University of Southern Denmark

Department of Engineering

# Robotics Mandatory Exercise 2

RMROVI1 Introduction to Robotics and Computer Vision

*Authors:*
Arkadiusz Adrian Bogdanowski arbog17@student.sdu.dk
Sergi Grau Moya segra17@student.sdu.dk

October 22, 2017

# 1 Introduction

The main objective of this assignment is to implement in *Robwork* interface a path planning based on *RRT-Connect* (Rapidly Random Tree) algorithm. It is provided a *Kr16WallWorkCell* which is the enviroment of the robot to perform the task. The goal is to take a bottle from a box to a table using the *RRT* algorithm.

The algorithm grows a tree from a starting configuration and uses random samples to keep growing to the final configuration. The connection length is delimited by a parameter that is called *Extended* which will limit the grow factor of the tree.

The implementation of this algorithm can be seen in *pathplanning.cpp* which is a *C++* script that loads all the workspace from *Robwork* and generates a *LUA* script with the configurations of the path planner that the robot will follow.

# 2 Setup

The exercise was done using the given workcell *Kr16WallWorkCell*, by modifying the given script *pathplanning.cpp*. The script was also extended to generate a *.lua* file with the results of the path planner. Only the most important parts of the code are highlighted in this section, the rest can be interpreted by looking at the comments in the code.

## 2.1 Running the path planner

This section highlights the most important changes done w.r.t. the provided *pathplanning.cpp*.
The command

```
rw::math::Math::seed();
```

added at the beginning of the *main* function was used to set the random seed generator used later by the RRT planner.
The following was then set to give the correct joint configurations to the planner.

```
Q from(6,-3.142, -0.827, -3.002, -3.143, 0.099, -1.573);
Q to(6,1.571, 0.006, 0.03, 0.153, 0.762, 4.49);
```

Then, the below was used to set the state and assign the frame joint variables.

```
State state = wc->getDefaultState();
device->setQ(from, state);
```

To grip the bottle, the following was used:

```
const string bottle = "Bottle";
const string tool = "Tool";
rw::kinematics::Kinematics::gripFrame
    (wc->findFrame(bottle),wc->findFrame(tool),state);
```

The RRT planner was then set up as in the original *pathplanning.cpp*.

### 2.2 Exporting to the *.lua* script

The function *pathplanning.cpp* was modified to save the found path to the file in the *.lua* format. Created file was saved with the current date. To achieve that, the iterative loop printing the content of the found path was called again, but printing to the generated file, instead of the console:

```
for (QPath::iterator it = path.begin();
        it < path.end(); it++) {
        std::string itString =
            boost::lexical_cast<std::string>(*it);
        //Remove first 4 characters, e.g. Q[6]:
        itString = itString.erase(0,4);
        LUAfile << "setQ(" << itString << ")" <<endl;
        }
```

The whole structure of the *.lua* script was also modified, to grip the bottle. The following code was used to attach the bottle to the gripper or to the table:

```
attach(bottle,gripper)
attach(bottle,table)
```

## 3 Results

The results achieved with the planner for Section 2 showed that the robot moves in a similar manner for different values of the parameter *extend*. Three *.lua* scripts are attached to the report, for the value of *extend* parameter of 1, 0.1 and 0.01.
More detailed discussion about the influence of the parameter *extend* on the results and the deviations between found solutions is shown in Section 4.
Section 5 tries to explain, why the robot path was found to be '*non-optimal*' and proposes a different solution to the pick-and-place task, mostly as a problem reformulation.

## 4 Statistics

All the generated *.lua* scripts, used for statistical analysis in this section, are attached to the assignment.

## 4.1 Parameter *extend* vs. the path length and the search time

| Extend ($\epsilon$) | 0.01 | 0.1 | 1 |
|---|---|---|---|
| Path length | 1672 | 145 | 13 |
| Search time ($s$) | 8.65 | 0.48 | 0.11 |

Table 1: Comparison between path lengths and execution time for several *extend* values.

Table 1 presents found path length and search time for the average of 5 runs of the RRT planner for 3 different values of *extend*.
The path length raises with the decreasing $\epsilon$ parameter. This is intuitively expected, as any found path is formed by the sections with a length of $\epsilon$. The same concerns the search time, as the planner needs to go through more iterations to find a path.
One could argue, which value of $\epsilon$ is optimal for the assignment. It is observed in RobWork Studio, that increasing the parameter leads to further and further discretization of the path, which might then lead to information loss.

## 4.2 Parameter *extend* vs. the quality of the found path

| Extend ($\epsilon$) | 0.01 | 0.1 | 1 |
|---|---|---|---|
| $\mu$ ($q_1$) | $-0.22$ | $-0.40$ | $-0.42$ |
| $\mu$ ($q_2$) | $-0.27$ | $-0,33$ | $-0.31$ |
| $\mu$ ($q_3$) | $-1.31$ | $-1.34$ | $-1.36$ |
| $\mu$ ($q_4$) | $-0.34$ | $-1.00$ | $-1.18$ |
| $\mu$ ($q_5$) | 0.31 | 0.32 | 0.35 |
| $\mu$ ($q_6$) | 1.13 | 0.72 | 1.49 |

Table 2: Comparison between the average joint value for several *extend* values.

Table 2 illustrates the mean value (for 5 different *.lua* scripts) of the joint configuration for all 6 joints for different $\epsilon$ values.
The table illustrates that the found solutions differ from each other.
Table 3 illustrates the standard deviation from the mean value (for 5 different *.lua* scripts) of the joint configuration for all 6 joints for different $\epsilon$ values.
In the table, a semi-manually found path (attached as *semi-manual.lua*) is compared to the other paths. The path has a different end robot joint configuration, which results in a shorter distance travelled by the end-effector (as further discussed in Section 5). The last row of Table 3 is of the most interest. It shows that the RMS values of all joint deviations are similar, regardless the $\epsilon$ value, which means that the '*quality*' of the path, expressed by the amount

| Extend ($\epsilon$) | 0.01 | 0.1 | 1 | Semi-manual |
|---|---|---|---|---|
| $\sigma$ ($q_1$) | 1.34 | 1.40 | 1.76 | 0.79 |
| $\sigma$ ($q_2$) | 0.23 | 0.24 | 0.29 | 0.16 |
| $\sigma$ ($q_3$) | 0.80 | 0.86 | 1.02 | 0.56 |
| $\sigma$ ($q_4$) | 1.84 | 1.34 | 1.07 | 0.55 |
| $\sigma$ ($q_5$) | 0.16 | 0.22 | 0.31 | 0.41 |
| $\sigma$ ($q_6$) | 2.52 | 2.11 | 2.15 | 1.08 |
| RMS ($q_n$) | 0.58 | 0.50 | 0.53 | 0.27 |

Table 3: Comparison between the standard deviation from the mean value for several *extend* values.

of path travelled by all joints, is comparable. However, the value is significantly smaller for the manual path, which means that the path is considerably shorter.

# 5 Conclusions

The found path of the robot was considered as '*non-optimal*'. By that the authors mean that the path travelled by the bottle could be a way shorter. The path travelled by the bottle is not nearly close to the shortest collision-free path between its start position and end position. The robot does a lot of unexpected moves, one of them shown in Figure 1, where the robotic arm is almost straight and putting the bottle upwards.

The path, being not optimal for the distance travelled by the bottle is, how-

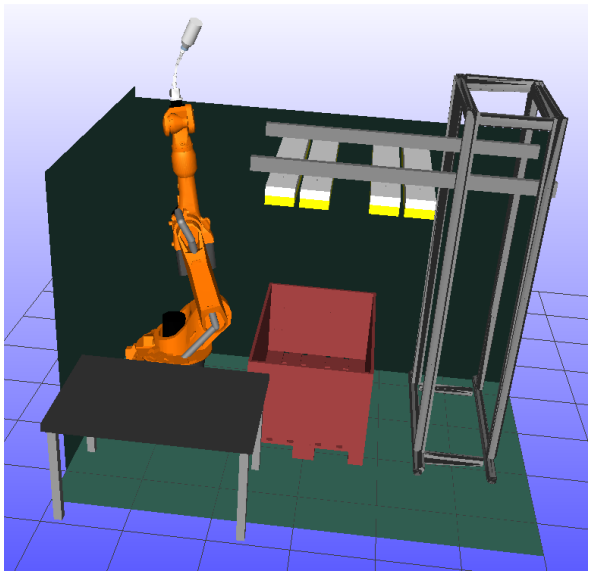

Figure 1: One of the 'weird' robot positions along the path found by the RRT planner

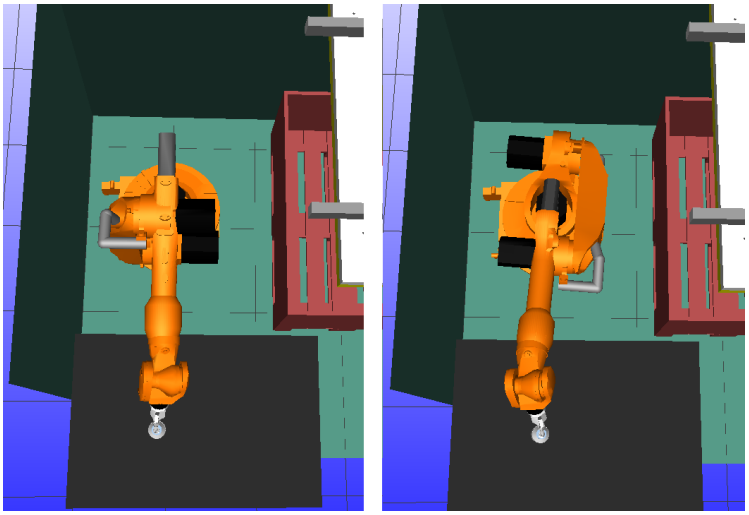

Figure 2: Comparison between the end position of the robot links at the final configuration. Left: configuration requested in the assignment; right: configuration used by the authors in *semi-manual.lua*

ever, the only possible path to fulfill the problem given to the planner. The problem is defined by reaching the goal position of **all 6 joints**. Therefore, the robot does not only have to deliver the bottle to the desired location in the desired orientation, but also reach the defined final configuration. This is not possible without doing 'weird' movements, as on Figure 1.
The optimal solution of the problem would be to change the planner constraints, by **only defining the end position and orientation of the bottle** (or the tool attached to the bottle). Then, the path travelled by the end-effector of the robot and the path travelled by all the robot joints would be significantly smaller.
More optimal solution (than the one found by the RRT planner with end joint configuration vector as the constraint) was found semi-manually (partially by the RRT planner and partially manually) by the authors of the assignment and is attached as *semi-manual.lua*. This illustrates that there exist different goal robot configurations, which lead to a shorter path of the end-effector between the start position and the end position. Figure 2 graphically shows that the end configurations are indeed different.