Lab 8

Kasper Høj Lorenzen

University of Southern Denmark

kalor@mmmi.sdu.dk

November 10, 2022

Overview

Programming exercise 8

Programming exercise 8 - Task Constrained Motion Planning

TASK_CONSTRAINED_RRT($\mathbf{q}_{init}, \Delta t$)
1 $\mathcal{T}.init(\mathbf{q}_{init})$;
2 **for** $a = 1$ **to** A
3 **do** $\mathbf{q}_{rand} \leftarrow$ RANDOM_CONFIG;
4 $\mathbf{q}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{q}_{rand}, \mathcal{T}$);
5 $\mathbf{q}_{dir} \leftarrow (\mathbf{q}_{rand} - \mathbf{q}_{near})/|\mathbf{q}_{rand} - \mathbf{q}_{near}|$;
6 $\mathbf{q}_s = \mathbf{q}_{near} + \mathbf{q}_{dir}\,\Delta t$;
7 **if** *CONSTRAINED*_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
8 **then** $\mathcal{T}$. add_vertex ($\mathbf{q}_s$);
9 $\mathcal{T}$. add_edge ($\mathbf{q}_{near}, \mathbf{q}_s$);
10 **return** $\mathcal{T}$

COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$)
1 $(\mathbf{C}, \mathbf{T_0^t}) \leftarrow$ RETRIEVE_CONSTRAINT($\mathbf{q_s}, \mathbf{q_{near}}$);
2 $\mathbf{T}_e^0 \leftarrow$ FORWARD_KINEMATICS($\mathbf{q}_s$);
3 $\mathbf{T}_e^t \leftarrow \mathbf{T}_0^t \mathbf{T}_e^0$;
4 $\Delta\mathbf{x} \leftarrow$ TASK_COORDINATES($\mathbf{T}_e^t$);
5 $\Delta\mathbf{x}_{err} \leftarrow \mathbf{C}\Delta\mathbf{x}$
6 **return** $\Delta\mathbf{x}_{err}$;

Fig. 4. Pseudo-code for the Task-Constrained RRT (TC-RRT) construction algorithm. The word *CONSTRAINED* represents either RGD, TS or FR. COMPUTE_TASK_ERROR is common among all three subroutines.

RGD_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
1 $i \leftarrow 0$; $j \leftarrow 0$;
2 $\Delta\mathbf{x}_{err} \leftarrow$ COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$);
3 **while** $i < I$ **and** $j < J$ **and** $|\Delta\mathbf{x}_{err}| > \epsilon$
4 **do** $i \leftarrow i + 1$; $j \leftarrow j + 1$;
5 $\mathbf{q}'_s = \mathbf{q}_s +$ RANDOM_DISPLACEMENT($\mathbf{d_{max}}$);
6 $\Delta\mathbf{x}'_{err} \leftarrow$ COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$);
7 **if** $\Delta\mathbf{x}'_{err} < \Delta\mathbf{x}_{err}$
8 **then** $j \leftarrow 0$; $\mathbf{q}_s = \mathbf{q}'_s$; $\Delta\mathbf{x}_{err} = \Delta\mathbf{x}'_{err}$;
9 **if** $\Delta\mathbf{x}_{err} \leq \epsilon$
10 **then if** IN_COLLISION($\mathbf{q}_s$)
11 **then return false**;
12 **else return true**;
13 **return false**;

TS_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
1 $(\mathbf{C}, \mathbf{T_0^t}) \leftarrow$ RETRIEVE_CONSTRAINT($\mathbf{q_s}, \mathbf{q_{near}}$);
2 $\mathbf{J} \leftarrow$ JACOBIAN($\mathbf{q}_{near}$);
3 $\Delta\mathbf{q} = \mathbf{q}_s - \mathbf{q}_{near}$;
4 $\Delta\mathbf{q}' = \Delta\mathbf{q} - \mathbf{J}^\dagger\mathbf{C}\mathbf{J}\Delta\mathbf{q}$;
5 $\mathbf{q}_s \leftarrow \mathbf{q}_{near} + \Delta\mathbf{q}'$;
6 **return** RGD_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$);

Programming exercise 8 - Task Constrained Motion Planning

- Tips for programming exercise 8:
  - We have constrained the movement of the bottle in the orientation (RPY)
  - Find the right hyperparameters to enable the algorithm to solve the problem
  - Change the goal frame "BottleGoal" in Scene.wc.xml to change the difficulty

Programming exercise 8 - Task Constrained Motion Planning

- Task:
  - Play around with the hyperparameters (step size, max error, random displacement, etc.)
- Optional:
  - Implement RRT-connect in the main loop to boost the performance of the algorithm significantly