Discussion 5/16/2008

Gabriele Nataneli
Character’s Rest Pose

Degrees of freedom: 3 translational at the root.

  3 rotational at the shoulder: (first around x, then y, then z)
  2 rotational at the elbow (first around x then around y)
  2 rotational at the wrist (y,z): first around y then around z

The figure shows the positions of the joints and the axis or rotation.
The lengths of the limbs are up to you. Try to use numbers that make sense.
IK Problem

IN GENERAL
• Given the position of the end effector, compute the joint angles of the articulated body

SPECIFICALLY
Given the position of the hand, compute the position of the elbow and wrist
End Effector

• Need to parameterize the position of the end effector in terms of the joint angles of the other links

\[ f(\theta_1, \theta_2, \ldots, \theta_7) \]
End Effector

- the function $f$ effectively converts points from the world frame to the end effector frame

\[
p_{\text{world \_ frame}} = g(\theta_1, \theta_2, \ldots, \theta_7) p_{\text{effector \_ frame}}
\]

\[
p_{\text{world \_ frame}} = T_h R_w r T_{wr} R_{el} T_{el} R_{sh} T_{sh} T_{\text{root}} p_{\text{effector \_ frame}}
\]

\[
p_{\text{world \_ frame}} = f(\theta_1, \theta_2, \ldots, \theta_7)
\]
End Effector

\[ T_{\text{root}} \] translation to the root of character
\[ T_{\text{sh}} \] translation to from root to shoulder
\[ R_{\text{sh}} \] rotation of the shoulder
\[ T_{\text{el}} \] rotation from shoulder to elbow
\[ R_{\text{el}} \] rotation of the elbow
\[ T_{\text{wr}} \] translation from elbow to wrist
\[ R_{\text{wr}} \] rotation of the wrist
\[ T_{h} \] translation from wrist to tip of the hand (end effector)
End Effector

- Translation matrices are constant
- Rotation matrices depend on the joint angles
- Using Euler angles

\[
R_{sh} = R_z(\theta_3) * R_y(\theta_2) * R_x(\theta_1) \\
R_{el} = R_y(\theta_5) * R_x(\theta_4) \\
R_{wr} = R_y(\theta_7)R_z(\theta_6)
\]
Reminder

\[ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
End Effector

Note About Transformations

- We use homogeneous coordinates to represent the end effector so each transformation is a 4x4 matrix and the position of the end effector is a 4 vector.

\[
p_{\text{end Effector}} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}
\]

- As a result our function is a mapping:

\[ f : \mathbb{R}^7 \rightarrow \mathbb{R}^4 \]
End Effector

Note about the position of the end effector

• If we assume the end effector lies at the origin of its own coordinate frame, we can use:

\[
p_{\text{end effector}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\]

• You can use this to get numbers out the function \( f \). Please think about the meaning of this. What if you wanted to place a particle system at the location of the end effector?
Why do we need the Jacobian?

- 1 DOF Problem

\[ x = f(\theta) \]

\[ \Delta \theta \]

\[ \Delta x \]

\[ \sim \Delta x \]
Why do we need the Jacobian?

• From the figure we infer

\[ \Delta x \approx \frac{df}{d\theta} \Delta \theta \]

• For a multivariate function

\[ \Delta x \approx J \Delta \theta \]

• In the limiting case

\[ \dot{x} = J \dot{\theta} \]
Compute the Jacobian

The Jacobian should account for the position of the end effector, resulting in a 3X7 Jacobian.
Compute the Jacobian

- Need to compute the partials

\[
\begin{bmatrix}
\frac{\partial f_x}{\partial \theta_1} & \frac{\partial f_x}{\partial \theta_2} & \cdots & \frac{\partial f_x}{\partial \theta_7} \\
\frac{\partial f_y}{\partial \theta_1} & \frac{\partial f_y}{\partial \theta_2} & \cdots & \frac{\partial f_y}{\partial \theta_7} \\
\frac{\partial f_z}{\partial \theta_1} & \frac{\partial f_z}{\partial \theta_2} & \cdots & \frac{\partial f_z}{\partial \theta_7}
\end{bmatrix}
\]
Compute the Jacobian

• For convenience you may use matrix operations in your code to compute the jacobian, so we better consider one column at a time

\[
\begin{bmatrix}
\frac{\partial f}{\partial \theta_1} & \frac{\partial f}{\partial \theta_2} & \cdots & \frac{\partial f}{\partial \theta_7} \\
\end{bmatrix}
\]
Example

• First column of the Jacobian
\[
\frac{\partial f}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \left( T_h R_{wr} T_{wr} R_{el} T_{el} R_{sh} T_{sh} T_{root} P_{effector\_frame} \right)
\]

• Remember that the rotation matrices are functions of the DOF. Then:
\[
\frac{\partial f}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \left( T_h R_{wr} (\theta_6, \theta_7) T_{wr} R_{el} (\theta_4, \theta_5) T_{el} R_{sh} (\theta_1, \theta_2, \theta_3) T_{sh} T_{root} P_{effector\_frame} \right)
\]
\[
= T_h R_{wr} (\theta_6, \theta_7) T_{wr} R_{el} (\theta_4, \theta_5) T_{el} \frac{\partial}{\partial \theta_1} \left( R_{sh} (\theta_1, \theta_2, \theta_3) \right) T_{sh} T_{root} P_{effector\_frame}
\]
\[
= T_h R_{wr} (\theta_6, \theta_7) T_{wr} R_{el} (\theta_4, \theta_5) T_{el} \frac{\partial}{\partial \theta_1} \left( R_{sh\_x} (\theta_1) R_{sh\_y} (\theta_2) R_{sh\_z} (\theta_3) \right) T_{sh} T_{root} P_{effector\_frame}
\]
\[
= T_h R_{wr} (\theta_6, \theta_7) T_{wr} R_{el} (\theta_4, \theta_5) T_{el} \frac{\partial}{\partial \theta_1} \left( R_{sh\_x} (\theta_1) \right) R_{sh\_y} (\theta_2) R_{sh\_z} (\theta_3) T_{sh} T_{root} P_{effector\_frame}
\]
How to code this up

1. Implement functions that give you a rotation matrix for each component axis
2. Implement functions that give you the derivative of a rotation matrix for each component axis
3. Implement a function that calls the previous one to produce the proper sequence of matrix multiplications for each column of the Jacobian
4. Multiply the resulting 4x4 transformation matrix by $p_{\text{end\_effector}}$ to get one column of the Jacobian
5. Call the function in (3) multiple times to compute the actual Jacobian
Solve the IK problem

• Need to relate a change in the position of the end effector to a corresponding change in the joint angles

\[
\dot{x} = J \dot{\theta} \\
\dot{\theta} = J^{-1} \dot{x} \quad \text{We cannot compute this, since the Jacobian is underdetermined}
\]

Instead use the pseudo-inverse

\[
\dot{\theta} = J^+ \dot{x} = J^T (JJ^T)^{-1} \dot{x}
\]

Define

\[
\beta = (JJ^T)^{-1} \dot{x} \Rightarrow \dot{\theta} = J^T \beta
\]

Evaluate \( \beta \)
Update the Joint angles

- With the Jacobian you can now estimate the change in joint angles required to achieve a corresponding change in the position of the end effector.
- For simplicity let’s consider the discrete case

\[ \Delta \theta = J^+ \Delta x \]

- Update the joint angles as follows

\[ \theta_{\text{current}} + \Delta \theta \]
Summary of Procedure

Initial case

- Move the arm so that the end effector is positioned at the beginning of the spline
- Do this by **manually** setting the initial values of joint angles
- Possibly animate the arm as it goes from the rest pose to the initial pose
Summary of Procedure

1. Determine the value of $\Delta x$ by moving forward on the spline by a small step
2. Evaluate the Jacobian at the current value of joint angles. Use the initial joint angles for the first step
3. Use the value of the Jacobian to get the pseudo-inverse. You need to perform one matrix inversion for this
4. Multiply the pseudo-inverse by the current value of $\Delta x$ to get $\Delta \theta$
5. Add $\Delta \theta$ to the current value of joint angles to get the next value of joint angles
6. Go back to (1) until you reach the end of the spline.

NOTE: When you finish tracing the spline go back to the beginning and do it again so the animation will repeat continuously