5-DOF Robot Arm Analysis

Panagiotis Kotsias

pcko@kth.se

June 28, 2018

Abstract

In this paper, a detailed kinematics analysis of a five degrees-of-freedom medical robotic arm is demonstrated. The forward kinematics are derived using the method proposed by Denavit and Hartenberg and a constrained nonlinear optimization algorithm is used to solve the inverse kinematics for a desired trajectory, given the joint limits. Finally, a cubic-spline interpolation in joint space is implemented, to satisfy the requirements of a stop-and-go spiral trajectory execution. This work comes along with $MATLAB$ code that supports its claims.

1 Introduction

Robotics has been introduced to a variety of fields, including medical applications, where the Da Vinci robot is probably the most notable example. Given the complexity and severity of surgical conditions, the analysis of such automated systems is critical and still undergoes heavy research.

2 Analysis

2.1 Forward Kinematics

The forward kinematics equation T describes the function that maps points in the joint space to points in the Cartesian space. To derive the forward kinematics function, the frame convention that was proposed by Denavit and Hartenberg is followed, as shown in Figure 1. All x-axes are shown in red, whereas all z-axes are in blue color. A circle with a dot inside denotes a vector pointing towards the reader. Frame O_8 is considered the reference frame or global frame for the rest of this report.

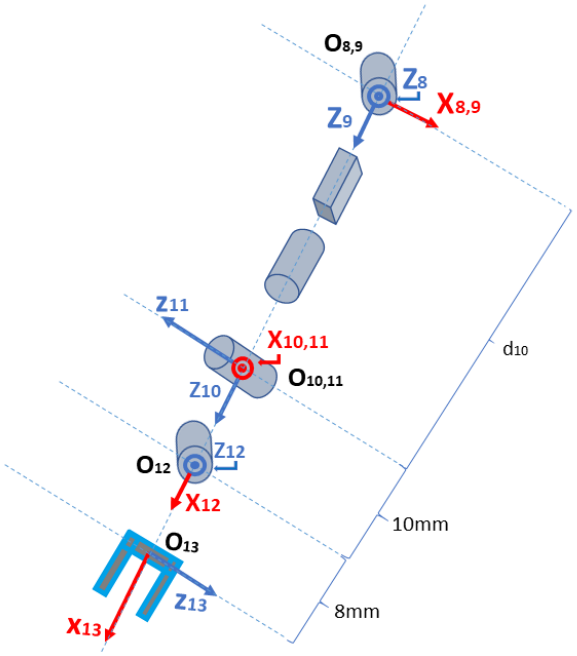

Figure 1: Coordinate frames according to the DH convention.

The parameters according to the convention are:

i	θ_i	d_i	a_i	α_i
9	${\theta_9}^\star$	0	0	$+90^o$
10	90^o	${d_{10}}^\star$	0	0
11	${\theta_{11}}^\star$	0	0	-90^o
12	${\theta_{12}}^\star$	0	10mm	-90^o
13	${\theta_{13}}^\star$	0	8mm	-90^o

Table 1: DH Parameters.

where $\star$ denotes the variable of each joint.

Since there are 6 different frames in the problem, there shall be 5 transformations between subsequent frames, each one describing the pose of a frame with respect to its directly previous one, forming the kinematic chain from the reference frame O_8 until the tool frame O_{13}. These transformations are denoted by A_i and are calculated

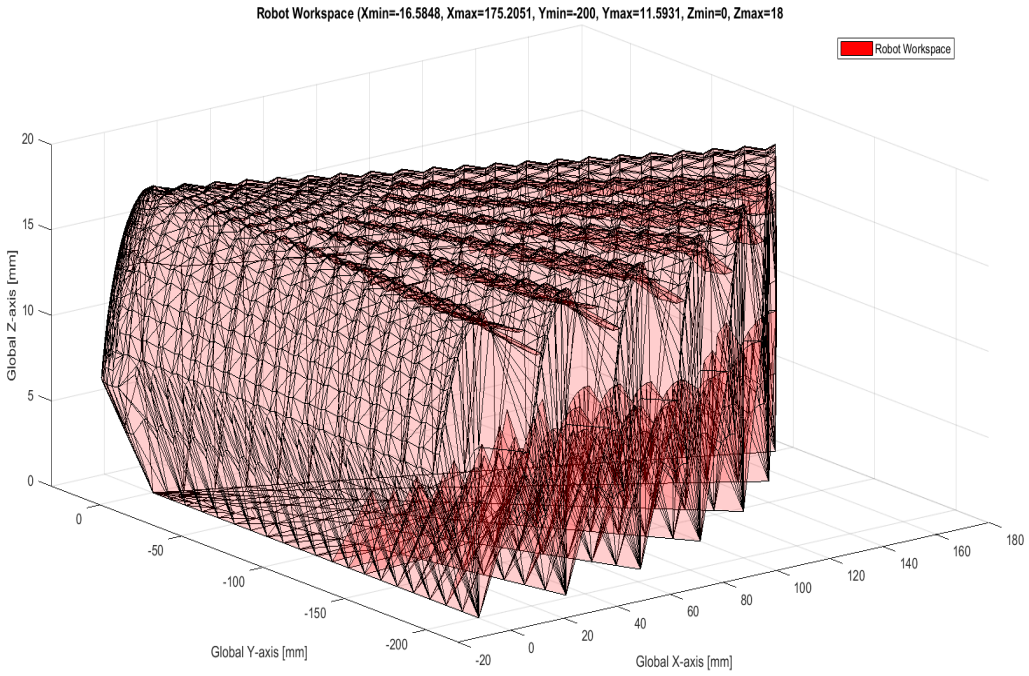

Figure 2: Robot Workspace.

using the formula:

$$\left[A_i(q_i)\right] = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

by simply substituting the DH parameters of the manipulator, where the components q_i of the 5×1 joint vector $\vec{q}$ are either θ_i or d_i depending on whether joint $i-1$ is revolute or prismatic, respectively.

The product $T_i = T_{i-1} \cdot A_i$ for $i \in [9, 13]$ describes the pose of frame O_i with respect to reference frame O_8, thus T_{13} describes the pose of the tool with respect to the reference frame, which is the solution to the forward kinematics problem.

All A_i and T_i matrices are calculated symbolically in the provided $MATLAB$ file under "Question 1", yet they are too complex to be analytically reported on paper.

2.2 Robot Workspace

According to the robot's specifications, the joint limits in degrees and millimeters are:

$$\vec{0} \leq \vec{q} \leq \vec{q}_{max} \tag{1}$$

where $\vec{q}_{max} = [60, 200, 60, 45, 30]^T$.

The workspace of the robot is calculated using a simple iterative procedure of nested loops, where each loop variable changes with a step of 10 degrees or 10 millimeters, depending on it being revolute or prismatic respectively, starting from 0 until it reaches its limit $q_{max,i}$. All these configurations of 5 values are fed to the forward kinematics function that was calculated in Section 2.1, which outputs the transformation matrix that includes the information about the Cartesian coordinates of the tool. In order to have a complete representation of the workspace, the Cartesian coordinates of all joints are calculated in each step as well, using the T_i matrices that were computed in Section 2.1. Finally, the complete workspace of the robot is shown in Figure 2.

2.3 Inverse Kinematics

There are various methods to approach the solution of the nonlinear Inverse Kinematics problem in related robotics literature. First of all, the analytical solution is the one that performs the fastest when solving for a big number of points. Its main disadvantage though is that it is non-trivial, especially if the axes of actuation of the last three degrees of freedom do not coincide, as it is the case in this problem. Another very common approach is utilizing the Jacobian matrix, which essentially is a linearization of the problem around the current configuration. This is an approximate numerical solution, which is very prune to the distance between subsequent setpoints. However, the aforementioned technique, does not, in its basic form, include any way to deal with constraints, such as the joint limits. Therefore, a third approach is chosen in this work, which relies on a different formulation of the task at hand and solves the inverse kinematics problem for a desired tool position in the Cartesian space. Let us assume that the current joint values form a 5×1 vector $\vec{q}$ and the position of the tool in the three dimensional space is given by a 3×1 vector $\vec{p}(\vec{q})$. An arbitrary setpoint with respect to the global frame is denoted by the 3×1 vector $\hat{\vec{p}}$. There are numerous ways to describe the difference between the setpoint and the current configuration, yet the author chose the Euclidean distance, which is thereafter proven to be good enough for the purpose. Thus, the difference between the two vectors is defined by the following scalar *error function* (or *cost function* or *objective function*):

$$e(\vec{q}) = \sqrt{\sum_{i=1}^{3} \left(\hat{p}_i - p_i(\vec{q})\right)^2} \tag{2}$$

Hence, the problem is reformulated into a constrained nonlinear optimization problem of minimizing the error function given the joint constraints, or:

$$\begin{aligned} \underset{\vec{q}}{\text{minimize}} \quad & e(\vec{q}) \\ \text{subject to} \quad & 0 \leq q_i \leq q_{max,i},\ i = 1, \ldots, 5. \end{aligned} \tag{3}$$

The solution $\hat{\vec{q}}$ to eq. (3) approximates the solution of the inverse kinematics problem:

$$\hat{\vec{q}} = \underset{0 \leq \vec{q} \leq \vec{q}_{max}}{\operatorname{argmin}} \ e(\vec{q}) \tag{4}$$

The cost function of eq. (2) is included in the *costfun.m* file and the solution described by eq. (4) is reached by utilizing the solver *fmincon*, which is included in MATLAB, for every single setpoint $\hat{\vec{p}}$ that is fed to it. That solver requires an initial "guess" of the solution, which directly affects its runtime duration and its convergence. Given that in the current problem, the tool is required to follow a continuous trajectory without sudden "jumps", a "good" initial guess would be the solution that was found in the previous step, since it should be close to the desired setpoint in the Cartesian space and, hopefully, in the joint space *if* the configuration is away from singularities. As it is evident, the first point of the trajectory has no previous solution to use as an initial guess, hence the "center" of the joint space is inserted once as a conservative initial guess.

Given a desired trajectory $f(x(t), y(t), z(t))$ that is continuous in the 3d space and discretized by n points for the robot to follow, these initial guess conditions can be summarized as:

$$\hat{\vec{q}}_{guess,init}(k) = \begin{cases} \frac{\vec{q}_{max}}{2}, & k = 1 \\ \hat{\vec{q}}(k-1), & k = 2, 3, \ldots, n \end{cases} \tag{5}$$

where $\hat{\vec{q}}(k-1)$ is the 5×1 solution to eq. (3) for the previous setpoint.

The solution to the constrained inverse kinematics problem of eq. (3) is summarized by the combination of eqs. (4) and (5) for a given continuous trajectory that is discretized by n setpoints.

2.4 Spiral Trajectory

In order to test the performance of the proposed solution to the inverse kinematics problem, a spiral trajectory inside the workspace is generated by the following equations:

$$x_s(t) = 75 + 50\cos(2\pi t) \tag{6a}$$

$$y_s(t) = -100 + 50\sin(2\pi t) \tag{6b}$$

$$z_s(t) = 13 + 0.2\pi t \tag{6c}$$

$$t \in [0, 2] \tag{6d}$$

which is shown in Figure 3. This trajectory is discretized by sampling $n = 2 \cdot 76 + 1 = 145$ points which are fed to the inverse kinematics solver to calculate the joint values, see Figure 4. The joint limits, which are represented by red lines, are never violated.

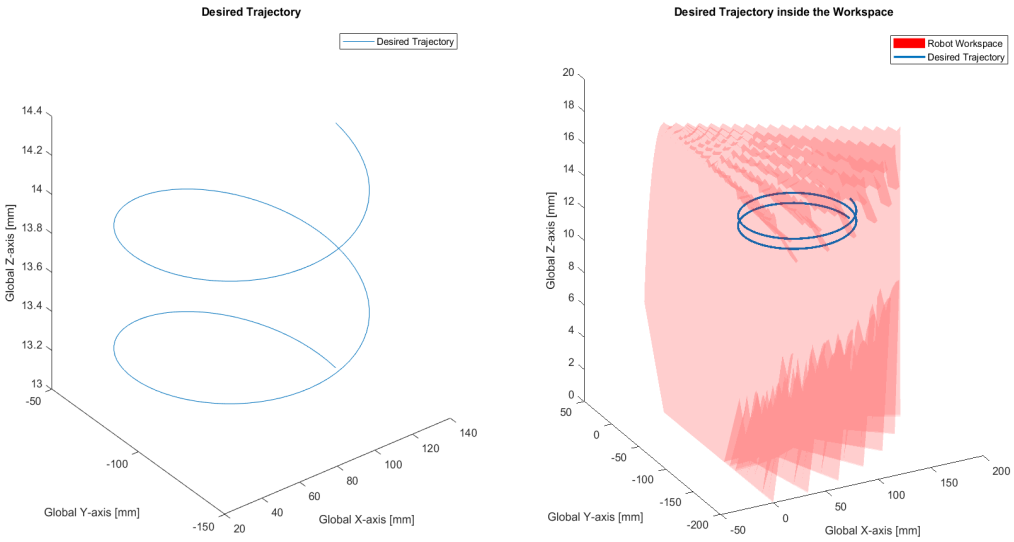

Figure 3: Desired Spiral Trajectory.

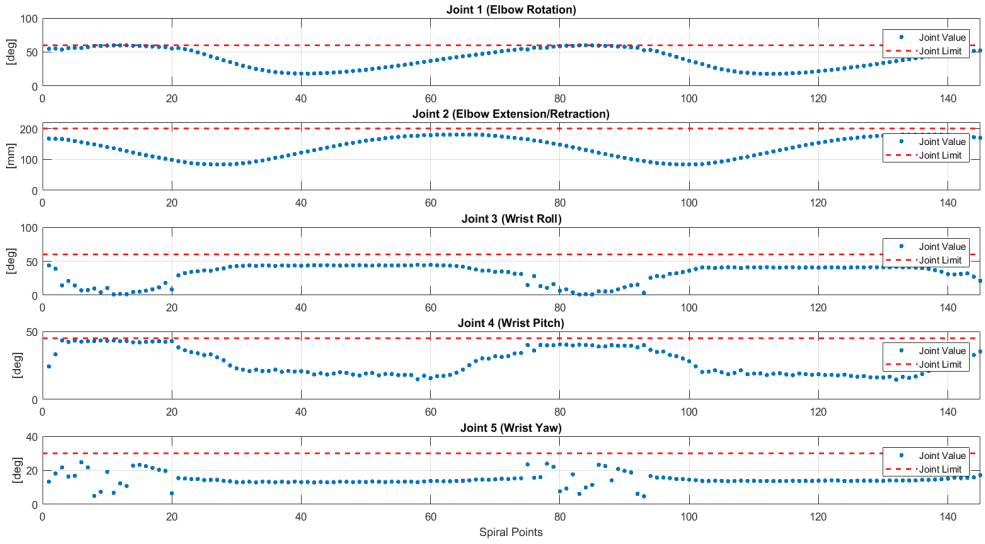

Figure 4: Inverse Kinematics Solution for All Joints.

To evaluate the solution, the calculated joint values are fed to the forward kinematics function of Section 2.1 that outputs the transformation that includes the Cartesian coordinates of the tool in the three-dimensional space. These configurations are thereafter plotted against the desired path, as shown in Figure 5.

2.5 Interpolation in Joint Space

Solving the inverse kinematics problem for a high number of points can be computationally expensive, thus an interpolation of points in between calculated configurations is required. For the purposes of this problem, a simple cubic splines interpolation for the points of each joint has been used. This method allows for satisfying various trajectory constraints, such as velocity or acceleration specifica-

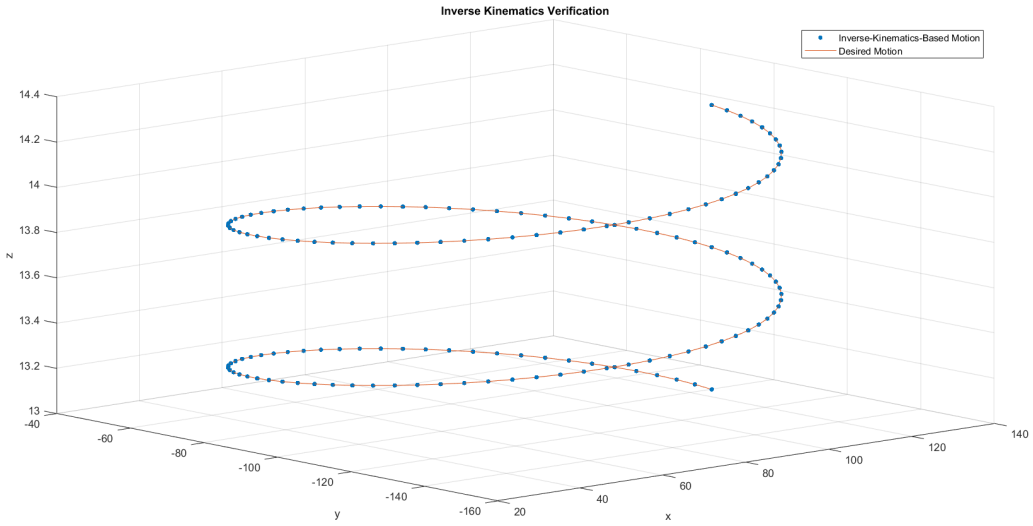

Figure 5: Inverse Kinematics Solution.

tions, which in this case are to have zero velocity of joints at every solution of the inverse kinematics problem; a *stop-and-go* trajectory.

For any trajectory $q_i(t)$ of the 5 joints, let us assume two subsequent solutions of the inverse kinematics problem:

$$q_i(t_0) = q_{i,0} \tag{7a}$$
$$q_i(t_f) = q_{i,f} \tag{7b}$$

that belong to it. If $q_i(t)$ is a cubic spline, then it is of the form:

$$q_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{8}$$

Thus, the velocity profile is given as:

$$\dot{q}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{9}$$

The stop-and-go requirements dictate that:

$$\dot{q}_i(t_0) = \dot{q}_i(t_f) = 0 \tag{10}$$

By inserting eq. (7) and eq. (10) into eqs. (8) and (9), a linear system of 4 equations with 4 unknowns is obtained:

$$q_{i,0} = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \tag{11a}$$
$$0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 \tag{11b}$$
$$q_{i,f} = a_f + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \tag{11c}$$
$$0 = a_1 + 2a_2 t_f + 3a_3 t_f^2 \tag{11d}$$

Or in matrix form:

$$\underbrace{\begin{bmatrix} q_{i,0} \\ 0 \\ q_{i,f} \\ 0 \end{bmatrix}}_{\vec{q}_{i,0f}} = \underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix}}_{\mathrm{M}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}}_{\vec{a}_i}$$

Therefore, the coefficients for each spline are calculated as:

$$\vec{a}_i = M^{-1} \vec{q}_{i,0f} \tag{12}$$

A total of $m = 5$ points are chosen to be plotted for each spline, the first and last of which belong to the inverse kinematics solution and are used in eq. (12) in order to calculate the polynomial coefficients. Therefore, 3 *new* points represent the calculated cubic interpolation between 2 subsequent solutions. All the points, including the solution of the inverse kinematics problem and the cubic interpolation for each joint, are shown in Figure 6.

Finally, by utilizing the forward kinematics function, the resulting tool trajectory based on the points of Figure 6, is shown in Figure 7.

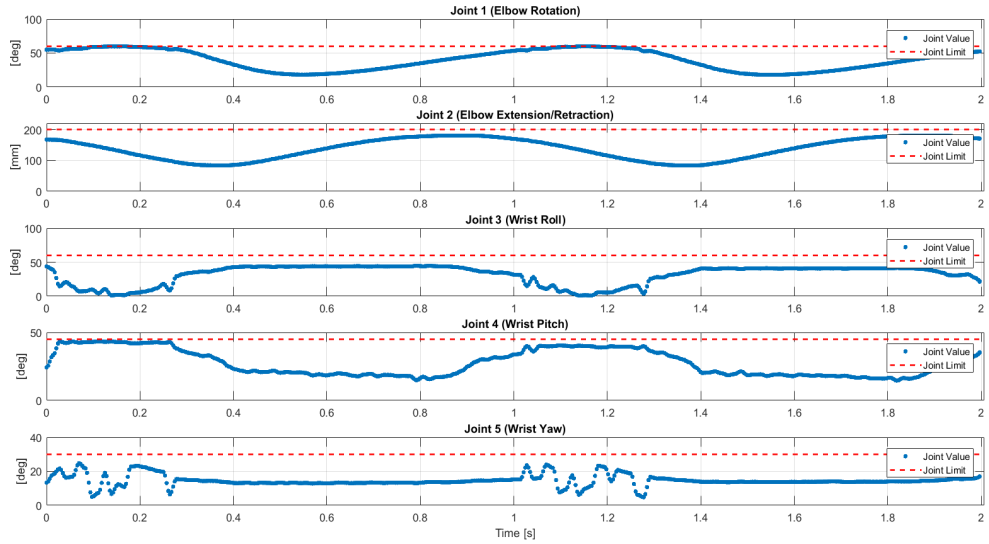

Figure 6: Cubic Splines Interpolation in Joint Trajectories.

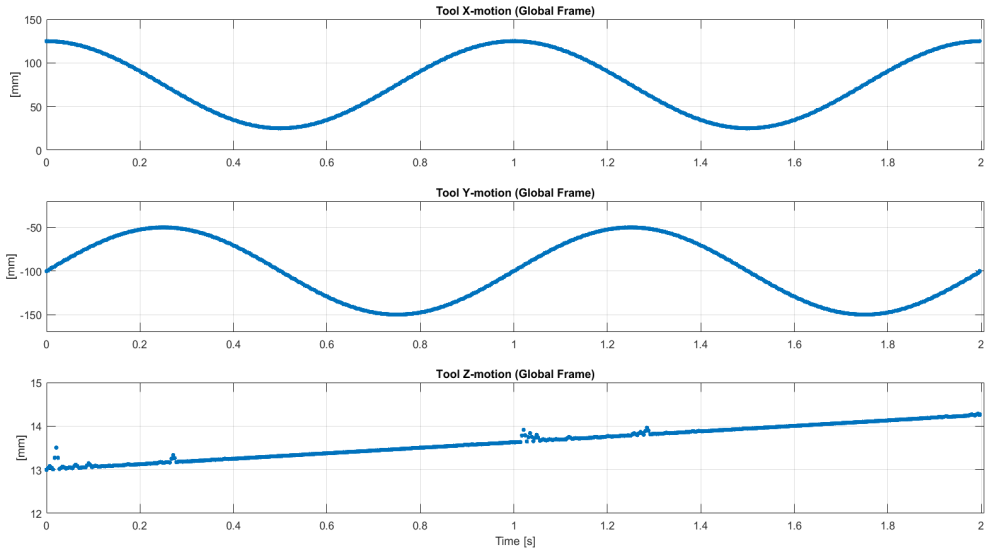

Figure 7: Resulting Tool Trajectory.

3 Conclusions

Solving the inverse kinematics for a manipulator analytically, where the last three joint axes do not coincide, may be considerably demanding. The implemented solution to the optimization problem demonstrated that there can be alternative approaches that satisfy physical constraints, such as joint limits. The cost function can be extended to include the difference in orientation between configurations, so that the inverse kinematics problem is solved for all 6 degrees of freedom in the Cartesian space. The resulting tool trajectory, shown in Figure 7 complies with the desired trajectory of eq. (6), as it was verified by Figure 5. There are some minor "jumps" in the z-direction in Figure 7 but this can be further improved by experimenting with the parameters at hand, such as the number of points n in the discretization of the desired path, a different formulation of the cost function or even a different nonlinear solver.