RRT-based motion planning for a SCARA robot (Project 1)

ROBT615 Optimal Control and Planning / ROBT703 Advanced Optimal Control and Planning
Nazarbayev University

Dr. Ton Duc Do

1 System and task description

The objective is to solve, entirely using Matlab, a motion planning problem for a SCARA robot that has the same structure of the FANUC SR-3iA shown in Fig. 1, although with slightly different parameters in terms of link lengths and limits on the angular displacements of the links. Our objective is to pick up

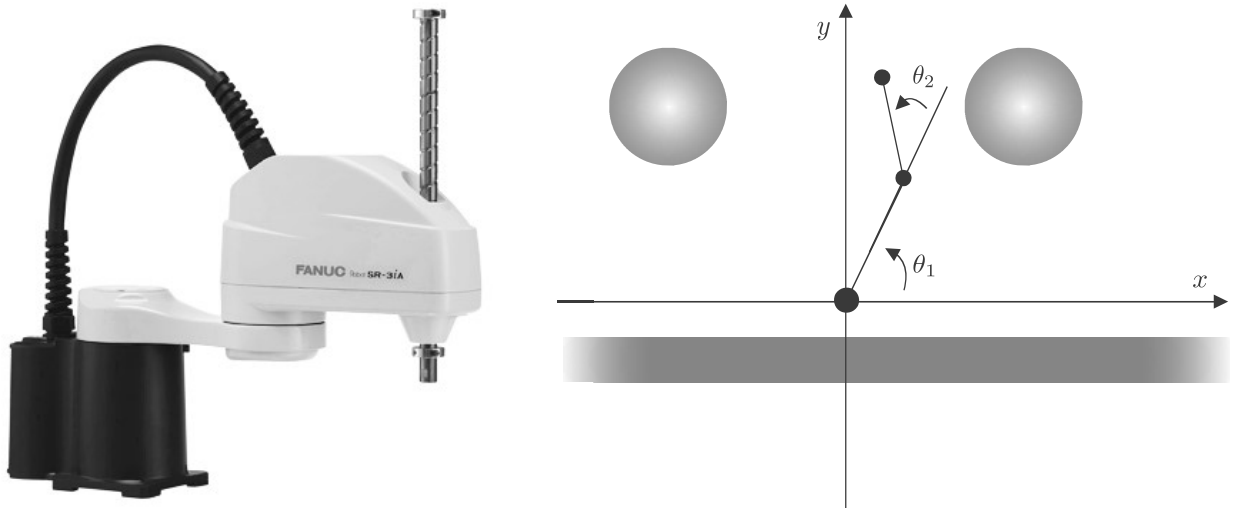

Figure 1: FANUC SR-3iA (left) and schematic of the robot and obstacles modeling for a reduced 2R configuration (right).

an object at a given position in space, and place it at a different position, using an electromagnet at the end effector, which can be activated or deactivated at will. The vertical motion of the robot can happen at any time without affecting collisions with obstacles: as a consequence, the vertical motion is planned independently, and we only have to determine the motion of the manipulator in the horizontal plane. The robot can thus be modeled as a simple 2R planar manipulator. The robot base (i.e., where link 1 is fixed to the ground) is at $(x, y) = (0, 0)$. The links have lengths $\ell_1 = 0.5$ m and $\ell_2 = 0.4$ m, respectively. The first obstacle to be avoided is a wall, which runs parallel to the x axis, keeping a distance of 0.1 m from it (see Fig. 1). Also, there are two other obstacles: these have a fixed position and have been conservatively

represented by two circles, both with radius $B = 0.2$ m, and with center at $(x_{c1}, y_{c1}) = (-0.6, 0.7)$ m and $(x_{c2}, y_{c2}) = (0.6, 0.7)$ m, respectively. The thickness of the links can be neglected, as the sizes of all obstacles have been already augmented to account for robot link thickness. The angular motion of link 1 is only limited by the presence of the wall (so no additional constraints have to be inserted), while link 2 can only move within a range of $\pm 90^\circ$ with respect to the configuration in which it is perfectly aligned with link 1 (i.e., $\theta_2 \in [-\pi/2, \pi/2]$). Our task is to plan a motion from any given initial configuration (where the object is picked) to any final configuration (where the object is placed), chosen in the free space, avoiding any collision during the robot motion.

2 Definition and representation of free space and obstacle space

First of all, we have to represent the robot configuration space, the configuration being $q = (\theta_1, \theta_2)$. The boundaries of the space have to be defined on both angles in a range of 2π (using a grid representation). Hint: it is better to chose the intervals for the two angles such that the free space is connected, without the need of using a torus representation: for example, rather than representing the range of both angles from 0 to 2π, one could do it between $-3\pi/4$ and $5\pi/4$. You will use the simple methods for collision detection seen during class time. From a visual inspection, we notice that link 1 can collide with the wall, but not with any of the circular obstacles: as a consequence, there is no need to define spheres around link 1. As for link 2, this will instead be necessary: you can choose an arbitrary number of spheres, but the suggestion is to use at least 3 of them to cover the whole link length. The aim of this part of the project is to represent free and obstacle space by showing them with two different colors, for example using the `countour` function of Matlab, used to draw a function that is 1 if a point in the grid belongs to the obstacle space, and 0 if it belongs to the free space. The points of the grid used to draw this figure will not be used for motion planning, however a fine grid will be useful to obtain a good representation of free and obstacle spaces, in which your path planning algorithms will operate.

3 Path planning

All algorithms should be implemented for $q_{\text{start}} = (3, -0.8)$ and $q_{\text{goal}} = (0, 1)$, getting samples via a uniform distribution within the boundaries of the rectangular configuration space defined in the previous section. Use a step $d = \pi/50$ and define $\mathcal{C}_{\text{goal}}$ as a circle of radius $\pi/25$ centered at q_{goal}.

- First, implement the standard RRT algorithm seen in class. Keep running it for several iterations even after $\mathcal{C}_{\text{goal}}$ is reached, to be able to show the improvement with respect to the first found trajectory. Remember to connect the closest point to q_{goal} in $\mathcal{C}_{\text{goal}}$ to q_{goal} itself with a straight line to complete the path (the same has to be done for RRT* described in the following).

- Then, implement RRT* using a radius $r > d$ for $\mathcal{C}_{\text{near}}$ determined by you via trial and error. Run the algorithm for the same number of iterations as in the standard RRT case, and show that a shorter path is found.

In both cases, plot the generated tree and the shortest path on top of the figure representing free and obstacle space: this will also help verifying if the solution that you found is likely to be the shortest path, or if something is wrong in the algorithm implementation.

4 Program output and grading

Your program overall can be composed of an arbitrary number of scripts and functions, which should be properly commented (including the comments at the beginning of each script/function, which define how it is used): assume that I am the engineer who needs to design the vertical motion of the SCARA robot, and for that I need to clearly understand what you did. Please name the main script to be run by the user as "main.m". The overall program, when run, has to generate the following figures:

1. representation of free and obstacle space;
2. representation (on top of the previous figure but as a separate figure), of start and goal configurations, generated tree, and shortest path found by RRT and RRT* (each on a separate figure).

You have to submit the Matlab files together as a zip file, plus a pdf file containing a short report including the figures generated previously.

The grading will be determined based on how well the following tasks are executed: definition and representation of free space and obstacle space (20%), path planning algorithms (45%), report clarity (15%), overall clarity of the code (20%).