CourseWare Wiki

```
/ b181 / courses / b4m36uir / labs / lab11
NAVIGATION
 ⋒B4M36UIR
 ■Classification
  > Tasks
 ∨Laboratory Exercises
    ■Lab01 - Introduction to V-REP and Open-Loop Robot Locomotion Control
    ■Lab02 - Exteroceptive sensing and Reactive-based Obstacle Avoidance
    ■Lab03 - Grid and Graph-based Path Planning Methods
    Lab04 - Incremental Path Planning
    ■Lab05 - Randomized Sampling-based Algorithms
    ■Lab06 - Curvature-constrained local planning in RRT
    Lab07 - Asymptotically Optimal Randomized Sampling-based Path Planning
    🖺 Lab08 - Multi-goal path planning and data collection path planning - TSP-like formulations
    □Lab09 - Data Collection Path Planning with Remote Sensing (TSPN)
    □Lab10 - Curvature-constrained Data Collection Path Planning ((D)TSPN)
    ■Lab11 - Game Theory in Robotics
    ■Lab12 - Game Theory in Robotics contd.
    ■Lab13 - Game Theory in Robotics contd.
 Lectures
  > For students
  > Tutorials
ALL COURSES
 Winter 2018 / 2019
 Summer 2017 / 2018
 Older
```

B4M36UIR & BE4M36UIR

-Table of Contents

- Lab11 Game Theory in Robotics
 - Pursuit Evasion
 - Greedy Policy

Lab11 - Game Theory in Robotics

Motivations and Goals

Become familiar with pursuit-evasion scenario

Be able to establish a greedy policy for pursuit-evasion problem

Tasks (teacher)

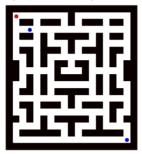
Implement and verify the functionality of greedy policy for pursuit-evasion problem (2 points)

Lab resources

Lab sripts: lab11 resource files

Pursuit Evasion

A problem in computer science where a set of pursuers is trying to catch a set of evaders.



Greedy Policy

In greedy policy the next-best state is selected in each discrete step of the game simulation without considering longer prediction horizon. Usual policies incorporate distances between individual agents as follows:

- For evaders select the most distant node to all pursuers
- For pursuers select the closest node to the closest evader

Tasks (2 points) - Lab11G

- 1. Implement a greedy policy according to the above-described rules
- 2. Note, you can improve the performance of the algorithm by employing more efficient distance function calculation method (e.g. Floyd-Warshall)
- 3. Submit archived player.py and planner.py classes into the BRUTE upload system

courses/b4m36uir/labs/lab11.txt · Last modified: 2018/09/06 14:06 (external edit)