Jacob Mose Hansen - 201270410
Alex Andbæk Nielsen - 201270408
Mads Gad Krogsgaard - 201270392
Morten Hoffmann Christensen - 201270118

ITROB
18. september 2015

ITROB Journal – Group 7

Forward Kinematics (Denavit-Hartenberg Convention)

Prerequisite:

- Run startup_rvc to be able to use the toolbox

```
%% Starting the toolbox
>> mydir = pwd; %Save current directory
>> cd('C:\Users\Mads\Desktop\ITROB\rvctools') %Change the path to the location of rvctools
>> startup_rvc
>> cd('C:\Users\Mads\Desktop\ITROB\Week 4') %This changes the directory to the location of the script
```

Exercises

1. **Measure lengths and angles of links and joints in the robot.**

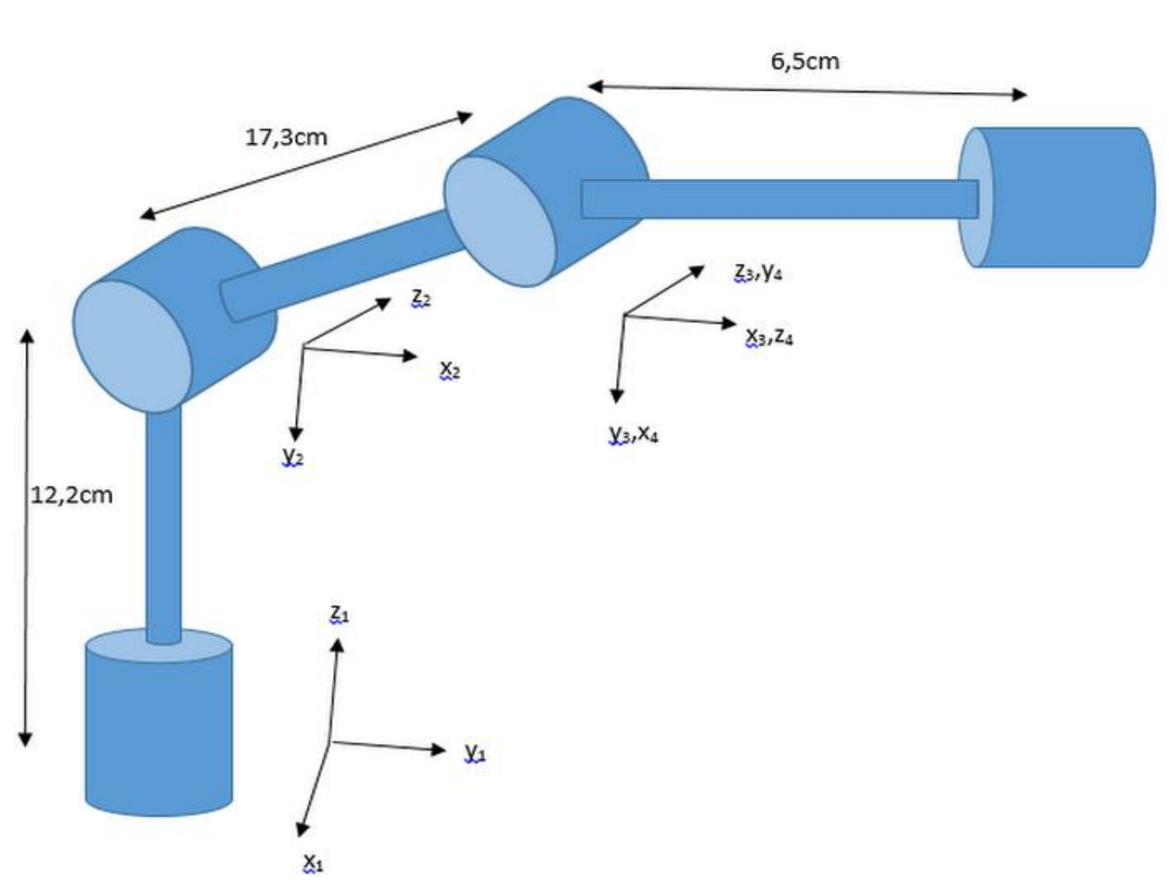


Figure 1 - Crustcrawler model

In Figure 1, the measurements and the coordinate frames of the robot are plotted onto the drawing. These are the physicical specifications of the Crustcrawler. The actual Matlab model shown in Figure 2 combines joint 3 and 4 and therefore the last link is 17.3 + 6.5 = 23.8 and this is set as an offset in order to combine the two joints.

2. Use Denavit-Hartenberg convention to establish coordinate frames for the robot.

In order to establish the coordinate frames we have to use the measured lengths and angles of the Crustcrawler to use as input parameters for the robot toolbox's Link function to create a vector of Link objects. The following code snippet is from Matlab:

```
L(1) = Link([0 4.1 0 pi/2]);%The value 4.1 is the measured base-height of the robot
L(2) = Link([0 0 12.2 0]);
L(3) = Link([0 0 0 pi/2]);
L(4) = Link([0 23.8 0 0]);
L

>> L =
 theta=q1, d=      4.1, a=        0, alpha=    1.571, offset=        0 (R,stdDH)
 theta=q2, d=        0, a=     12.2, alpha=        0, offset=        0 (R,stdDH)
 theta=q3, d=        0, a=        0, alpha=    1.571, offset=        0 (R,stdDH)
 theta=q4, d=     23.8, a=        0, alpha=        0, offset=        0 (R,stdDH)
```

The 1st parameter is theta and indicates the joint angle. The 2nd parameter *d* indicates the links offset. The 3rd parameter *a* indicates the links length and the 4th parameter alpha indicates the link rotation. Details of how to use the Link function is in the toolbox documentation[1].

[1] In the directive \rvctools\robot\robot.pdf.

3. Extract the Denavit-Hartenberg parameters from the model.

The DH parameters is extracted by establishing a serial link between the model and the toolbox by using the SerialLink function. We pass the vector of Link objects L to the constructor which returns a SerialLink object we can display. The matlab code:

```
Crustcrawler = SerialLink(L, 'name', 'Crustcrawler');

>> Crustcrawler =

Crustcrawler (4 axis, RRRR, stdDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|        4.1|          0|      1.571|          0|
|  2|         q2|          0|       12.2|          0|          0|
|  3|         q3|          0|          0|      1.571|          0|
|  4|         q4|       23.8|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1  0  0  0   tool =  1  0  0  0
           0         0  1  0  0           0  1  0  0
        9.81         0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```

The function creates a 4-link robot by connecting the four Links (L(1), L(2), L(3), L(4)). Further details on the SerialLink function can be seen under the toolbox documentation.

4. Calculate the transformation A-Matrices between frames.

To calculate the transformation matrix between frames we use the fkine function on the Crustcrawler SerialLink. This is done by only defining Link L(1) and L(2) etc. and then calling the fkine function with zero angles. See the matlab code:

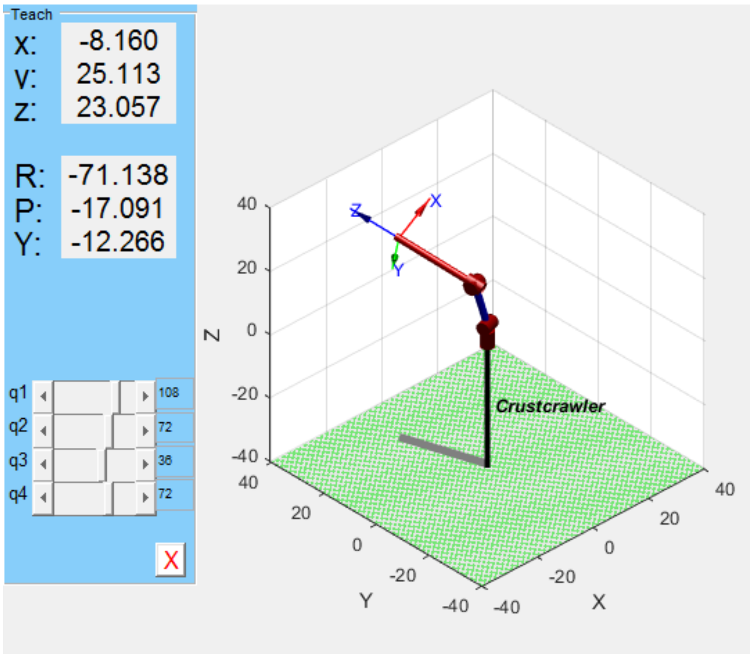
```
L(1)  = Link([0 4.1 0 pi/2]);
L(2)  = Link([0 0 12.2 0]);

…

>> Crustcrawler.fkine([0 0])

ans =

    1.0000         0         0   12.2000
         0    0.0000   -1.0000         0
         0    1.0000    0.0000    4.1000
         0         0         0    1.0000
```

```
L(1) = Link([0 0 12.2 0]);%%Orginial L(2) change to L(1) to get Matrix-A for 2-3
L(2) = Link([0 0 0 pi/2]);%%Orginial L(3) change to L(2) to get Matrix-A for 2-3
```

```
For Link 2 to Link 3 we get the following A-matrix:

>> Crustcrawler.fkine([0 0])
ans =

    1.0000         0         0   12.2000
         0    0.0000   -1.0000         0
         0    1.0000    0.0000         0
         0         0         0    1.0000

For Link 3 to Link 4 we get the following A-matrix:

L(1) = Link([0 0 0 pi/2]);%%Orginial L(3) change to L(1) to get Matrix-A for 3-4
L(2) = Link([0 23.8 0 0]);%%Orginial L(4) change to L(2) to get Matrix-A for 3-4

>> Crustcrawler.fkine([0 0])
ans =

    1.0000         0         0         0
         0    0.0000   -1.0000  -23.8000
         0    1.0000    0.0000    0.0000
         0         0         0    1.0000
```

Running the below command displays the image of the Crustcrawler-model

```
>> Crustcrawler.teach([0 0 0 0])
```



Teach
x: -8.160
y: 25.113
z: 23.057

R: -71.138
P: -17.091
Y: -12.266

q1 108
q2 72
q3 38
q4 72

Figure 2 - Crustcrawler model in Matlab

5. Calculate the full transformation T_n^0 between base and end effector frames.

To calculate the full transformation matrix between base and end effector frames, we simply use the above technique, just with fkine called with four zero angles. As such:

```
>> Crustcrawler.fkine([0 0 0 0])

ans =

    1.0000         0         0   12.2000
         0   -1.0000   -0.0000   -0.0000
         0    0.0000   -1.0000  -19.7000
         0         0         0    1.0000
```

For more information on how to use the fkine function, write `doc fkine` in the matlab command window. The same goes for `Link` and `SerialLink`.