Robot Perception and Control

Kinematics and Control

Last updated: Jul / 25 /2024

Kashu Yamazaki
kyamazak@andrew.cmu.edu

Kinematics

Kinematics

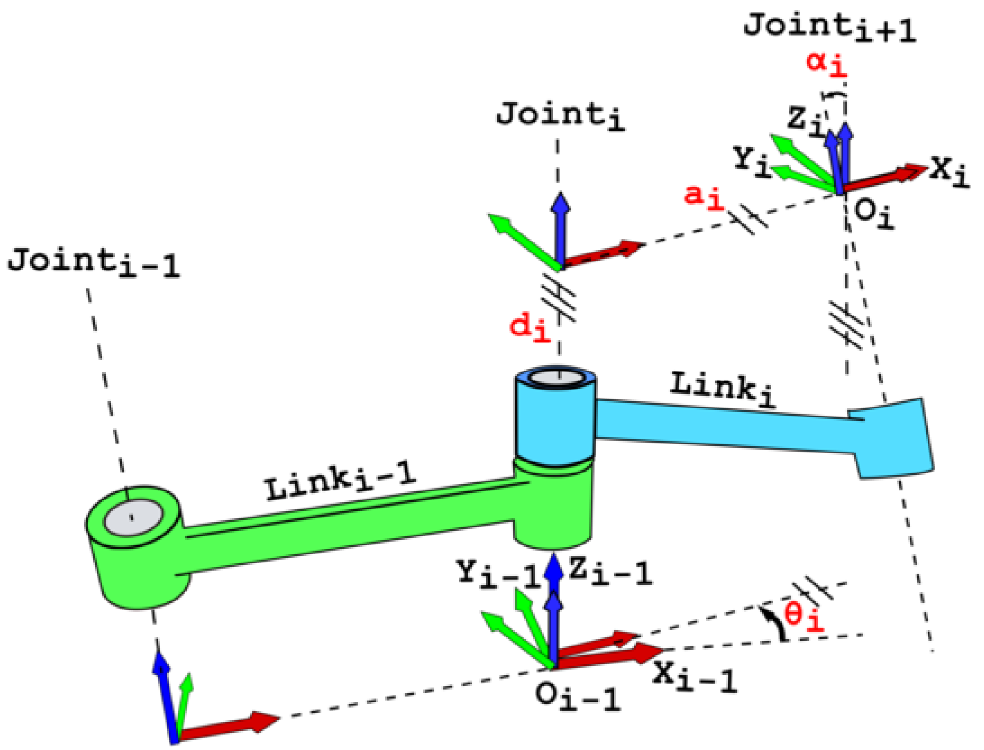
Kinematics: study of a motion of the robot without considering the forces and torques producing the motion.

Denavit–Hartenberg convention

The following four transformation parameters are known as D–H parameters:

- a_i: distance from z_{i-1} to z_i along x_i.
- α_i: angle from z_{i-1} to z_i about x_i.
- d_i: distance from x_{i-1} to x_i along z_i.
- θ_i: angle from x_{i-1} to x_i about z_i.

Usually, a_i and α_i are **constants** that describe the geometry of the robot, while d_i and θ_i are the **variables** that describe the motion of prismatic and revolute joints, respectively.



Denavit–Hartenberg convention

In this convention, coordinate frames are attached to the joints between two links such that one transformation is associated with the joint $[Z]$, and the second is associated with the link $[X]$. The coordinate transformations $[T]$ along a serial robot consisting of n links form the kinematics equations of the robot as:

$$[T] = [Z_1][X_1][Z_2][X_2]\ldots[Z_n][X_n]$$

where each transformation $[Z][X]$ can be implemented as a 4×4 matrix using the DH parameters as:

```
def transform(a, alpha, d, theta):
    return Rot(theta, axis="z") @ Trans(d, axis="z") @ Trans(a, axis="x") @ Rot(alpha, axis="x")
```

Forward Kinematics

Forward kinematics is the problem of finding the end-effector position and orientation $x(t)$ of a robot manipulator given the joint angles $\theta(t)$ and link lengths.

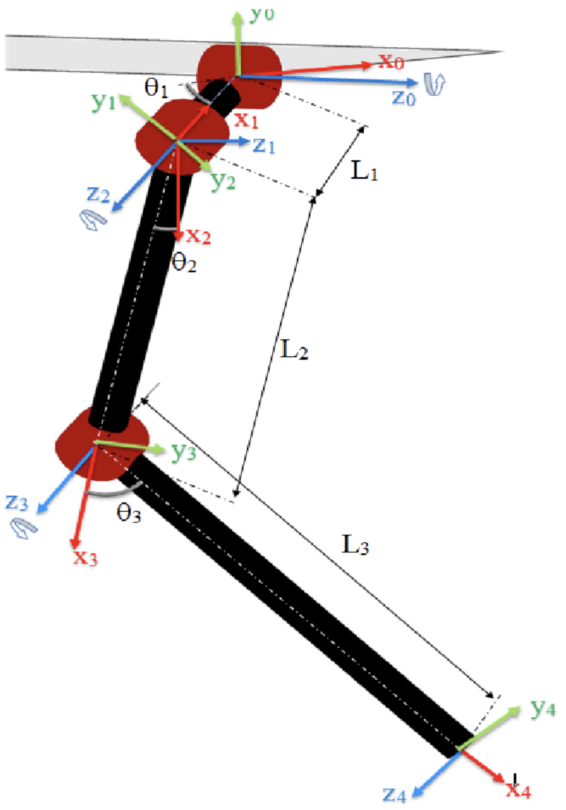
$$x(t) = f(\theta(t))$$

We can use the DH parameters of a robot to simply represent the forward kinematics as a chain of transformations starting from a **base link**, which connects to the origin.

```
def fk(theta):
    trans = np.identity(4)
    for (_a, _alpha, _d, _theta) in dh_params(theta):
        trans = trans @ transform(_a, _alpha, _d, _theta)
    return trans
```

Forward Kinematics

Let's consider the *right front leg* joints (coordinate systems below):



The DH parameters of the leg:

Link	a	α	d	θ
0-1	L_1	0	0	θ_1
1-2	0	$-\pi/2$	0	$-\pi/2$
2-3	L_2	0	0	θ_2
3-4	L_3	0	0	θ_3

From the table, we can construct the each transformation T and the forward kinematics is given by the chain of those transformations as T_0^4.

$$T_0^1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & -L_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & -L_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & L_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & L_3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0^4 = T_0^1 T_1^2 T_2^3 T_3^4$$

Inverse Kinematics

Inverse kinematics (IK) is essentially the reverse operation of FK: computing configuration(s) to reach a desired workspace coordinate. Unlike forward kinematics, inverse kinematics cannot be solved in a closed-form expression (in general). If we can derive a closed-form expression through symbolic manipulations, we can use Analytical IK, otherwise we need to use numerical approach.

Analytical IK

- Once the equations are derived, solutions are very fast to compute.
- Often difficult or tedious to derive.
- Only applicable to non-redundant robots (# DOFs = # of task space dimensions).

Numerical IK

- need to define solution parameters or initial guesses
- More generalizable

Analytical Inverse Kinematics

Let's consider a 2D arm in 2D space as in the Figure.

From law of cosines:

$$\cos(\pi - \theta_2) = -\frac{x_2^2 + y_2^2 - L_1^2 - L_2^2}{2L_1L_2}$$

Assuming solution exists ($-1 \leq RHS \leq 1$),

$$\theta_2 = \pm\cos^{-1}\left(\frac{x_2^2 + y_2^2 - L_1^2 - L_2^2}{2L_1L_2}\right)$$

- Note that *elbow down* and *elbow up* solutions ($\pm$) exist.

Then using the θ_2, we can find a θ_1 as:

$$\theta_1 = \operatorname{atan2}(y_2, x_2) - \operatorname{atan2}(L_2 \sin\theta_2, L_1 + L_2\cos\theta_2)$$

Analytical Inverse Kinematics

Given the foot position (x_4, y_4, z_4), the joint positions $\theta_1, \theta_2, \theta_3$ for a typical quadruped leg is given as in paper:

$$\theta_1 = -\operatorname{atan2}(-y_4, x_4) - \operatorname{atan2}\left(\sqrt{x_4^2 + y_4^2 - L_1^2}, -L_1\right)$$

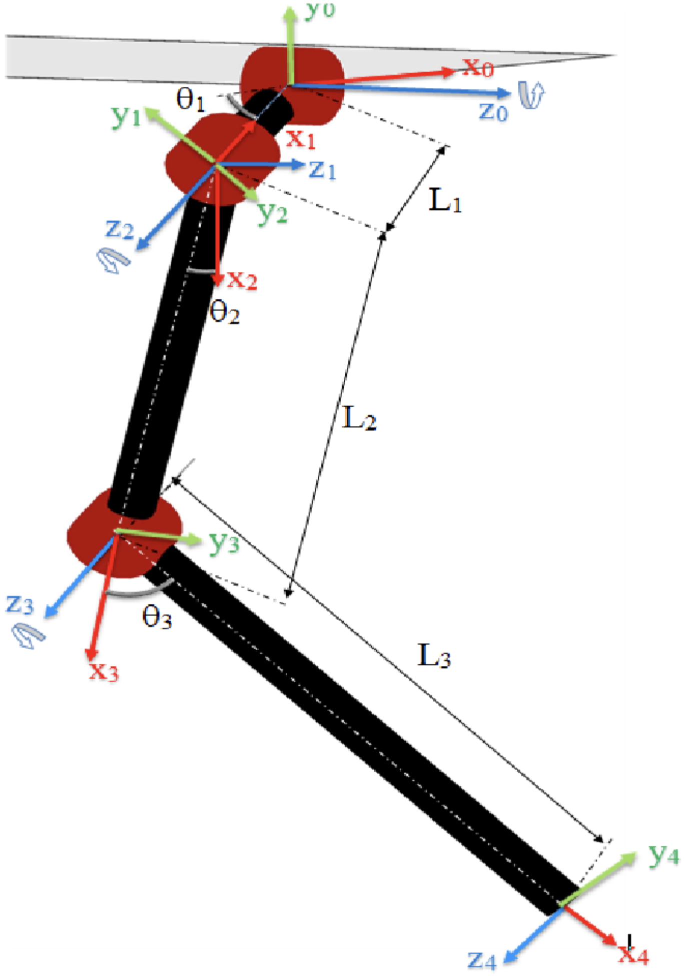
$$\theta_2 = \operatorname{atan2}\left(z_4, \sqrt{x_4^2 + y_4^2 - L_1^2}\right) - \operatorname{atan2}\left(L_3 \sin(\theta_3), L_2 + L_3 \cos(\theta_3)\right)$$

$$\theta_3 = \operatorname{atan2}\left(\pm\sqrt{1 - D^2}, D\right)$$

where

$$D = \frac{x_4^2 + y_4^2 + z_4^2 - L_1^2 - L_2^2 - L_3^2}{2L_2L_3}$$

- the $\pm$ sign in θ_3 determines the knee direction if the quadruped.



Jacobian

The Jacobian matrix is a matrix of partial derivatives that describes how the robot's configuration affects the robot's end-effector position. The Jacobian matrix is defined as:

$$J = \frac{\partial x}{\partial \theta}$$

where x is the end-effector position and θ is the joint angles.

If we consider the differentiation w. r. t. time, we can write the relationship between $\dot{x}$ (or v) and $\dot{\theta}$.

$$\dot{x} = J(\theta)\dot{\theta}$$

$$\dot{\theta} = J^{-1}(\theta)\dot{x}$$

Basic Jacobian

Numerical Inverse Kinematics

Given an initial guess θ^0 that is close to a solution θ_d, the kinematics can be expressed as the Taylor expansion:

$$\begin{aligned} x_d = f(\theta_d) &= f(\theta^0) + \frac{\partial f}{\partial \theta}\bigg|_{\theta^0} (\theta_d - \theta^0) + \text{h.o.t.} \\ &= f(\theta^0) + J(\theta^0)\Delta\theta + \text{h.o.t.} \end{aligned}$$

where $J(\theta^0)$ is the coordinate Jacobian at θ^0.

By truncating the Taylor expansion at first order, we can obtain the approximation as:

$$J(\theta^0)\Delta\theta = x_d - f(\theta^0)$$

Assuming that $J(\theta^0)$ is square and invertible, we can solve for $\Delta\theta$ as:

$$\Delta\theta = J^{-1}(\theta^0)(x_d - f(\theta^0))$$

- In practice, **pseudo-inverse** of a Jacobian J^+ is used and we do not need to assume square and invertible.

We will **iteratively update** the guess until it converges to a solution.

$$\theta^{i+1} \longleftarrow \theta^i + \eta\Delta\theta$$