Inverse Kinematics

The kinematics we have learned so far is forward kinematics, in which given the settings of the joints, we infer the location and orientation of the end of the arm. Now, we will learn the opposite problem, inverse kinematics, in which we want a position and orientation, and must calculate the joint angles.

Let’s consider this problem by looking at an example. Consider our 2D arm of Figure 1, which we will now interpret as a 3D arm, consisting of these two elbow joints. Hopefully, it is clear that this consists of a rotation around the Z axis by the red angle \( \theta_1 \), a translation along the X axis of \( l_1 \), a rotation around the Z axis by the blue angle \( \theta_2 \), and a translation along the X axis of \( l_2 \). Our transformation matrices look as follows (we use \( c_1 = \cos(\theta_1) \)):

\[
\begin{bmatrix}
  c_1 & -s_1 & 0 & l_1 c_1 \\
  s_1 & c_1 & 0 & l_1 s_1 \\
  0 & 0 & 1 & 0
\end{bmatrix} \cdot \begin{bmatrix}
  c_2 & -s_2 & 0 & l_2 c_2 \\
  s_2 & c_2 & 0 & l_2 s_2 \\
  0 & 0 & 1 & 0
\end{bmatrix} = \begin{bmatrix}
  c_1 c_2 - s_1 s_2 & -c_1 s_2 - s_1 c_2 & 0 & l_2 c_1 c_2 - l_2 s_1 s_2 + l_1 c_1 \\
  s_1 c_2 + c_1 s_2 & -s_1 s_2 + c_1 c_2 & 0 & l_2 s_1 c_2 + l_2 c_1 s_2 + l_1 s_1 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\]

Now, remember, this is an instance of our general transformation matrix \( T \). If we make our desired end of arm location \((p_x, p_y, p_z)\), and our roll, pitch, and yaw \( \phi, \psi, \) and \( \theta \), respectively, then

\[
\begin{bmatrix}
  c_1 c_2 - s_1 s_2 & -c_1 s_2 - s_1 c_2 & 0 & l_2 c_1 c_2 - l_2 s_1 s_2 + l_1 c_1 \\
  s_1 c_2 + c_1 s_2 & -s_1 s_2 + c_1 c_2 & 0 & l_2 s_1 c_2 + l_2 c_1 s_2 + l_1 s_1 \\
  0 & 0 & 1 & 0
\end{bmatrix} = \begin{bmatrix}
  c_\phi c_\psi - s_\phi s_\psi & c_\phi s_\psi + s_\phi c_\psi & p_x \\
  s_\phi c_\psi + c_\phi s_\psi & s_\phi s_\psi - s_\phi c_\psi & p_y \\
  -s_\phi c_\psi & c_\phi s_\psi & p_z
\end{bmatrix}
\]

Given our desired position and roll/pitch/yaw, we can write \( T \) exactly, giving us a system of equations to solve. Unfortunately, these are nonlinear, transcendental equations, making them difficult to solve. But not impossible! Let’s look back at our example.

We’re going to need our trig identities. For example, \( \sin(\alpha) \cos(\beta) + \cos(\alpha) \sin(\beta) = \sin(\alpha + \beta) \), and \( \cos(\alpha) \cos(\beta) - \sin(\alpha) \sin(\beta) = \cos(\alpha + \beta) \). If we write \( \cos(\theta_1 + \theta_2) \) as \( c_{12} \) (and similarly with \( \sin \)), we can write our above matrix as:

\[
\begin{bmatrix}
  c_{12} & -s_{12} & 0 & l_2 c_{12} + l_1 c_1 \\
  s_{12} & c_{12} & 0 & l_2 s_{12} + l_1 s_1 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\]

Using the entry in the upper right, which must equal \( p_x \) in \( T \), and using the fact that \( T(0,0) = c_{12} \), we can now see that \( c_1 = \frac{p_x - l_2 T(0,0)}{l_1} \).
Great! Now, remember that we don’t want to do $\arccos(c_1)$, because it’s numerically unstable. Instead, we’ll solve for $s_1 = \frac{p_y - l_2 T(1,0)}{l_1}$, and do

$$\theta_1 = \arctan2\left(\frac{p_y - l_2 T(1,0)}{l_1}, \frac{p_x - l_2 T(0,0)}{l_1}\right).$$

Hooray! One down, one to go. Fortunately, there are now a number of ways to set up a system of equations to solve for $s_2$ and $c_2$, to allow us to do $\theta_2 = \arctan2(s_2, c_2)$.

**Closed-Form Solutions vs. Numerical Solutions**

The above solutions, in which a single equation is written to solve for $\theta_1$ and $\theta_2$, were a bit of work to find. Now that we have them, however, we can just hardcode in these equations for any $(x,y,z,\text{roll},\text{pitch},\text{yaw})$ tuple in order to perform inverse kinematics. This is known as the *closed-form solution*.

Some alert student, well versed in numerical analysis, might not see the purpose behind solving for these equations. In their mind, they see a series of equations, and they are aware techniques exist to calculate $\theta_1$ and $\theta_2$ to arbitrary precision. This is true! We could do away with all the trig identities, and variable manipulation, and just let a computer find these values. This is the *numerical solution*.

It turns out that numerical solutions, while pleasingly easy on the human, are a bit of a dud in real robotics. The techniques for numerically calculating our parameters are slow, iterative processes, compared to a very fast calculation of the closed-form solution. As we want robots to perform quickly, getting as much work done as possible, we usually discount the idea of numerical solutions, sentencing us to finding the closed-form solution. Fortunately, this only needs to be done once for a given robot.

**Multiple Solutions**

What if, for some $\theta$, we are able to find $\cos(\theta)$, but not $\sin(\theta)$ (or the other way around)? Should we just give up on $\arctan$, and use $\arccos$? Well, remember that $\cos^2(\theta) + \sin^2(\theta) = 1$. So, we can calculate $\theta = \arctan2(\pm\sqrt{1 - \cos^2(\theta)}, \cos(\theta))$. So... which one do we use? Plus or minus? Well, this situation commonly arises when there are multiple solutions to the problem. For example, consider the arm in Figure 2. In this, there are two possible solutions for the given manipulator position and orientation.

![Figure 2: Multiple solutions to arrange a 3-link planar arm. From Craig, Introduction to Robotics](image)

So, is one better? Mathematically, no, both work. Physically, maybe. Notice how one of the joints on the arm in the lower solution of our example is below the table. This may or may not be a problem; it depends upon the physical space, so mathematically, we have no real way to tell. How many solutions are there? It depends upon the robot arm, and the number of joints that turn in the same plane. Manufactured robotic arms are often constructed to limit this as much as possible.

Furthermore, some solutions won’t be possible for other physical reasons because the motors won’t turn that far. If we know that a joint won’t turn past $\pi/2$, then any solutions that include that can be automatically discarded.
Multiple solutions is another reason why numerical approaches are inappropriate. They will only find one of the possible solutions, and it might be unusable for physical reasons.

So how is this handled in the real world? Well, for a given robot arm, even if it’s a pretty complicated one, there are likely only 8-16ish possible solutions for an arbitrary \((x,y,z,\text{roll},\text{pitch},\text{yaw})\), which isn’t so bad. It’s not unreasonable for manufacturers to include the closed-form inverse kinematics for all of these solutions.

**The Moral of the Story**

Inverse kinematics is a simple enough idea, but unavoidably complicated in practice. There’s no magic algorithm that can be followed for a general arm, you just have to sit down with your table of trigonometric properties, and have a go at it.