MCE/EEC 647/747: Robot Dynamics and Control

Lecture 3.5: Sumary of Inverse Kinematics Solutions

Reading: SHV Sect.2.5.1, 3.3

Mechanical Engineering
Hanz Richter, PhD

Inverse Orientation: Euler Parameterization

- Suppose a desired orientation is specified between any two frames (numerically, through a 3x3 rotation matrix R)
- A good approach is to select a particular decomposition (parameterization) and try to solve for its independent parameters.
- With the Euler parameterization, we saw that

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

- SHV develops a well-reasoned, step-by-step solution of $R_{ZYZ} = R$ to find θ, ϕ and ψ. (See Sect. 2.5.1)

Inverse Orientation by Euler: Solutions

- Let r_{ij} be the numerical entries of R, for $i, j = 1, 2, 3$.
- If at least one of r_{31} and r_{32} is not zero, there will be two solutions for θ:

$$\theta^+ = \text{atan2}(r_{33}, \sqrt{1 - r_{33}^2})$$

$$\theta^- = \text{atan2}(r_{33}, -\sqrt{1 - r_{33}^2})$$

- Atan2 is the two-argument arctangent function defined in Appendix A. *Warning: Not exactly the same as Matlab's* `atan2`
- With θ^+, the remaining solutions are

$$\phi^+ = \text{atan2}(r_{13}, r_{23})$$

$$\psi^+ = \text{atan2}(-r_{31}, r_{32})$$

Inverse Orientation by Euler: Solutions...

- With θ^-, the remaining solutions are

$$\phi^- = \text{atan2}(-r_{13}, -r_{23})$$

$$\psi^- = \text{atan2}(r_{31}, -r_{32})$$

- If r_{31} and r_{32} are zero, θ can be either 0 or π.
- In this case, the system is underdetermined (infinite number of solutions for ϕ and ψ).
- Only the sum $\phi + \psi$ is determined by problem information.

Example

1. Find a way to use Matlab's `atan2` so that the results match SHV's Atan2.
2. Write code to assist you in finding solutions to the inverse orientation problem by the Euler parameterization

Decompose

$$\mathrm{Rot}_{x,\pi/4}\mathrm{Rot}_{y,\pi/3}\mathrm{Rot}_{y,-\pi}\mathrm{Rot}_{z,\pi}$$

into Euler angles and verify both solutions.

The solution for the pitch, roll and yaw parameterization is closely-related to the Euler solution (just a permutation of angles).

Inverse Position: 2-Link Planar Arm

See Figure 1.22 in SHV.

- Any desired endpoint position can be projected onto the world frame as x and y.
- Let $D = \frac{x^2+y^2-a_1^2-a_2^2}{a_1 a_2}$. Then two solutions exist for q_2:
- Elbow up

$$q_2 = \tan^{-1}(-\sqrt{1-D^2}/D)$$

- Elbow down

$$q_2 = \tan^{-1}(\sqrt{1-D^2}/D)$$

- Once q_2 is determined:

$$q_1 = \tan^{-1}(y/x) - \tan^{-1}\left(\frac{a_2 \sin(q_2)}{a_1 + a_2 \cos(q_2)}\right)$$

Inverse Position and Orientation

1. To meet a simultaneous end frame position and orientation requirement, we must match H_n^o to a given numerical 4x4 matrix H.
2. There will always be 12 equations, since the last row of H and H_n^0 is always 0 0 0 1.
3. The number of unknowns depends on the robot (n unknowns for n joints).
4. The problem is in general difficult to solve. There could be no solutions, or one or more solutions.
5. Some configurations have relatively simple geometries leading to known solutions.

Inverse Position and Orientation by Decoupling

Special case: 6 DOF (very useful)

1. The last 3 joints have actuation axes intersect at a single point o_c (the wrist center).
2. The point of interest (whose world position is being requested) is o. We assume that o is obtained by translation by d_6 units starting from o_c, along the z_6 axis.
3. The frame of interest (whose world orientation is being requested) is centered at o_c. The desired orientation relative to the world is R.
4. Under these assumptions we have $o^0 = H_6^0[0\ 0\ d_6|1]^T$, where the structure of H_6^0 is:

$$\begin{bmatrix} R & o_c^0 \\ 0 & 1 \end{bmatrix}$$

Inverse Position and Orientation by Decoupling

- Therefore $o = o_c^0 + R[0\ 0\ d_6]^T$. If the data for o is $[o_x, o_y, o_z]$ and the world components of o_c are $[x_c, y_c, z_c]$:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

- The above will give us the required o_c. Decoupling: o_c depends only on the first three joint variables!
- We can then find q_1, q_2 and q_3 to obtain o_c independently (an inverse position sub-problem).

Inverse Position and Orientation by Decoupling...

- Decoupling: R_3^0 depends only on the first three joints. Assume inverse position has been solved.
- Then R_3^0 is known and

$$R = R_3^0 R_6^3, \quad R_6^3 = (R_3^0)^{-1} R$$

- Once R_6^3 is obtained, we solve an inverse orientation sub-problem (using Euler) for q_4, q_5 and q_6.
- SHV provides details for the inverse position sub-problem for various common configurations.
- Multiple solutions may exist (up to 4 for the PUMA robot, 8 total in combination with 2 inverse orientation solutions).
- Understanding and applying these solutions is left as a homework problem.

Kinematics with Corke's Robotics Toolbox

Some functions

- Forward Euler calculation `eul2r`
- Inverse orientation by Euler (try our example with the toolbox) `tr2eul`. Use the flip option for the negative solution.
- Create a robot link using DH parameters: `L=Link([theta,d,a,alpha,jtype])` Use `jtype=0` for revolute, 1 for prismatic.
- Use `theta=0` to leave the joint coordinate unspecified.
- `Link` defines the frame attached to the link per DH. Recover the corresponding H with `L.A(thetavalue)`
- Similarly, use `R.a`, `R.offset` to recover or redefine length and offset.

Kinematics with Corke's Robotics Toolbox

Some functions...

- Building a robot: use `L(1)=Link(..),L(2)=Link(...)`
- Then `myrobot=SerialLink(L,'name','chosenname')`
- Find the value of $H_n^0(q)$: `myrobot.fkine(q)`
- Plot the robot pose at q: `myrobot.plot(q)` (list q as a row vector!)
- Built-in robot: `mdl_puma560` (creates `p560` serial link object
- Inverse kinematics by decoupling (PUMA 560 satisfies all assumptions): `p560.ikine6s(H,'opt')`
- `H` is the desired position+orientation. Options to choose among various solutions.

Kinematics with Corke's Robotics Toolbox

Some functions...

- General inverse kinematics (numerical search):
  `myrobot.ikine(H,guess)`
- Example: For the 2-link planar manipulator with unit link lengths, we compute the required $q(t)$ so the endpoint describes a circle of radius 0.5 centered at (1,1). We verify using forward kinematics.