intern_interview_project

May 18, 2019

1 Path Planner

Wall-E needs your help

He is trying to find his way to Eve. He knows his surrounding map environment and all the obstacles on the map. He also knows the Eve's coordinates on the map. However, he doesn't have an algorithm that will tell him the efficient path to reach Eve.

2 Objective

Your mission is to build a web server for Wall-E. He will make HTTP requests to your server to obtain an efficient route to reach Eve. The following is the set of endpoints your server needs to support. For the sake of simplicity, endpoints are not strictly RESTful. There is only one map on the server and it should be stored in memory.

2.1 Create Map

Request to instantiate a map on the server. We can imagine that a map is essentially a 2D Cartesian plane

- Path: /api/maps
- Method: **POST**
- Request Data
  - `row`: integer
  - `col`: integer
- Response Data
  - `http_status`: integer
  - `row`: integer
  - `col`: integer

[1]:
```
import requests

r = requests.post("http://localhost:3000/api/maps/", json={'row': 5, 'col': 5})
print(r.status_code, r.reason, r.json())
```

```
201 Created {'row': 5, 'col': 5}
```

2.2 Create Start

Request to set a start position on the map, i.e. where Wall-E currently is.

- Path: /api/paths/start
- Method: **POST**
- Request Data
  - `i`: integer
  - `j`: integer
- Response Data
  - `http_status`: integer
  - `i`: integer
  - `j`: integer

2.3 Create Goal

Request to set a goal position on the map, i.e. where Eve currently is.

- Path: /api/paths/goal
- Method: **POST**
- Request Data
  - `i`: integer
  - `j`: integer
- Response Data
  - `http_status`: integer
  - `i`: integer
  - `j`: integer

[2]:
```
import requests

r = requests.post("http://localhost:3000/api/paths/start", json={'i': 0, 'j':␣
↪0})
print(r.status_code, r.reason, r.json())

r = requests.post("http://localhost:3000/api/paths/goal", json={'i': 4, 'j': 4})
print(r.status_code, r.reason, r.json())
```

```
201 Created {'i': 0, 'j': 0}
201 Created {'i': 4, 'j': 4}
```

2.4 Create Heuristic Cost

Request to a list of cost values on the map, i.e. where the obstacles are.

- Path: /api/costs
- Method: **POST**
- Request Data
  - `costs: []{ i: integer, j: integer, value: float }`
- Response Data
  - `http_status: integer`
  - `costs: []{ i: integer, j: integer, value: float }`

[3]:
```
import requests

r = requests.post("http://localhost:3000/api/costs", json={
    'costs': [
        {'i': 0, 'j': 1, 'value': 10},
        {'i': 1, 'j': 1, 'value': 10},
        {'i': 3, 'j': 0, 'value': 10},
        {'i': 3, 'j': 1, 'value': 10},
        {'i': 1, 'j': 3, 'value': 10},
        {'i': 2, 'j': 3, 'value': 10},
        {'i': 3, 'j': 3, 'value': 10},
    ]
})
print(r.status_code, r.reason, r.json())
```

```
201 Created {'costs': [{'i': 0, 'j': 1, 'value': 10}, {'i': 1, 'j': 1, 'value':
10}, {'i': 3, 'j': 0, 'value': 10}, {'i': 3, 'j': 1, 'value': 10}, {'i': 1, 'j':
3, 'value': 10}, {'i': 2, 'j': 3, 'value': 10}, {'i': 3, 'j': 3, 'value': 10}]}
```

2.5 Find Path

Request to find the optimal path to reach goal (Eve).

- Path: /api/paths
- Method: **GET**
- Request Data
  - none
- Response Data
  - `http_status`: integer
  - `steps`: integer
  - `path`: `[]{ i: integer, j: integer }`

[4]:
```
import requests

r = requests.get("http://localhost:3000/api/paths")
print(r.status_code, r.reason, r.json())
```

```
200 OK {'steps': 9, 'path': [{'i': 4, 'j': 4}, {'i': 4, 'j': 3}, {'i': 4, 'j':
2}, {'i': 3, 'j': 2}, {'i': 2, 'j': 2}, {'i': 2, 'j': 1}, {'i': 2, 'j': 0},
{'i': 1, 'j': 0}, {'i': 0, 'j': 0}]}
```

3 Comments

Wall-E and Eve

3.1 Requirements

At Fetch Robotics, we use Go, Python and C++ for our backend services. However, they are not required for this project. We are language agnostic. What we want to see is how you structure

your code and how you approach the problem. Here's a list of recommended languages and frameworks to speed up your development time.

- Express with Node.js
- Flask with Python 2 or 3
- Golang
- Sinatra (or Rails) with Ruby
- Spring with Java

3.2 Bonus

If you find this project is too easy, we got some bonus points for you.

- Handle error gracefully
- Write unit tests for your algorithms
- Visualize the path with a frontend application

3.3 Submission

Please upload your code to GitHub or zip it and email it to cfeng@fetchrobotics.com.