16-642 Fall 2017: Reference Notes for Part 1
Spatial Representations
(Lectures 1, 2, and 3)

George Kantor

28,30 August and 6 September 2017

1 Introduction

Hand out syllabus and review.

Note that the first topic we will cover will be robotic manipulation. If you want material to supplement the lectures and lecture notes, the best source is:

Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, Robot Modeling and Control, John Wiley and Sons, 2006.

This book covers much of the same material and uses the same conventions that we will use in class.

2 Some Terminology

Before we can begin discussing how robotic manipulators work, we need to introduce some mathematical terminology and tools.

Ambient Space: the space that a robot operates in, usually the plane (E^2) or 3 dimensional Euclidean space (E^3).

System: a set of point in ambient space.

Configuration: a description of the location of every point in a system.

Configuration Space: the space of all possible configurations

Degrees of Freedom: the dimension of the configuration space

Rigid Body: a (possbily infinite) collection of points that is constrained such that

1. the distance between any two points remains constant (i.e., no distortions allowed)
2. the handedness of the collection always stays the same (i.e, no reflections allowed)

Displacement: a possbile motion of a rigid body (rotations, translations, and combinations thereof).

Some examples of degrees of freedom:

- a point in a plane: 2 DOF

- a point in 3-space: 3 DOF
- a rigid body in a plane: 3 DOF
- a rigid body in 3-space: 6 DOF
- a rigid body in 4-space: ???

The configuration of a rigid body in n-space is called the Special Euclidean Group and denoted $SE(n)$.

2.1 Linkages

A linkage is a combination of *links* and *joints* where a

link is a rigid body

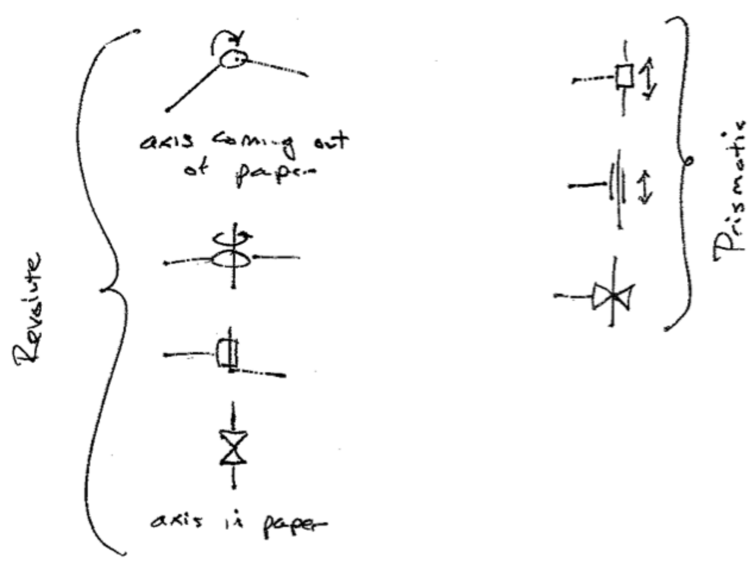
joint constrains the relative motion between two links

We mostly care about two types of joints:

revolute allows only rotation around some axis

prismatic allows only translation along some axis

We have lots of ways of drawing schematic sketches of joints:



There are two basic categories of linkages:

serial: serial linkages are composed of links and joints connected in such a way that no closed loops are formed. For grounded serial linkages, the number of degrees of freedom is equal to the number of joints.

parallel: parallel linkages are composed of links and joints that are connected to form one or more loops. The number of degrees of freedom is complicated.

2.2 Task Space and Joint Space

Task space is the space that is used to describe the problem you are trying to solve. In most cases, the task is to position the last link of a serial linkage at some desired pose (position plus orientation) in 3-space, so the task space is $SE(3)$. But sometimes we define other task spaces, especially in homework problems! Generically, we denote the task space as $\mathbb{X}$.

Joint space is the space of all joint configurations. For a grounded serial linkage, joint space is identical to configuration space. We denote the joint space as Θ.

The discipline of *manipulation* is primarily interested in defining ways to get back and forth between $\mathbb{X}$ and Θ.

3 Representating $SE(3)$

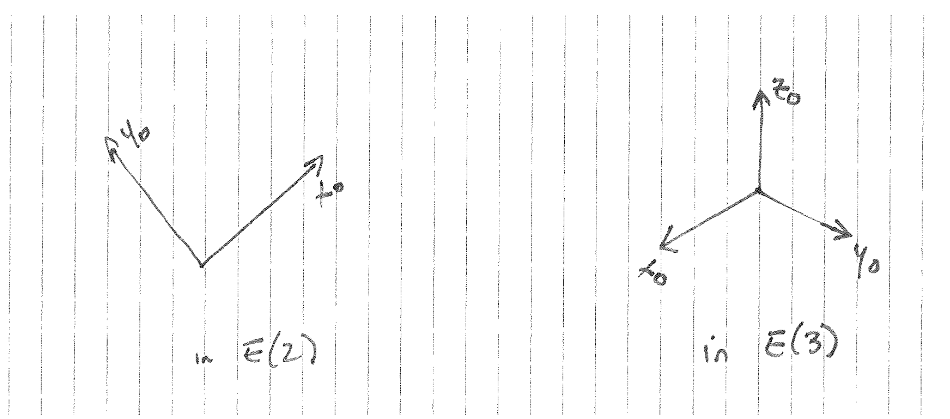
Our first job in understanding kinematics is to learn how to represent $SE(3)$, i.e., the combination of orientation and postion in 3-space (E^3). Position is relatively straightforward, it can just be represented with a 3-vector. Orienation is a bit stranger. The set of orientations in E^3 is in fact 3 dimensional, but it cannot be represented everywhere with a 3-vector. So we will have to learn some tricks. And while we're at it, we'll learn the same tricks for $SE(2)$ (actually, for $SE(n)$). But first, some background:

3.1 Frames

A **frame** in n-space is a collection of n mutually orthogonal unit vectors, for example:

- in E^2 the frame $\{0\}$ is given by two unit vectors x_0 and y_0 with the property that $x_0^T y_0 = 0$ (i.e., x_0 and y_0 are orthogonal) and $x_0 \times y_0 = 1$ (i.e., the frame is right handed).
- in E^3 the frame $\{0\}$ is given by three unit vectors that satisfy
  1. $x_0^T y_0 = 0$, $y_0^T z_0 = 0$, and $x_0^T z_0 = 0$ (i.e., the vectors are mutually orthogonal)
  2. $x_0 \times y_0 = z_0$ (i.e., the frame is right handed)

Here are sketches of example frames in the plane and in 3-space:



The unit vectors that form a frame are called basis vectors. In E^3 you are probably used the frame defined by the standard basis vectors:

$$x_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad y_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

but any collection of vectors that satisfy the conditions above will work.

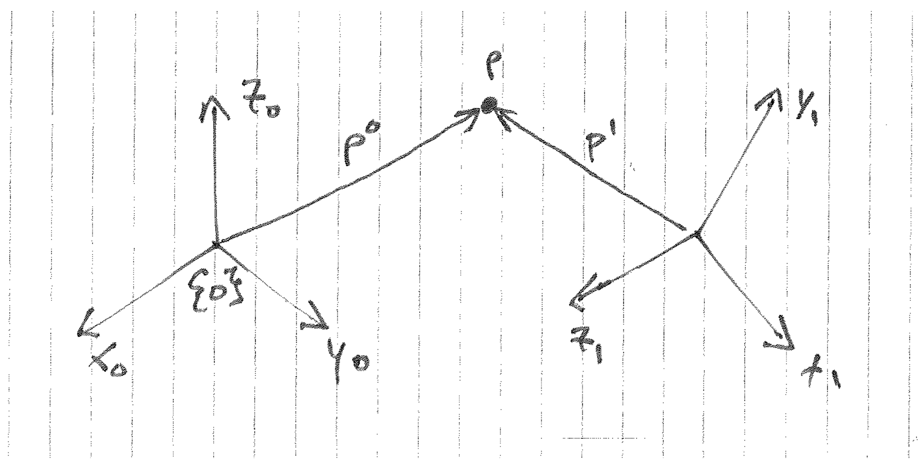
Now, for a given frame, we can represent a point p in that frame as a linear combination of the frame vectors:

$$p^0 = p^0_x x_0 + p^0_y y_0 + p^0_z z_0,$$

where p^0_x, p^0_y, and p^0_z are scalars. Another more common way to represent p^0 is to use a 3-vector containing these scalars:

$$p^0 = \begin{bmatrix} p^0_x \\ p^0_y \\ p^0_z \end{bmatrix}.$$

Here the superscript 0 means that p is begin represented in the $\{0\}$ frame. But p can be represented in other frames as well, and it will have a different represenation in different frames.



Here p^0 and p^1 are different vectors, but they represent the same point.

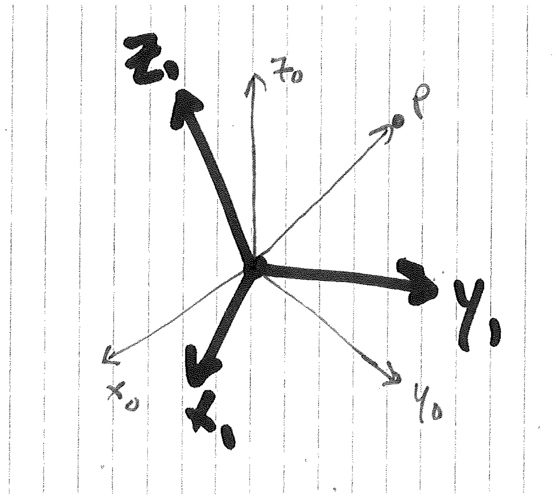
You may be used to picking *fixed* or *intertial* frames, however it is also possible to have frames that move. In fact, it is quite useful.

Here's a quick preview of where we are going with this. Robotic manipulators are composed of multiple rigid bodies. They often interact with multiple other rigid bodies in space. So it is important to keep track of the positions and oriantations of all of those bodies.

To handle this, we will rigidly attach a frame to every rigid body we care about. Now we will look for a mathematical (analytical) way to describe the displacements between the frames. Once we can describe displacements between frames, the same representation can be used to describe the positions and orientations of the rigid bodies they are attached to.

3.2 Rotations

Consider two frames whose origins coincide:



The point p has two representations: p^0 and p^1.

idea: we can use the coordinate transformation that maps p from frame $\{1\}$ to frame $\{0\}$ to represent the rotation between the two frames.

fact: consider the matrix whose columns are the vectors of frame $\{1\}$ represented in the $\{0\}$ frame:

$$R_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix},$$

then

$$p^0 = R_1^0 p^1$$

for any point p. I will leave it to you to convince yourself that this is true.

We call R_1^0 a rotation matrix, and we use it to represent the relative rotation between frame $\{0\}$ and frame $\{1\}$.

Some remarks:

1. R_1^0 is an orthogonal matrix. This is true by construction: all of the columns are mutually orthogonal to one another because the columns are vectors that make up a frame. This results in the useful fact that $(R_1^0)^T R_1^0 = I$.

2. $\det(R_1^0) = 1$. To see that this is true, you need to know the following fact about the determinate of any 3×3 matrix:

$$\det(A) = \det\left(\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}\right) = (a_1 \times a_2)^T a_3$$

Since the columns of R_1^0 are defined to be x_1^0, y_1^0, and z_1^0, we get

$$\det\left(R_1^0\right) = \left(x_1^0 \times y_1^0\right)^T z_1^0,$$

and since we know that frame $\{1\}$ is right handed, we get

$$\det\left(R_1^0\right) = \left(z_1^0\right)^T z_1^0 = 1.$$

The fact that the determinant is one means that multiplication by R_1^0 preserves handedness. In other words, if we multiply all of the vectors in a right handed frame by R_1^0, the resulting vectors will also be right handed. Intuitively, this means that R_1^0 cannot reflect (i.e., flip or mirror) a frame.

Any matrix that satisfies 1 and 2 is a rotation matrix, and any rotation matrix satisfies 1 and 2. This holds in every dimension as well. Which leads us to define the folowing set:

$$SO(n) = \left\{ A \in \mathbb{R}^n \mid A^T A = I, \quad \det(A) = 1 \right\}.$$

$SO(n)$ is called the *special orthogonal group*, and it is equivalent to the set of rotation matrices in $E(n)$. We care most about $SO(3)$ and $SO(2)$.

Note that for any $A, B \in SO(n)$, the matrix product $AB \in SO(n)$ too. To see this you just need to check the two conditions in the definition:

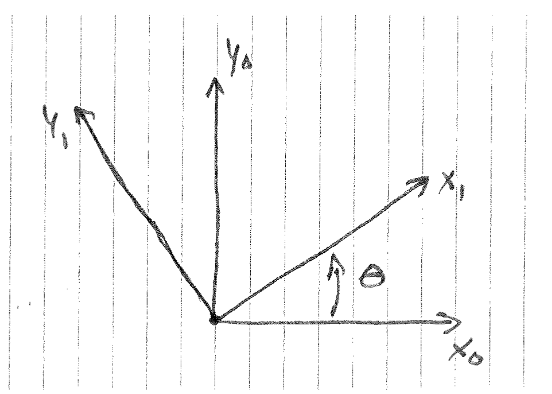
1. $(AB)^T(AB) = B^T A^T AB = B^T B = I$ (OK!)
2. $\det(AB) = \det(A)\det(B) = 1$ (OK!)

Three Intepretations: There are three distinct ways we can use and interpret rotation matrices, and we will use them all in this class. A rotation matrix can be thought of as:

1. a description of the relative orientation between two frames
2. a coordinate transform between two frames. Specifically, R^i_j is the coordinate transformation from frame $\{j\}$ to frame $\{i\}$, i.e. $p^i = R^i_j p^j$.
3. a motion of a point (or collection of points) within a single frame, i.e., the rotation matrix R can map the point p to a new point q in the same frame, $q = Rp$. Likewise, a rotation can be interpreted a motion from one frame to another (because frames are really just collections of points). In particular, R^i_j can be thought of as the motion that moves frame $\{i\}$ to frame $\{j\}$.

Note that the motion and coordinate transform interpretations are opposites of each other. The coordinate transformation maps from $\{j\}$ to $\{i\}$, while the motion moves from $\{i\}$ to $\{j\}$. This makes sense if you think about it it, but it can easily get confusing if you are not precise in the language you use.

In 2-space, it is easy to explicitly write out a rotation matrix. Consider the following two frames:



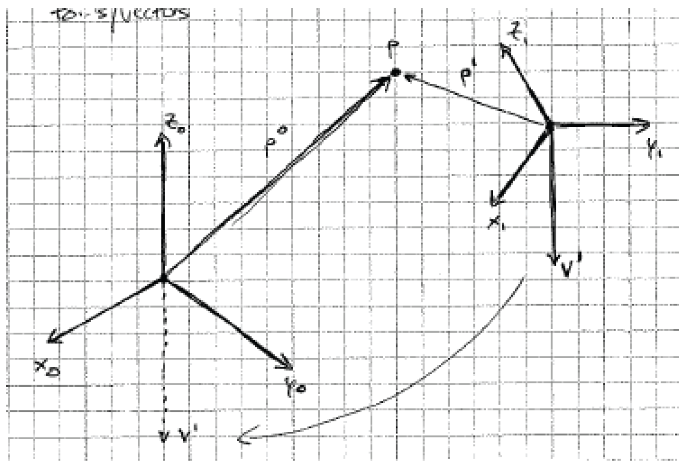
Here

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

4 Points and Vectors

In the previous section, we used rotation matrices to describe relative orientations of two frames with a coincident origin. Will rotation matrices still work for frames with different origins? **Yes**, but be have to be careful to distinguish between *points* and *vectors*. Recall that

1. **points** describe the location of a point (duh!). In some other books points are called *fixed vectors*.
2. **vectors** specify direction and magnitude only. They can be moved around freely (and they're sometimes called *free vectors*).



When we map a point p^1 into frame $\{0\}$, we really want to generate p^0 because p represents a position, the location of the tip is what is important. This is clearly more than just a rotation.

When we map the vector v^1 into frame zero, we only care about the direction and magnitude, i.e., the difference between the tip of v^1 and the origin o_1. Hence I can imagine moving frame one so that the origins are coincidental, then v^0 can be obtained by a rotation:

$$v^0 = R_1^0 v^1$$

So a rotation matrix is good enough to transform vectors, but we need something more to transform points!

5 Describing Position and Orientation

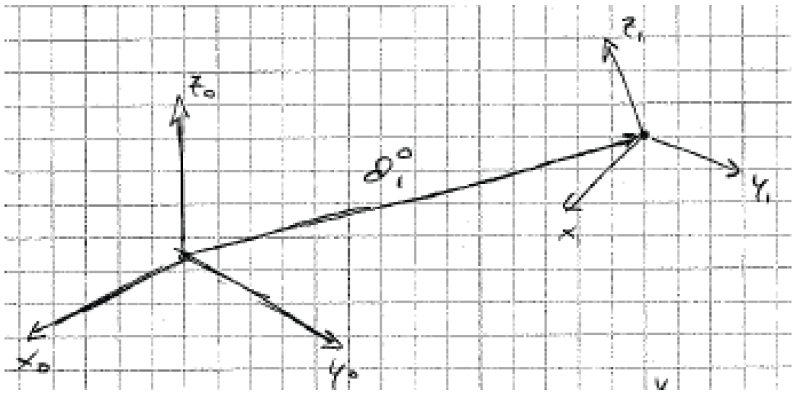
So how do we represent the position of frame $\{1\}$ relative to frame $\{0\}$? That is easy, we represent it with a 3-vector that describes the location of the origin of frame $\{1\}$ with respect to frame $\{0\}$. This vector can be denoted o_1^0.

In the book, they also sometimes use d_1^0 to be the same thing, and I like that notation a little better because an o looks like a zero.

Note the the origin is a point since it describes a position.

Now we can completely describe the relative pose between frame $\{0\}$ and frame $\{1\}$ with the pair

$$\{R_1^0, d_1^0\}$$



The coordinate transformation from frame $\{1\}$ to $\{0\}$ can be written as

$$p^0 = d_1^0 + R_1^0 p^1.$$

Now check this out: define H_1^0:

$$H_1^0 = \begin{bmatrix} R_1^0 & d_1^0 \\ 0\,0\,0 & 1 \end{bmatrix}$$

Then

$$\begin{bmatrix} p^0 \\ 1 \end{bmatrix} = H_1^0 \begin{bmatrix} p^1 \\ 1 \end{bmatrix}$$

Here's how you verify that:

$$\begin{bmatrix} R_1^0 & d_1^0 \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} p^1 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1^0 p^1 + d_1^0 \\ 0 + 1 \end{bmatrix}$$

The matrix H_1^0 is called a **homogeneous transformation matrix**.

We will use homogeneous transformation matrices to describe the position and orientation of frames and the rigid bodies they are attached to.

Just like rotation matrices, homogeneous transformation matrices have three interpretations:

1. relative pose (position and orientation) between frame 0 and frame 1
2. coordinate transform from frame 1 to frame 0
3. motion of a point or a rigid body within a frame.

6 More on Points and Vectors

When using homogeneous transforms, we can more easily distinguish between points and vectors.

A point will always be written with a 1 in the last place:

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

note that if we use the homogeneous transform

$$HP = \begin{bmatrix} R & d \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} Rp + d \\ 1 \end{bmatrix}$$

The result still has a 1 on the end so it is still a point.

A vector will always be written with a 0 in the last place:

$$V = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

note that if we use the homogeneous transform

$$HV = \begin{bmatrix} R & d \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} Rv \\ 0 \end{bmatrix}$$

The result still has a 0 on the end so it is still a vector.

7 Basic Displacements

Recall that we will use homogeneous transforms to represent relative displacements between frames:

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

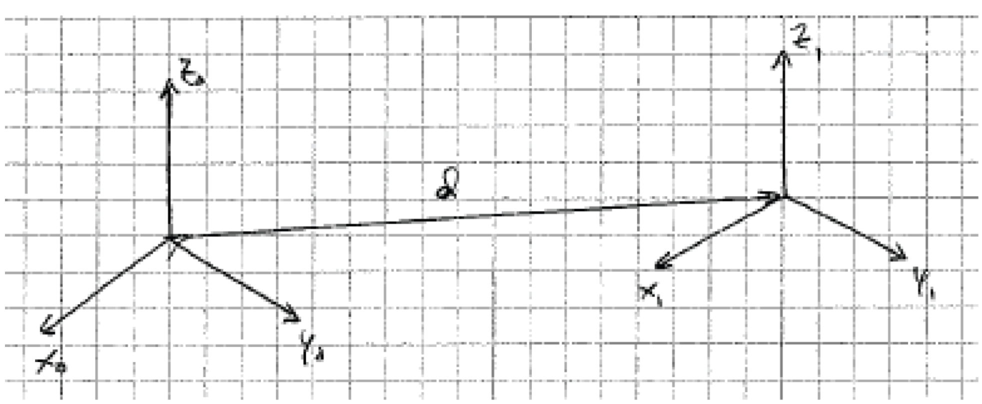
and H_i^j will transform points from frame i to frame j, i.e., $P^j = H_i^j P^i$, where

$$P^i = \begin{bmatrix} p^i \\ 1 \end{bmatrix}$$

The set of displacements is equal to the set of homogeneous matrices and is called the Special Euclidean Group, denoted $SE(n)$. Like $SO(n)$, it is also a matrix Lie group.

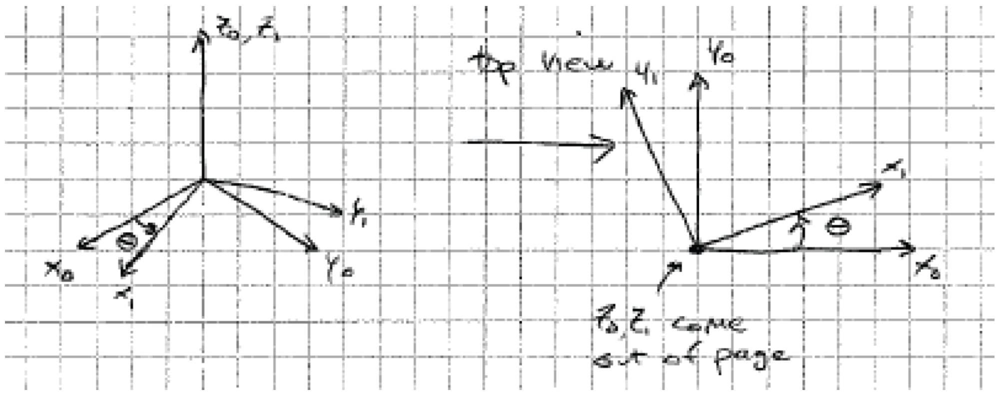
We'll start with some displacements that are so simple we can just write down what they are by inspection.

Pure Translation:



$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

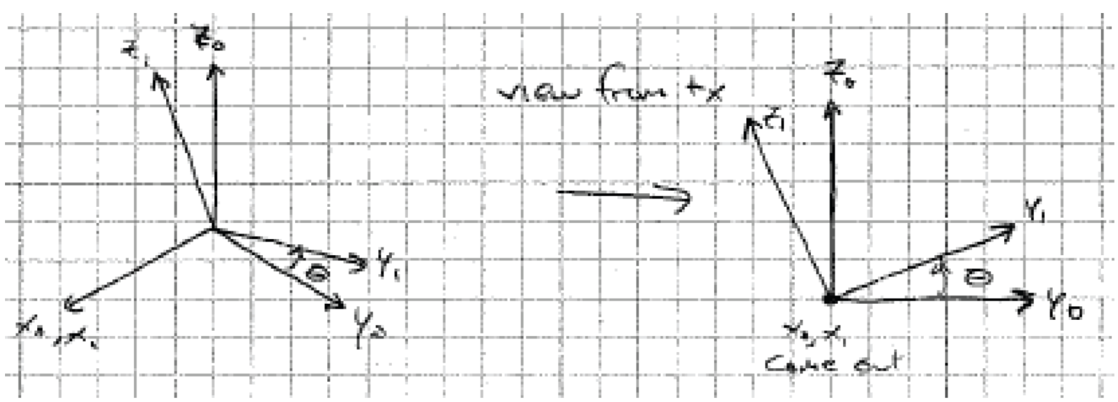
Rotation About z-axis: Positive rotation is defined by right hand rule.



Note that z coordinate does not change, so we only need to worry about the (x, y) plane.

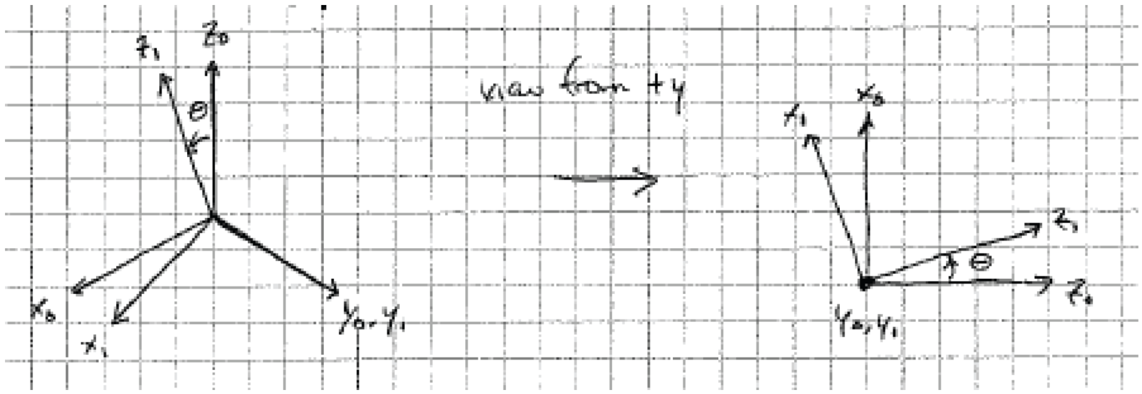
$$H_1^0 = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & R_{z,\theta} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation About x-axis:



$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & R_{x,\theta} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation About y-axis:



$$H_1^0 = \begin{bmatrix} c\theta & 0 & s\theta & 0 \\ 0 & 1 & 0 & 0 \\ -s\theta & 0 & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & R_{y,\theta} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8 Inverses of Simple Displacements

for any displacement:

$$H_i^j = (H_j^i)^{-1}$$

This is obvious from the coordinate transform interpretation:

$$P^0 = H_1^0 P^1 \quad \Longrightarrow \quad P^1 = \underbrace{(H_1^0)^{-1}}_{\triangleq H_1^0} P^0$$

pure translation:

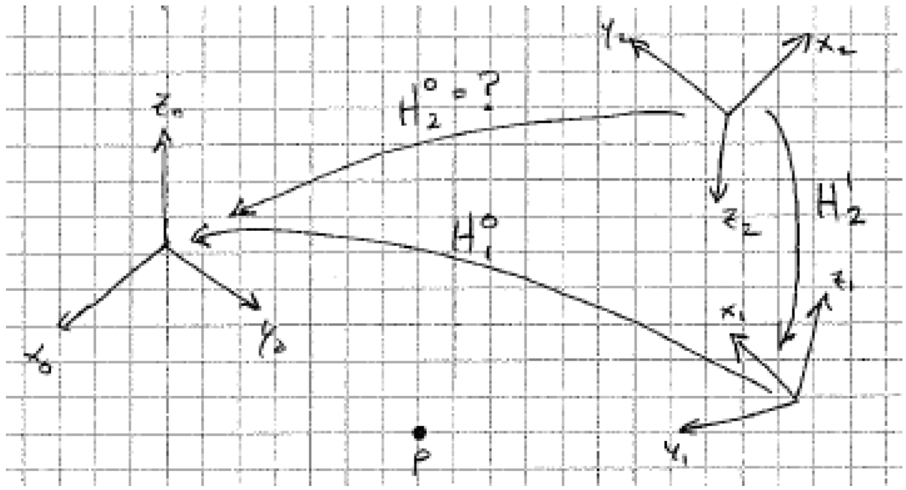
$$H_0^1 = \begin{bmatrix} I & d \\ 0\,0\,0 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad H_1^0 = (H_0^1)^{-1} = \begin{bmatrix} I & -d \\ 0\,0\,0 & 1 \end{bmatrix}$$

pure rotation:

$$H_0^1 = \begin{bmatrix} & 0 \\ R & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad H_1^0 = (H_0^1)^{-1} = \begin{bmatrix} & 0 \\ R^T & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{bmatrix}$$

9 Multiple Frames

Consider three frames:



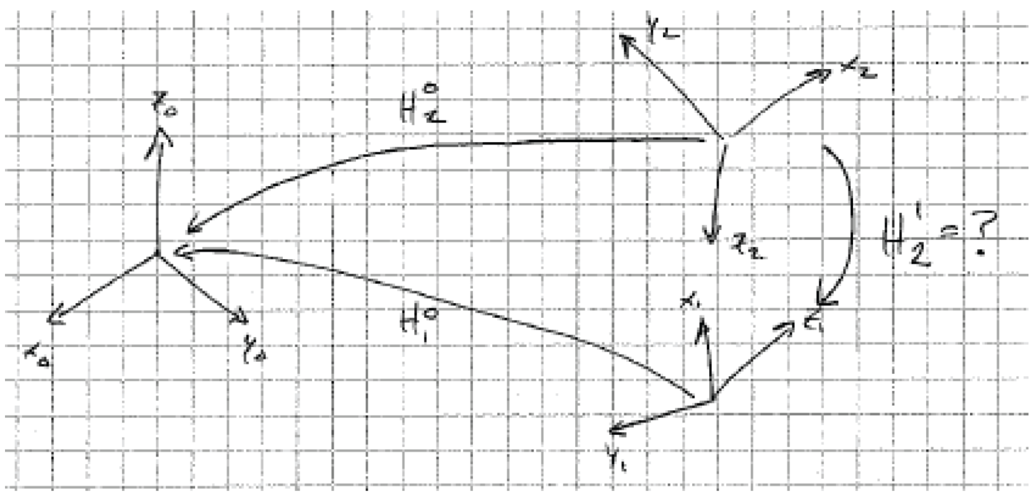
Assume H_1^0 and H_2^1 are known, what is H_2^0?

$$p^1 = H_2^1 p^2$$

$$p^0 = H_1^0 p^1 = H_1^0 H_2^1 p^2$$

$$\boxed{\Longrightarrow H_2^0 = H_1^0 H_2^1}$$

Three frames again:



$$H_2^1 = (H_1^0)^{-1} H_2^0 = H_0^1 H_2^0$$

This idea can be extended to any number of frames.

Note that this means that complicated displacements can be decomposed into a sequence of simpler displacements. We'll use this a lot.

10 "Moving" Frames

Here's a common way to describe the relative orientation between two frames:

- start with the two frames (0 and 1) co-located
- perform a sequence of simple displacements on frame 1 to get it to its "final" position

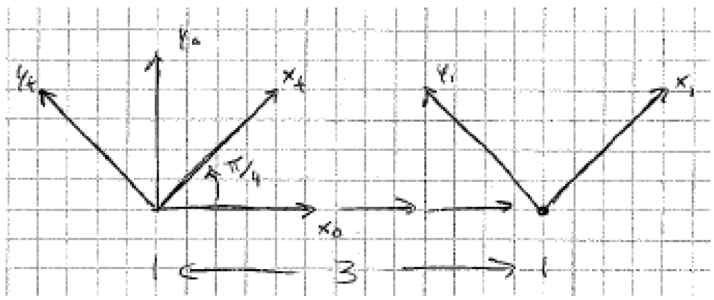
Note that the sequence of displacements can be specified two ways:

1. with respect to frame 0 (the fixed frame, sometimes called *absolute motion*)
2. with respect to the current position of frame 1 (the current frame, sometimes called *relative motion*)

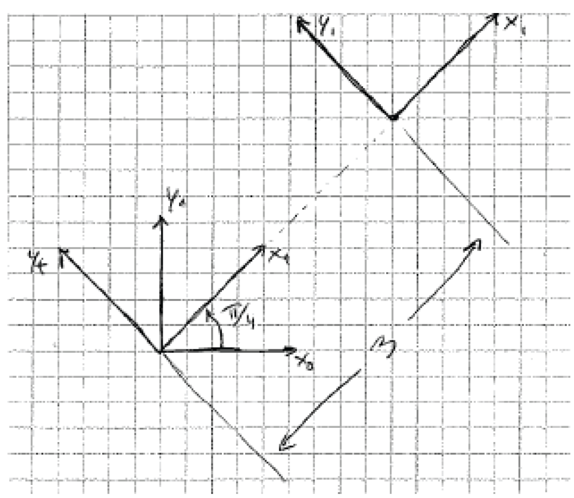
2D example: rotate by $\pi/4$, then translate 3 units along x-axis. Or "do G_1 then do G_2", where:

$$G_1 = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

wrt fixed frame (frame 0, absolute)



wrt current frame (frame 1, relative)



So how do we mathematically get the value of H_1^0 that represents what happens after you do the whole sequence? Obviously, it is different whether motion was specified relative to fixed frame or current frame.

Obviously, there is no confusion about what the first transform means since both frames start off aligned. So here's another way to ask the same questions: Suppose G_1 describes some initial relative displacement between frame 1 and frame 0. Now suppose a second transformation G_2 is applied to frame 1 to move it to it's "final" position. What is the resulting value of H_1^0?

In the example, we've labeled the intermediat frame t.

if G_2 is applied wrt current frame (frame t): G_2 describes the relative displacement between frame t and frame 1, i.e., G_2 is the coordinate transform from frame 1 to frame t. This is just like the first "multiple frames" case above:

$$H_1^0 = H_t^0 H_1^t = G_1 G_2$$

In other words: **multiply on the right!**

if G_2 is applied wrt to fixed frame: It's easiest to think about the G's as rigid body motions in this case (recall the "three interpretations").

G_1 moves the point p_0 to p_t. (both points are written in the 0 frame)

$$p_t = G_1 p_0$$

Now G_2 moves p_t to p_1:

$$p_1 = G_2 p_t = G_2 G_1 p_0$$

So H_1^0 is the motion that moves p_0 to p_2, i.e.,

$$H_1^0 = G_2 G_1.$$

in other words: **mulitply on the left!**

11 Other Representations of Orientation

There are other ways to represent orientation in E^3. They can be easily understood in terms of rotation matrices and compositions:

11.1 Axis-Angle Representation

Any rotation in 3-space can be represented by a unit vector $n = [n_1, n_2, n_3]^T$, and a rotation about that axis θ. Here, as always, the positive rotation direction is determined by the right hand rule. This result is due to Euler, whose name we will see a lot in this class. This gives us a 4 dimensional representation of the space of rotations, though it has the drawback every rotation has an infinity of possible representations.

11.2 Quaternions

Quaternions are a commonly used four dimensional representation of rotations (as opposed to the 9-dimensional rotation matrices that we use). We give a brief overview here, there are plenty of good references out there if you want to learn more.

A quaternion is a kind of extended complex number, with imaginary variables i, j, and k. Mathematically, a quaternion is expressed as

$$Q = q_0 + iq_1 + jq_2 + kq_3,$$

where the rules for the imaginary variables are:

$$i^2 = j^2 = k^2 = -1, \qquad i = jk = -kj, \qquad j = ki = -ik, \qquad k = ij = -ji.$$

This is often represented as a 4-vector:

$$Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

A unit quaternion (i.e. $\|Q\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$) can be used to represent a rotation. Specifically, for a rotation of θ about an axis defined by the unit vector $n = [n_1, n_2, n_3]^T$, the corresponding quaternion is

$$Q = \begin{bmatrix} \cos\frac{\theta}{2} \\ n_1 \sin\frac{\theta}{2} \\ n_2 \sin\frac{\theta}{2} \\ n_3 \sin\frac{\theta}{2} \end{bmatrix}$$

The “inverse” of Q is just its conjugate (i.e., negate the signs of all of the imaginary variables)

$$Q^{-1} = Q^* = \begin{bmatrix} \cos\frac{\theta}{2} \\ -n_1 \sin\frac{\theta}{2} \\ -n_2 \sin\frac{\theta}{2} \\ -n_3 \sin\frac{\theta}{2} \end{bmatrix}$$

If you multiply Q and Q^* using the rules above, you get

$$QQ^* = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

which if you think about it is the identity rotation.

To rotate a point P, we represent is as a quaternion with zero real part,

$$P = ip_1 + jp_2 + kp_3,$$

and then the rotated version of P can be expressed as the quaternion product

$$P^{\text{rotated}} = QPQ^*.$$

11.3 Euler Angles

Following in the vein of representing complex displacement by composing simple ones, we use three simple rotations to represent any orientation in E^3. Start by imagining frames 0 and 1 to be aligned. Then do the following:

1. rotate frame 1 about z_1 by ϕ ($R_{z,\phi}$)
2. rotate frame 1 about y_1 by θ ($R_{y,\theta}$)
3. rotate frame 1 about z_1 by ψ ($R_{z,\psi}$)

The three angles (ϕ, θ, ψ) are called **ZYZ Euler angles**.

Question 1: Given (ϕ, θ, ψ), what is the equivalent rotation matrix?

Well, each of the steps listed above is wrt the *current* frame, so the resulting rotation matrix is achieved by successively multiplying on the right:

$$R = R_{z,\phi} R_{y,\theta} R_{z,\psi}.$$

Question 2: Given a rotation matrix R, what are the equivalent ZYZ Euler angles?

Short answer: you have to solve a big system of nonlinear equations. There is always at least one solution. In some cases there are infinite solutions (see pages 55-56 in SHV).

11.4 Roll, Pitch, Yaw

Roll, pitch, yaw are similar to Euler angles, but they are rotations specified about the *fixed* coordinate frame. Start by imagining frames 0 and 1 to be aligned. Then do the following:

1. rotate frame 1 about x_0 by ψ $(R_{x,\psi})$
2. rotate frame 1 about y_0 by θ $(R_{y,\theta})$
3. rotate frame 1 about z_0 by ϕ $(R_{z,\phi})$

Question 1: Given roll pitch yaw (ψ, θ, ϕ), what is the equivalent rotation matrix?

Well, each of the steps listed above is wrt the *fixed* frame, so the resulting rotation matrix is achieved by successively multiplying on the left:

$$R = R_{z,\phi} R_{y,\theta} R_{x,\psi}.$$

Question 2: Given a rotation matrix R, what are the equivalent roll, pitch, and yaw?

answer: similar to the Euler angles answer.