SNU School of Engineering BS Thesis Template

(서울대학교 공과대학 학사학위논문 템플릿)

지도교수 : 홍길동

이 보고서를 공학학사 학위 논문 대체 보고서로 제출함.

2025년 월 일

서울대학교 공과대학

컴퓨터공학부

장준서

2025년 8월

SNU School of Engineering BS Thesis Template

(서울대학교 공과대학 학사학위논문 템플릿)

지도교수 : 홍길동

이 보고서를 공학학사 학위 논문 대체 보고서로 제출함.

2025년 월 일

서울대학교 공과대학

컴퓨터공학부

장준서

2025년 8월

Abstract

Abstract of the thesis

Keywords: Seoul National University, College of Engineering, Bachelor's Thesis, Template

Contents

List of Tables

List of Figures

제 1 장 Introduction

With advancements in autonomous driving technology, various autonomous robots are being utilized in everyday life. Representative examples include service robots that shuttle between kitchens and tables to deliver food and utensils [1], cleaning robots that roam specific areas to perform cleaning and disinfection tasks [2], and logistics robots that transport goods in large warehouses [3].

These robots typically operate within designated areas, performing simple, repetitive tasks. As the demand for autonomous robots continues to grow, the size of the areas each robot must manage has also expanded. Whereas traditional service robots were confined to operating within the boundaries of a small restaurant, they are now required to transport food or goods across large shopping malls with multiple restaurants or multi-story office buildings [4].

The expansion of operational areas has introduced several technical challenges. One such challenge is the inability to cover all areas using a single map. Autonomous robots rely on pre-constructed maps to determine their current location and calculate routes to their destinations. However, if a large operational area is represented using a single map, the computational load becomes excessively high, leading to inefficiency. A solution to this problem is dividing the operational area into multiple maps. This approach reduces the size of each map, enabling more efficient localization and easier maintenance.

However, when the starting point and destination are located on different maps, conventional path-planning methods such as A* or RRT cannot compute a path. These methods operate under the assumption that both points lie within the same map. Therefore, a path-planning algorithm for new data structure, multiple maps, is necessary. The paper [5] proposes a path-planning method for navigating between different floors of multi-storey buildings. In this study, the robot's operational area is represented as a hybrid map composed of multiple layers. The

upper layer contains topological information, such as the building and floor details of specific locations, while the lower layer holds metric information, such as an occupancy grid map. Using this hybrid map, the robot performs hierarchical path planning to identify the waypoints required for navigation.

This report builds upon the aforementioned study and explains the implementation of a path-planning method for navigating between arbitrary start and destination points across multiple maps—a task the author undertook at Bear Robotics. The path-planning process consists of two main stages: 1) High-level path planning, which operates on a graph composed of nodes that connect different maps. This stage determines which maps to traverse and which nodes to pass through to move from the starting point to the destination. 2) Low-level path planning, which calculates the global path within a single map to travel between nodes. By combining high-level and low-level path planning, the method ensures efficient navigation across multiple maps while managing the complexity of larger operational areas.

The remainder of this report is organized as follows: Chapter 2 explains the background knowledge for autonomous robot navigation; Chapter 3 details the data structure to represent operational areas and path planning method across multiple maps; and Chapter 4 concludes the report.

제 2 장 Body

The implemented path-planning method operates on a hybrid map consisting of two layers. The lower layer consists of occupancy grid maps, each uniquely identified by an ID and possessing its own independent coordinate system. During navigation, the robot uses a map corresponding to the current location, and its position is expressed in the coordinate system of that specific map. When a horizontal area is too large and must be divided into multiple maps, overlapping regions are introduced between adjacent maps. These common regions ensure seamless localization when the robot transitions from one map to another, allowing for accurate position estimation even after switching maps.

The upper layer is a graph, where nodes are categorized into three types:

1. Destination: Represents final destinations such as tables in a restaurant or storage locations in a warehouse.

2. Horizontal Map Change Point: Represents transition points between different maps on the same floor. These points exist within the overlapping regions of two maps and are conceptually identical locations but have separate nodes for each map. Multiple points can connect two specific maps, and a single point may connect three or more maps.

3. Elevator: Represents the location of a specific elevator on a given floor. Although an elevator may serve multiple floors, a separate node is defined for each floor's boarding point.

Each node contains various information such as location coordinates, floor number and the map ID to which it belongs, and the connected map IDs.

To make a graph, nodes should be connected by edges. The conditions for creating edges between nodes are as follows:

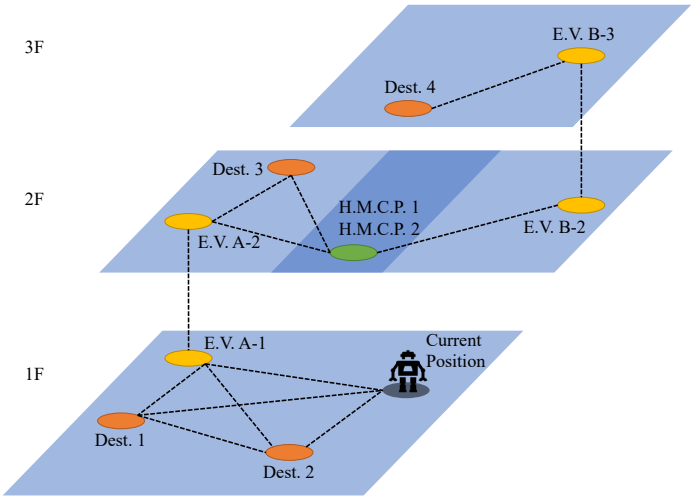

Figure 2.1: This figure illustrates an example of the upper layer of a hybrid map for a three-story building. The building is structured such that the 1st and 2nd floors are connected by Elevator A, while the 2nd and 3rd floors are connected by Elevator B. On the 1st floor, where the robot is currently located, the map includes two destination nodes and one elevator node representing Elevator A. The black node represents the robot's current location. Since it is a temporary node, it will be removed when the path planning is finished. The 2nd floor is divided into two separate maps: the left map contains one destination node, one elevator node for Elevator A, and one horizontal map change point (H.M.C.P. 1) that connects the left and right maps. The right map includes one elevator node for Elevator B and one horizontal map change point (H.M.C.P. 2) that connects the two maps on this floor. Although H.M.C.P. 1 and H.M.C.P. 2 are defined in the coordinate systems of their respective maps, they represent the same physical location. On the 3rd floor, the map includes one elevator node for Elevator B and one destination node.

1. Within the same map: Nodes on the same map are connected. The edge weight between two nodes is determined by the path's length between them. The path is calculated using an occupancy grid map in the lower layer and traditional path planning method, like A*.

2. Between horizontal map change points: Nodes representing the same location across different maps are connected. As the actual distance between the two nodes is 0, the edge weight between such nodes is set to 0.

3. Between elevator nodes: Nodes representing the same elevator across different floors are connected. As the robot won't move while elevator is moving, the edge weight between these nodes is also set to 0.

An example of this graph structure can be seen in Figure **??**.

Path planning consists of three main stages. The first stage involves adding the current position as a new node of type "destination" to the upper-layer graph. Following the edge

creation rules described earlier, edges are then connected between this new node and other nodes within the same map. If the starting point is not the current position but an existing node, this step is skipped.

The second stage calculates the shortest path from the starting node to the destination node in the upper layer. A single-source, single-destination shortest path algorithm, such as Dijkstra' s algorithm, is used for this purpose. This stage determines the waypoints the robot must pass through to reach its destination. Importantly, this step is performed only once, before the robot begins its journey.

The third stage involves performing global path planning in the lower layer to navigate from the current position to the next waypoint. This stage is similar to navigation within a single map, where real-time information such as the robot's current position and obstacles is used to compute the global path and local path. This process continues until the robot reaches the final destination.

An example is illustrated in Figure **??**. Suppose the destinations are given as Dest. 4, Dest. 3, Dest. 2, and Dest. 1. First, the robot's current position is added to the graph, as shown in the figure. Then, Dijkstra' s algorithm is used to calculate the shortest path from the current position to Dest. 4. The resulting node array is "E.V A-1, E.V A-2, H.M.C.P 1, H.M.C.P. 2, E.V B-2, E.V B-3, Dest. 4". The process is repeated by updating the starting point to the current destination and calculating the shortest path to the next destination. The results of these calculations can be found in Table **??**.

After high-level path planning is completed, low-level path planning is performed. Figure **??** illustrates the robot performing path planning to reach the first waypoint, E.V A-1. Low-level path planning continues iteratively until no waypoints remain.

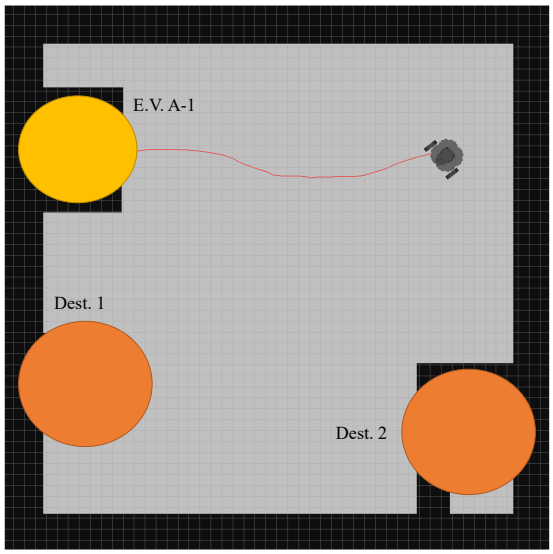

Figure 2.2

Table 2.1: An example of a table.

Start Node	End Node	Waypoints
”Current Position”	”Dest. 4”	”E.V. A-1”, ”E.V. A-2”, ”H.M.C.P. 1”, ”H.M.C.P. 2”, ”E.V. B-2”, ”E.V. B-3”, ”Dest. 4”
”Dest. 4”	”Dest. 3”	”E.V. B-3”, ”E.V. B-2”, ”H.M.C.P. 2”, ”H.M.C.P 1”, ”Dest. 3”
”Dest. 3”	”Dest. 2”	”E.V. A-2”, ”E.V. A-1”, ”Dest. 2”
”Dest. 2”	”Dest. 1”	”Dest. 1”

제 3 장 Conclusion

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

References

[1] "Servi +." Accessed: 2025-01-11. (), [Online]. Available: https://www.bearrobotics.ai/servi-plus (visited on 01/11/2025).

[2] J. Bok. "병원 방역도 인공지능 로봇이." Accessed: 2025-01-11. (2022), [Online]. Available: http://sjbnews.com/news/news.php?number=749827 (visited on 01/11/2025).

[3] "Carti 100." Accessed: 2025-01-11. (), [Online]. Available: https://www.bearrobotics.ai/carti (visited on 01/11/2025).

[4] "Enhanced hospitality with servi lift." Accessed: 2025-01-11. (), [Online]. Available: https://www.bearrobotics.ai/hotels (visited on 01/11/2025).

[5] Q. Zhang, X. Wu, B. Liu, A. H. Adiwahono, T. A. Dung, and T. W. Chang, "A hierarchical topological planner for multi-storey building navigation," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2015, pp. 731–736. DOI: 10.1109/AIM.2015.7222624.

국문초록

하수는 두 산 틈에서 나와 돌과 부딪쳐 싸우며, 그 놀란 파도와 성난 물머리와 우는 여울과 노한 물결과 슬픈 곡조와 원망하는 소리가 굽이쳐 돌면서, 우는 듯, 소리치는 듯, 바쁘게 호령하는 듯, 항상 장성을 깨뜨릴 형세가 있어, 전차 만승과 전기 만대나 전포 만가와 전고 만좌로써는 그 무너뜨리고 내뿜는 소리를 족히 형용할 수 없을 것이다. 모래 위에 큰 돌은 홀연히 떨어져 섰고, 강 언덕에 버드나무는 어둡고 컴컴하여 물지킴과 하수 귀신이 다투어 나와서 사람을 놀리는 듯한데, 좌우의 교리가 붙들려고 애쓰는 듯싶었다. 혹은 말하기를, “여기는 옛 전쟁터이므로 강물이 저같이 우는 것이다” 하지만 이는 그런 것이 아니니, 강물 소리는 듣기 여하에 달렸을 것이다.

주요어:서울대학교, 공과대학, 학사학위논문, 템플릿