EENG 428 Laboratory --- Lab Session 3

Description:

In this session, mathematical derivation for forward and inverse kinematic problems are to be discussed, starting from simple coordinate systems, and ending with the Denavit-Hartenberg convention.

Prerequisites:

Attending students are expected to know:

- Different forms of transformation matrices (Rotation about / Translation along an axis) and their Matlab implementations using the robotic toolbox.
- D-H convention.

Contents:

1- Forward and inverse kinematics in simple coordinate systems.

2- Forward kinematic equation for a serial manipulator using D-H convention.

1- Forward and inverse kinematics in simple coordinate systems:

Representing all points of 3D space can be achieved through infinite number of different sets of coordinates. The most preferable coordinates are known to be orthogonal. The most common orthogonal coordinates are the Cartesian, cylindrical and the spherical coordinates. These coordinates offer to represent any point (zero dimensional) in space.

Representing all directions of 3D space can be achieved through infinite number of different sets of coordinates. These coordinates are also 3 dimensional and articulated (only rotational). The most common coordinates are the Euler, and Roll-Pitch-Yaw (RPY) coordinates. These coordinates offer to represent any direction (with zero distance from the origin) in space.

1.1- Cartesian robot, forward and inverse kinematics:

A Cartesian robot consist of 3 prismatic joints (linear joints), that are placed in orthogonal planes. The position of the tool (the last frame) w.r.t the base is given by:

$${}^{B}T_{H} = \text{Trans}\ (P_x,P_y,P_z)$$

Both forward and inverse kinematics (for positions) are very simple to compute, the Matlab representation for forward and inverse kinematics for the Cartesian robot are as follows:

```
syms px py pz

T=transl(px,py,pz) % Forward kinematics

X=T(1:3,4); % Inverse kinematics
x=X(1)
Y=X(2)
z=X(3)
```

1.2- Cylindrical robot, forward and inverse kinematics:

A cylindrical robot consist of one revolute joint and two prismatic joints that are placed in orthogonal planes. The position of the tool (the last frame) w.r.t the base is given by:

$$ {}^{R}T_{P} = T_{cyl}(r,\alpha,l) = \text{Trans}(0,0,l)*\text{Rot}(z,\alpha)*\text{Trans}(r,0,0) $$

$$ \text{Or: } {}^{R}T_{P} = T_{cyl}(r,\alpha,l) = \text{Rot}(z,\alpha)*\text{Trans}(r,0,l) $$

Forward and inverse kinematics for cylindrical robots are only valid to define the position of the last frame with respect to the base, where the orientation of the last frame cannot be controlled.

```
syms r al l

Tcyl=transl(0,0,l)*trotz(al)*transl(r,0,0) %forward
kinematics
%Example:
Tcyl=subs(Tcyl,[r al l],[2 pi/3 5]);

%inverse kinematics
z= Tcyl(3,4)
theta= atan2(Tcyl(2,4),Tcyl(1,4))
```

```
if(Tcyl(1,4)==0)
    R=Tcyl(2,4)/sin(theta)
else
    R=Tcyl(1,4)/cos(theta)
End
```

1.3- Spherical robot, forward and inverse kinematics:

A cylindrical robot consist of two revolute joints and one prismatic joint that are placed in orthogonal planes. The position of the tool (the last frame) w.r.t the base is given by:

$$^{R}T_{P} = T_{sph} = Rot(z,\gamma)*Rot(y,\beta)*Trans(0,0,r)$$

Forward and inverse kinematics for spherical robots are only valid to define the position of the last frame with respect to the base, where the orientation of the last frame cannot be controlled.

Exercise 1 (Homework):

Write a simple Matlab code that allows to find forward and inverse kinematics for a spherical manipulator. (go with symmetry with the code written for the cylindrical manipulator in the lab sheet).

1.4- Forward and inverse kinematics in Roll-Pitch-Yaw (RPY) coordinates:

Any orientation in 3D space can be achieved by three sequential rotations about z, y and x axes respectively.

$$T_{RPY} = Rot(z, \theta_z)*Rot(y, \theta_y)*Rot(x, \theta_x)$$

The angles θ_z , θ_y and θ_x are known as the Roll-Pitch-Yaw angles respectively.

There is a possibility to achieve any orientation by reversing the order of rotations (about x then y then z). Yet, the first order is more used in the course book.

- Forward kinematics: if RPY angles are given, the orientation of the last frame can be directly computed, and it is unique.

```
syms tx ty tz
Trpy=trotz(tz)*troty(ty)*trotx(tx)
```

Substituting the angles with their numerical values gives the final orientation.

- Inverse kinematics: if the desired orientation is given, the values of the RPY angles can be calculated to achieve the same orientation. Usually, there are two set of solutions that give the same orientation.

$$\text{For any } {}^{R}T_{p} = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_{RPY} should be equal to ${}^{R}T_{p}$, this will lead us to the following two sets of solutions:

$\theta_{y(1)} = ATAN2(-n_z,+\sqrt{1-n_z{}^2})$	$\theta_{y(2)} = ATAN2(-n_z,-\sqrt{1-n_z{}^2})$
$\theta_{x(1)} = ATAN2(o_z/\sqrt{1-n_z{}^2},a_z/\sqrt{1-n_z{}^2})$	$\theta_{x(2)} = ATAN2(-o_z/\sqrt{1-n_z{}^2},-a_z/\sqrt{1-n_z{}^2})$
$\theta_{z(1)} = ATAN2(n_y/\sqrt{1-n_z{}^2},n_x/\sqrt{1-n_z{}^2})$	$\theta_{z(2)} = ATAN2(-n_y/\sqrt{1-n_z{}^2},-n_x/\sqrt{1-n_z{}^2})$

Example:

Given ${}^{R}T_{p} = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 0 \\ 0.505 & 0.722 & 0.475 & 0 \\ -0.788 & 0.160 & 0.595 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, find all sets of the RPY angles that achieve the same orientation.

Solution:

```
clear all
close all
syms tx ty tz
T=[0.354 -0.674 0.649 0;0.505 0.722 0.475 0;-0.788 0.160
0.595 0; 0 0 0 1]
Trpy=trotz(tz)*troty(ty)*trotx(tx)

ty1=atan2(-T(3,1),sqrt(1-(T(3,1)^2)))
ty2=atan2(-T(3,1),-sqrt(1-(T(3,1)^2)))

tx1=atan2(T(3,2)/cos(ty1),T(3,3)/cos(ty1))
tx2=atan2(T(3,2)/cos(ty2),T(3,3)/cos(ty2))

tz1=atan2(T(2,1)/cos(ty1),T(1,1)/cos(ty1))
tz2=atan2(T(2,1)/cos(ty2),T(1,1)/cos(ty2))

sets=[180*[tx1 ty1 tz1]'/pi 180*[tx2 ty2 tz2]'/pi]
```

Matlab robotic toolbox offers two ready functions that can find both of forward and inverse kinematics for RPY coordinates, these functions are *rpy2tr()*and *tr2rpy()*, an example for the usage is given below:

```
rpy2tr(0.9594,0.9076,0.2627,'zyx') % forward kinematics
for RPY
rpy2tr(55,51,15,'deg','zyx')%  forward  kinematics  for
RPY, given the angles in degree

T=[0.354 -0.674 0.649 0;0.505 0.722 0.475 0;-0.788 0.160
0.595 0; 0 0 0 1];
```

```
% inverse kinematics for RPY
tr2rpy(T,'zyx')
tr2rpy(T,'deg','zyx')
```

1.5- Forward and inverse kinematics in Euler coordinates:

Any orientation in 3D space can be achieved by three sequential rotations about z, y then z axes respectively.

$$T = \mathrm{Rot}(z, \theta_1)*\mathrm{Rot}(y, \theta_2)*\mathrm{Rot}(z, \theta_3)$$

The angles θ_1 , θ_2 and θ_3 are known as the Euler angles.

- Forward kinematics: if Euler angles are given, the orientation of the last frame can be directly computed, and it is unique.

```
syms t1 t2 t3
Trpy=trotz(t1)*troty(t2)*trotz(t3)
```

Substituting the angles with their numerical values gives the final orientation.

- Inverse kinematics: if the desired orientation is given, the values of the Euler angles can be calculated to achieve the same orientation. Usually, there are two set of solutions that give the same orientation.

$$\text{For any } {}^{R}T_{p} = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T_{Eul} should be equal to ${}^{R}T_{p}$.

Exercise 2 (Homework):

1- Derive the inverse kinematic formulas for the Euler angles that achieve any desired orientation. (find two sets similar to the RPY angles) (Depend on pages 64-66 in the course book).

2- Given ${}^{R}T_{p} = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 0 \\ 0.505 & 0.722 & 0.475 & 0 \\ -0.788 & 0.160 & 0.595 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, find all sets of the Euler angles that achieve the same orientation.

3- Matlab robotic toolbox offers two ready functions that can find both of forward and inverse kinematics for Euler coordinates, these functions are *eul2tr()*and *tr2eul().* Give one example for the usage of these functions (similar to the given example for RPY).

4- Both RPY and Euler angles can achieve any orientation in 3D space, these angles are connected to each other by one transformation and some offset(s). By inspection, find the angles that replaces the question marks in the following formula:

$$RPY(\theta_1, \theta_2, \theta_3) * rot(y, 90°) = Euler(?,?,?)$$

(state clearly how did you find the unknowns).

1.6- Rotation about an arbitrary axis:

In previous, we discussed sequential rotations about x,y and z axes that can achieve any orientation in 3D space (RPY and Euler coordinates).

It is also possible to achieve any orientation in 3D space by only one rotation about a certain axis (a vector) with the correct angle. This is known by the rotation about an arbitrary axis, and has many applications.

The transformation matrix that correspond to the rotation about a non-zero unit vector ***k*** with an angle ***Ɵ*** is given by:

Rot (***k***, ***Ɵ***) =

$$\begin{bmatrix} \cos(\theta) + {k_x}^2(1-\cos(\theta)) & k_xk_y(1-\cos(\theta)) - k_z\sin(\theta) & k_xk_z(1-\cos(\theta)) + k_y\sin(\theta) & 0 \\ k_xk_y(1-\cos(\theta)) + k_z\sin(\theta) & \cos(\theta) + {k_y}^2(1-\cos(\theta)) & k_yk_z(1-\cos(\theta)) - k_x\sin(\theta) & 0 \\ k_xk_z(1-\cos(\theta)) - k_y\sin(\theta) & k_yk_z(1-\cos(\theta)) + k_x\sin(\theta) & \cos(\theta) + {k_y}^2(1-\cos(\theta)) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- <u>Forward kinematics</u> is very easy to compute (by substitution in the above matrix).
- <u>Inverse kinematics:</u>

  Given a transformation matrix (the desired orientation) ${}^{R}T_{p}$, finding the angle of rotation about a vector that achieve the same orientation in the inverse kinematic problem.

  For any ${}^{R}T_{p} = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$:

  The sum of the diagonal elements:

$$\cos(\theta) + {k_x}^2(1 - \cos(\theta)) + \cos(\theta) + {k_y}^2(1 - \cos(\theta)) + \cos(\theta) + {k_y}^2(1 - \cos(\theta))$$

$$= n_x + o_y + a_z$$

As k is a unit vector:

$$2\cos(\theta) + 1 = n_x + o_y + a_z$$

And this gives:

$$\boxed{\theta = \cos^{-1}[(n_x + o_y + a_z - 1)/2]}$$

Then, by subtracting each non-diagonal element from its mirror gives:

$$2k_z \sin(\theta) = n_y - o_x$$

$$2k_y \sin(\theta) = a_x - n_z$$

$$2k_x \sin(\theta) = o_z - a_y$$

Then:

$$\boxed{\begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} = \begin{bmatrix} (o_z - a_y)/(2\sin(\theta)) \\ (a_x - n_z)/(2\sin(\theta)) \\ (n_y - o_x)/(2\sin(\theta)) \end{bmatrix}}$$

There are two values that θ can take, each value correspond to a different unit vector.

Mathematically, the values of theta are can be found as:

$$\theta_1 = \cos^{-1}[(n_x + o_y + a_z - 1)/2]$$

$$\theta_2 = -\cos^{-1}[(n_x + o_y + a_z - 1)/2]$$

Exercise 3 (Homework):

1- Given $^{R}T_{p} = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 0 \\ 0.505 & 0.722 & 0.475 & 0 \\ -0.788 & 0.160 & 0.595 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, find all sets of possible unit vectors with corresponding rotational angles that achieve the same orientation.

2- Matlab robotic toolbox offers two ready functions that can find both of forward and inverse kinematics for rotation about an arbitrary axis, these functions are *angvec2tr()*and *tr2angvec().* Give one example for the usage of these functions.

3- The ready function *tr2angvec()* gives only one set of answers for the inverse kinematic problem (rotation about an arbitrary axis). Generalize it by writing a function that gives the two sets of vector-angle that can achieve a desired (given) orientation.

2- Denavit-Hartenberg convention in matrix representation:

Denavit-Hartenberg convention is used to facilitate and standardize the way of representing two or more sequential frames of serial robotic manipulators.

Using this convention, any two sequential frames are related to each other by only two rotations and two translations as follows:

$$^{n}T_{n+1} = A_{n+1} = \mathrm{Rot}(z,\theta_{n+1})*\mathrm{Trans}(0,0,d_{n+1})*\mathrm{Trans}(a_{n+1},0,0)*\mathrm{Rot}(x,\alpha_{n+1})$$

(Refer to your lecture notes and pages 67-75 in the course book for all details)

Using Matlab, the D-H matrix (the *A* matrix) can be written as:

```
syms t al a d
A= trotz(t)*transl(a,0,d)*trotx(al)
```

If we have the D-H parameters, the function *subs()* will allow us to find the representation of each link as in the following example.

Example:

Given the manipulator in Figure 2.26 in the course book, D-H parameters are given the following table, find all the A matrices, then find ${}^{U}T_{H}$.

#	θ	d	a	α
1	θ_1	0	0	90
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	0	a_4	-90
5	θ_5	0	0	90
6	θ_6	0	0	0

Solution:

```
syms t al a d
A= trotz(t)*transl(a,0,d)*trotx(al);
```

```
syms t1 t2 t3 t4 t5 t6 a2 a3 a4
A1=subs(A,[t d a al],[t1 0 0 pi/2])
A2=subs(A,[t d a al],[t2 0 a2 0])
A3=subs(A,[t d a al],[t3 0 a3 0])
A4=subs(A,[t d a al],[t4 0 a4 -pi/2])
A5=subs(A,[t d a al],[t5 0 0 pi/2])
A6=subs(A,[t d a al],[t6 0 0 0])
Atotal=A1*A2*A3*A4*A5*A6
Atotal=simplify(Atotal)
```

Forward kinematics: Finding ${}^{U}T_{H}$ given all joint variables is straight forward (only substitution in the product of the *A* matrices).

Note that, there are no ready functions in the Matlab robotic toolbox that can help you to derive the forward of inverse kinematics in *all symbolics formulas*. The method stated in the lab sheet will be very useful for future topics (Jacobian analysis).

Exercise 4 (Homework): (Exercise 21 in chapter 2 in the course book).

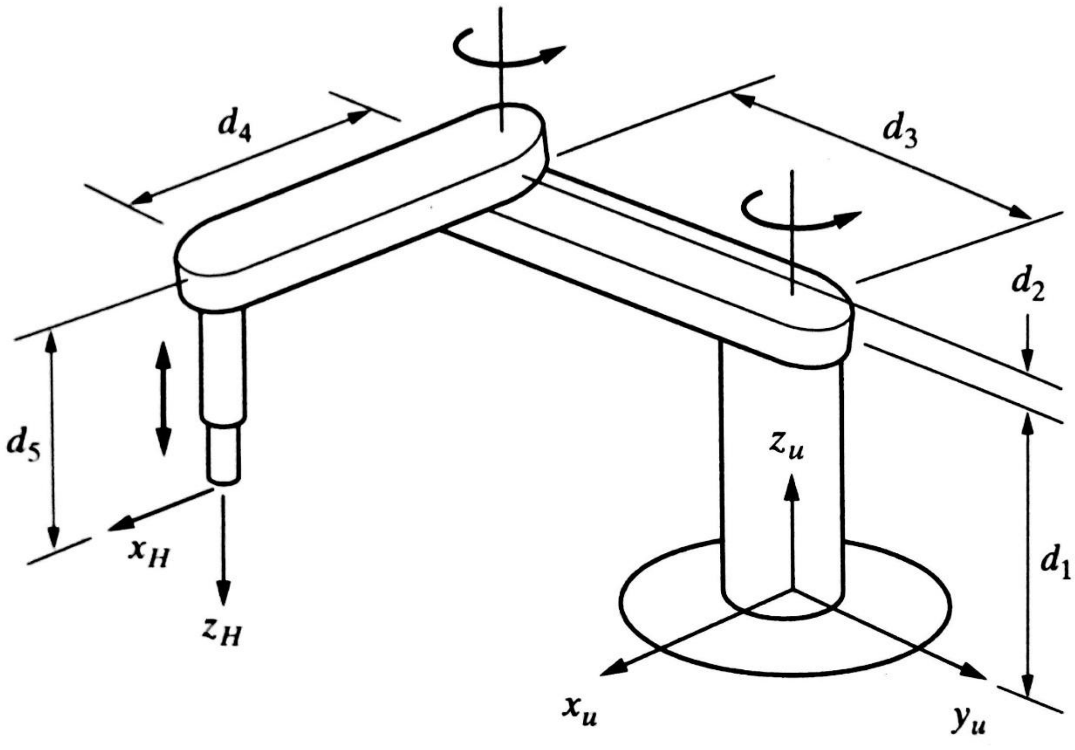
For the SCARA-type robot shown in the next figure,

1- Assign coordinate frames for its links based on D-H representation.

2- Fill out the D-H parameters table.

3- Write all the *A* matrices separately.

4- Find the transformation matrix for the tool-frame coordinates w.r.t the base (the product of the *A* matrices).



<u>Only Email submissions are accepted (you have exactly 6 days to submit)</u>

<u>Paper solutions are not accepted under any circumstances</u>

- Take clear photos of your paper solutions and combine them all in a single pdf file.
- Send your Matlab codes in a separate file (m-file or word document).

  Submit to the Email: lab.eeng428@gmail.com
- In case of emergency, contact me on the Email: (Don’t visit me in the office before writing an Email explaining your problem)

Mohamad.Harastani@emu.edu.tr