So now we're pretty good at 2D kinematics, and we can expand this to 3D kinematics.

Transformation Matrices

The derivation of these follows the derivation of the 2D transformation matrices closely; you can grind your way through the math on your own if you'd like. Since we're in 3D, and since we'll still need heterogeneous matrices to perform translation, our points will now be represented with a 4×1 vector $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$, and our transformation matrices will have to be 4×4.

Translation

$$Trans(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around the X axis

$$RotX(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around the Y axis

$$RotY(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around the Z axis

$$RotZ(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Coordinate Frames

We are going to use right-handed frames in this course. This means that if you take your right hand, put your palm at the origin, and point your fingers along the X-axis, then curl your fingers in the direction of the Y-axis, then your thumb will be pointed in the direction of the Z-axis. See Figure 1 for an illustration of this.

When we talk about a rotation around an axis, that axis will remain stationary, and the other two will rotate around it. A rotation in the positive direction means that if we look down the axis at the origin, the other axes spin in a counter-clockwise direction. An example of rotating around the Y-axis in a positive direction is shown in Figure 2.

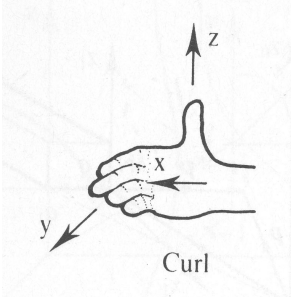

Figure 1: Right-handed axes (from McKerrow, Introduction to Robotics)

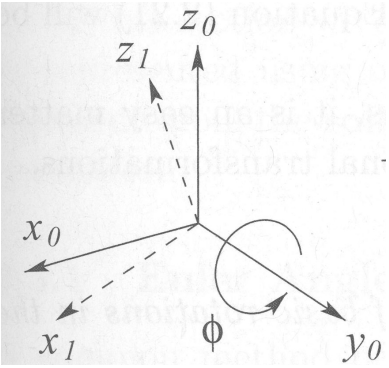

Figure 2: Rotation around the Y-axis (from Spong et al., Robot Modeling and Control)

Assigning Frames to Joints

There are an infinite number of ways to assign coordinate frames to robotic manipulators. However, it makes it easier for everybody to talk to each other and share results if there is a standard way of doing these assignments. This standard way is known as the Denavit-Hartenberg Convention.

In the DH Convention, each transformation from one joint to another is a product of the following basic transformations:

$$^{i-1}T_i = RotZ(\theta_i)Trans(l_i, 0, d_i)RotX(\alpha_i),$$

where $\theta_i, l_i, d_i, \alpha_i$ are the **joint angle**, **link length**, **link offset**, and **link twist**, respectively. These will be discussed further.

But, an astute student might say, in moving a frame we have six degrees of freedom (three directions of translation, three axes to rotate around), and the DH convention only gives us four parameters. How can we represent all possible joints this way? For two coordinate frames N_0 and N_1, we declare the following two things must be true:

- x_1 is perpendicular to z_0
- x_1 intersects z_0

It turns out that when we limit ourselves in this way, we no longer have to transform between arbitrary frames, but between a subset where four degrees of freedom is enough.

We're going to be dealing primarily with *revolute joints*, or joints with a rotation around a single axis, like an elbow, or the rotation on a wrist. There are other kinds, of course, like a *prismatic* joint, which lengthens and shortens like the arm of a crane, or a *ball and socket* joint like a shoulder.

On revolute joints, we will add the additional constraint on our frames that the joint will rotate around the Z-axis.

Revolute joints which meet the above criteria can be broken down into the following three cases.

1. z_{i-1} **and** z_i **are parallel**

   An example of this is a link connecting two elbow joints which turn in the same plane. We can recycle our 2D arm for this. Figure 3 illustrates the two frames. In both frames, the Z-axis is coming out at the reader, and so is not illustrated. θ_1 and l_1 are both labelled. We have no twist, meaning $\alpha_1 = 0$. Additionally, we require no translation along the Z-axis, so $d_1 = 0$.

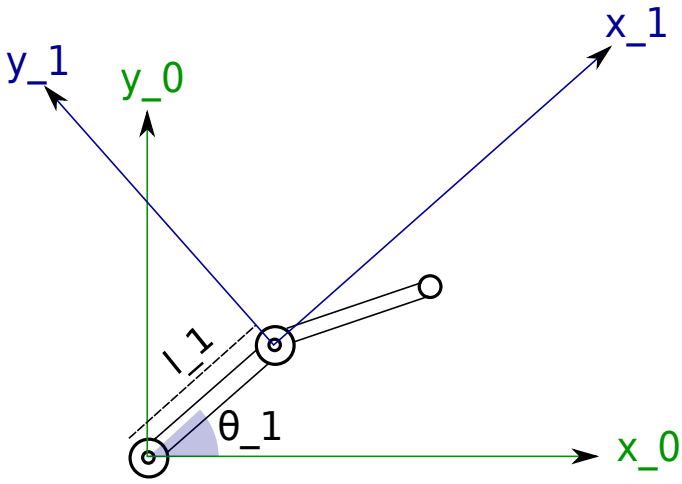

Figure 3: Illustration of z_0 and z_1 being parallel. Both Z-axes are pointing at the reader, and are not included.

2. z_{i-1} **and** z_i **are not co-planar**

   An example of this is a link connecting two elbow joints which turn in different planes. In Figure 4, you can see an example of this. z_0 is coming directly at the viewer, and y_1 is pointing exactly away from the reader.

   Referring to the DH convention, we first rotate around z_0 by θ. We then translate along the new X-axis l_1. We do no translation along the Z-axis, so $d_1 = 0$. We then rotate around x_1 so that z_1 is pointed in the correct direction. In this case, we are looking down the X-axis at the origin, and we rotate it $\pi/2$ radians clockwise, making $\alpha_i = -\pi/2$. This is where α gets its name of a link twist.

3. z_{i-1} **and** z_i **intersect**

   This is likely the trickiest case. Our arm is a good example; we have an elbow, and then we have a wrist at the end of a link. The wrist's rotation is around the center of the forearm, meaning the Z-axis runs along the forearm as well, perpendicular to the elbow's Z-axis.

   It's natural to want to put the origin for our new frame in the middle of our wrist joint. However, then there's no way for x_1 to intersect z_0, and that's one of our rules for the DH convention. So instead, we do a weird trick, and put the origin of our new frame in the same place as the old frame.

   Now, where to point x_1? Well, it needs to be perpendicular to both z_1 and z_0. This leaves you with two choices. One of them is illustrated in figure 5.

   So, how does this correspond to our parameters for our transformation? Well, first we rotate around z_0 $\phi + \pi/2$ radians (look at our picture, and see why. If we had chosen for x_1 to go down and to the right, it would be $\phi - \pi/2$ radians.), so that's our θ_1. We then do no translation at all, so $l_1 = d_1 = 0$. We then rotate around x_1 $\pi/2$ radians to get z_1 pointed in the correct direction. Weird!

   We can make up the link's length by adding it to l_2 at the next transformation.

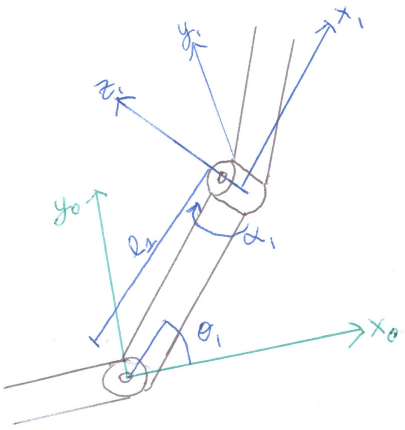

Figure 4: Illustration of z_0 and z_1 being not co-planar.

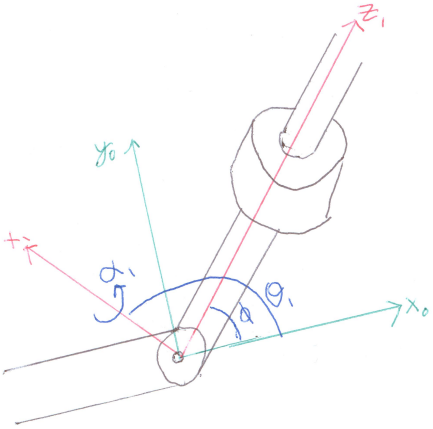

Figure 5: Illustration of z_0 and z_1 intersecting. z_0 and y_1 are coming out at the viewer.

> Another example of this type is Figure 5, but reversed, where joint 0 is a wrist, and joint 1 is an elbow. How do you think that would work? Draw an elbow-wrist-elbow, and figure out how the frames should be arranged.

So if those are our three examples, why is d a parameter? What's its purpose? Well, d comes in handy when our links are not straight, easy links; the DH convention works just as well with curvy, twisted links.

Performing 3D Kinematics

Finding a point on the end of the robot is an easy transition from 2D kinematics. For each joint i, define θ_i, l_i, d_i, and α_i, create the transformation matrices, and multiply them all up, in the correct order, and then multiply by the point $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$. That's it!

But, of course, we're not interested only in the position of the end of the robot, but also the orientation, referred to in 3D as its roll, pitch, and yaw. What we're interested in is the total rotation around each of the three axes in the original reference frame. The first thing to note is that when we assign DH values to every joint, when all joint angles are 0, the end frame must have no rotational component when compared to the reference frame. In other words, **in its default position, the arm must have no roll, pitch, or yaw**. Sometimes, this may mean an additional set of DH parameters.

In this class, we'll refer to roll as the rotation ψ around the X axis, pitch as the rotation ϕ around the Y axis, and yaw as the rotation θ around the Z axis. These definitions, unfortunately, aren't well standardized. What is standardized is that these occur in the reference frame, in the order yaw, pitch, roll. Notice this is different than how I understand it is used in reference to aircraft; in robotics, these terms refer to rotations around the axes of the reference frame.

Let's think about the general transformation matrix in 3D. Let's shorten things like $\cos(\theta)$ to c_θ.

$$\begin{aligned} T =& Trans(p_x, p_y, p_z) RotX(\psi) RotY(\phi) RotZ(\theta) \\ =& \begin{bmatrix} c_\theta c_\phi & -s_\theta c_\psi + c_\theta s_\phi s_\psi & s_\theta s_\psi + c_\theta s_\phi c_\psi & p_x \\ s_\theta c_\phi & c_\theta c_\psi + s_\theta s_\phi s_\psi & -c_\theta s_\psi + s_\theta s_\phi c_\psi & p_y \\ -s_\phi & c_\phi s_\psi & c_\phi c_\psi & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

So, when we have calculated our general transformation matrix, from the reference frame to the frame at the end of the arm, we can solve for ϕ, ψ, and θ by just teasing these terms out of this matrix. However, just as in 2D, we don't trust arcsin or arccos, and only trust arctan2.

This means that before we solve for any of our angles, such as θ, we will instead solve for $\cos(\theta)$ and $\sin(\theta)$, so that we can call $arctan2(\sin(\theta), \cos(\theta))$ and avoid all numerical issues.

Consider the general transformation matrix above. Given such a matrix, we could take the 3rd row, 1st column, (hereby denoted $T(2,0)$) and call *arcsin* on it, in order to calculate ϕ. However, we now know this isn't what we want to do. Instead, we're going to negate this, and use that term as the first argument in our *arctan*2 function. This leaves us needing to calculate $\cos(\phi)$. Using the handy trigonometric identity that $\cos^2(x) + \sin^2(x) = 1$, we note that $\cos^2(\phi) = T(0,0)^2 + T(1,0)^2$. So, $\phi = arctan2(-T(2,0), \sqrt{T(0,0)^2 + T(1,0)^2})$.

Note we could have used plus-or-minus for the 2nd argument of the above function; this will rear its head in inverse kinematics.

Assuming $\phi \neq -\pi/2$ or $\pi/2$, the three angles are pulled out easily:

$$\begin{aligned} \phi =& arctan2\left(-T(2,0), \sqrt{T(0,0)^2 + T(1,0)^2}\right) \\ \theta =& arctan2\left(\frac{T(1,0)}{\cos(\phi)}, \frac{T(0,0)}{\cos(\phi)}\right) \\ \psi =& arctan2\left(\frac{T(2,1)}{\cos(\phi)}, \frac{T(2,2)}{\cos(\phi)}\right) \end{aligned}$$

If ϕ is $\pi/2$ or $-\pi/2$, then we can calculate the sum or difference of ψ and θ, but cannot narrow it down further than that. So, we use a convention in which $\theta = 0$.

$$\begin{aligned} \phi =& \frac{\pi}{2} \\ \theta =& 0 \\ \psi =& arctan2(T(0,1), T(1,1)), \end{aligned}$$

and,

$$\begin{aligned} \phi =& -\frac{\pi}{2} \\ \theta =& 0 \\ \psi =& -arctan2(T(0,1), T(1,1)), \end{aligned}$$