機器導航與探索 HW1
110033225 黃興佑

一、關於讀完 A*、RRT、RRT* 後的一些整理

場景:網格地圖、地圖已知、目的地已知

- Dijkstra's: 導航場景下的廣度優先搜尋(BFS). 速度慢但一定是最佳路徑解.
- Best-First-Search: 只走當下代價 g(v) 最低的路徑, 是種貪心策略.
  它不會去回頭考慮沒走到的路徑, 是否會比當前有更高的期望值.此算法求解速度快, 但經常不是最佳解. 代價是自定義的, 可以是移動到該點的路徑長, 或是該點與目標的距離等.

<u>名詞解釋</u>

greedy policy: 只考慮當下最優解的算法. 但會忽略先苦後甘且整體代價更小的路徑解.

heuristic(啟發式): 意指有使用經驗法則的演算法. 像是 Best-First Search 考慮節點與目標點的距離函數, 是歐式距離、曼哈頓距離還是其他啟發函數h(v) 才是最有效率的, 取決於場景跑圖經驗, 沒有標準答案, 會稱作是 heuristic 的. Dijkstra's 就是一套制式標準的走法, 並不是 heuristic 的.

- A* algorithm = Dijkstra's + Best-First-Search. A* 每次迭代, 會像 BFS 展開當前節點能行走到的點, 並考慮 spanning tree 中所有末節點的 h(v)+g(v) 最低的走.

在啟發函數定合適下, *A* 會盡可能快地找到最優路徑.

也就是說, A* 每走下步都會回頭考慮所有路徑可能性, 並搭配啟發函數合理地往正確方向移動或止損. 不會像 greedy policy 一定要撞牆才反悔, 但又不像 BFS 那樣純暴力展開.

場景:**dense map**、地圖已知、目的地已知

- RRT: 從起點以一定長度為半徑, 設置暫時的目標點(q_rand), 並從最接近 q_rand 的節點 q_nearest, 往 q_rand 以一個固定步長(delta_q)前進至新節點(q_new), 直至到達q_rand, 就會再重複同樣的事情; 假如這之間的直線連線遇到障礙, 就會重新設置新的暫時目標點.
- RRT*: RRT改良版, 在每次計算出 q_new 時, 不會直接和 q_nearest 連接, 而是尋找 q_new 周圍的 q_near 節點們, 比起目前的 q_parent1 -> q_nearest-> q_near 有沒有代價更小的 q_parent2 -> q_near, 有的話就做 rewiring, 即更新 q_nearest 為 q_parent2.

二、程式碼

A* algorithm

變數:
- 用 queue 放代價最低的節點, 以 heapq 進行排序, min-heap 的O(N)大約是 NlogN。
- g(n): 起點到目前節點的實際距離。
- h(n) (啟發函數): 目前節點到目標的歐式距離。
- parent: 紀錄每個節點的父節點, 用來回溯最短路徑。

搜尋過程:
1. 取當前代價最小 (g(n) + h(n)) 的節點; 若該節點為目標, 結束迭代。

2. 否則，擴展該節點的 8 個鄰近節點 (上下左右 + 4 個對角)。
3. 過濾無效節點 (超出範圍或遇到障礙物)。
4. 計算新節點的 g(n)，若新路徑較短，則更新 g(n)、h(n)，並加入 queue。

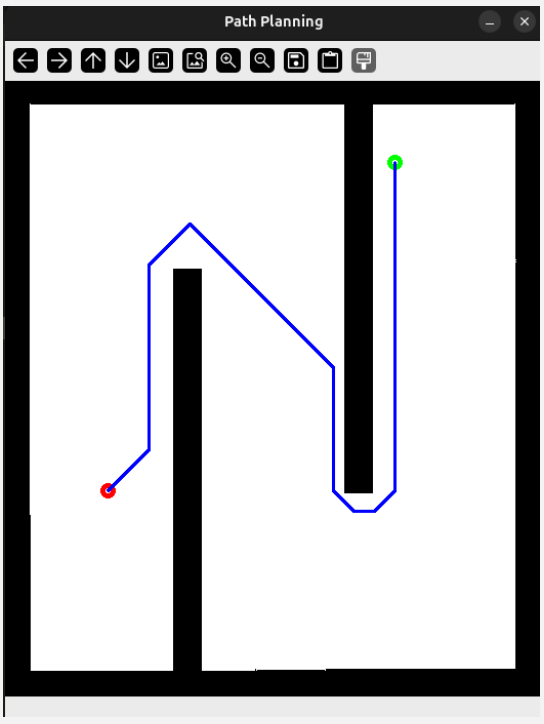
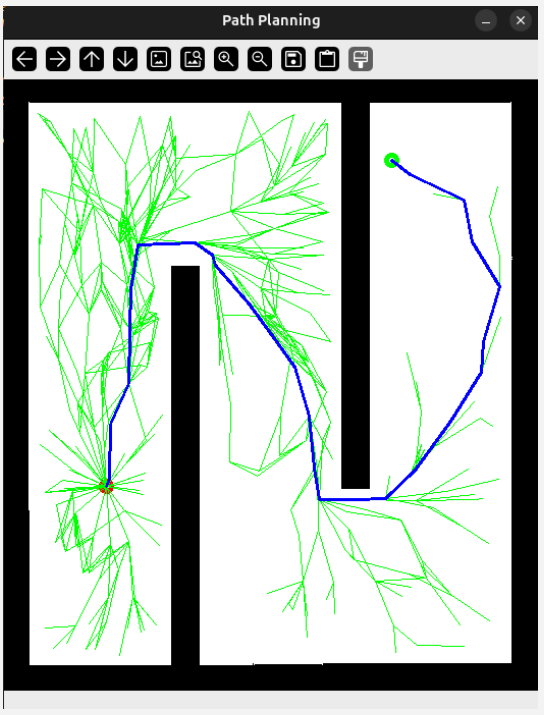
RRT* algorithm

變數:

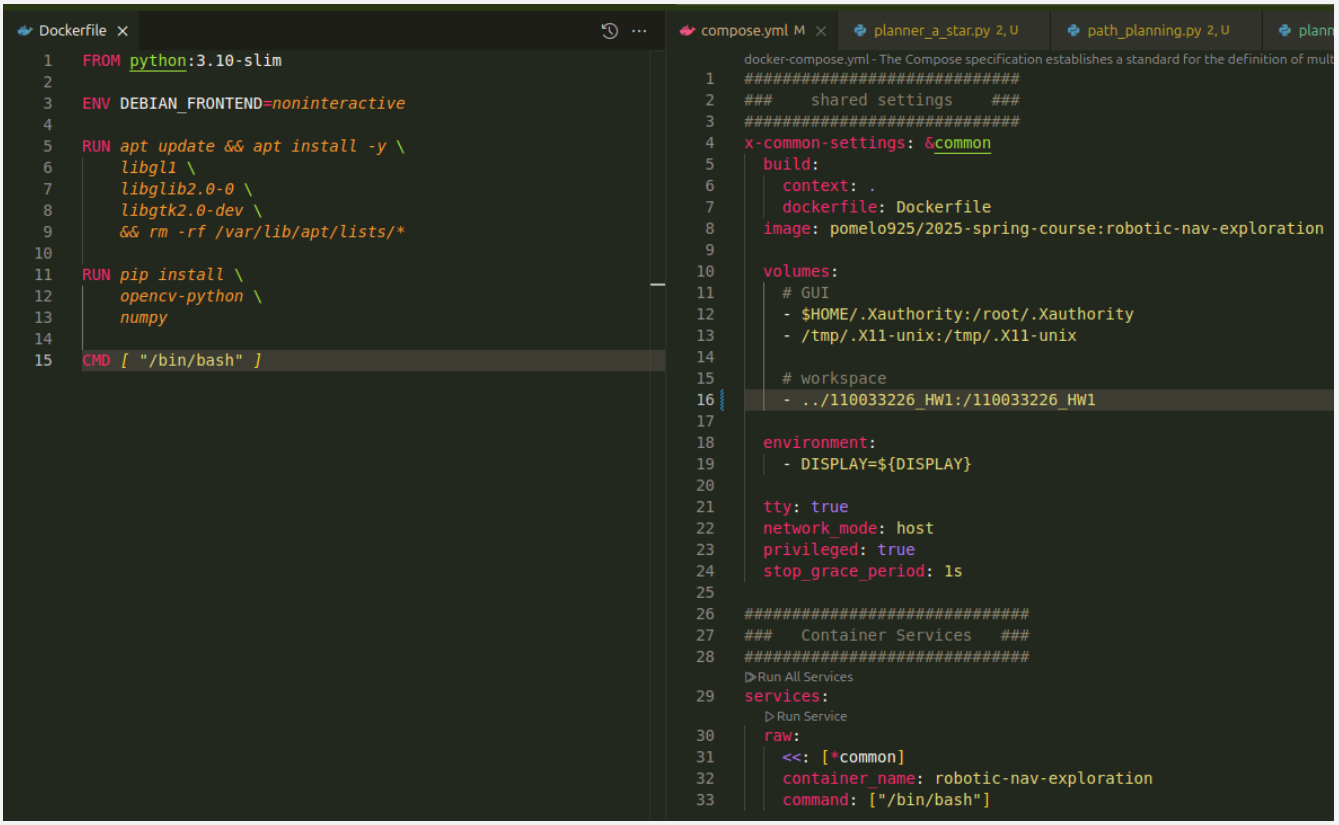
- ntree: 儲存路徑樹，每個節點對應其父節點。
- cost: 記錄每個節點的累積路徑成本。
- extend_len: 每次擴展的最大距離。
- r_near: 重新連結時的鄰近半徑 (通常是 extend_len * 2)。
- _random_node: 隨機採樣，50% 機率直接選擇目標點 (增加收斂速度)。其他是在地圖範圍內隨機取點。
- *nearest*_node: 找到最近的已存在節點
- _steer: 嘗試擴展，計算方向向量，並延伸最多 extend_len 距離.

搜尋過程:

1. 隨機取樣點 (samp_node)，找到最近節點 (near_node)。
2. 嘗試擴展 (_steer)，確保不碰撞後加進 tree 內。
3. 尋找鄰近節點 (near_neighbors)，在 r_near 半徑內找滿足條件的節點。
4. 重新選擇最佳父節點 (Re-parenting)：比較所有鄰近節點，選擇連接後累積成本最小的節點作為新節點的父節點。
5. 重新調整連結 (Rewiring)：更新 tree。
6. 若新節點夠靠近 goal，停止擴展並回溯路徑，由 goal_node 反向追蹤 ntree，構建完整路徑。

A* algorithm	RRT* algorithm
	

備註: 運行環境為 docker



Dockerfile

```
FROM python:3.10-slim

ENV DEBIAN_FRONTEND=noninteractive

RUN apt update && apt install -y \
    libgl1 \
    libglib2.0-0 \
    libgtk2.0-dev \
    && rm -rf /var/lib/apt/lists/*

RUN pip install \
    opencv-python \
    numpy

CMD [ "/bin/bash" ]
```

compose.yml M

```
docker-compose.yml - The Compose specification establishes a standard for the definition of mult
############################
###    shared settings    ###
############################
x-common-settings: &common
  build:
    context: .
    dockerfile: Dockerfile
  image: pomelo925/2025-spring-course:robotic-nav-exploration

  volumes:
    # GUI
    - $HOME/.Xauthority:/root/.Xauthority
    - /tmp/.X11-unix:/tmp/.X11-unix

    # workspace
    - ../110033226_HW1:/110033226_HW1

  environment:
    - DISPLAY=${DISPLAY}

  tty: true
  network_mode: host
  privileged: true
  stop_grace_period: 1s

############################
###   Container Services   ###
############################
▷Run All Services
services:
  ▷Run Service
  raw:
    <<: [*common]
    container_name: robotic-nav-exploration
    command: ["/bin/bash"]
```