Exercises on direct kinematics

Raquel Urtasun
TTI Chicago

March 1, 2010

1 Direct kinematics

This is the first of a series of exercises to introduce the basic concepts. The experiments will be perform in Matlab. Return the code that you generate and the answer to the questions on the submission date that appears on the website.

1.1 Euler angles

Create a function on matlab that takes as input 3 angles $\theta_x, \theta_y, \theta_z$ and computes the corresponding rotation matrix.

$$function[R] = computeRotationFromEuler(\theta_x, \theta_y, \theta_z)$$

Note that matlab uses radians to for computing cos and sin. You can convert to degrees by:

$$\theta_{degrees} = \theta_{rad} \frac{1\dot{8}0}{\pi}$$

Create a random vector $\mathbf{x} \in \Re^{\mathbf{3}}$ on the interval $[0, 1]$, by using $x = rand(1, 3)$. Visualize the vector using the function

$$plot3([0, x(1)], [0, x(2)], [0, x(3)], 'ro-', 'markersize', 10);$$

Convert x to homogeneous coordinates by appending a one to the end of the vector,

$$x = [x, 1];$$

Rotate this vector using the rotation matrix computed by the function *computeRotationFromEuler* to compute the new vector y. Plot the result of the rotation using the following command

$$hold\ on; \quad plot3([0, y(1)], [0, y(2)], [0, y(3)], 'bo-', 'markersize', 10);$$

Use the *hold on* command to visualize $\mathbf{x}$ and $\mathbf{y}$ vectors on the same plot. Make sure that what you did is a rotation by computing the norm of both vectors x and y. Is the norm the same? why? Create a new function that computes rotations and translations

$$function[RT] = computeRotationTranslationFromEuler(\theta_x, \theta_y, \theta_z, \mathbf{t})$$

where t is a 3D translation. Do it by calling the previous function, and multiplying the result by a matrix that does only translation. What's the difference when you multiply left and right

by the translation? What happen in each case? Generate an RT matrix and apply the rotation to $\mathbf{x}$ to generate $\mathbf{z}$. Plot the result

$$plot3([0, z(1)], [0, z(2)], [0, z(3)], 'go-', 'markersize', 10);$$

Compute the norm now. What's it's value? has it changed? why?

Use the function *computeRotationTranslationFromEuler* to compute a case of Gimbal lock. What happens if you apply $R(\theta_1, \frac{\pi}{2}, \theta_2)$?

Compute a case where the euler angles are not a unique representation. What's the rotation matrix in this case? why the non-uniqueness is a bad thing?

1.2 Quaternions

A quaternion with rotation θ along the $\hat{v}$ axis is defined as

$$\mathbf{q} = [\cos\frac{\theta}{\mathbf{2}}, \sin\frac{\theta}{\mathbf{2}}\hat{\mathbf{v}}]$$

with $\hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}$. Proof that If $\hat{\mathbf{v}}$ is unit length, then $\mathbf{q}$ will also be.

1.3 Kinematic chain

Compute a function that creates a kinematic chain of 3 joints, and that initializes the rotations to the identity matrix, and the translation to t_1,t_2,t_3.

$$function[kinematic] = computeKinematicChain(t_1, t_2, t_3)$$

Play with different rotations of the kinematic chain. See the effects by computing a function that visualizes the kinematic chain

$$function[\,] = visualizeKinematicChain(kinematic)$$

If the position of the end-effector is computed by recursively applying the local rotations, what rule should you use in order to compute the Jacobian?

1.4 Inverse kinematics

Proof that the pseudo-inverse satisfies the following properties

$$\begin{aligned}
\mathbf{A}\mathbf{A}^\dagger\mathbf{A} &= \mathbf{A} \\
\mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger &= \mathbf{A}^\dagger \\
(\mathbf{A}\mathbf{A}^\dagger)^* &= \mathbf{A}\mathbf{A}^\dagger \\
(\mathbf{A}^\dagger\mathbf{A})^* &= \mathbf{A}^\dagger\mathbf{A}
\end{aligned}$$

by using the fact that the SVD of a matrix is

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

and the SVD of the pseudo-inverse is

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^*$$