TÉCNICO LISBOA

Robotics

Winter 2022

Departamento de Engenharia Electrotécnica e de Computadores

1st lab assignment - v1.0 - Last updated 16/11/2022

Introduction to Serial Manipulators

(To be handed – by e-mail – no later than 30 December 2022, 23:59:59)

João S. Sequeira
joao.silva.sequeira@tecnico.ulisboa.pt

1 Syllabus

The aim of this lab assignment is to demonstrate the importance of kinematic models in the execution of a typical task by the serial manipulator with 5 rotational degrees-of-freedom (dof), Scorbot ER-7, shown in figure 1.

Figure 1: The 5-dof, plus gripper, serial manipulator Scorbot ER-7

From a mathematical perspective, this serial manipulator maps the set of 5 joint angles, $\theta_1, \ldots, \theta_5$, into the gripper's 3 position coordinates, x, y, z, and 3 orientation coordinates, α, β, γ. This map is called **direct kinematics** or **forward kinematics**.

Given that only 5 dof are available it is not possible to put the end-effector in arbitrary x, y, z position and arbitrary α, β, γ orientation. Which is more important, position or orientation, depends on the specific task the manipulator has to execute.

The inverse of the **direct kinematics** is called **inverse kinematics** or **backwards kinematics**. This map returns sets of joint angles that correspond to a given position and orientation.

All the geometric information, necessary for the development of kinematics models is contained in the physical dimensions shown in figure 2.

The explicit computation of both the direct and inverse kinematics is a topic to be covered in the theory classes. However, it must be emphasized that it relies, exclusively, in the careful definition of reference frames (i.e., coordinate systems where to measure the joint angles, i.e., the dof of the manipulator) and on the computation of homogeneous transformations between consecutive reference frames. Also to remark is that this is a topic that was already been discussed in previous Linear Algebra courses.

Note that this lab statement does not require explicitly that either the direct or inverse kinematics has to be computed. However, depending on the solution strategy chosen by each group, it may be useful to compute those functions.

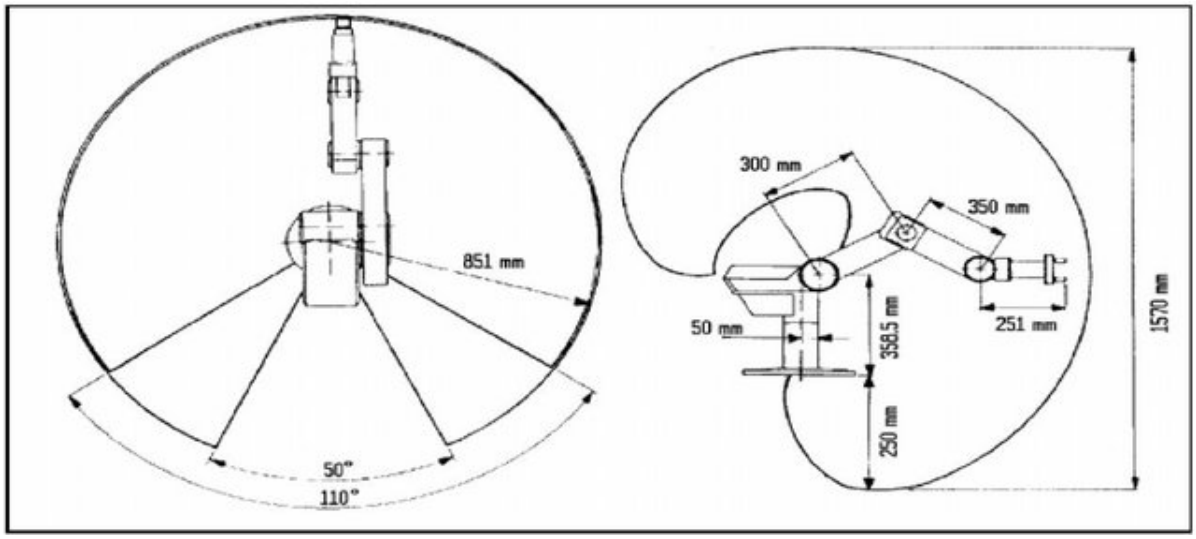

Figure 2: Scorbot physical dimensions.

A user manual for the robot is available at the course webpage. Also check https://tinyurl.com/28o9klb9 for useful suggestions on how to program the robot.

2 Task

The task proposed amounts to have the robot drawing over a flat paper surface. A board marker will be used to draw (thick writing point). The marker will be grabbed with the gripper of the robot. The grabbing force is not high and hence the manipulator must be operate at low speeds so that the marker stays in place.

Given an initial position over the paper surface, the robot must execute the drawing while keeping the marker in contact with the paper surface at all times.

The drawing will be a black-white image given in the form of a file (png, or jpeg formats).

1. Input the file with the drawing image
2. From the black-white image given select a set of adequate points the marker will have to touch.
3. Have the robot touching the drawing paper surface at the selected via points
4. Make the marker (and the robot , of course) following a path containing the via points such that, in the end, the black-white image in the file is "transferred" to the drawing surface

The drawing paper surface will be mounted over a compliant, synthetic foam/plastic based, material.

3 Connecting to the robot

The robot communicates with a computer through a serial RS232 line, or through USB (via a USB-to-serial converter). You can operate the robot through a teach pendant device or by sending motion commands to the serial line.

The robot is commanded by an instruction set called ACL (information widely available on the web and in the course webpage). Sending commands to the robot is simply to send the ASCII characters of the command to the robot, via the serial line. To receive information from the robot simply listen for ASCII characters in the serial line.

In principle, you can create your own program in any language of your preference, e.g., Python, provided it can access the serial line so that it can send commands to the robot and receive information from it. The course webpage will provide example scripts (in Python 3.8) showing how to send/listen the serial line.

The robot can be operated either in joint space coordinates, e.g., the joint angles, or directly in workspace coordinates, e.g., $x, y, z, \alpha, \beta, \gamma$. There are sets of commands to move each joint and read their current values. These commands are part of the ACL language.

At the beginning of a working session it is advisable to execute the HOME command in order to calibrate the robot (this is an operation that may take up to 2 minutes to complete).

4 Expected outcome

- Python^{TM} (or whatever language is used) scripts. Code shall be handed, by e-mail, ready for automated demonstration, no later than 30 December 2022, 23:59:59.
- A report, in pdf format, detailing all the steps and assumptions taken, and how to operate the software developed, must be handed also by 30 December 2022, 23:59:59. The report must have no more than 8 A4 pages (strict).

5 Evaluation guidelines

The following points will be checked to determine the grading of the work.

- Similarity between the original (source) drawing and the one produced by the robot.
- Existence/absence of continuity in the drawing lines.
- Execution time.
- Evidence of excessive pressure between the tip of the marker and the paper sheet.

A Python based example to send commands to the robot

```
# Robotics 22-23, Lab1 serial communications with the Scorbot example
```

```
import serial
import time
import datetime

# This function listens the serial port for wait_time seconds
# waiting for ASCII characters to be sent by the robot
# It returns the string of characters
def read_and_wait(ser, wait_time):
    output = ""
    flag = True
    start_time = time.time()

    while flag:
        # Wait until there is data waiting in the serial buffer
        if ser.in_waiting > 0:
            # Read data out of the buffer until a carriage return / new line is found
            serString = ser.readline()

            # Print the contents of the serial data
            try:
                output = serString.decode("Ascii")
                print(serString.decode("Ascii"))
            except:
                pass
        else:
            deltat = time.time() - start_time
            if deltat>wait_time:
                flag = False

    return output


def main():
    print("Starting")

    # Open the serial port COM4 to communicate with the robot (you may have to adjust
    # this instruction in case you are using a Linux OS)
    # (you must check in your computer which ports are available are, if necessary,
    # replace COM4 with the adequate COM)
    ser = serial.Serial('COM4', baudrate=9600, bytesize=8, timeout=2, parity='N', xonxoff=0, stopbit
    print("COM port in use:  {0}".format(ser.name))

    print("Homing the robot (if necessary)")
    #ser.write(b'home\r')
    #time.sleep(180)  # homing takes a few minutes ...

    serString = ""  # Used to hold data coming over UART


    ##########################################################################
    # ATTENTION: Each point used was previously recorded with DEFP instruction
    #(from a terminal console - you can use, for example, putty or hyperterminal
```

```
    # as terminal console)
    ##############################################################################

    print('going to point P1')
    ser.write(b'MOVE P1\r')
    time.sleep(0.5)
    read_and_wait(ser,2)

    print('going to point P2')
    ser.write(b'MOVE P2\r')
    time.sleep(0.5)
    read_and_wait(ser,2)

    # closing and housekeeping
    ser.close()

    print('housekeeping completed - exiting')


##############################################################################
if __name__ == "__main__":
    main()
```