B4M36UIR & BE4M36UIR

Task07 - Asymptotically optimal randomized sampling-based path planning

The main task is to obtain experience with the implementation of the state-of-the-art asymptotically optimal randomized sampling-based motion planning algorithms.

Deadline	01. December 2018, 23:59 PST
Points	4
Label in BRUTE	Task07
Files to submit	archive with `samplingplanner` directory
	Minimal content of the archive: `samplingplanner/OMPLPlanner.py`
Resources	Task07 resource files

Additional environments to test the planner

The assignment and deadline for this task are postponed for one week to Lab08 and 01.12. respectively

Assignment

Extend the previous work on Task05 - Randomized sampling-based algorithms to planning using the Open Motion Planning Library (OMPL). Use the Informed RRT* OMPL implementation to provide a **collision free** path through the environment represented by a geometrical map.

In file `OMPLPlanner.py` implement the Informed RRT* algorithm using the OMPL library.

The implementation requirements are similar to the ones of the Task05 - Randomized sampling-based algorithms.

Approach

Adjust the provided minimum example of the Open Motion Planning Library (OMPL) path planning and fuse it into your code from Task05 - Randomized sampling-based algorithms. Two adjustments has to be made in the code:

1. You need to provide the planner with a callback function that will be used for collision checking:
   `task.setStateValidityChecker(ob.StateValidityCheckerFn(isStateValid))`
   This function basically does only proper interfacing of the `environment`)
2. Don't forget that you need to appropriately set the sampling resolution for the collision checking based on the given constraints
   `task.getSpaceInformation().setStateValidityCheckingResolution(resolution)`

Evaluation

The simplified evaluation script for testing of the implementation is following

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
import math
import time
import numpy as np
import matplotlib.pyplot as plt
from collections import deque

sys.path.append('environment')
sys.path.append('samplingplanner')

import Environment as env
import OMPLPlanner as op


if __name__ == "__main__":
    #define the planning scenarios
    #scenario name
    #start configuration
    #goal configuration
    #limits for individual DOFs
    scenarios = [("environments/simple_test", (2,2,0,0,0,0), (-2,-2,0,0,0,math.pi/2), [(-3,3), (-3,3), (0,0), (0,0), (0,0), (0,2*math
                 ("environments/squares", (2,-3,0,0,0,0), (4,8,0,0,0,0), [(0,5), (-5,10), (0,0), (0,0), (0,0), (0,2*math.pi)]),
                 ("environments/maze1", (2,2,0,0,0,math.pi/4), (35,35,0,0,0,math.pi/2), [(0,40), (0,40), (0,0), (0,0), (0,0), (0,2*ma
                 ("environments/maze2", (2,2,0,0,0,math.pi/4), (35,26,0,0,0,3*math.pi/2), [(0,40), (0,40), (0,0), (0,0), (0,0), (0,2*
                 ("environments/maze3", (2,2,0,0,0,0), (25,36,0,0,0,0), [(0,40), (0,40), (0,0), (0,0), (0,0), (0,2*math.pi)]),
                 ("environments/maze4", (2,2,0,0,0,0), (27,36,0,0,0,math.pi/2), [(0,40), (0,40), (0,0), (0,0), (0,0), (0,2*math.pi)])
                 ("environments/cubes", (-1,-1,1,0,0,0), (6,6,-6,math.pi,0,math.pi/2), [(-2,7), (-2,7), (-7,2), (0,2*math.pi), (0,2*m
                 ("environments/cubes2", (2,2,2,0,0,0), (19,19,19,0,0,0), [(-10,30), (-10,30), (-10,30), (0,2*math.pi), (0,2*math.pi)
                 ("environments/alpha_puzzle", (0,5,0,0,0,0),(25,25,25,0,0,0), [(-40,70),(-40,70),(-40,70),(0,2*math.pi),(0,2*math.pi

    #enable dynamic drawing in matplotlib
    plt.ion()

    ########################################
    ## EVALUATION OF THE OMPL PLANNER
    ########################################
```

```
    for scenario in scenarios:
        name = scenario[0]
        start = np.asarray(scenario[1])
        goal = np.asarray(scenario[2])
        limits = scenario[3]

        print("processing scenario: " + name)

        #initiate environment and robot meshes
        environment = env.Environment()
        environment.load_environment(name)

        #instantiate the planner
        planner = op.OMPLPlanner(limits)

        #plan the path through the environment
        path = planner.plan(environment, start, goal)

        #plot the path step by step
        ax = None
        for Pose in path:
            ax = environment.plot_environment(Pose, ax=ax, limits=limits)
            plt.pause(0.01)
```

courses/b4m36uir/hw/task07.txt · Last modified: 2018/11/20 09:29 by cizekpe6