–Table of Contents

Lab07 - Asymptotically Optimal Randomized Sampling-based Path Planning

Motivations and Goals
Become familiar with advanced methods of randomized sampling-based motion planning
Be able to plan the path using the framework of the Open Motion Planning Library
Tasks (teacher)
Implement the RRT-based planning approach using the OMPL framework
Lab resources
OMPL example instance

Open Motion Planning Library

The Open Motion Planning Library contains implementations of many sampling-based algorithms such as PRM, RRT, EST, SBL, KPIECE, SyCLOP, and several variants of these. Details on the installation of the OMPL library are at Open Motion Planning Library tutorial. Below is a simple annotated example of motion planning in continuous space:

```
#!/usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt
from ompl import base as ob
from ompl import geometric as og

def plot_points(points, specs='r'):
    """
    method to plot the points

    Parameters
    ----------
    points: list(float, float)
        list of the pint coordinates
    """
    x_val = [x[0] for x in points]
    y_val = [x[1] for x in points]
    plt.plot(x_val, y_val, specs)
    plt.draw()

def isStateValid(state):
    """
    method for collision detection checking

    Parameters
    ----------
    state : list(float)
        configuration to be checked for collision

    Returns
    -------
    bool
        False if there is collision, True otherwise
    """
    return (state[0] >= state[1])

def plan(start, goal):
    #set dimensions of the problem to be solved
    stateSpace = ob.RealVectorStateSpace()  #set state space
    stateSpace.addDimension(0.0, 10.0) #set width
    stateSpace.addDimension(0.0, 10.0) #set height

    #create a simple setup object
```

```
    task = og.SimpleSetup(stateSpace)

    #set methods for collision detection and resolution of the checking
    task.setStateValidityChecker(ob.StateValidityCheckerFn(isStateValid))
    task.getSpaceInformation().setStateValidityCheckingResolution(0.001)

    #setting start and goal positions
    start_pose = ob.State(task.getStateSpace())
    goal_pose = ob.State(task.getStateSpace())
    start_pose()[0] = start[0]
    start_pose()[1] = start[1]
    goal_pose()[0] = goal[0]
    goal_pose()[1] = goal[1]
    task.setStartAndGoalStates(start_pose, goal_pose)

    #setting particular planner from the supported planners
    info = task.getSpaceInformation()
    planner = og.RRT(info)   # RRT, RRTConnect, FMT, ...
    task.setPlanner(planner)

    #find the solution
    solution = task.solve()

    #simplify the solution
    if solution:
       task.simplifySolution()

    #retrieve found path
    plan = task.getSolutionPath()

    #extract path and plot it
    path = []
    for i in range(plan.getStateCount()):
        path.append((plan.getState(i)[0], plan.getState(i)[1]))

    plot_points(path,'r')
    plt.show()

if __name__ == "__main__":
    plan([0, 0], [10, 10])
```

Besides the setup of the planner, the programmer needs to deliver the validity checking function. In the problem of robotic motion planning this function usually consists of collision checking which can be based on different approaches.

Further information of the planner can be obtained using access through PlannerData object as demonstrated below.

```
    #retrieve planner data
    plannerdata = ob.PlannerData(info)
    planner.getPlannerData(plannerdata)

    #getting vertices
    nv = plannerdata.numVertices()
    g = []
    for i in range(0, nv):
        vert = plannerdata.getVertex(i)
        g.append((vert.getState()[0], vert.getState()[1]))

    #getting edges
    e = []
    for i in range(0, nv):
        for j in range(0, nv):
            if plannerdata.edgeExists(i, j):
                e.append((g[i], g[j]))
    #show edges
    plot_edges(e)
```

```
        plt.show()

    def plot_edges(edges):
        """
        method to plot the edges of the randomized sampling-based path planning approach graph

        Parameters
        ----------
        edges: list((float,float),(float,float))
            list of edges (coordinate1, coordinate2)
        """
        ed = np.asarray(edges)
        for edge in ed:
            plt.plot([edge[0,0], edge[1,0]],[edge[0,1], edge[1,1]],'g',linewidth=0.6)
        plt.draw()
```

Motion planning benchmarking

A set of motion planning benchmarking examples available online.

courses/b4m36uir/labs/lab07.txt · Last modified: 2018/11/19 15:21 by cizekpe6