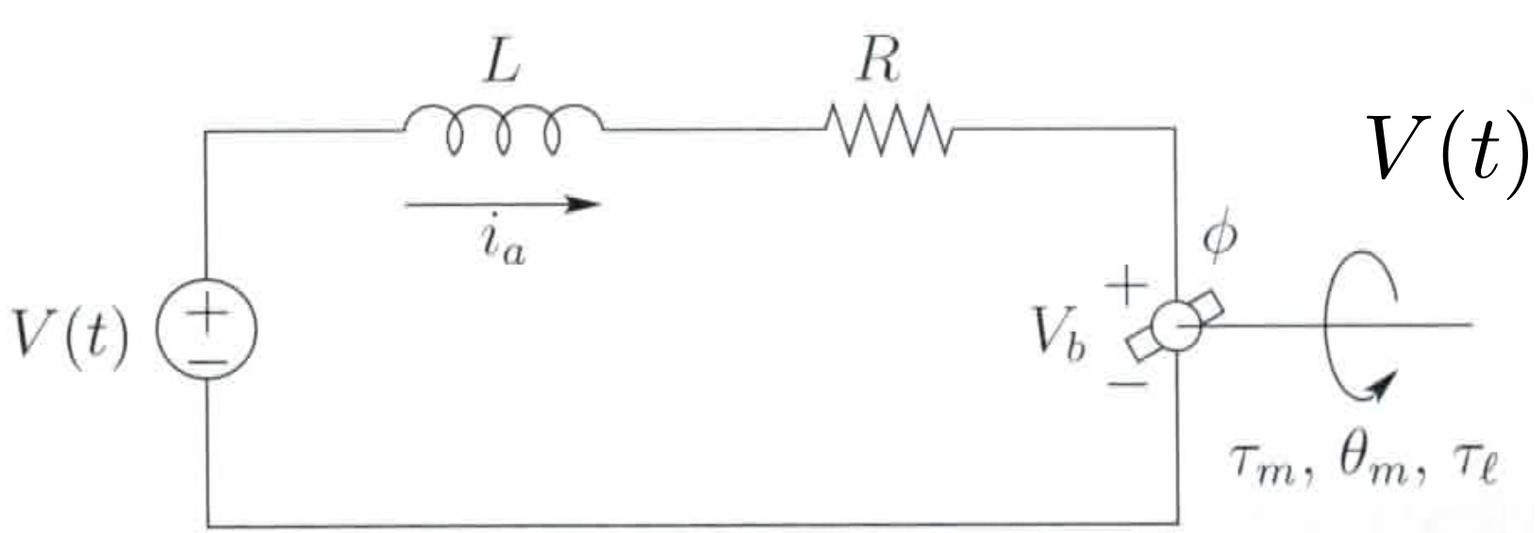
$$V_b = k_v \, \omega_m$$

motor velocity (rad/s)

$$k_t = k_v$$

if using meters, kilograms and seconds

Electrical Circuit Diagram of a DC Brushed Motor



Use Kirchoff's Voltage Law:

$$V(t) = L\frac{di_a}{d_t} + R\,i_a + \overbrace{k_v\,\omega_m}^{V_b}$$

If I hold the motor's shaft stationary so it cannot rotate,
the motor is merely an inductor in series with a resistor.

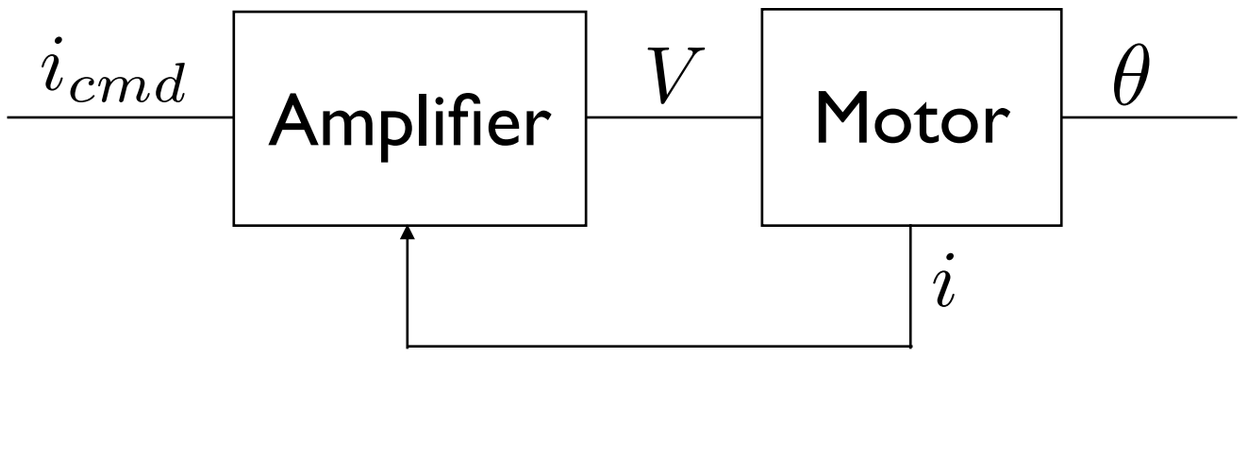
Usually we want to allow motors to rotate!

Want to control torque output.... $\tau_m = k_t\,i_a$

I could just ignore the inductance and back-emf, assuming $i_a \approx \frac{V(t)}{R}$

Not a terrible solution, but inductance is often non-negligible,
back-emf is large when speeds are high, and R changes with temperature.

These electrical dynamics can affect performance, so engineers sometimes try to overcome them
by using a current controlled circuit (current drive) instead of voltage drive.


i_{cmd}
Amplifier
V
Motor
θ
i