Aalto University
School of Electrical
Engineering

ELEC-E8126

ROBOTIC MANIPULATION

Report: Exercise -2

Author:
Aravind Swaminathan

Student Number:
769383

January 30, 2020

1 Mathematical equation for plan length computation

The plan length was computed using the Forward Kinematics provided by the moveit package. The trajectory generated will have the joint angles. Each trajectory will have 7 joint values(7 joints in robot). Each trajectory point and its joint values are used to compute the end-effector pose in the cartesian system. Each end-effector pose is then pushed into a vector for plan length calculation.

To calculate the end-effector using forward Kinematics , the follwoing code is used:

```
std::vector<Eigen::Vector3d> trajecory_points_in_cartesian;
for(int i=0;i<plan.joint_trajectory.points.size();i++){
  trajectory_state.setVariablePositions(plan.joint_trajectory.join t_names,
            plan.joint_trajectory.points.at(i).positions);
  Eigen::Affine3d current_end_effector_state =
            trajectory_state.getGlobalLinkTransform("lumi_ee");
  trajecory_points_in_cartesian.push_back(current_end_effector_state.translation());
}
```

Here in this snippet, we use the end-efector pose using the getGlobalLinkTransform() API from the moveit.

To compute the Path length, we use the Euclidean distance formula which is given by

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

and the code snippet to compute the path length using 2 consecutive points in trajectory vector is given below

```
    double computePlanLength(vector<Eigen::Vector3d> traj){
    double length=0;
    for (int i=0;i<traj.size()-1;i++)
    {
            double euclidean_distance = pow(pow(traj[i][0] -traj[i+1][0], 2)  +
                                        pow(traj[i][1] -traj[i+1][1], 2) +
                                        pow(traj[i][2] -traj[i+1][2], 2), 0.5);
            length += euclidean_distance;
    }
    return length;
}
```

2 Necessary Information to replicate results:

1. Number of plans computed per plan: 5 2. Starting position : Default position which is set by calling

```
ros::service::call("/lumi_mujoco/reset", srv_reset);
```

3. Goal position: The goal position is set to follwing coordinates

$$\begin{bmatrix} 1.15 & -1.55 & -1.68 & -2.43 & -0.14 & 2.03 & 0.68 \end{bmatrix}$$

4. The code has to be run only once, It will run multiple planners one by one and each planner will be run 5 times. All the process is automated and the code finally pproduces a csv file with all the results in the work space folder with the file name "Multiple_planner_test_log.csv".

5. The code will run in following manner, PRM, RRT,RRT*,KPIECE and all the results are given in same order

6. To have some settling time between trials, the sleep time between the trials was increased to 10s, because the execution of previous path is taking time, as we can see in the rviz.

3 Results of Planners:

1. Results of PRM planner:

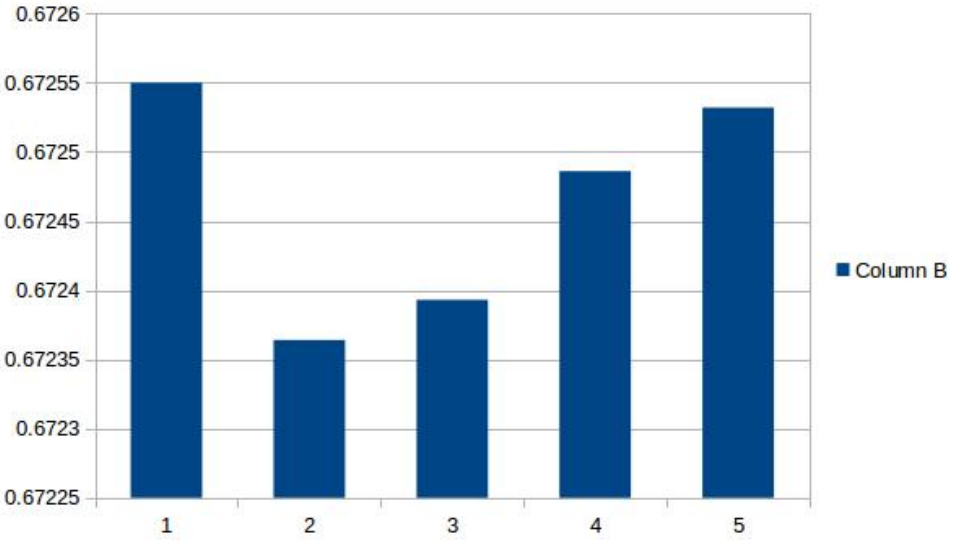

Figure 1: Plan length of PRM for 5 trials

2. Results of RRT planner:

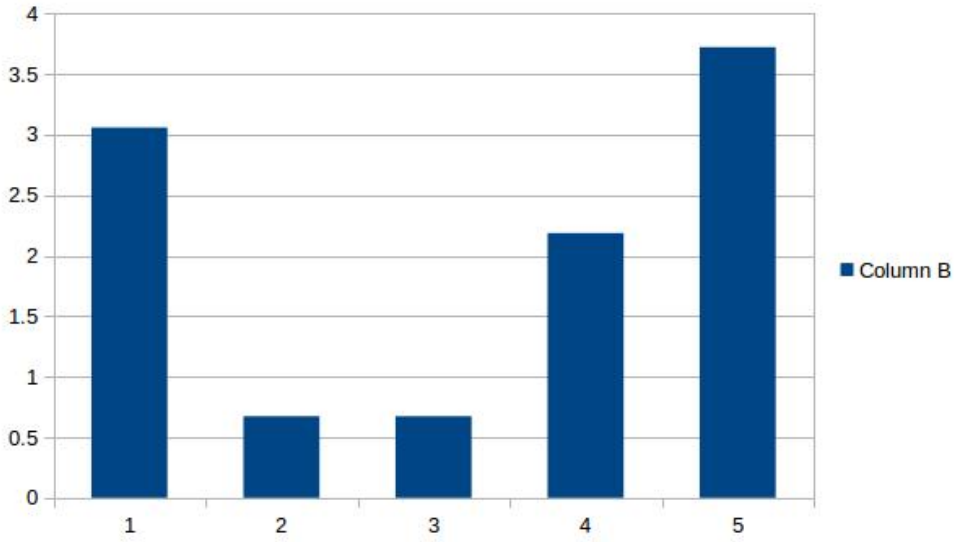

Figure 2: Plan length of RRT for 5 trials

3. Results of RRT* planner:

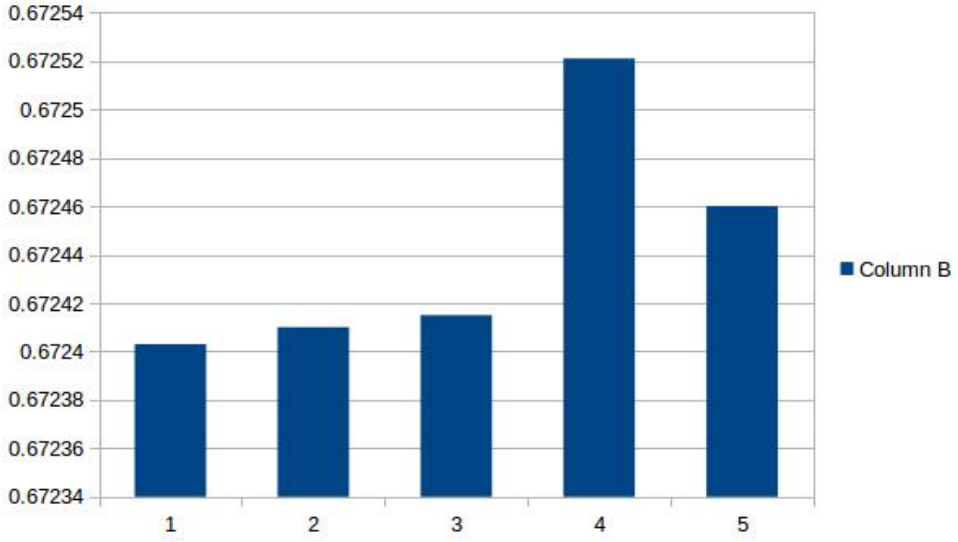

Figure 3: Plan length of RRT* for 5 trials

4. Results of KPIECE planner:

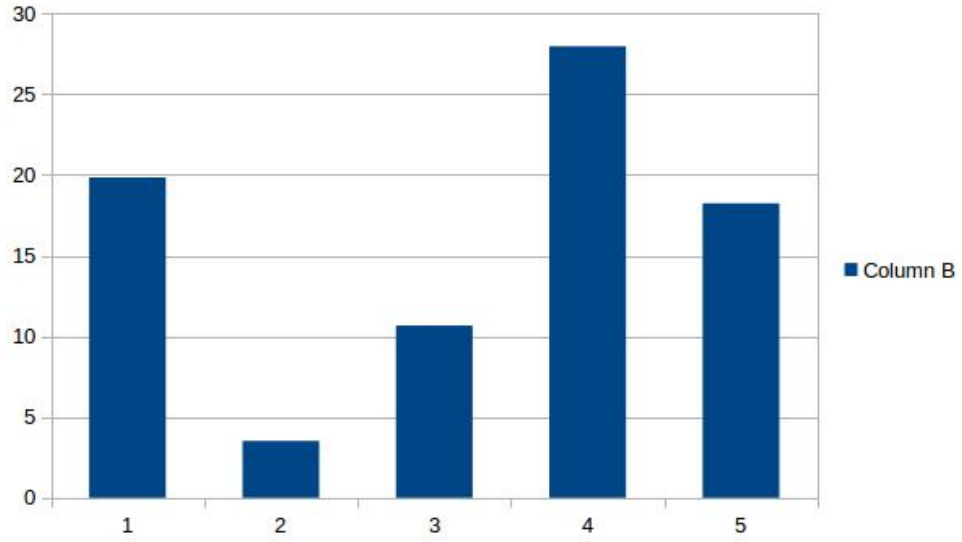

Figure 4: Plan length of KPIECE for 5 trials

5. Comparison of all the Planners(Average plan length comparison):

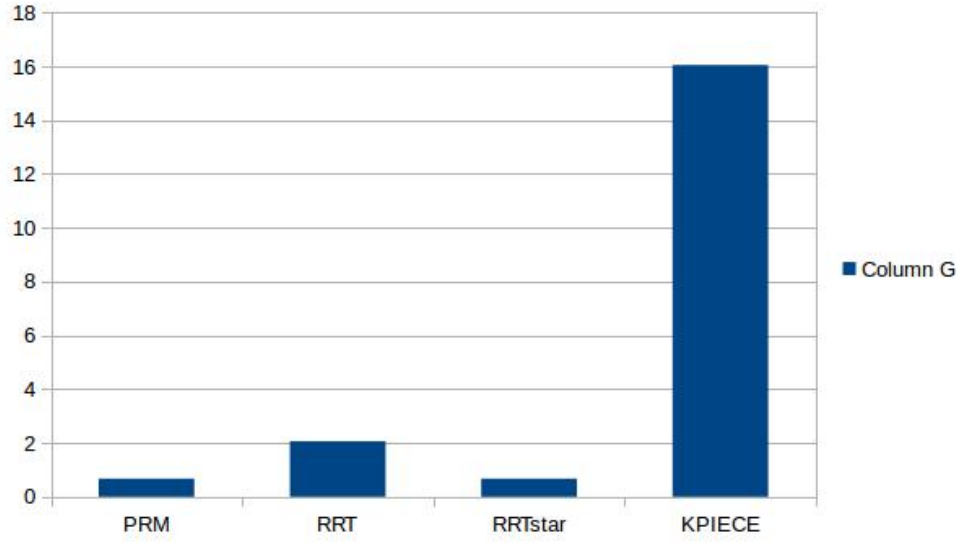

Figure 5: Average Plan length of All planners

6. Comparison of Average planning time for all the planners:

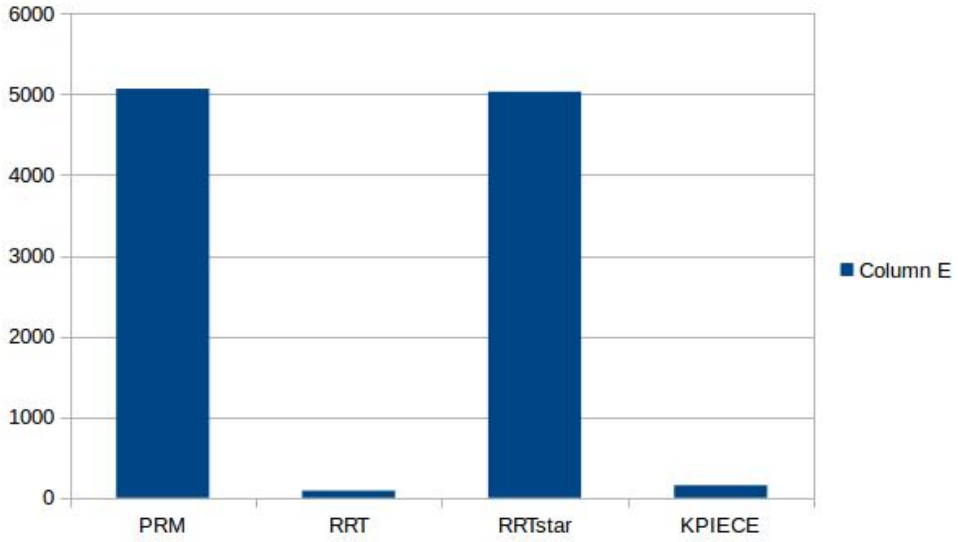

Figure 6: Average Planning time of All planners

The results from which these graphs are generated are given below

PRM

Planning time	Plan length
5057	0.67255
5046	0.672364
5055	0.672393
5083	0.672486
5095	0.672532

RRT

Planning time	Plan length
94	3.05841
115	0.672431
56	0.672408
56	2.18634
116	3.72184

RRTstar

Planning time	Plan length
5031	0.672403
5034	0.67241
5036	0.672415
5034	0.672521
5022	0.67246

KPIECE

Planning time	Plan length
247	19.8397
62	3.51655
85	10.6634
240	27.9647
136	18.2383

Planner	Avg_Plan_length
PRM	0.672465
RRT	2.0622858
RRTstar	0.6724418
KPIECE	16.04453

Planner	Avg_Plan_time
PRM	5067.2
RRT	87.4
RRTstar	5031.4
KPIECE	154

4 Discussion on Results:

1. PRM planner was better than RRT and KPIECE specifically with the plan length planner because it creates more nodes of the configuration space and it works better with the many samples. Although it is time consuming because, it has a local planner as well for the local nodes connection and it has to sample a huge space for the giving even a short path.

2. RRT planner is short in processing than PRM and KPIECE but the plan length results are not good. It has a shorter planning time because of it is more suitability for high degrees of freedom. It is generally a sub-optimal planning algorithm with less heuristics.

3. RRTstar is showing better performances than RRT and KPIECE with plan length, because of its holonomic property. It always tries to find a optimal solution using the rewiring feature. RRT* will consume time because of this rewiring feature and that is one of major difference from the RRT

4. KPEICE was very poor in terms of plan length than any other planner. The path generated was very complex and way far from being optimal. The reason behind this maybe due to its lack of probabilistic completeness.

5 Answers to the questions:

1. If we have given infinite time, the RRT* algorithm will take a lot of time, even if it had found the path, because it tries to still optimize the path, until the rewiring is done. But RRT doesn't have this constraint, once it found the path, it will stop planning instead of optimizing.

But the planned paths will not be same, finally RRT* will be giving close to optimal solution. Lets say for example if we plan for 20-100s, RRT* will give output, once it has explored the full configuration area and found a optimal solution. But RRT will give an output even if it is not optimal. So the plan lengths will be different

2. It is always necessary to run the planner atleast 5 times to comment on it. Because if we see at the results of RRT planner, it finds path at different planning time (say 115ms for 0.67m plan length, 56ms for 2.18m plan length). To decide on a planner, we need to find the average performance of the planner. Because, these planners are probabilistic approaches, and they explore the joint space in a random way. Since it explores the space randomly, the exploration will not be same every trial.

So it is always better to run a planner multiple times to take it for a discussion