Direct and Inverse Kinematics for the ROB3/TR5 manipulator. Robotics, 1st lab. assignment

Nicolchi - Folkert Franzen

May 8, 2013

1 Introduction

This is the report for the first assignment of the Robotics course. The task is the creation of the direct and inverse kinematics of the *ROB3/TR5* manipulator. This report shows how we achieved this task and the main practical/technical choices during the development phase. The report is divided in 4 sections.

The first section describes the frames assignment and the Euler's transformations following the Denavit-Hartenberg convention.

In the second and third section describe how we have implemented the direct and the inverse kinematics for the manipulator: how it's possible to find the position and orientation of the final manipulator using the given angles of each joint in the first case. And the second case, about the inverse kinematics, shows how to reach a certain position and orientation for the gripper using the joint angles.

The fourth section introduces some tests of our architecture, for showing how it works and checking the correctness of the computations.

In the last section we discuss some conclusions about the work.

In the appendix there is a little manual that explains how one can use the functions that we have implemented.

In the following we use these notations to abbreviate the equations:

- $s_\alpha = \sin(\alpha)$
- $c_\alpha = \cos(\alpha)$

2 D-H Configuration

During the theory classes we discovered that there isn't a perfect method for assigning the frames for the joint in every singular case. But we can

still do something, for example we can use a convention that can help us choosing the frames orientation and position. For that reason we decided to use the Denavit-Hartenberg convention[**?**][**?**].

It allows to represent a geometric transformation in the three-dimensional Euclidean space with the minimum number of parameters, namely four. In this convention, each homogeneous transformation is represented by the product of four basic transformations.

We can choose the arrangement of the frames using the following rules:

- the z-axis is in the direction of the joint axis
- the x-axis is parallel to the common normal: $x_n = z_n \times z_{n-1}$.
- the y-axis follows from the x- and z-axis by choosing it to be a right-handed coordination system.

The parameters for the Euler transformation (knowns as D-H parameters) are the following, where index i denotes the i-th frame considered:

- a_{i-1}: distance from Z_{i-1} to Z_i along X_{i-1}
- α_{i-1}: angle between Z_{i-1} and Z_i around X_{i-1}
- d_i: distance from X_{i-1} to X_i along Z_i
- θ_i: angle between X_{i-1} and X_i around Z_i

If there is no unique common normal (parallel z-axes), then d is a free parameter and the direction of x_n is from z_{n-1} to z_n. For the Euler angles convention a right-handed Z-Y-Z convention for the axis is chosen, where the first Euler angle corresponds to a rotation over the Z-axis and so on. We chose this convention, because it simplifies the inverse kinematics.

Following all the directives of the D-H convention we created the following frame configuration (figure (**??**)) and table of values of the parameters.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$0°$	0	275	θ_1
2	$-90°$	0	0	θ_2
3	$0°$	200	0	θ_3
4	$0°$	130	0	θ_4
5	$90°$	0	130	θ_5

Table 1: D-H parameters for the ROB3/TR5 manipulator

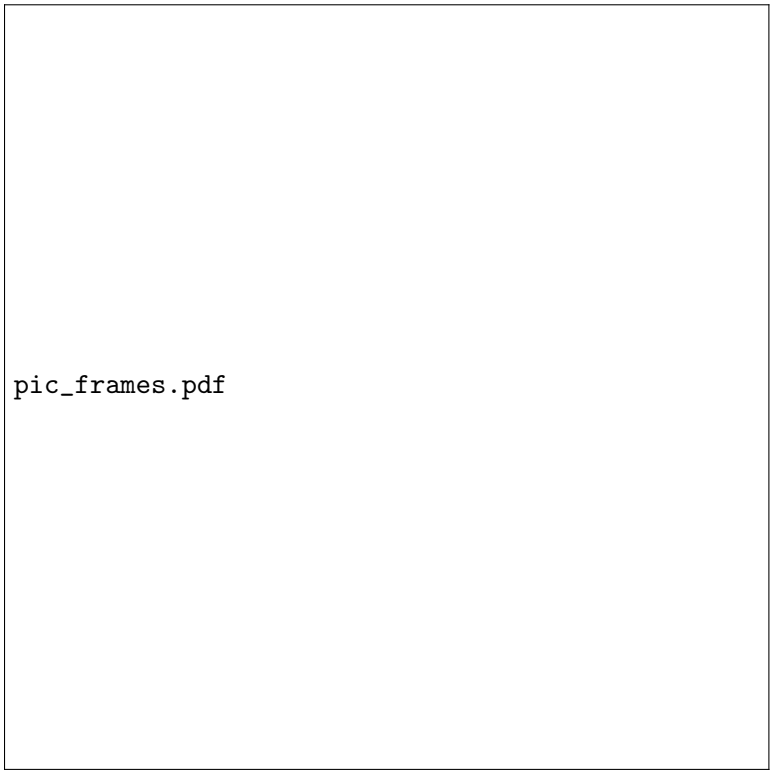

Figure I: Configuration of the used frames representing the manipulator

3 Direct Kinematics

After the frame assignment the next step is to model the direct kinematics for the manipulator. The direct kinematics refer to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters. In this case, for the D-H convention, we create a trasformation matrix for every frame, like in (**??**). It represents the transformation from one frame to the next (from $i-1$ to i).

$$ {}^{i}_{i-1}T = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i}c_{\alpha_{i-1}} & c_{\theta_i}c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -s_{\alpha_{i-1}}d_i \\ s_{\theta_i}s_{\alpha_{i-1}} & c_{\theta_i}s_{\alpha_{i-1}} & c_{\alpha-1} & c_{\alpha-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The angle θ is the input angle for the particular joint. We use the parameters of the D-H table for computing the matrices of all joints. These matrices contain information about the rotation and the position. When we have all the matrices, we have to multiply all the matrices from the global

frame to the frame of the end effector, like in (**??**).

$$ {}^{5}_{0}T = {}^{1}_{0}T \cdot {}^{2}_{1}T \cdot {}^{3}_{2}T \cdot {}^{4}_{3}T \cdot {}^{5}_{4}T \tag{2} $$

The resulting matrix describes the transformation from the global frame (index 0) to the one of the end-effector (index 5). From that matrix we can retrieve all the information about the end-effector, like position, rotation and orientation.

4 Inverse Kinematics

The goal of the inverse kinematics is to computate all the possible angles values of the joints for a given position and orientation of the end-effector. The number of solutions for each position and orientation depends on the physical constraints of the manipulator.

The chosen method for modeling the inverse kinematics of the ROB3/TR5 manipulator is a combination of the algebriac and geometric method. We retrieve the inverse matrix of the first level - the matrix between the global frame and the first frame - multiplying the standard form of the transformation matrix for the first level with the global matrix, like in (**??**). From that matrix we can retrieve useful data for our computations.

$$ [{}^{1}_{0}T]^{-1} \cdot {}^{5}_{0}T = \begin{bmatrix} c_1 r_{11} + s_1 r_{21} & c_1 r_{12} + s_1 r_{22} & c_1 r_{13} + s_1 r_{23} & c_1 p_x + s_1 p_y \\ c_1 r_{21} - s_1 r_{11} & c_1 r_{22} - s_1 r_{12} & c_1 r_{23} - s_1 r_{13} & c_1 p_y - s_1 p_x \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

In our construction of the inverse kinematics we pass in input six values, that are the three coordinates for the position, and the three values for the orientation, α, β and γ. It is possibile to see that we pass six values, but we have only 5 degrees of freedom. So one of these values is useless. For our construction we assume that the value that we don't need is the value of α, because we retrieve α directly from the values of the x coordinate and the y coordinate, using (**??**).

Using the convention of the *Z-Y-Z* Euler angles we can easily compute the values of the angles. For the first joint it's a simple operation, shown in (**??**)

$$ \theta_1 = atan\left(\frac{p_y}{p_x}\right) \tag{4} $$

For the angle of the fifth joint the situation is similar, but we use the data from the matrix previously described in (**??**). In (**??**) we can see two values,

called s_5 and c_5. These two values are the result of the equation in position $(2,1)$ and $(2,2)$ of the matrix, and for equation (**??**) it's the same. The only difference are the equations in the matrix, that in these case are in (3,3) and (1,3).

$$\theta_5 = atan2\left(s_5, c_5\right) \tag{5}$$

$$\theta_{234} = \theta_2 + \theta_3 + \theta_4 = atan2\left(s_{234}, c_{234}\right) \tag{6}$$

Because of the choice of the *Z-Y-Z* convention we can simply retrieve the values for θ_{234} and θ_5 from the orientation angles of the end-effector β and γ (compare equations (**??**) to (**??**)).

We decided to retrieve θ_1 from the x and y coordinates. If we had not chosen this, we could retrieve the value for θ_1 from the α value of the orientation of the end-effector, because of the convention *Z-Y-Z*.

Now it's clear that θ_{234} is not the angle for one specific joint, but the sum of the angles of the second, third and fourth joint of the manipulator. We can consider the isolated three joints as a planar structure because they all exclusively turn around the same axis. This trick allows us to find the particular three angles using the geometric method.

Now we have to split this "compounded angle" to retrieve the second, third and fourth angle of the manipulator. In first instance we retrieve the position of the fourth joint. In doing that we inspect figure (**??**). The result of the inspection is described by the following equations (**??**), (**??**) and (**??**).

$$\Delta x_{24} = p_x - d_5 \cdot \sin(\theta_{234}) \cdot \cos(\theta_1) \tag{7}$$

$$\Delta y_{24} = p_y - d_5 \cdot \sin(\theta_{234}) \cdot \sin(\theta_1) \tag{8}$$

$$\Delta z_{24} = p_z - d_5 \cdot \cos(\theta_{234}) - d_1 \tag{9}$$

From the results of these computations and the cosine law represented in (**??**), we can retrieve the value of θ_3 (see equation (**??**)).

$$\left(\Delta x_{24}\right)^2 + \left(\Delta y_{24}\right)^2 + \left(\Delta z_{24}\right)^2 = a_2^2 + a_3^2 - 2 \cdot a_2 \cdot a_3 \cdot \cos(\pi - \theta_3) \tag{10}$$

$$\theta_3 = acos\left(\frac{\left(\Delta x_{24}\right)^2 + \left(\Delta y_{24}\right)^2 + \left(\Delta z_{24}\right)^2 - a_2^2 - a_3^2}{2 \cdot a_2 \cdot a_3}\right) \tag{11}$$

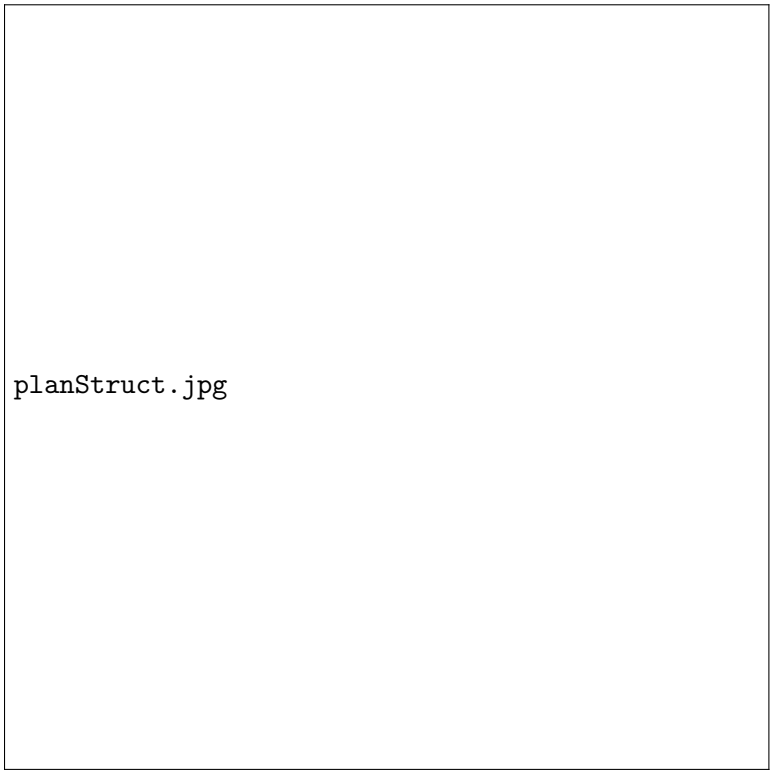

Figure II: Planar structure formed by the 2nd, 3rd and 4th joint

Equation (**??**) has two solutions: $0 \leq \theta_3 \leq \pi$ and $-\pi \leq \theta_3 \leq 0$, but only the first is feasible, because of the physical restrictions of the manipulator. For calculating θ_2 we first calculate the two angles ϵ and δ in figure (**??**). By inspecting the geometry and again using the cosine law we retrieve equations (**??**) and (**??**). Finally, θ_2 results of these two angles (compare equation (**??**)).

$$\epsilon = atan2\left(\Delta z_{24}, \sqrt{(\Delta x_{24})^2 + (\Delta y_{24})^2}\right) \tag{12}$$

$$cos\left(\delta\right) = \frac{(\Delta x_{24})^2 + (\Delta y_{24})^2 + (\Delta z_{24})^2 + a_2^2 - a_3^2}{2 \cdot a_2 \cdot \sqrt{(\Delta x_{24})^2 + (\Delta y_{24})^2 + (\Delta z_{24})^2}} \tag{13}$$

$$\theta_2 = \begin{cases} \delta - \epsilon & , \theta_3 < 0 \\ -\delta - \epsilon & , \theta_3 \geq 0 \end{cases} \tag{14}$$

Now it's trivial to compute the last angle that we need, θ_4, using the equation in (**??**)

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \tag{15}$$

We can easily discover that in the inverse kinematics we have 4 different solution. This is simple to understand, because we can compute θ_1 using the same formula that we used in (**??**), but we can change the sign of the coordinates, like in (**??**).

$$\theta_1 = atan2(-p_y, -p_x) = atan\left(\frac{p_y}{p_x}\right) - \pi \tag{16}$$

The result now it will be the same result of the other operation but traslated of 180°, so in a different quadrant. We therefore conclude that for this type of manipulator, for each value of θ_1 there are two possibilities for θ_2, thus pertaining 4 theoretical solutions possible for this type of manipulator. To find this solutions, in the case of the ROB3/TR5 manipulator, we can operate adding and subtracting 180° in the right angles. We don't do that computations because all of these solution are unfeasible for the manipulator, due to physical restriction.

5 Test

5.1 Direct Kinematics

Angles					Position			Orientation		
θ_1	θ_2	θ_3	θ_4	θ_5	X	Y	Z	α	β	γ
0	0	0	0	0	330	0	405	0	0	0
45	45	-45	-45	45	126.92	126.92	225.50	-135	45	-135
-90	0	90	0	0	0	-330	145	-90	90	0
-90	0	90	180	90	0	-70	145	90	90	-90
-90	45	45	90	45	0	-141.42	-126.42	0	180	0
30	-30	45	60	180	367.49	212.17	375	30	75	180

Table 2: Tests using the `DirectKinematics` function

5.2 Inverse Kinematics

Position	Orientation	Angles	
		v_1	v_2
330 0 405	0 0	0	0
		0	0
		0	0
		0	0
		0	0

367.4942 212.1729 375	75 180	0 30 -30 45 180	0 30 4.9574 -45 180
0 -70 145	90 -90	-90 24.8028 134.4270 -69.2298 -90	-90 105.6469 -134.4270 118.7801 -90
126.9239 126.9239 225.5025	45 -135	45 17.9647 124.2034 -97.1681 -135	45 98.5005 -124.2034 70.7029 -135
89 200 400	0 0	66.0109 -34.4827 99.9153 -65.4326 0	66.0109 37.0996 -99.9153 62.8157 0

Table 3:Tests using the `InverseKinematics` function

6 Conclusion

All the functions has been implemented using MATLAB. The results obtained are quite satisfactory to validate the suitability of the theoretical kinematics models to the ROB3/TR5 serial manipulator. All the functions seems to work correctly. The results of the functions, besides being consistent, seems correct and satisfactory for both functions; the position and orientation of the end-effector was always computed correctly, as far as measurement and calibration precision allowed. As for the inverse kinematics, it seems to obtain all possible solutions for any given input. When there is no solution available it detects so, but the most common case is that two possible mathematical solutions are available. In practice, the manipulator can only perform one of them due to its physical constraints. It seems like there are some errors that arise in converting degree values into the $[0 \ldots 255]$ interval values, and about the normal usury of the robot, which can distort the teoric values from the observed movement of the robot.

A precise calibration system could avoid this problem and improve the quality of the results.

A Manual

A.1 Use of the Direct Kinematics

The function that computes the direct kinematicts is called `DirectKinematics`. The signature of the function is as follows.

```
[ Position, Orientation ] = DirectKinematics( theta )
```

The input value *theta* is an array that contains the 5 angles of the joints (except for the end-effector). The output values are two arrays, *Position* and *Orientation*, that contain respectively the position (x-, y- and z-coordinates in the global frame) and the orientation (angles around x-,y- and z-axis of the global frame) of the of the end-effector.

A.2 Use of the Inverse Kinematics

The function that computes the inverse kinematics is called `InverseKinematics`. The signature of the function is as follows.

```
[ Angles ] = InverseKinematics( Position, Orientation )
```

The input values, *Position* and *Orientation*, are two arrays that contain the position and the orientation that we want to positioning the end-effector. Precisely the *Orientation* array contains only two values, the angles β and γ, for the reason explained in (**??**). The output value is an array called *Angles* that contains the solutions of the 5 angles for the joints for moving the end-effector in the desired position and orientation. It is possible that not all the solutions are feasible. For check if a solution is feasible, use the function described in (**??**).

A.3 Use of the function for check the angles

The function that checks if an array of angles is feasible or not is called `transform_ang_to_rob_val`.

```
[ rob_val ] = transform_ang_to_rob_val( theta )
```

The input value is an array called *theta* that contains the five angles for the joints. The function checks if these values are feasible for the robot, and translate the values in an integer included in the interval $[0 \dots 255]$. The output value it's called *rob_val*, and will be an array with the values ready for send to the robot, or an error message if the solution passed in input is not feasible.

References

[1] Denavit J. Hartenberg R.S. (1955), "A kinematic notation for lower-pair mechanisms based on matrices", *Trans ASME J. Appl.*

Consulted Sites

[2] Denavit-Hartenberg convention, `http://en.wikipedia.org/wiki/Denavit-Hartenberg_parameters`