Modern Robotics Practical

Per Arne Kjelsvik, *p.a.kjelsvik@student.utwente.nl, s2049201, Erasmus exchange, M-EE*
Frans Skarman, *f.j.b.skarman@student.utwente.nl, s1982001, Erasmus exchange, M-CS*

I. Introduction

THE goal of this practical task was to implement a working position control for a manipulator arm. We believe this practical was intended to make the students visualize and also understand what they're actually doing when implementing geometric (coordinate-free) control. With the practical, you can understand how unit twists are defined, how the Jacobian of an end-effector will be in relation to the reference frame and configuration chosen, and understand how the kinematics of angle-dependent joints work. In this report a short breakdown of the practical is given, firstly in how the system was modeled and represented. Secondly the control law chosen and the implementation of the full system, and lastly a short discussion of the results of the practical.

II. Forward kinematics

The first step in solving the forward kinematics problem was to chose a reference configuration of the robot. We chose a reference configuration with all 3 limbs of the arm pointing straight up along the Z-axis as detailed in figure 1 This configuration was chosen as it makes selection of the initial H-matrices and unit twists easy.

As the initial pose contains no rotations ($q_i = 0$), the H matrices were chosen as

$$H_n^{n-1}(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where L_n is the length of arm n.

Since the joint at the base is a pure rotation along the z-axis through the origin of Ψ_0 and the two other joints rotate along the x-axis through the center of their frames, the unit twists were chosen as

$$\hat{T}_1^{0,0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{T}_n^{n-1,n-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

With the reference configuration chosen, the resulting H-matrices of a given configuration q were calculated using joint relations formula as:

$$H_n^0(q) = H_{n-1}^0(q) \exp\left(\tilde{\hat{T}}_n^{(n-1),(n-1)} \ q_n\right) \cdot H_n^{n-1}(0)$$

It should be noted that this is not brockett's formula as we use twists expressed in the frame of the joint rather than Ψ_0.

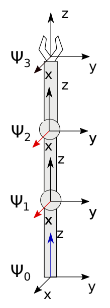

Fig. 1: Reference configuration of the arm

However, equation (1) shows that it does yield the same result as brockett's formula.

$$\begin{aligned} H_2^0(q) &= e^{\tilde{\hat{T}}_1^{0,0} q_0} H_1^0(0) e^{\tilde{\hat{T}}_2^{1,1} q_1} H_2^1(0) \\ &= e^{\tilde{\hat{T}}_1^{0,0} q_0} H_1^0(0) H_0^1(0) H_1^0(0) e^{\tilde{\hat{T}}_2^{1,1} q_1} H_0^1(0) H_1^0(0) H_2^1(0) \\ &= e^{\tilde{\hat{T}}_1^{0,0} q_0} e^{H_1^0(0) \tilde{\hat{T}}_2^{1,1} H_0^1(0) q_1} H_2^0(0) \\ &= e^{\tilde{\hat{T}}_1^{0,0} q_0} e^{\tilde{\hat{T}}_2^{0,1} q_1} H_2^0(0) \end{aligned} \tag{1}$$

III. Differential forward kinematics

The first step in calculating the desired velocity of the individual joints was to calculate the jacobian of each joint.

The jacobian is defined as

$$J = (T_1^{0,0}, T_2^{0,0}, T_3^{0,0})$$

Since the unit twist of the first joint is a twist from frame 1 to frame 0, no calculations were required for it. The rest of the twists in the jacobian are given by

$$T_n^{0,0} = Ad_{H_n^0} T_n^{n-1,n-1}$$

The desired velocity of the end-effector was calculated by $\dot{p}_{ee} = K_V \cdot (p_{sp} - p_{ee})$ where p_{sp} is the desired position, p_{ee} is the current position and K_V is a constant that can be used to control how quickly the arm tries to approach the reference position.

In order to make calculations easier, a fourth frame Ψ_4 was added which has the same position as Ψ_3, but with the same orientation as Ψ_0.

A second jacobian J' was then defined as $J' = Ad_{H_0^4} J(q)$ which directly maps changes in joint rotations $\dot{q}$ to a the twist $T_3^{4,0}$.

Since we don't care about the angles of the joints, we defined a third jacobian J_v which only consists of the velocity part of J'. This is a mapping between $\dot{q}$ and $v_3^{4,0}$

The desired value for $v_3^{4,0}$ is $\dot{p}_{ee}$ which means that a change to q which takes the robot closer to the desired location can be calculated as

$$\dot{q}_{set} = J_v^{\dagger} \cdot \dot{p}_{ee}$$

IV. Position control

The objective of the position control was to make the end-effector follow the setpoint as accurately as possible. A faster tracking can be achieved with a higher gain at the cost of large actuator inputs resulting in wear-and-tear and stuttering of the robot. So a trade-off can be made between fast tracking with not-well scaled actuator inputs, or a slower tracking of the reference with smoother actuator inputs. Initially no changes were made to the control law, which means that it is only a simple proportional controller regulating the error signal:

$$\dot{p}_{ee} := K_v \cdot e = K_v \cdot (p_{sp} - p_{ee}).$$

The controller is a linear feedback control where gain is applied to what you're controlling, proportional to the difference between the setpoint value and the measured feedback value. The proportional gain K_v was set to 20, but the output of the controller was saturated with an additional limiting factor as suggested by exercise 2 of the practical, which limits the input $\dot{p}_{ee}$ to an absolute value of 10 cms^{-1}. The threshold was set to 20 cms^{-1} when running on the robot, as it seemed to be a safe limit too. The controller is limited from setting as large inputs as it would like to, which can especially be seen in parts where the setpoint is accelerating. The start-up phase is an example, where the arm is lagging behind more than normal before reaching close to the setpoint. If the error of the system is constant, the controller will stop applying input, resulting in an offset error so arm position is offset from desired position. This is a problem when using proportional control only. The offset could be removed by adding integral action, and derivative action could help smooth the controller outputs. However, with the conservative saturation of 20 cms^{-1}, there was not a noticeable improvement with integral and/or derivative action, so the simpler proportional control was kept in the end.

V. Implementation

For efficiency of code, the whole practical is quite simple from a programmatic view, but an example where our code is efficient is when calculating H_0^4. Rather than defining H_4^0 and then taking the inverse, we defined H_0^4 directly, using the relations of homogeneous matrices rather than the inverse of a homogeneous matrix with the `MATLAB`-function `inv`. Of course, making a function that uses the knowledge of homogeneous matrices to invert properly could easily have been used instead. Also, for the project when implementing Brockett, the `MATLAB`-function `expm` was used to calculate the exponential of $\tilde{\hat{T}}_n^{n-1,n-1}$. Implementing a custom function that uses the mathematical properties we know of the exponential matrix would have been a lot more computationally effective for this part of the practical.

VI. Results

Figure 5 shows the movement and set point of the end-effector in the x-z plane. Figure 2 and figure 3 shows the difference between the setpoint and actual position of the end-effectors in the x- and z-planes.

Finally, figure 4 shows the authors with the robot.

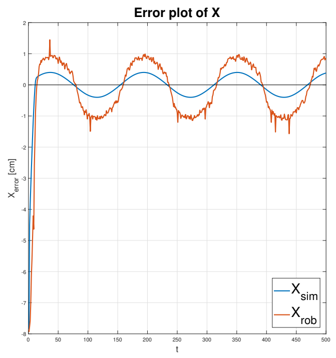

Fig. 2: plot showing the error in the x-axis over time of the simulated robot, shown in blue and the physical robot, shown in red

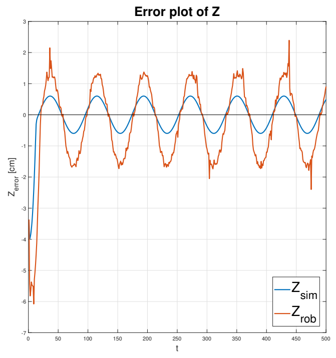

Fig. 3: plot showing the error in the z-axis over time of the simulated robot, shown in blue and the physical robot, shown in red

Fig. 4: Picture of the authors with the robot

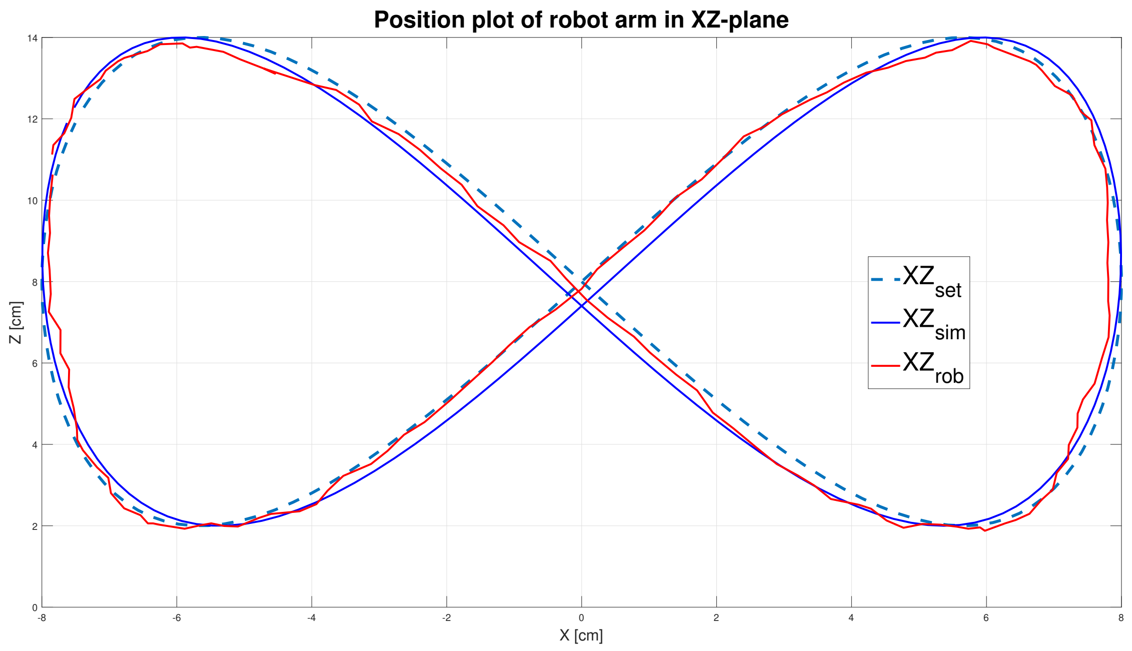

Fig. 5: Plot showing the movement of the robot arm in one cycle around a figure 8. The dashed line is the desired position, the blue line is the simulated position and the red line is the actual position