11117|扬州大学
硕士学术学位论文评阅书

学号:	2024022004
论文名称:	结合空间聚类与旅行商规划的高效全区域探索
作者姓名:	Sultan Mamun
作者学科专业:	电气工程
作者研究方向:	机器人技术与自动化

论文题目	结合空间聚类与旅行商规划的高效全区域探索		
学科(专业)	电气工程		
评议项目	评价要素	分数|评分	
论文选题	选题的学科前沿性，理论意义、现实意义和应用价值等。	10	7
文献综述	熟悉与本课题相关领域的国内外学术动态与发展趋势，综述全面系统，评述得当，富有见解；体现作者掌握本学科及相关领域的理论基础和专门知识。	20	16
论文设计方法与手段	研究目标明确，方法得当，手段先进，数据来源可靠，分析方法科学，论证验证可信；运用新的视角、新的方法探索和研究新的问题，论点明确，论证论据丰富准确；工作量饱满，能反映作者独立从事科学研究工作的能力。	30	25
论文成果的创新性	论文成果具有原创性。理论上有突破，实验方法或运算方法或分析方法上有创新，论文成果的应用具有可推广性。	30	24
论文写作	学风严谨，论文格式符合学科规范。论文结构合理，层次分明，概念明确，条理清晰，语言表达准确、流畅，且有学理性和逻辑性，引文规范，图表清晰。	10	7
总分	79		
是否同意答辩	基本达到硕士学位论文要求，需进一步修改，经导师审核后答辩（70分-79分）		
推荐论文评选	不推荐		

对论文熟悉程度	很熟悉

对学位论文的学术评语
The idea of combining spatial clustering and TSP-based planning for exploration is novel and interesting. The work is well-motivated, highlighting the limitations of existing frontier/viewpoint-based exploration methods. The dual-stage local and global exploration approach is a reasonable strategy. The experimental results demonstrate the potential efficacy of the proposed method in various simulated and real-world environments.

论文的不足之处和建议
Shortcomings: The thesis lacks a comprehensive review and theoretical analysis of diffe rent spatial clustering techniques considered for this problem. The criteria and justification for selecting the specific clustering algo rithm(s) used are not clearly explained. There is no detailed analysis of the computational complexity and scalabi lity of the overall algorithm, especially for large-scale environments. The impact of various parameter choices (e.g., cluster sizes, planning ho rizons) on the performance is not thoroughly investigated. The robustness of the approach to navigation uncertainties, localization errors, and sensor noise is not evaluated. The thesis primarily focuses on static environments and does not consider dynamic scenarios where obstacles/frontiers may continuously evolve. A formal theoretical analysis of the algorithm's convergence, optimality guarantees, and completeness is lacking. Comparisons are made against a limited set of baseline exploration strate gies; a more comprehensive comparative evaluation against other state-of-the-art or learning-based techniques is desirable. Suggestions: Provide a more comprehensive review and theoretical analysis of different clustering techniques, justifying the chosen approach. Conduct a detailed computational complexity analysis and scalability stud y of the proposed algorithm. Investigate the sensitivity of the approach to different parameter choice s and provide guidelines for parameter tuning. Evaluate the robustness of the algorithm to navigation uncertainties, loc alization errors, and sensor noise through targeted experiments. Extend the approach to handle dynamic environments where obstacles and fr ontiers evolve over time. Include a formal theoretical analysis of the algorithm's convergence, opt imality, and completeness properties. Compare the proposed method against a broader set of state-of-the-art exp loration techniques, including learning-based approaches. Explore the extension of the approach to multi-robot exploration scenario s with coordination strategies.

论文的不足之处和建议
Perform more extensive real-world validation across diverse environments and application domains.

编号：MH20062

扬州大学

硕士研究生学位论文评阅书（学术学位）

论文题目：	CombiningSpatialClusteringandTSP-basedPlanningforEfficientFullAreaExploration
作者姓名：	Sultan Mamun
所在学院：	电气与能源动力工程学院
学科专业：	电气工程
专业代码：	0808
研究方向：	机器人技术与自动化
填表时间：	2024-07-18

扬州大学研究生院制

一、硕士学位论文评阅评分指标

序号	评阅内容	评阅参考要素	项目分值	实际得分
1	**论文选题**	选题的学科前沿性，理论意义、现实意义和应用价值等	10	8分
2	**文献综述**	熟悉与本课题相关领域的国内外学术动态与发展趋势，综述全面系统，评述得当，富有见解；体现作者掌握本学科及相关领域的理论基础和专门知识。	20	14分
3	**论文设计方法与手段**	研究目标和研究问题明确，研究方法使用得当，研究手段先进，数据来源可靠，分析方法科学，论证验证可信；运用新的视角、新的方法探索和研究新的问题，论点明确，论证论据丰富准确；论文研究工作量饱满，能反映作者独立从事科学研究工作的能力。	30	21分
4	**论文成果的创新性**	论文成果具有原创性。理论上有突破，实验方法或运算方法或分析方法上有创新，论文成果的应用具有可推广性。	30	21分
5	**论文写作**	学风严谨，论文格式符合学科规范。论文结构合理，层次分明，概念明确，条理清晰，语言表达准确、流畅，且有学理性和逻辑性，引文规范，图表清晰。	10	6分
总分		综合评定等级：优秀90～100分；良好80～89分；中等70～79分；及格60～69分；不及格<60分。	100	70分

评阅人对论文是否同意答辩的意见

（请在相应的 □ 内打“ √ ”）

( )已经达到硕士学位论文要求，同意经少量的修改后答辩（80分以上）。

(√)基本达到硕士学位论文要求，需进一步修改，经导师审核后答辩（70分-79分）。

( )与硕士学位论文要求有一定差距，需进行较大的修改后重新评审（60分-69分）。

( )已未达到硕士学位论文要求，不同意答辩（60分以下）。

二、硕士学位论文评阅意见

请您参照以下几个方面提出评阅意见：

1、对论文选题的应用价值和理论意义的评价；

2、对文献综述能力的评价；

3、对掌握基础理论、专业知识情况及独立从事科研工作能力的评价；

4、对论文成果创新性的评价；

5、对论文写作水平的评价。

对论文的综合评价：

该硕士学位论文主要研究了基于结合空间聚类和旅行商问题的自主全局探索规划问题，导向正确，选题立足电机与电器工程专业领域中移动机器人自主探索的现实问题，具有一定的理论意义和应用价值；文献综述资料较丰富，基本符合专业质量要求；正文写作安排合理，能够运用本专业基础理论进行建模、计算和仿真，综合分析问题和解决问题，研究方法恰当，工作量较饱满；内容组织符合要求，逻辑构建结构合理，层次分明，重点突出，逻辑性较强，表达较准确，写作较规范，图表清晰；达到电机与电器专业硕士学位论文要求。

问题如下：

1.中英文摘要撰写较省略，建议增加主要创新点和指标参数；

2.中英文关键词不对应，需修改完善；

3.正文中所有图下方的图题需严谨规范，图表的说明应客观，去掉口语化描述；

4.正文6、7和8中，均为文字说明，缺乏相应的性能指标和参数的指标对比。

综上所述，论文基本达到硕士学位论文的水平，可以作为硕士学位论文安排答辩。

对论文的修改建议：

问题如下：

1. 中英文摘要撰写较省略，建议增加主要创新点和指标参数；
2. 中英文关键词不对应，需修改完善；
3. 正文中所有图下方的图题需严谨规范，图表的说明应客观，去掉口语化描述；
4. 正文6、7和8中，均为文字说明，缺乏相应的性能指标和参数的指标对比。

是否推荐参加优秀专业学位硕士论文评选	( )省级	( )校级	(√)不推荐

评阅日期 2024年 07月 19日

三、论文评阅专家基本情况

评阅人姓名：	匿名	职称：		
是否博导：	( )是		( )否	
是否硕导：	( )是		( )否	
目前从事专业或领域：				
所在工作单位（盖章）：				
通讯地址：				
E-mail/联系电话（办）： /				
您对论文涉及的领域的熟悉程度：	(√)很熟悉	( )较熟悉	( )一般	( )不熟悉

注：1、除作者自评部分外，评阅书其他部分由评阅人填写；
　　2、学院应及时将评阅结果复印件转交论文作者本人，不得透漏评阅人基本信息；
　　3、论文作者应根据评阅人提出的修改意见进行修改后，方可进行学位论文答辩；
　　4、论文评阅书原件需交至校档案馆归档。

分类号________	学　号 MH20062
U D C________	密　级 __________

揚州大學
YANGZHOU UNIVERSITY

硕 士 学 位 论 文

（全日制学术学位）

结合空间聚类与旅行商规划的高效全区域探索

学　　　　院：	电气与能源动力工程学院
专　　　　业：	电气工程及其自动化
姓　　　　名：	Sultan Mamun
指　导　教　师：	包加桐副教授
答辩委员会主席：	李生权　教授
答　辩　日　期：	2024 年 9 月 5 日

Combining Spatial Clustering and TSP-based Planning for Efficient Full Area Exploration

A thesis submitted to
Yangzhou University
in partial fulfillment of the requirements
for the degree of
Masters of Science in Electrical Engineering

By
Sultan Mamun
Supervisor: Prof. Bao Jiatong
Electrical Engineering
September 2024

扬州大学学位论文原创性声明和版权使用授权书

学位论文原创性声明

本人声明：所呈交的学位论文是在导师指导下独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含其他个人或集体已经发表的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：[illegible]

签字日期：2024 年 03 月 14 日

学位论文版权使用授权书

本人完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交学位论文的复印件和电子文档，允许论文被查阅和借阅。本人授权扬州大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

本学位论文属于（请在以下相应方框内打“√”）：

1、保密□，在___年解密后适用本授权书。

2、不保密□。

作者签名：[illegible]　　　　导师签名：包加桐

签字日期：2024 年 03 月 14 日　　　　签字日期：2024 年 3 月 14 日

Catalogue

摘要

未知环境下的自主探索已成为移动机器人的关键能力。许多方法严重依赖于信息增益来确定探索目标而使得计算效率低，以及缺乏有效的线路优化。基于强化学习的方法没有考虑全区域覆盖，并且无法保证将学习策略转移到新环境的表现。为了解决这些问题，本文研究提出了一种双阶段探索方法，该方法结合了对所有可能的探索目标进行空间聚类和基于旅行商问题（TSP）进行局部与全局尺度的线路规划，旨在在高度复杂的环境中进行高效的全区域探索。我们的方法涉及两个阶段：探索和重定位。在探索阶段，我们直接从所有可能的本地探索目标的集群生成局部导航目标候选集。然后基于 TSP 框架进行线路规划来确定最终的局部导航目标。此外，在重定位阶段，我们提出对所有可能的全局探索候选目标进行空间聚类，并再次应用基于 TSP 的旅行规划来有效地将机器人引导至先前检测到但尚未探索的区域。所提出的方法在各种具有挑战性的模拟和现实环境中得到了验证，实验结果证明了其有效性和效率。

关键词：移动机器人；自主探索；空间聚类；旅行商规划

Abstract

Autonomous exploration in unknown environments has become a critical capability of mobile robots. Many methods often suffer from problems such as exploration goal selection based solely on information gain and inefficient tour optimization. Recent reinforcement learning based methods do not consider full area coverage and the performance of transferring learned policy to new environments cannot be guaranteed. To address these issues, a dual-stage exploration method has been proposed, which combines spatial clustering of possible exploration goals and Traveling Salesman Problem (TSP) based tour planning in both local and global scales, aiming for efficient full area exploration in highly convoluted environments. Our method involves two stages: exploration and relocation. During the exploration stage, we introduce to generate local navigation goal candidates straight from clusters of all possible local exploration goals. The local navigation goal is determined through tour planning, utilizing the TSP framework. Moreover, during the relocation stage, we suggest clustering all possible global exploration goals and applying TSP-based tour planning to efficiently direct the robot towards previously detected but yet-to-be-explored areas. The proposed method is validated in various challenging simulated and real-world environments. Experimental results demonstrate its effectiveness and efficiency.

Key Words: Mobile Robot; Autonomous exploration; Spatial Clustering; Traveling Salesman Problem (TSP) Planning.

Chapter 1 Introduction

1.1 Background

A basic issue in many fields, including robotic exploration, urban planning, and environmental monitoring, is the efficient exploration of a given area. When resources like time, money, or energy are scarce and thorough coverage of the area is required or desired, efficient exploration becomes necessary. Conventional exploration techniques typically entail a methodical or haphazard survey of the region, which might not be the most effective strategy, particularly in expansive or complicated settings. In data analysis and pattern recognition, spatial clustering techniques are widely used to divide a dataset into groups or clusters based on the spatial proximity of data points. These methods can be used to divide an area into manageable regions for exploration in spatial exploration problems by looking for significant structures within the data.

On the other hand, tour planning algorithms are employed to determine the best order of waypoints or destinations to visit in order to reduce travel time or distance. In the context of logistics, travelling salesman issues, and vehicle routing, these algorithms have been thoroughly examined. But their use in spatial exploration tasks is relatively limited, especially when combined with spatial clustering. The goal of this thesis is to effectively explore an area by bridging the gap between tour planning and spatial clustering techniques and utilizing their synergies. Better coverage of the area can be achieved while consuming less resources by combining spatial clustering to divide the area into clusters and tour planning algorithms to optimize the exploration path within each cluster. This work intends to make a significant contribution to the fields of environmental monitoring, urban planning, and exploration robotics by offering a novel method that can increase the efficacy and efficiency of exploration tasks. It is anticipated that the suggested methodology will find use in a number of real-world scenarios, such as autonomous exploration of uncharted territory, sensor network deployment for environmental monitoring, and route planning for urban surveying [27][28].

Mobile robots are being increasingly deployed across a wide range of applications [7][1], such as search and rescue operations, hospital services, and office deliveries. In these scenarios, the robots are required to autonomously navigate and explore unknown environments in order to gather information and efficiently accomplish their tasks. For instance, search and rescue robots must rapidly and autonomously search through disaster-stricken areas, while hospital service and office delivery robots need to efficiently explore and map large and complex environments without the need for additional human intervention. Therefore, the autonomously exploration and navigation through unknown environments represent critical capabilities for mobile robots.

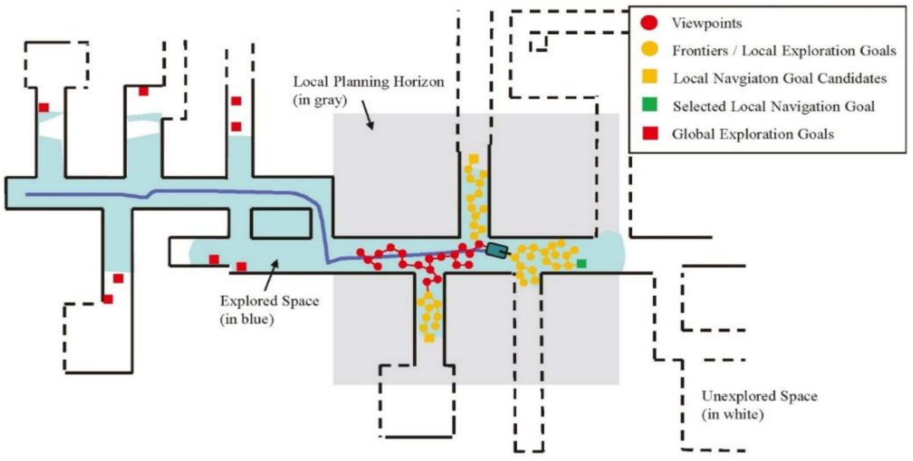

Viewpoints
Frontiers / Local Exploration Goals
Local Navgiaton Goal Candidates
Selected Local Navigation Goal
Global Exploration Goals
Local Planning Horizon (in gray)
Explored Space (in blue)
Unexplored Space (in white)

Fig. 1-1. Overview of the Autonomous Exploration Methodology. The example environment which the robot is required to explore is shown as a 2D map. Explored areas are depicted in blue, while unexplored regions are represented by white spaces. Our method comprises two stages: the exploration stage and the relocation stage. During the exploration stage, an RRT is expanded within the local planning horizon, where each node serves as a viewpoint. Frontiers are identified from these viewpoints. The identified local frontiers are considered as possible exploration goals. They are clustered, with each cluster corresponding to a local navigation goal candidate. By using local tour planning, the optimal local navigation goal is selected for execution, considering the remaining goals as possible global exploration goals. Once no clusters in the local planning horizon remain, the robot switches to the relocation stage. In this stage, all updated global exploration goals are clustered and employed for global tour planning. The robot is then guided towards the selected global navigation goal. These two stages are executed back-and-forth until no global exploration goals remain.

Conventional exploration methods typically involve detecting frontiers [21][9][13][17], sampling viewpoints, and navigating the robot towards the viewpoint with the highest information gain. These methods often rely on either frontiers or randomly sampled viewpoints. Frontiers are special locations that separate explored areas from unexplored ones and were first introduced for autonomous exploration by Yamauchi et al. [21]. However, the challenge remains in how to detect frontiers and select optimal frontiers for exploration to maximize navigation efficiency and area coverage. On the other hand, most viewpoint-based approaches tend to be greedy, with viewpoints densely sampled around the robot and maintained throughout the exploration process. Without proper tour planning, the robot may spend considerable time navigating back to viewpoints that are close to already explored regions. Therefore, both frontier-based and viewpoint based exploration methods frequently encounter challenges such as relying exclusively on information gain for the selection of navigation goals and not optimizing the exploration path effectively. This leads to reduced efficiency in exploration, particularly in highly convoluted environments. Recent reinforcement learning based methods [11][20] focus on learning exploration policies and do not consider full area coverage. The performance cannot be guaranteed when transferring learned policy to new environments.

In this paper, we propose a dual-stage exploration method that combines spatial clustering and tour planning to address these challenges. Instead of individually assessing the information gain for each potential exploration goal, spatially clustering all possible densely distributed exploration goals allows the robot to rapidly determine sparse navigation goals. Tour planning based on Traveling Salesman Problem (TSP) framework is executed on these sparse navigation goals, optimizing the robot's path by considering place-to-place costs and reducing computation load by limiting the number of tour locations. Spatial clustering and tour planning are performed in both local and global scales.

Figure 1-1 offers an overview of our method. During the exploration stage, frontiers are identified from an RRT without any bias, all within the local planning horizon. These frontiers are always densely distributed and considered as possible local exploration goals. They are subsequently clustered and transformed into individual local navigation goal candidates. Planning tours based on the local navigation goal candidates and the robot's home position allows for the rapid expansion of the exploration boundary. When no local exploration goal clusters remain, the robot transitions to the relocation stage. Our method again clusters all possible global exploration goals that have not been visited before and performs global tour planning. This empowers the robot to optimally navigate to different recognized but unexplored sub-areas. These two stages are repeated until no global exploration goals remain. In addition, a retrying mechanism is introduced to employ before the robot switches to the relocation stage from the exploration stage, in order to enhance the exploration robustness.

1.2 Problem Statement

In many domains, such as robotic exploration, urban planning, and environmental monitoring, efficiently exploring an entire region is a fundamental challenge. Conventional exploration techniques frequently depend on oversimplified approaches, like random traversal or pre-established pathways, which might not be the best in terms of coverage, time, or resource usage. These methods could result in overly frequent trips to specific locations, ineffective use of resources, or insufficient coverage of the intended area.

Geographic heterogeneity refers to the fact that many real-world environments have distinct features, such as resource distributions, terrain features, and environmental conditions, in different regions. This heterogeneity may go unnoticed by traditional exploration techniques, leading to inadequate coverage or wasteful resource use. Tasks involving exploration frequently have constraints on their time, energy, and senses. Efficient exploration requires making the most of the target area's coverage while adhering to these resource limitations.

Creating the best possible exploration route requires taking into account a number of variables, such as the overall exploration goal, resource limitations, coverage requirements, and spatial proximity between locations. All of these factors may be difficult for traditional methods to properly incorporate, resulting in less-than-ideal outcomes. The complexity of the exploration task increases exponentially with the size of the exploration area. Conventional techniques might not work well over wide regions, leading to computational bottlenecks or unworkable exploration plans. The exploration environment can be dynamic in many applications, changing over time as things change. This could involve adjustments to the topography, the allocation of resources, or the presence of environmental dangers.

Conventional approaches might not adjust well to these changing environments, resulting in antiquated exploration strategies or ineffective use of available resources.

To overcome these obstacles, a thorough strategy combining spatial clustering methods and tour planning algorithms is needed to divide the region into manageable clusters and organize effective exploration routes within each cluster. The proposed approach aims to optimize the exploration process and increase overall efficiency in full area exploration tasks by taking into account the spatial distribution of features, resource constraints, and exploration objectives [23][24][25][26].

1.3 Objectives

This thesis aims to address the main contributions and goals of the proposed research on tour planning and spatial clustering combined for effective full area exploration. These goals are set up in such a way as to direct the research process and offer a precise path towards accomplishing the intended results. To divide the given area into clusters of spatially proximate regions, the first goal is to investigate and evaluate different spatial clustering techniques. This entails researching more sophisticated approaches like hierarchical clustering or Gaussian mixture models, as well as more conventional ones like K-means clustering and density-based clustering (like DBSCAN). The goal is to comprehend the advantages and disadvantages of each clustering strategy and determine which one is best for the particular exploration task at hand.

The second goal is to look into various tour planning algorithms that can effectively navigate within each cluster in order to thoroughly explore each of its constituent regions. This entails learning about well-known algorithms like the genetic algorithm, ant colony optimization, and nearest neighbor algorithm in addition to possibly investigating cutting-edge techniques specific to the exploration setting. Evaluating each algorithm's performance with respect to computational efficiency, scalability, and exploration completeness is the goal. The third goal is to create a framework for combining tour planning and spatial clustering to maximize the area's exploration path. In order to create the best possible exploration path, this entails developing algorithms and techniques that seamlessly integrate the tour planning strategies with the area's clustered structure, which is derived from spatial clustering. The aim is to guarantee that the integrated framework can effectively investigate the whole region, taking into account variables like exploration coverage, travel distance, and cluster boundaries.

The proposed approach's performance under various scenarios and parameter settings will be assessed through a series of in-depth simulations, which constitute the fourth objective. This entails putting the integrated framework into practice and assessing its effectiveness in relation to alternative or baseline methodologies. The aim of this study is to evaluate the efficacy of the suggested methodology with respect to exploration efficiency, coverage, computational complexity, and environmental adaptability.

Validating the suggested method in real-world experiments conducted in realistic exploration scenarios is the fifth goal. This entails putting the integrated framework to use in actual settings, like cities, rural regions, or environmental monitoring stations. The aim is to evaluate the approach's practicality and suitability for actual exploration tasks and confirm its efficacy in the face of real-world uncertainties and constraints. The ultimate goal is to evaluate the outcomes of real-world experiments and simulations, analyses the data, and make judgements about the applicability and efficacy of the suggested strategy. This entails evaluating the method's advantages and disadvantages, pointing out possible directions for development or additional study, and talking about how it might affect a variety of industries, including robotics, environmental monitoring, and urban planning. All things considered, our goals are intended to direct the investigation towards gaining a thorough grasp of the suggested methodology, proving its efficacy via simulation-based assessments and practical validations, and offering perceptions into its possible uses and consequences for effective whole-area exploration.

1.4 Contributions

Our contributions can be summarized as follows:

(1) A novel exploration method is proposed to enhance the DSVP method by incorporating frontier clustering and tour planning, which leads to more efficient and reliable exploration in highly convoluted environments.

(2) We validate the efficiency and stability of our method in space exploration by testing it in a range of challenging simulated and real-world environments. The experimental results demonstrate that our method has advantages over the original DSVP method and other compared methods.

(3) Our code is open-source, allowing others to reproduce our results and conduct comparative analyses with various exploration techniques.

Chapter 2 Exploration

2.1 Related Work

Autonomous exploration of unknown environments has been extensively investigated in mobile robots, ranging from unmanned ground vehicle (UGV) [21][17][16] to unmanned aerial vehicle (UAV) [20][5], and others. The exploration task can be executed either by a single-robot or in a multi-robot configuration [8][14][2]. Typically, these methods can be categorized into two groups: traditional approaches and machine learning-based techniques. Traditional methods mainly rely on frontiers and viewpoints, employing a greedy strategy for exploring unknown environments. Most traditional techniques ensure completeness by guaranteeing full space coverage [16]. In contrast, machine learning-based methods [10][18] may not offer completeness, and the performance cannot always be guaranteed when transferring learned policy to new environments.

The central challenge in frontier-based exploration lies in frontier detection and selection. Typically, frontiers are identified as centroids of frontier edges, which are the lines separating known from the unknown space [19]. One approach to detecting frontiers [16] involves expanding an RRT [12] until its nodes extend into the unknown region of the map. Nodes within the unknown region are then considered as frontiers. To speed up the search for frontiers, multiple independently growing trees are utilized. In [15], an adaptive frontier detection method is proposed to enhance the successful sampling rate of RRT and solve the oversampling issue of RRT in sliding windows. Once frontiers are detected, a simple policy for selecting navigation goals is to choose the nearest frontier as the exploration target. Another policy involves selecting frontiers by minimizing map entropy concerning occupancy probabilities [5][17]. Instead of choosing a single frontier, some works [13] exploit all frontiers and employ tour planning to determine the sequence for visiting all frontiers. However, as the number of frontiers increases, the computation load of tour planner escalates.

Viewpoint-based exploration methods focus on the generation of viewpoints and the calculation of their exploration gain. The "Next-Best-View" Planner (NBVP) [3] is a well-known method that treats nodes in the RRT as viewpoints, assessing their exploration gain based on the number of unknown voxels observed from each viewpoint. The robot then plans its path towards the viewpoint with the highest gain. Another approach is the Graph-Based Exploration Planner (GBP) [6], which constructs a global rapidly exploring random graph to guide the robot towards unexplored spaces. The DSVP method [21] draws ideas from both frontier-based and viewpoint-based methods. It extends the local map boundary though an RRT biased by local frontiers. In DSVP, each RRT node serves as a viewpoint, and the RRT branch with the highest gain is employed for exploration. When no local frontiers remain, the robot is directed to different areas based on global frontiers during the relocation stage. DSVP has shown great performance across various unknown environments. In [20], viewpoints capable of covering local frontier clusters are generated, and a tour planner is employed to determine the optimal sequence for visiting these viewpoints. The tour planner operates on a small scale, ensuring computational efficiency. In [11], a rapid autonomous exploration method (FAEL) is introduced, taking into account factors such as information gain, movement distance, and coverage efficiency simultaneously. However, it falls short in ensuring complete space coverage. Furthermore, it does not address the requirement of returning to a home position, which is crucial in many scenarios, especially those with only one exit.

Our work presents an improvement to the DSVP method, involving rapid clustering of RRT nodes providing frontier information, selection of candidate navigation goals for each cluster, and efficient tour planning at both local and global scales. Through extensive experiments in a variety of highly convoluted environments, we demonstrate that our method outperforms the compared approaches in terms of efficiency and stability.

2.2 Problem Description

Let $S \subset R$ 3 be the full space being explored, which comprises of the known occupied space Socc, the known free space S_{free}, and the currently unknown space Sunk. The primary objective of the exploration task is to navigate autonomously within S and discover as much of the known space $S_{occ} \cup S_{free}$ as possible within a given time limit T_{lim}. The evaluation criteria for assessing the exploration performance include metrics such as area coverage (i.e., explored area volume, travel distance, and overall time) and exploration efficiency (i.e., explored area volume per second). In addition, the evaluation assumes precise robot localization and mapping, when measuring the exploration performance.

2.3 Our Approach

As shown in Figure 2-1, the proposed framework consists of several blocks of data structures and functions that can be classified into two stages: exploration and relocation. In the exploration stage, a local RRT is expanded within the free space of the environment, concurrently maintaining a local viewpoint graph. The identification of local frontiers is accomplished either by examining the graph vertices or the RRT nodes (Section 4.1.1). Additionally, these frontiers are further processed into individual clusters, yielding candidate goals (Section 4.1.2) which, in turn, serve as input for local tour planning (Section 4.1.3). The planned tour destination is taken as the navigation goal, with the local viewpoint graph facilitating the determination of the shortest path from the current robot position to the destination. When no clusters are within the current local planning horizon, the robot transitions to the relocation stage. In this stage, global frontier clusters are generated (Section 4.2.1) from the remaining local frontier clusters, and the global viewpoint graph is updated (Section 4.2.2). Each global frontier cluster is also represented by a global candidate goal. All global candidate goals are further organized into groups, and tour optimization is applied to ascertain the next navigation goal (Section 4.2.3). The robot transitions between the two stages until no candidate navigation goals remain. Furthermore, to enhance the method's robustness, a retrying mechanism is introduced, enabling the regeneration of the RRT for a repeated attempt at the exploration stage before transitioning to the relocation stage.

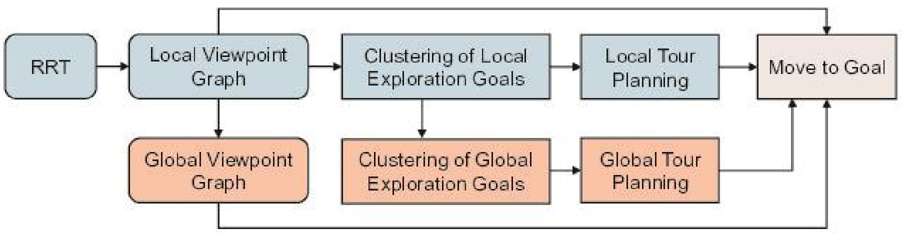


Fig. 2-1. Exploration Method Framework: Modules Active in Exploration (Blue) and Relocation (Orange) Phases.

2.3.1 Exploration Stage

2.3.1.1 Detecting Local Exploration Goals

In traditional frontier-based exploration [19], frontiers $\mathcal{F}$ are defined as known-free voxels adjacent to unknown voxels. Here, a known-free voxel v is inspected by examining its neighborhood voxels $\{u\}$ within a cuboidal region $\mathcal{B}_v \subset \mathbb{R}^3$ centered at v. If there exist unknown neighbor voxels with the number exceeding a given threshold λ_{num}, the voxel is classified as:

$$v \in \mathcal{F} \Leftrightarrow n(u) > \lambda_{num}, \forall u \in \mathcal{B}_v \wedge v \in \mathcal{S}_{free} \wedge u \in \mathcal{S}_{unk} \tag{2-1}$$

Local frontiers $\mathcal{F}_L$ are those found within a local planning horizon denoted as $\mathcal{H} \subset \mathbb{R}^3$, centered at the robot position at the start of each exploration planning iteration. A voxel v is considered a local frontier if it meets:

$$v \in \mathcal{F}_L \Leftrightarrow v \in \mathcal{F} \wedge v \in \mathcal{H} \tag{2-2}$$

Figure 1-1 also illustrates the process of detecting local frontiers. Initially, an RRT is dynamically expanded from the robot position to generate viewpoints within $\mathcal{H}$. Each node on the tree corresponds to a viewpoint, and a local viewpoint graph G_L is constructed using these viewpoints as vertices. The local viewpoint graph helps plan a local navigation path from the current position to a local destination. Subsequently, frontiers are chosen from these viewpoints, defined as $\{\mathcal{F}_i\}$, $i = 1, \cdot\cdot\cdot, M$, where M is the number of frontiers. Specifically, any vertex v on the graph that meets Equation (2-2) is regarded as a local frontier. Each local frontier $\mathcal{F}_i$ is associated with a gain value:

$$V(\mathcal{F}_i) = n(u), \forall u \in \mathcal{B}_{\mathcal{F}_i} \wedge u \in \mathcal{S}_{unk} \tag{2-3}$$

which is used to select the most promising frontiers for building the global viewpoint graph.

2.3.1.2 Clustering Local Frontiers

The identified local frontiers FL are divided into K clusters based on their spatial proximity:

$$\mathcal{F}_L = \mathcal{F}_L^1 \cup \cdots \cup \mathcal{F}_L^K, \mathcal{F}_L^i \cap \mathcal{F}_L^j = \varnothing, 1 \le i, j \le K, i \ne j \tag{2-4}$$

Frontiers are assigned to the same cluster if their Euclidean distance is less than a tolerance parameter ϵ, specifying the minimum distance between any two clusters:

$$\mathcal{F}_L^i \cap \mathcal{F}_L^j = \varnothing \Leftrightarrow d(u,v) > \epsilon, \forall u \in \mathcal{F}_L^i, \forall v \in \mathcal{F}_L^j \tag{2-5}$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance between two spatial points. Note that there is another inherent constraint for frontier clustering. The frontiers are sampled from a local RRT, and the robot can travel freely between any two frontiers without encountering obstacles. This means that, in our context, two frontier clusters positioned on opposite sides of a thin wall can be merged into a single cluster, as long as they originate from the same RRT sampling.

For each local frontier cluster $\mathcal{F}_L^k$, its centroid is calculated by averaging the 3D positions of all frontiers within the cluster:

$$c_L^k = \frac{1}{\left|\mathcal{F}_L^k\right|}\sum_i \mathcal{F}_i, \mathcal{F}_i \in \mathcal{F}_L^k \tag{2-6}$$

where $\left|\mathcal{F}_L^k\right|$ is the number of local exploration goals in cluster $\mathcal{F}_L^k$.

As shown in Figure 2-2, a navigation goal candidate g_L^k is selected for each local cluster $\mathcal{F}_L^k$. This goal is the furthest point along the direction from the robot position P_{rob} to c_L^k.

In addition, a slight tolerance of direction angle between $\frac{\overrightarrow{g_L^k - c_L^k}}{\left\|g_L^k - c_L^k\right\|}$ and $\frac{\overrightarrow{c_L^k - P_{rob}}}{\left\|c_L^k - P_{rob}\right\|}$ is allowed. This is done to ensure that the local navigation goal is distant from the current robot position, promoting the coverage of more unknown areas for the next iteration of planning. In the end, K local exploration goal clusters will result in K local navigation goal candidates $\{g_L^k\}, 1 \le k \le K$.

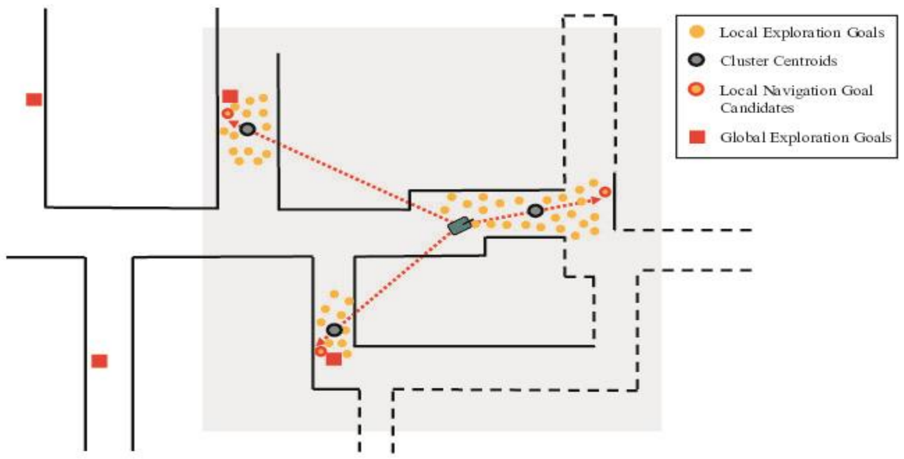

● Local Exploration Goals
● Cluster Centroids
● Local Navigation Goal Candidates
■ Global Exploration Goals

Fig. 2-2. The process of generating local navigation goal candidates. For each exploration goal cluster, the centroid is calculated. The vector from the robot position to the centroid is determined. The furthest point in the vector direction with a slight angle tolerance is selected as the navigation goal

candidate.

2.3.1.3 Local Tour Planning

The next question is how to determine the best local exploration goal from the candidate set. Inspired by [20], we propose solving it as a variant of the Traveling Salesman Problem (TSP). The local candidate goals $\{g_L^k\}, 1 \le k \le K$ and the global home position ghome serve as places to be visited by a salesman, starting from the current place P_{rob}. The goal is to compute an optimal open-loop tour that passes through all the local candidate goals and ends at the global home position. Unlike the standard TSP, where the final place to visit is the starting place, here the final place is a specified place (i.e., $g_{\mathrm{hom}e}$). We model this as an Asymmetric TSP (ATSP) by designing the cost matrix C^{ATSP} accordingly. $C_{r,c}^{ATSP}$ is a $(K + 1)$ × $(K + 1)$ square matrix, where the entry $C_{r,\ c}^{ASTP}$ at row r and column c denotes the traveling cost from place r to place c. The cost to travel from the current robot position $Prob$ to the k-th local candidate goal g_L^k is:

$$C_{0,k}^{ATSP} = L(P_{rob}, g_L^k) + H(P_{rob}, g_L^k), k \in \{1, \cdots, K\}, \tag{2-7}$$

where L(·,·) calculates the shortest path length between two positions according to the local viewpoint graph, and H(·,·) calculates the cost of changing the robot heading to the next position. The cost between any two local goals is:

$$C_{r,c}^{ATSP} = C_{c,r}^{ATSP} = L(g_L^r, g_L^c), r, c \in \{1, \cdots, K\}. \tag{2-8}$$

Considering the first column of C^{ATSP}, the item $C_{k,\ 0}^{ASTP}$ normally denotes the cost traveling from g_L^k to Prob. We substitute it with the cost traveling from g_L^k to $g_{\mathrm{hom}e}$:

$$C_{k,0}^{ATSP} = L(g_L^k, g_{home}), k \in \{1, \cdots, K\}, \tag{2-9}$$

where the shortest path length is calculated according to the global viewpoint graph. The advantage of including $ghome$ in local tour planning will be investigated in Section 5.3. Based on the cost matrix, the ATSP can be solved using existing algorithms. Once the optimal open-loop tour is obtained, the robot proceeds to the first destination on the tour. After reaching the initial goal, the local planning horizon H is adjusted accordingly, and the exploration stage is repeated until no frontier clusters remain within H.

It’s important to note that the TSP is kept small-scale and easy to solve by controlling the number of places in the tour, determined by the size of the local planning horizon and the clustering process. Additionally, the global home position imposes a rigid constraint on the

TSP, ensuring the robot returns home regardless of the order in which local candidate goals are visited. However, the global home position is not selected as the next navigation goal during the exploration stage.

2.3.2 Relocation Stage

When there are no remaining exploration goal clusters within H, the robot enters the relocation stage. However, it's important to note that the algorithm's parameters may not always suit all types of terrains and environmental structures. Randomly sampling spatial points can sometimes result in an RRT with limited variation or even degeneration. In such cases, the robot may fail to explore certain areas effectively. To enhance the success rate of exploration, we introduce a retrying mechanism before transitioning from exploration to relocation. Specifically, when there are no exploration goal clusters within H, the RRT is regenerated for further exploration. If the robot still cannot find any exploration goal clusters, it proceeds to the relocation stage. This stage involves clustering global exploration goals, updating the global viewpoint graph, and planning a global tour.

2.3.2.1 Clustering Global Exploration Goals

In each iteration of local tour planning, any remaining local navigation goal candidates that have not been visited before are added to the list of global exploration goals. However, because the information of navigation goals can change during the exploration process, each global exploration goal is double-checked at the beginning of the relocation stage to confirm whether it still represents an unexplored space. This verification can be quickly performed using Equation (2-1), where the searched space Bv is doubled in x and y directions. Any global exploration goals that no longer satisfy Equation (2-1) are removed from the list. The updated global exploration goals are then denoted as $\{g_G^n\}, 1 \le n \le N$.

In large-scale and convoluted environments, the number of global exploration goals, denoted as N, can become excessively large. This will pose a significant challenge for the tour planning process. To address this issue, we propose the utilization of clustering technology to reduce the number of travel destinations when N exceeds a predefined threshold, denoted as ρ_{num}. Specifically, each global exploration goal is assigned a cost value representing the shortest path length from the global home position to itself. Any two global exploration goals with a cost difference less than ϵ_{goal} are grouped into the same cluster. Within each cluster, we select the exploration goal that is furthest from the home position as a global navigation goal candidate. This approach significantly reduces the number of global navigation goal candidates.

2.3.2.2 Updating Global Viewpoint Graph

The global viewpoint graph plays an important role in computing the travel cost between any two global navigation goal candidates, as well as planning the shortest navigation path from the current robot position to a desired destination within a large-scale space. Following the approach in [21], in each iteration of the exploration stage, the local graph vertices on the shortest path from the robot to vertices with positive gain values are added to the global viewpoint graph. The edges are updated accordingly. The global viewpoint graph provides a sparse representation of the environment while still providing short paths between viewpoints.

2.3.2.3 Global Tour Planning

During the relocation stage, the primary objective is to find an optimal global tour that guides the robot to visit all global navigation goals and return to the global home position. This is also achieved by implementing an ATSP. The cost matrix, which has a size of (N +1)×(N + 1), is built using Equations (2-7) - (2-9) without considering the heading cost. Here, the local candidate goal $\{g_L^k\}, 1 \le k \le K$ are replaced by $\{g_G^n\}, 1 \le n \le N$.

Since the number of global navigation goal candidates is suppressed by the clustering step, the global tour planning process can be completed with a satisfactory computational load. The number of iterations required for global tour planning or relocation is also decreased, resulting in significant savings in navigation time. We will investigate the effectiveness of this approach in an ablation study, as discussed in Section V-C.

2.3.3 Algorithm Implementation

Algorithm 1 provides an overview of the entire exploration process. Throughout this process, a Lidar Odometry and Mapping (LOAM) system plays a important role in estimating the robot’s states and generating the resultant map. Additionally, a terrain travers ability analysis module comes into play, generating a terrain map that provides the robot with obstacle information. Based on the terrain map and globally aligned laser scans, two maps, M_{occ} and M_{octo}, are updated at both semantic and metric levels, respectively. Within a defined local planning horizon, an RRT is dynamically expanded based on the information from these semantic and metric maps. The subsequent steps (Lines 7 ～ 17) in the algorithm involve frontier detection, local exploration goal clustering, and local tour planning, as described in Section IV-A. If no local exploration goal clusters are found during the exploration stage, the algorithm allows for one more attempt before transitioning to the relocation stage. The following steps implement the relocation stage as described in Section IV-B. Line 23 shows the process of updating global exploration goals, which involves doublechecking already existing goals and incorporating newly detected local navigation goals. All global exploration goals are then clustered and employed for tour planning (Lines 25 ～ 30). The strategic application of clustering technology to reduce traveling destinations for tour planning significantly limits computation overhead, especially in large-scale and convoluted environments. Upon successfully navigating the robot to its planned target, it transitions back to the exploration stage. These two stages alternate until no global exploration goals remain. Eventually, the robot returns to its home position, signifying the completion of the exploration task.

Algorithm 1 Exploration in Unknown Environments

```
Input: LOAM result and terrain map
Output: Explored map of the environment
1:  while Try exploration stage do
2:      $\mathcal{H} \leftarrow$ updatePlanningHorizon()
3:      $M_{occ} \leftarrow$ updateOccupancyGridMap()
4:      $M_{octo} \leftarrow$ updateVolumetricMap()
5:      $\mathcal{T} \leftarrow$ dynamicRRT($\mathcal{H}$, $M_{occ}$, $M_{octo}$)
6:      $\mathcal{G}_L \leftarrow$ updateLocalGraph($\mathcal{T}$)
7:      $\mathcal{F}_L \leftarrow$ detectLocalFrontiers($\mathcal{T}$, $M_{octo}$)
8:      if $\mathcal{F}_L \neq \emptyset$ then
9:          $\mathcal{G}_G \leftarrow$ updateGlobalGraph($\mathcal{F}_L$)
10:         $\{\mathcal{F}_L^j\} \leftarrow$ clusteringLocalExplorationGoals($\mathcal{F}_L$)
11:         Remove clusters containing no more than $\tau_c$ (e.g. $\tau_c = 3$) points.
12:         if $\{\mathcal{F}_L^j\} \neq \emptyset$ then
13:             $\{g_L^k\} \leftarrow$ generateLocalNavigationGoalCandidates($\{\mathcal{F}_L^j\}$)
14:             $g \leftarrow$ planningTour($\{g_L^k\}$, $P_{rob}$, $g_{home}$, $\mathcal{G}_L$, $\mathcal{G}_G$)
15:             Do path planning and navigate the robot to target $g$.
16:         end if
17:     end if
18:     if $\mathcal{F}_L = \emptyset$ or $\{\mathcal{F}_L^j\} = \emptyset$ then
19:         Retry exploration stage one more time.
20:     end if
21: end while
22: Transitions to relocation stage.
23: $\{g_G^n\} \leftarrow$ updateGlobalExplorationGoals($\{g_G^n\}$, $\{g_L^k\}$)
24: if $\{g_G^n\} \neq \emptyset$ then
25:     if $|\{g_G^n\}| \geq \rho_{num}$ then
26:         $\{g_G^i\} \leftarrow$ clusteringGlobalExplorationGoals($\{g_G^n\}$)
27:         $g \leftarrow$ planningTour($\{g_G^i\}$, $P_{rob}$, $g_{home}$, $\mathcal{G}_G$)
28:     else
29:         $g \leftarrow$ planningTour($\{g_G^n\}$, $P_{rob}$, $g_{home}$, $\mathcal{G}_G$)
30:     end if
31:     Do path planning and navigate the robot to target $g$.
32:     Transitions to exploration stage.
33: else
34:     Return home and complete the exploration task.
35: end if
```

2.4 Experiments

We begin by evaluating the performance of our proposed method in various simulated environments and comparing it with the DSVP method [22], which is a challenging autonomous exploration method. Given that DSVP has already compared with the state-of-the-art methods such as NBVP [3] and GBP [6], demonstrating its superior performance, we do not replicate those results in this paper. Additionally, we compare our method with the FAEL method [12], which also employs tour planning for exploration. Furthermore, we conduct ablation studies to investigate the necessity of specific key operations within our method. Finally, we validate our proposed method through testing on our mobile robot in real-world environments.

2.4.1 Setup

For the evaluation, we employ a benchmark exploration dataset [4] containing five exploration scenarios: *indoor, campus, garage, tunnel, and forest*. These environments vary in characteristics, with indoor, tunnel and forest environments featuring narrow passages predominantly, while campus and garage environments offer more spacious areas. These environments are all large scale and highly convoluted. They are loaded in Gazebo simulator and are unknown to all the methods. All the methods are implemented in the Robot Operating System (ROS) framework. The benchmark exploration platform [4] includes an ROS package called "vehicle simulator" to simulate robot kinematics and provide accurate location information. The maximum linear and angular velocities of the simulated robot are set to 2m/s and 90deg/s, respectively. Other essential ROS packages for exploration, including collision avoidance, terrain travers ability analysis, and path following, are also utilized. DSVP and our planner are launched as independent ROS nodes.

In the implementation, we define $\mathcal{B}_v$ as a space of $10 \times 10 \times 0.8\ m^3$ in Equation (2-1) and H as a space of $30\times30\times10\ m^3$ in Equation (2-2). λ_{num} in Equation (2-1) is set to 10 for narrow environments and 40 for spacious environments. For clustering local exploration goals, we choose $\epsilon = 2m$ in Equation (2-5) for narrow environments and $\epsilon = 3m$ for spacious environments. Clusters with fewer than 3 points are removed. When solving the ATSP for local and global tour planning, a Lin-Kernighan traveling salesman heuristic solver [10] is employed. In Equation (2-7), the heading cost is defined as $\frac{\alpha}{\pi}\cdot 20$, where α represents the radians by which the robot should change its heading. If the number of global exploration goals for global tour planning exceeds $\rho_{num}=40$, the goals are spatially clustered with a tolerance of $\epsilon_{goal}=10m$. Both the DSVP method and our method share default parameters for operations such as RRT generation, graph updates, 3D occupancy mapping, and other relevant activities.

Each method is run 10 times, with a run ending under the following conditions [22]: the exploration algorithm reports completion, the robot moves less than 10m within 300s, or a time limit is reached. We compute the average performance across these 10 runs to compare the methods. All simulations and evaluations are conducted on a Desktop PC equipped with an Intel Core i9-10900KF CPU and 32GB of memory running Ubuntu 20.04.

2.4.2 Results

2.4.2.1 Area Coverage and Exploration Efficiency

Table 2-1 provides an overview of the volumes explored by the compared methods, with the maximum volume across all runs for each environment serving as the ground truth area volume. A run is considered unsuccessful if the explored volume is less than 98% of the ground truth. Table 2-2 compares the exploration performances of DSVP and our method. The performance metrics are calculated based on the successful runs.

TABLE 2-1 ENVIRONMENT VOLUME ESTIMATED BY THE RUNS.

Env.	Maximum Volume (m^3)			98% of the
	DSVP	Ours	All	Maximum Volume (m^3)
indoor	5378.8	5377.1	5378.8	5271.2
campus	46101.6	46196.3	46196.3	45272.3
garage	42445.5	42291.6	42445.5	41596.6
tunnel	22072.9	22024.6	22072.9	21631.4
forest	42580.4	43128.4	43128.4	42265.8

TABLE 2-2 COMPARISON OF EXPLORATION PERFORMANCE.

Env.	Method	Success Rate (%)	Explored Volume (m^3)		Traveling Distance (m)		Overall Time (s)		Explored Volume Per Second (m^3/s)	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
indoor	DSVP	80	5357.4	11.0	1469.3	92.0	887.4	67.3	6.07	0.41
	Ours	90	5364.8	10.5	1444.2	42.3	837.5	28.2	6.41	0.21
campus	DSVP	80	45804.4	175.4	2613.9	39.1	1348.6	27.9	33.98	0.68
	Ours	100	46020.1	122.8	2631.5	24.2	1354.6	12.3	33.98	0.33
garage	DSVP	70	42367.9	49.4	5105.7	247.1	2728.2	124.1	15.56	0.74
	Ours	100	42239.4	40.2	4114.7	204.3	2216.1	107.1	19.1	0.87
tunnel	DSVP	100	22046.9	22.3	6673.7	252.8	3612.2	141.9	6.11	0.23
	Ours	100	21956.5	90.4	6033.4	152.8	3238.6	88	6.78	0.17
forest	DSVP	50	42492	89.4	2104.6	44.3	1113.2	16.5	38.18	0.50
	Ours	100	42748.1	252.4	2053.5	103.7	1098.9	57.7	38.99	1.76

In the indoor environment, characterized by long and narrow corridors connected with lobby areas, both methods may fail to go through spaces with guard rails, as shown in Figure 2-3, due to the randomness of RRT expansion. But our method demonstrates a slightly higher success rate of 90% compared to DSVP's 80%. This is attributed to our method's bias-free RRT expansion, which favors finding sparse local candidate goals that can be reached by the robot. DSVP, on the other hand, biases RRT expansion toward frontiers in large-scale spaces.

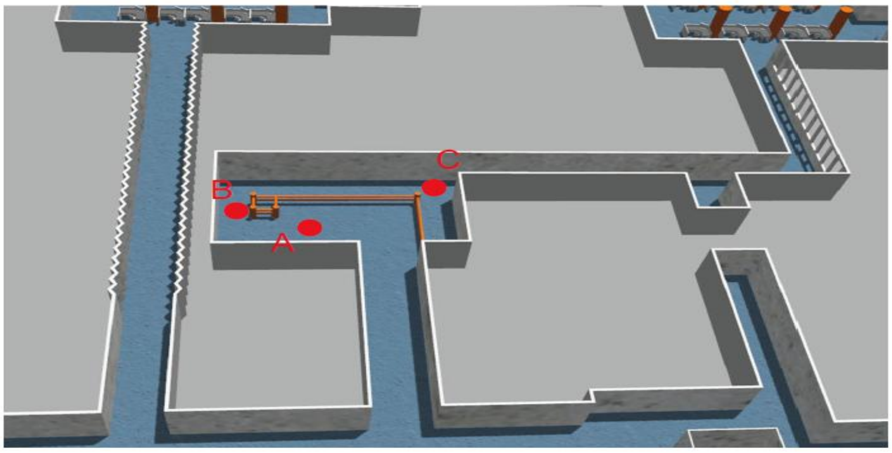


Fig. 2-3. The space with guard rails which occasionally hinders the robot's passage. During exploration failures, the robot may successfully reach Point A but faces difficulties progressing to Point C.

This is because the laser scans can penetrate the guard rails, resulting in fewer frontiers being detected at Point B. If the RRT cannot find a feasible path for extending from Point B to Point C, the robot will be unable to reach its destination at Point C via Point B.

Figure 2-4 shows the exploration trajectories that are the best of the 10 runs. The best trajectory is from the successful runs and with the highest exploration efficiency. On average, DSVP completes the exploration after traveling 1469.3 meters over 887.4 seconds, while our method takes 837.5 seconds and travels 1444.2 meters. With tour planning, our method facilitates more efficient travel to candidate goals. Despite the need to solve the ATSP, the heuristic solver efficiently handles the limited number of candidate goals in our method. In terms of exploration efficiency, our method covers 6.41 m^3/s, while DSVP covers 6.07 m^3/s. Notably, the standard variances of all metrics for our method are lower than those for DSVP, indicating that our method is more efficient and stable.

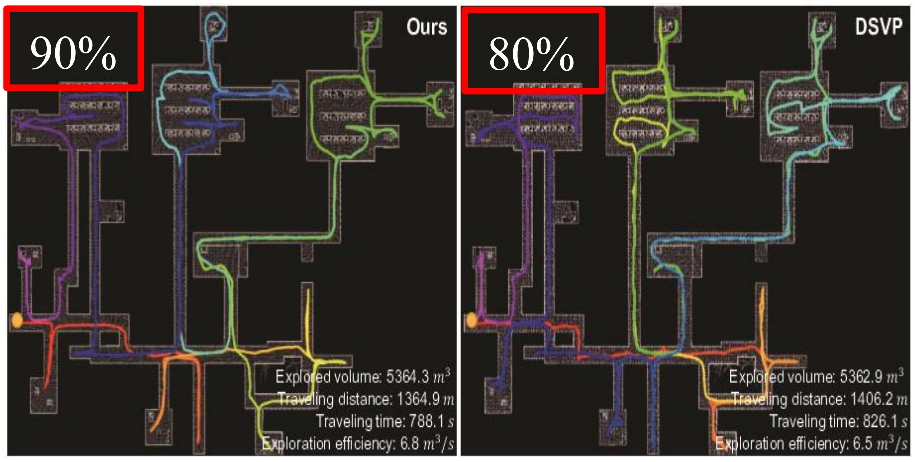

90%
Ours
Explored volume: 5364.3 m^3
Traveling distance: 1364.9 m
Traveling time: 788.1 s
Exploration efficiency: 6.8 m^3/s
80%
DSVP
Explored volume: 5362.9 m^3
Traveling distance: 1406.2 m
Traveling time: 826.1 s
Exploration efficiency: 6.5 m^3/s

Fig. 2-4. The resulting maps and trajectories of the compared methods for the indoor environment.

In the campus environment with undulating terrains, DSVP achieves a success rate of 80%, while our method achieves 100% by incorporating the retrying mechanism for RRT expansion. However, regenerating the RRT and re-finding local exploration goals add extra time to our method's process. On average, both methods spend comparable time and travel similar distances to complete the explorations, achieving an exploration efficiency of 33.98 m^3/s. As shown in Figure 2-5, the best trajectory in DSVP outperforms ours.

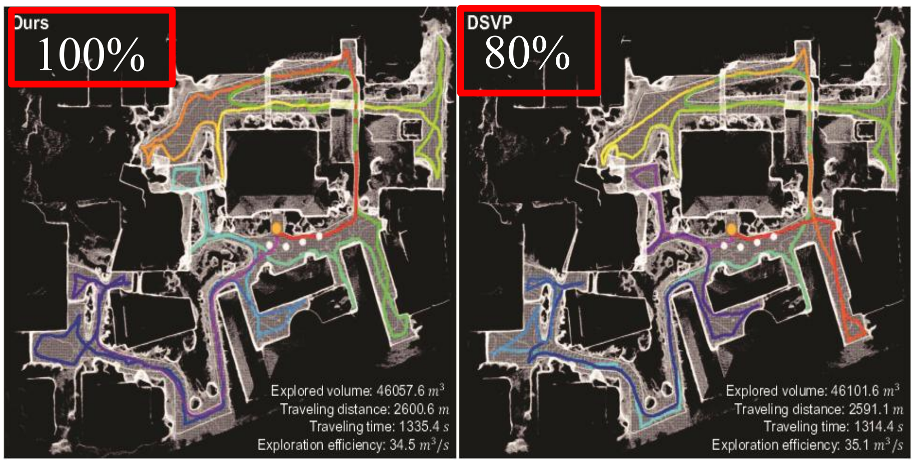

Ours
100%
Explored volume: 46057.6 m^3
Traveling distance: 2600.6 m
Traveling time: 1335.4 s
Exploration efficiency: 34.5 m^3/s
DSVP
80%
Explored volume: 46101.6 m^3
Traveling distance: 2591.1 m
Traveling time: 1314.4 s
Exploration efficiency: 35.1 m^3/s

Fig. 2-5. The resulting maps and trajectories of the compared methods for the campus environment.

In this environment, where lanes form a star network, tour planning does not significantly impact exploration in our method. Again, our method exhibits lower standard variances across all metrics, indicating greater stability. In the garage environment, featuring multiple floors and sloped terrains, DSVP faces challenges and achieves a success rate of only 70%. In contrast, our method successfully completes the exploration in all runs. On average, our method travels 991 meters less distance, takes 512 seconds less time, and achieves a 3.5 m3/s higher exploration efficiency compared to DSVP. Moreover, our method exhibits lower standard variances for explored volume, traveling distance, and overall time, indicating superior efficiency and stability. Qualitative comparisons of the best trajectories, as shown in figure 2-6, support these findings.

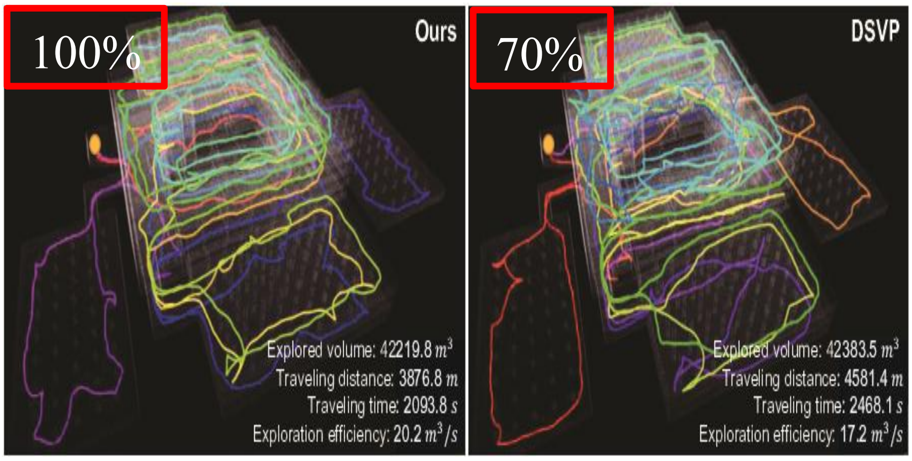

100%
Ours
Explored volume: 42219.8 m^3
Traveling distance: 3876.8 m
Traveling time: 2093.8 s
Exploration efficiency: 20.2 m^3/s
70%
DSVP
Explored volume: 42383.5 m^3
Traveling distance: 4581.4 m
Traveling time: 2468.1 s
Exploration efficiency: 17.2 m^3/s

Fig. 2-6. The resulting maps and trajectories of the compared methods for the garage environment.

In the tunnel environment, characterized by a complex network of tunnels, both methods complete the exploration task for all runs. However, our method outperforms DSVP in terms of traveling distance (640 meters less), time (374 seconds less), and exploration efficiency (0.67 m^3/s higher). Additionally, our method has lower standard variances for most metrics compared to DSVP. Figure 2-7 shows a qualitative comparison of the best trajectories, demonstrating that our method explores a comparable volume of space with less time and distance.

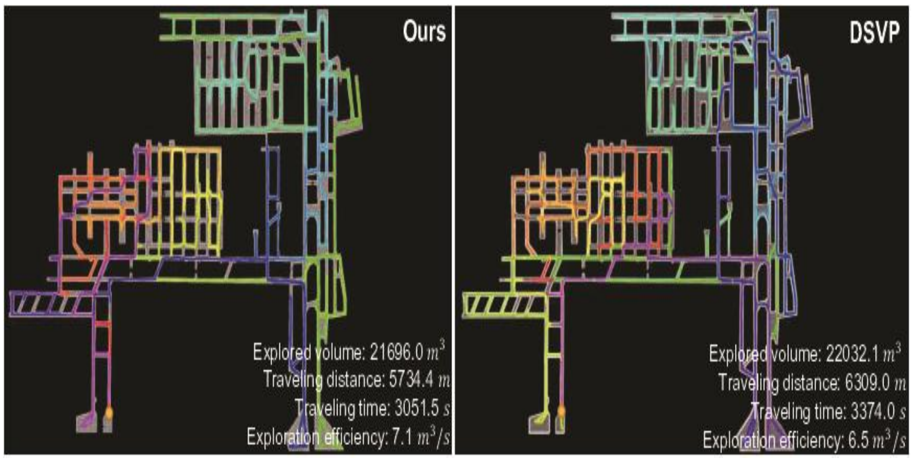

Ours
Explored volume: 21696.0 m^3
Traveling distance: 5734.4 m
Traveling time: 3051.5 s
Exploration efficiency: 7.1 m^3/s
DSVP
Explored volume: 22032.1 m^3
Traveling distance: 6309.0 m
Traveling time: 3374.0 s
Exploration efficiency: 6.5 m^3/s

Fig. 2-7. The resulting maps and trajectories of the compared methods for the tunnel environment.

In the forest environment with cluttered trees, DSVP tends to explore the space coarsely, resulting in the oversight of many small spaces and a 50% success rate. Our method is able to complete all exploration runs but exhibits slightly higher standard variance. In terms of other metrics, both methods demonstrate comparable performance overall. However, our method achieves a slightly higher exploration efficiency than DSVP, with an additional 0.8 m^3/s. Qualitative comparisons of the best trajectories for each method, as shown in Figure 2-8, further highlight our method's superior exploration efficiency.

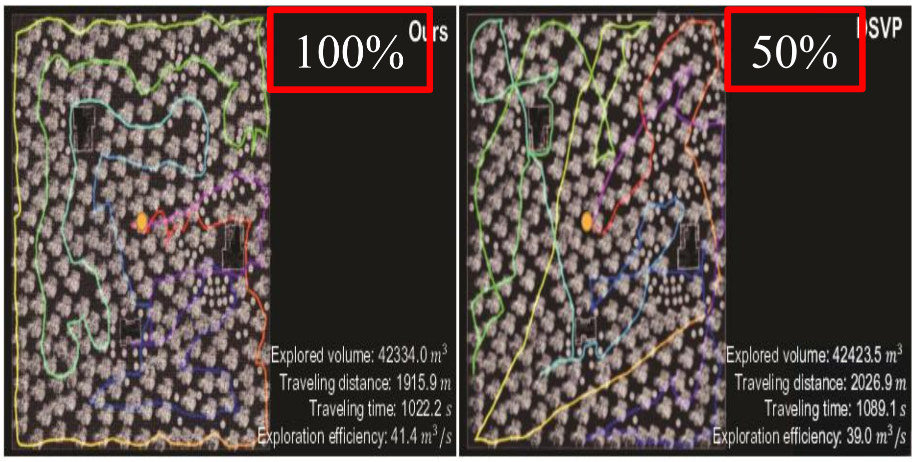

100%
Ours
Explored volume: 42334.0 m^3
Traveling distance: 1915.9 m
Traveling time: 1022.2 s
Exploration efficiency: 41.4 m^3/s
50%
DSVP
Explored volume: 42423.5 m^3
Traveling distance: 2026.9 m
Traveling time: 1089.1 s
Exploration efficiency: 39.0 m^3/s

Fig. 2-8. The resulting maps and trajectories of the compared methods for the forest environment.

2.4.2.2 Computational Efficiency

To investigate the computational efficiencies of the methods, the planning iteration count and average planning runtime for each exploration run are recorded. The average values across all runs are presented in Table 2-3. Our method generally requires fewer planning iterations compared to DSVP in most environments. In the indoor, tunnel, and forest environments, our method also exhibits shorter runtimes than DSVP. However, it's worth noting that in the campus and garage environments, which feature numerous wide spaces, our method generates a larger number of candidate goals, leading to additional time for planning and target selection. Nevertheless, the average planning runtime across all environments remains at approximately 0.45 seconds, highlighting the efficiency and suitability of our exploration algorithm for real-world mobile robot deployment.

TABLE 2-3 COMPARISON OF COMPUTATION EFFICIENCIES.

Env.	Method	Planning Iteration Count	Avg. Planning Runtime (s)	Total Planning Time (s)
indoor	DSVP	260	0.54	140.40
	Ours	242	0.48	116.16
campus	DSVP	239	0.44	105.16
	Ours	217	0.54	117.18
garage	DSVP	649	0.41	266.09
	Ours	602	0.46	276.92
tunnel	DSVP	779	0.37	295.63
	Ours	654	0.35	228.90
forest	DSVP	195	0.48	93.60
	Ours	227	0.41	93.07

2.4.2.3 Comparison with other methods

We also execute the open-source FAEL code [12] in the same five simulated environments for multiple runs. As shown in Figure 2-9, the FAEL method does not prioritize complete area coverage and returning home. It exhibits limitations in fully exploring highly convoluted environments. Sometimes, there may be frequent switches in navigation target selection, particularly within highly convoluted spaces. FAEL appears to be better suited for large unconvoluted spaces, where it balances factors like information gain, movement distance, and coverage efficiency effectively. Since FAEL fails in exploration of the garage environment, characterized by multiple floor and sloped terrains, its compatibility with multi-floor environments remains uncertain.

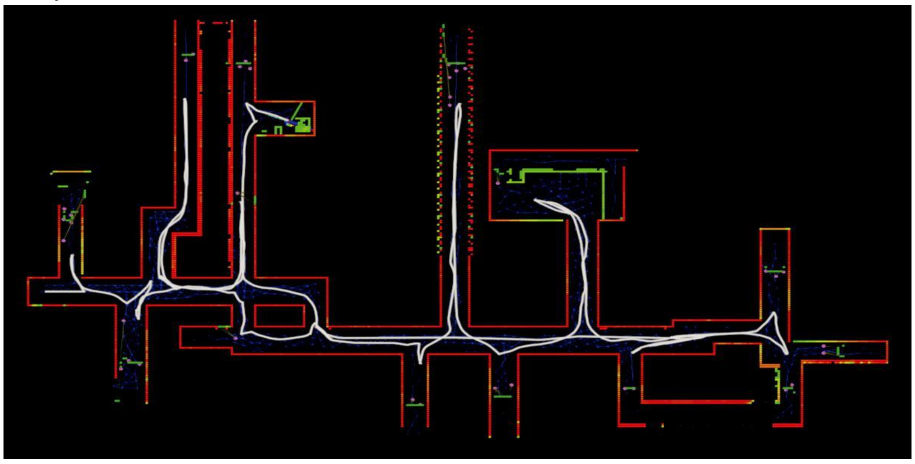


Fig. 2-9. Exploration result of the indoor environment. The white lines denote the trajectories of the robot. It shows that the FAEL method exhibits limitations in fully exploring the highly convoluted environment.

2.4.3 Ablation Studies

2.4.3.1 Local tour planning without considering global home position

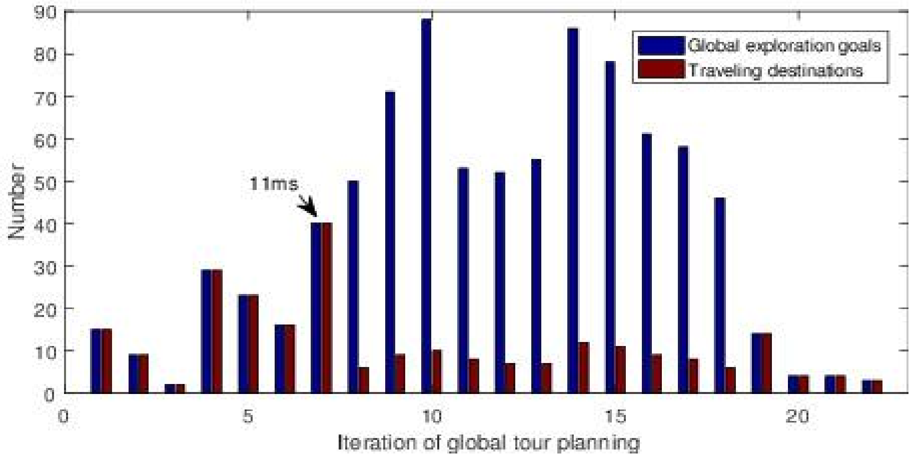
As indicated in Equation (2-9), the cost of traveling from any local candidate goal to the current robot position is defined as the distance from the candidate goal to the global home position. We investigate the scenario where the global home position is not taken into account, specifically by setting C AT SP k,0 to 0. The revised version of our method is referred to as "Ours_R1". We perform 10 exploration runs in the indoor environment. The average performance results are presented in Table 2-4. It shows that, when the global home position is not considered, the method yields comparable exploration coverage and traveling distance but requires more exploration time. Consequently, the exploration efficiency is lower compared to the method that takes the home position into account.

TABLE 2-4 COMPARISON OF EXPLORATION PERFORMANCE WHETHER CONSIDERING GLOBAL HOME POSITION OR NOT IN LOCAL TOUR PLANNING.

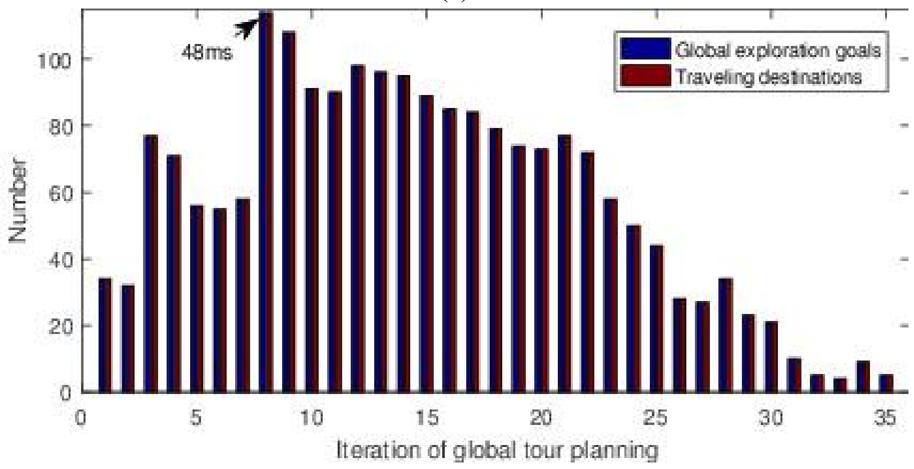
Method	Explored Volume (m^3)	Traveling Distance (m)	Overall Time (s)	Explored Volume Per Second (m^3/s)
Ours	5364.8	1444.2	837.5	6.41

Ours_R1	5372.3	1436.2	858.9	6.26

Global tour planning without clustering global exploration goals. In the global tour planning process, the traveling destinations are selected from the global exploration goals for the ATSP. In the experiment, if the number of global exploration goals exceeds 40, they are clustered, resulting in a significant decrease in the number of traveling destinations. Figure 2-10(a) shows the change in the number of selected traveling destinations throughout each iteration of global tour planning, when executing exploration in the garage environment. Solving the ATSP with 40 nodes takes about 11 milliseconds. Figure 2-10(b) shows the results when clustering is not utilized, during another exploration in the same environment. The maximum number of traveling destinations reaches 114, requiring 48 milliseconds to solve the ATSP. As the number of global exploration goals increases, more iterations of tour planning or relocation are required. Consequently, this leads to an extended overall exploration time.



(a)



(b)

Fig. 2-10. The number of traveling destinations in global tour planning. When utilizing clustering (a), the number of traveling destinations is significantly reduced, requiring a maximum of 11 milliseconds to solve the ATSP. Without clustering (b), the number of traveling destination increases significantly, resulting in longer processing times to solve the ATSP and more iterations of tour planning.

Switching to the relocation stage without using retrying mechanism. The effectiveness of employing the retrying mechanism is examined before the robot switches to the relocation stage when no local exploration goals are found. Since the campus and garage environments are characterized by many uneven terrains, they have a more significant influence on the RRT expansion during the exploration stage. We conduct 10 exploration runs in both environments without using the retrying mechanism. The success rates of exploration in the campus and garage environments are 90% and 0%, respectively. The experimental result shows that the RRT has a high possibility of failing to reach unexplored area when the robot is climbing the ramp between floors. Without attempting to generate the RRT once more, the robot stops exploration and is relocated to a previously recognized space that has not been explored. As shown in Table 2-2, the success rate of exploration reaches 100% in both environments when the retrying mechanism is employed, validating its effectiveness.

2.4.4 Test in Real-world Environments

We deploy our exploration package and the benchmark navigation stack [4] on our mobile robot platform, which is a four-wheel differential driving mobile robot as shown in Figure 2-11. For autonomous exploration, our robot is equipped with a Velodyne VLP-16 lidar and a 6-axis IMU. To handle robot localization and mapping, we utilize a modified version of the LIO-SAM package [15], which implements a Realtime lidar-inertial odometry. The robot’s maximum linear and angular velocities are set to 0.5 m/s and 15 deg/s, respectively. All packages run on an onboard computer with a 4.8GHz i7 CPU and 32GB RAM.

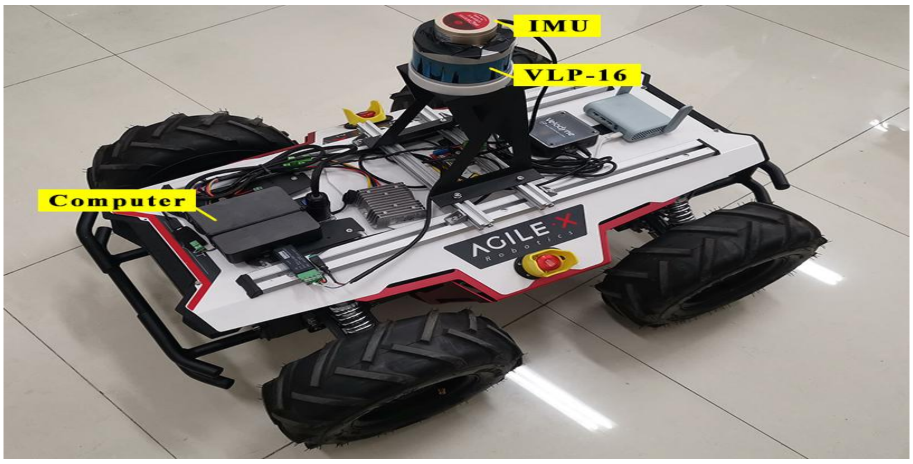


Fig. 2-11. Our mobile robot. For autonomous exploration, it is equipped with a Velodyne VLP-16 lidar and a 6-axis IMU. All packages run on an onboard computer with a 4.8GHz i7 CPU and 32GB RAM.

The first experiment is conducted in a single floor of a building on our university campus, as shown in Figure 2-12. The environment consists of several narrow corridors, intersections, doors and dead ends. Figure 2-12 also displays the resulting map and final trajectory obtained by our method. To prevent the robot from entering the bathroom, we placed some static obstacles within the environment. Overall, our method spends 760.2 seconds traveling a distance of 230.5 meters and covering an area of 1928.3 cubic meters. Given the low-speed setting of our mobile platform and the relatively enclosed corridor environment, the exploration efficiency is measured at 2.54 m^3/s.

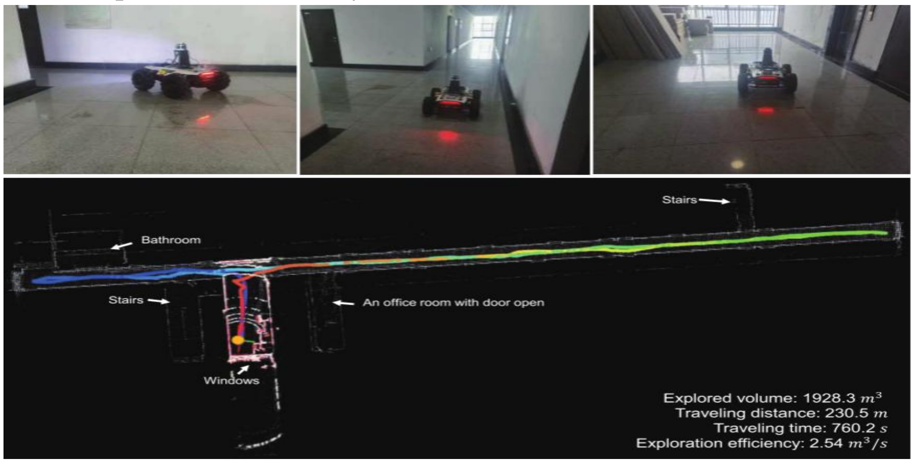

Stairs
Bathroom
Stairs
An office room with door open
Windows
Explored volume: 1928.3 m^3
Traveling distance: 230.5 m
Traveling time: 760.2 s
Exploration efficiency: 2.54 m^3/s

Fig. 2-12. The resulting map and trajectory of our method for a real corridor environment.

In another exploration experiment, we navigate the robot through an underground parking lot of a teaching building on our university campus, as illustrated in Figure 2-13. This environment comprises multiple parking spaces with many cars parked, along with building columns and other obstacles. The entry and exit of the parking lot were blocked with static obstacles. The robot begins the exploration near the entry point. After traveling a distance of 567.8 meters over a duration of 1243.3 seconds, the robot successfully completes the exploration. The explored space volume measures 15077.8 cubic meters, resulting in an exploration efficiency of 12.13 m^3/s. These real-world experiments demonstrate the effectiveness of our algorithm when applied to actual mobile robots highlighting its capacity to efficiently navigate and explore unfamiliar environments.

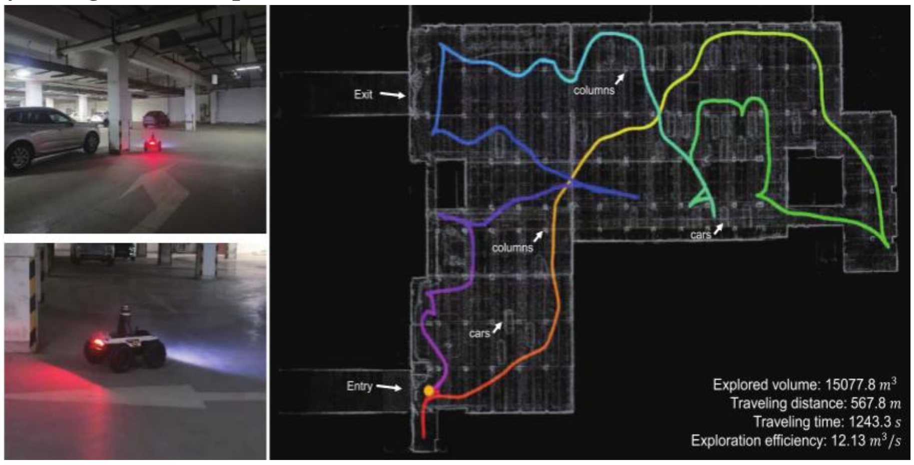

Exit
columns
columns
cars
cars
Entry
Explored volume: 15077.8 m^3
Traveling distance: 567.8 m
Traveling time: 1243.3 s
Exploration efficiency: 12.13 m^3/s

Fig. 2-13. The resulting map and trajectory of our method for a real underground parking lot environment.

Chapter 3 Navigation

3.1 Autonomous Navigation

The system, typically a car or robot, navigates autonomously in a given environment without direct human intervention. Autonomous navigation is crucial in exploration path planning as it combines spatial clustering and path planning.

Autonomous navigation relies on sensor data to interpret the surrounding environment. Sensors like cameras, LiDAR (Light Detection and Ranging), radar, and inertial measurement units (IMUs) are commonly used to gather environmental information including landmarks, obstacles, and terrain features. Perception algorithms analyze these sensor data to understand the environment and extract key data for navigation. Robots using localization techniques like Simultaneous Localization and Mapping (SLAM) can determine their position and orientation in the environmental map. Robots also utilize path planning algorithms to find the optimal path to a target location, considering obstacles, terrain, time, energy, and other factors. Autonomous navigation systems employ motion control algorithms to execute planned paths while avoiding obstacles and maintaining stability, adjusting parameters like steering and velocity to track the intended path accurately.

Autonomous navigation systems need to adapt to changing environments and handle unexpected situations like moving obstacles, terrain changes, sensor failures, to ensure reliable navigation performance.

In three-dimensional complex environments, autonomous navigation requires not only determining the robot's current position and orientation but also building a complete map model of the surrounding environment. Traditional two-dimensional navigation methods are insufficient to meet this demand. Therefore, we need to utilize 3D sensors to acquire rich environment 3D point cloud data and construct detailed 3D environment maps through advanced data processing algorithms.

Among these, map construction based on 3D point cloud data is a key technology. Firstly, we acquire 3D point cloud data of the environment through methods such as laser scanning, and then use point cloud registration algorithms such as Normal Distributions

Transform (NDT) to fuse and align multiple frames of point cloud data, establishing a complete 3D environmental model.

3.2 Robot Localization Based on Particle Filter

3.2.1 Concept and Working Principle of Particle Filter

Particle filtering is a probabilistic filtering method based on Bayesian estimation. It approximates the probability distribution of the target state using a weighted set of state samples called particles. At each time step, the particle filter performs two key steps: predicting updates to particles based on the motion model, and then adjusting the weights of particles based on the latest observation data. After multiple iterations, the particle set eventually converges near the true state, providing a probabilistic distribution form of state estimation.

Particle filtering was first proposed by Gordon, based on Monte Carlo methods and recursive Bayesian estimation filtering methods. Its core is to approximate the posterior probability density function using some random sampling points, replace the integral calculation with sample mean, and obtain the minimum variance estimate of the state. Particle filtering is a method that uses Monte Carlo methods to randomly extract a set of particles with weights to approximate the posterior probability density distribution, and sequentially predict and update the state.

The system model is defined as:

$$x_{k+1} = f_k(x_k, \varpi_k) \tag{3-1}$$

$$z_k = h_k(x_k, \upsilon_k) \tag{3-2}$$

In the equation, $x_{\mathrm{k+1}}$ represents the system state vector, z_k represents the measurement vector, $f(\cdot), h(\cdot)$ denotes the nonlinear function, Q_k represents the process noise covariance, and R_k represents the measurement noise covariance.

Let $x = \left\{x_{0:k}^i, w_k^j\right\}_{i-1}^N$ represent the set of particles $p(x_{0:k} \mid y_{1:k})$ representing the posterior probability density function of the system.

Where $x = \left\{x_{0:k}^i, i = 1,2 \wedge \mathrm{N}\right\}$ is the sample set, $\left\{w_{0:k}^i, i = 1,2 \wedge \mathrm{N}\right\}$ represents the corresponding weights, i.e., $\sum w_k^j = 1$, and N is the number of particles.

According to the Monte Carlo principle, the posterior probability density can be approximated as:

$$p(x_{0:k} | y_{1:k}) \approx \sum_{i=1}^{N} w_k^j \delta(x_{0:k} - x_{0:k}^j) \tag{3-3}$$

Where x_k^j is the ith particle randomly drawn from the importance distribution function $q(X_k | Y_k)$ at time k, w_k^j is the normalized particle importance weight, and $\delta(\cdot)$ is the Dirac delta function. Assuming the state follows a Markov process, the recursive formula for the particle importance weights given the state is:

$$w_k^i \propto w_{k-1}^i \frac{p(y_k | x_k^i) p(x_k^i | x_{k-1}^i)}{q(x_k^i | x_{k-1}^i, y_k)} \tag{3-4}$$

The normalized weights are obtained by normalizing the weights as follows:

$$w_k^{j^*} = \frac{w_k^j}{\sum_{i=1}^{N} w_k^j} \tag{3-5}$$

The estimated state is outputted as follows:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^i x_k^i \tag{3-6}$$

In particle filtering, particle degeneracy is inevitable, and the variance of importance weights increases over time. This increase in variance significantly affects accuracy, as after several iterations, only one particle may have a non-zero weight, while the weights of other particles are almost zero. This means that a considerable amount of effort is spent on updating particles that $p(x_k | y_{1:k})$ contribute almost nothing to the estimation.

3.2.2 State Representation and Models in Navigation

In robot navigation, the state typically includes six degrees of freedom, such as position (x, y, z) and orientation (roll, pitch, yaw).

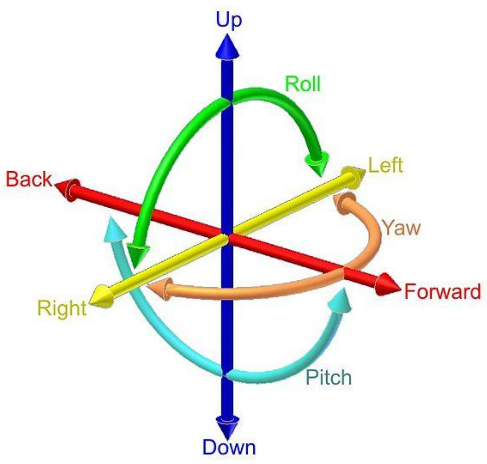


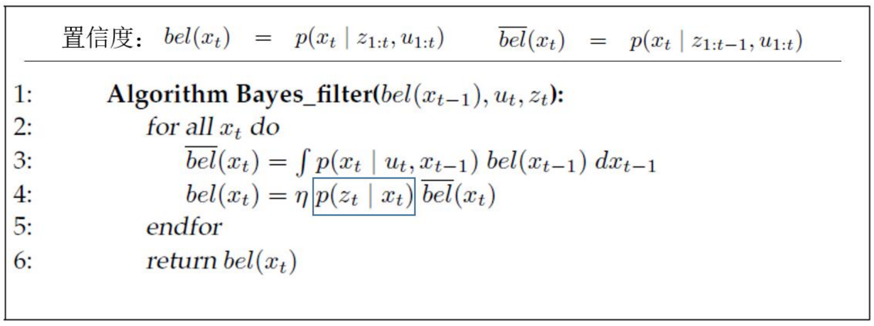
Fig. 3-1. Robot Coordinate System

In this schematic, position coordinates (x, y, z) as illustrated in the diagram (Up, Down, Forward, Back, Left, Right); Orientation angles (Roll, Pitch, Yaw); Here, x, y, and z represent the robot's position coordinates in three-dimensional space. Roll, pitch, and yaw represent the robot's orientation angles in space, corresponding to rotations around the x, y, and z axes, respectively.

These six degrees of freedom comprehensively describe the robot's position and orientation in the three-dimensional environment. During autonomous navigation, the robot needs to continuously monitor and control these state variables to ensure precise movement along the planned path.

The motion model describes how the robot's state changes from the previous time step to the next under given control inputs, often established using odometry data.

The observation model describes the relationship between current observation data (such as laser measurements) and the robot's state, used to compute the weights of particles.



置信度：$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$ $\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t})$

```
1:  **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
4:          $bel(x_t) = \eta\, \boxed{p(z_t \mid x_t)}\, \overline{bel}(x_t)$
5:      endfor
6:      return $bel(x_t)$
```

Fig. 3-2. Particle Filter

The complete definition of the observation model should be $p(z_t|x_t, m)$, where m represents the environment map. In mobile robotics, commonly used sensors include ranging sensors, which encompass ultrasonic and laser sensors. Both types of sensors inevitably introduce noise, and the observation model aims to quantitatively describe the various types of noise present in the sensors.

3.2.3 The specific implementation process of particle filtering in navigation

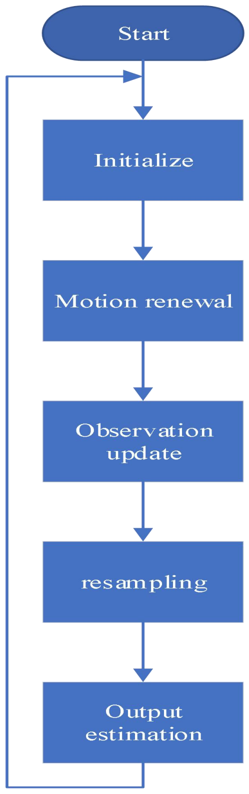


Fig. 3-3. Implementation Process

(1) Initialization: Randomly generate a set of particles representing the initial state of the robot based on prior information.

(2) Motion Update: Utilize the motion model to predictively update the state of each particle.

(3) Observation Update: Compute the weight of each particle based on the current observation data to reflect its likelihood probability.

(4) Resampling: Generate a new set of particles based on particle weights through random sampling or deterministic methods.

(5) Output Estimation: Obtain the final pose estimate of the robot from the current particle set using methods such as weighted averaging.

(6) Iterative Loop.

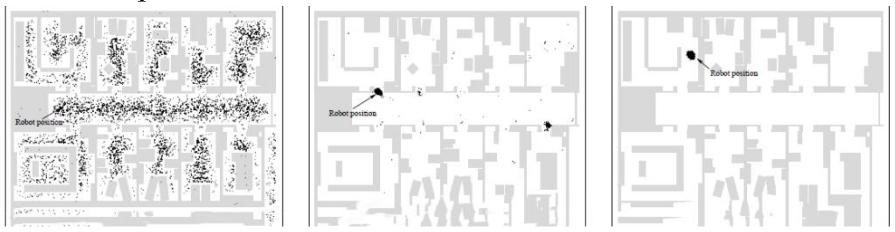


Fig. 3-4. Illustration of Particle Filter Localization Method

3.2.4 Analysis of Advantages and Disadvantages

Particle filtering is a Bayesian estimation technique based on Monte Carlo methods, which approximates the probability distribution of the target state by using a set of weighted state samples (particles). This method has several advantages in navigation localization: firstly, it can effectively handle nonlinear and non-Gaussian state estimation problems, whereas traditional methods like Kalman filtering often require linear and Gaussian assumptions. Secondly, the computational complexity of particle filtering is mainly linearly related to the number of particles, which can usually be controlled within the real-time performance requirements, resulting in lower costs compared to grid-based Monte Carlo localization methods. Additionally, particle filtering can effectively fuse observation data from multiple sensors to improve the robustness of localization.

Despite its many advantages, particle filtering also has some notable drawbacks. Firstly, this method requires the reasonable setting of several key parameters such as initial particle distribution, resampling strategy, process noise, etc., and parameter selection significantly influences the algorithm's performance, necessitating tuning according to specific application scenarios. Secondly, particle filtering relies on accurately modeling the initial state distribution and various noises; failure to accurately characterize these statistical properties in the real environment will affect the algorithm's convergence and final estimation accuracy. Lastly, in 3D space localization, due to the higher dimensionality of the state space, more particles are needed to effectively approximate the true distribution, placing higher demands on computational resources, and its applicability requires further research.

In conclusion, particle filtering is a flexible and powerful probabilistic localization algorithm that has advantages in handling nonlinear and non-Gaussian problems, as well as fusing data from multiple sensors. However, it also has limitations that require parameter adjustment and improvement tailored to specific application scenarios to achieve optimal performance.

3.3 3D Point Cloud Map Construction

3.3.1 Principle of 3D Laser Scanning

3D laser scanning is a commonly used three-dimensional sensing technology, which operates by scanning space with laser beams and measuring the intersection points of the laser beams with the target surface to obtain three-dimensional shape information of the target. Specifically, a laser emitter emits an infrared laser beam, which is directed horizontally and vertically through mechanical devices or optical components to achieve scanning. When the laser beam illuminates the target surface, the reflected or scattered laser is recorded by a receiver, and the distance between the laser and the target surface is calculated. By recording the position and distance of the laser beam, the system generates three-dimensional point cloud data of the target. Finally, the point cloud data undergoes a series of data processing algorithms, such as noise reduction, point cloud registration, surface reconstruction, etc., to generate high-quality three-dimensional models, which can be widely used in industrial inspection, architectural measurement, map creation, unmanned vehicle navigation, and other fields. The entire process achieves the acquisition of high-precision three-dimensional shape information of the target, possessing significant application value.

3.3.2 Point Cloud Data Formats and Processing

The obtained 3D point cloud data is typically stored in the form of XYZ coordinates. However, directly using raw point cloud data poses some challenges, necessitating preprocessing steps such as filtering, down sampling, and normal estimation to remove noise, reduce data size, and extract geometric feature information to enhance the efficiency and accuracy of subsequent processing algorithms. Commonly used point cloud data formats include PCD, PLY, OBJ, etc., which organize and store point cloud data in different ways for use by higher-level applications.

3.3.3 Point Cloud Map Construction Based on Scan Matching

To construct a complete 3D model of the environment, it is necessary to register and fuse multiple frames of 3D point cloud data. Commonly used point cloud registration algorithms include ICP (Iterative Closest Point) and NDT, which iteratively optimize to find the optimal rigid transformation parameters to align the point cloud data into the same coordinate system. This enables the gradual construction of a detailed 3D map representation of the environment, providing precise spatial information for subsequent navigation tasks.

When navigating using a 3D map, it is essential to represent the surrounding environment in three dimensions (3D) and utilize this representation for tasks such as path planning and navigation. A 3D map offers a more accurate and comprehensive description of the surrounding environment, facilitating more effective navigation for autonomous systems. Common methods for creating 3D maps of the environment involve using sensors capable of recording depth data, such as 3D cameras or LiDAR scanners. After processing these sensor data, mapping algorithms create a comprehensive 3D model of the surrounding environment, including surfaces, objects, and obstacles. Three-dimensional path planning algorithms consider not only the horizontal (x and y) dimensions but also the vertical (z) dimension, allowing autonomous systems to navigate through complex terrain at different elevations. 3D maps provide a rich visual representation of the surrounding environment, aiding human operators in tracking the robot's navigation progress or issuing advanced commands.

3.4 NDT Point Cloud Registration Algorithm

3.4.1 Theoretical Basis of NDT Algorithm

The NDT algorithm is a point cloud registration method that divides point cloud data into small voxels and fits a three-dimensional Gaussian distribution within each voxel. By minimizing the statistical distance between Gaussian distributions of neighboring voxels, the optimal rigid body transformation aligning two sets of point clouds is determined. This method does not require feature point extraction beforehand and performs registration directly on raw point cloud data, making it robust.

The steps involved in implementing the NDT algorithm are as follows:

(1) Discretize the data into different regions.

(2) For each region, calculate the centroid:

$$\mathbf{q} = \frac{1}{n}\sum_i \mathbf{x}_i \tag{3-7}$$

(3) Compute the covariance matrix based on the differences between each point and the centroid within each sub-region:

$$\Sigma = \frac{1}{n}\sum_i (\mathbf{x}_\mathbf{i} - \mathbf{q})(\mathbf{x}_\mathbf{i} - \mathbf{q})^t \tag{3-8}$$

(4) Obtain the NDT representation N for the discrete data, specifically represented as:

$$p(\mathbf{x}) \sim \exp(-\frac{(\mathbf{x}-\mathbf{q})^t \Sigma^{-1}(\mathbf{x}-\mathbf{q})}{2}) \tag{3-9}$$

3.4.2 Segmenting point clouds into voxels and fitting Gaussian distributions

NDT first divides the point cloud data into regular three-dimensional grid voxels. For each non-empty voxel, it calculates the mean and covariance matrix of the internal point cloud data to describe the Gaussian distribution characteristics of the point distribution within that voxel. This approach efficiently represents the geometric features of the point cloud data.

3.4.3 Iterative minimization of distribution distance to solve rigid body transformation

The NDT algorithm iteratively optimizes to find the rigid body transformation parameters that minimize the statistical distance between the Gaussian distributions of corresponding voxels in the source and target point clouds. Common distance measures include KL divergence, sequential distance, etc. This allows for the calculation of the optimal rigid body transformation aligning the two-point clouds.

3.4.4 Analysis of the Advantages and Disadvantages of the NDT Algorithm

The advantage of the NDT algorithm lies in its ability to register directly on raw point cloud data without the need for manual feature point extraction, thus exhibiting robustness. However, its disadvantage is the decrease in registration accuracy in environments lacking sufficient planar features, coupled with an increase in computational complexity with higher point cloud density. Overall, NDT provides an efficient solution for handling 3D point cloud data.

3.5 Combine Particle Filter with NDT

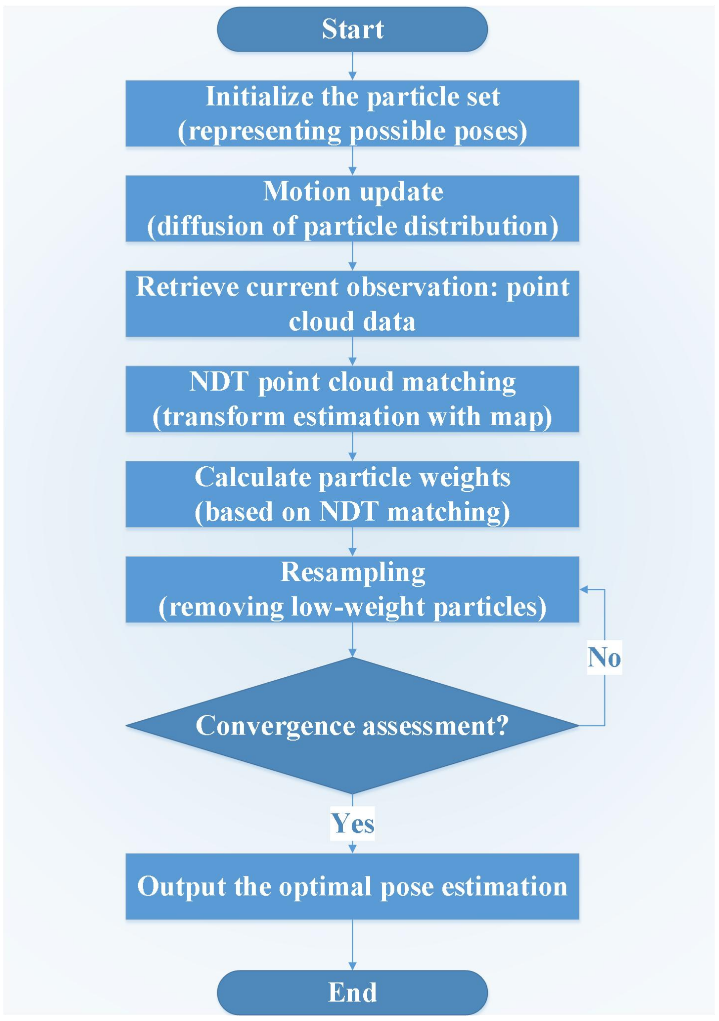

Start
Initialize the particle set (representing possible poses)
Motion update (diffusion of particle distribution)
Retrieve current observation: point cloud data
NDT point cloud matching (transform estimation with map)
Calculate particle weights (based on NDT matching)
Resampling (removing low-weight particles)
No
Convergence assessment?
Yes
Output the optimal pose estimation
End

Fig.3-5. Combining particle filtering with NDT point cloud matching

3.5.1 Constructing Observation Model Based on NDT

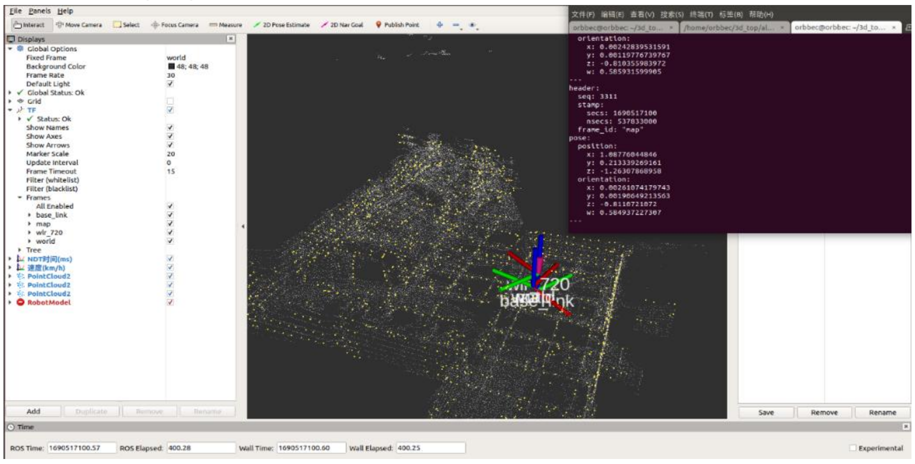
In the observation update step of particle filtering, the NDT algorithm is employed to match the current observed 3D point cloud data with the map model corresponding to the hypothesized pose represented by each particle. The obtained matching score can serve as the observation model for calculating the weight of each particle.

3.5.2 Updating Particle Weights Using NDT Observations in Localization

By incorporating the NDT matching score into the observation update formula of particle filtering, the particle weights can more accurately reflect the degree of alignment between the current observation data and the assumed pose, thereby enhancing localization accuracy.

3.5.3 The Localization Results in a Simulated Environment

Using simulation software to construct complex indoor 3D environments, the performance of the particle filter-NDT localization algorithm under ideal conditions is shown in the following figure 3-6 based on simulated LiDAR sensor data.


File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Add Duplicate Remove Rename
Save Remove Rename
wlr_720
base_link
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
orbbec@orbbec: ~/3d_to...
/home/orbbec/3d_top/al...
Time
Experimental

```
Global Options
  Fixed Frame          world
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: Ok
Grid
TF                     ✓
  Status: Ok
  Show Names           ✓
  Show Axes            ✓
  Show Arrows          ✓
  Marker Scale         20
  Update Interval      0
  Frame Timeout        15
  Filter (whitelist)
  Filter (blacklist)
  Frames
    All Enabled        ✓
    base_link          ✓
    map                ✓
    wlr_720            ✓
    world              ✓
  Tree
NDT时间(ms)            ✓
速度(km/h)             ✓
PointCloud2            ✓
PointCloud2            ✓
PointCloud2            ✓
RobotModel             ✓
```

```
  orientation:
    x: 0.00242839531591
    y: 0.00119776739767
    z: -0.810355983972
    w: 0.585931599905
---
header:
  seq: 3311
  stamp:
    secs: 1690517100
    nsecs: 537833000
  frame_id: "map"
pose:
  position:
    x: 1.08776044846
    y: 0.213339269161
    z: -1.26307868958
  orientation:
    x: 0.00261074179743
    y: 0.00190649213563
    z: -0.8110721072
    w: 0.584937227307
---
```

```
ROS Time: 1690517100.57   ROS Elapsed: 400.28   Wall Time: 1690517100.60   Wall Elapsed: 400.25
```

Fig.3-6. Localization Results in Simulated Environment

The experimental results indicate that the algorithm combining particle filtering with NDT achieves high localization accuracy and can be effectively applied in the experimental platform, demonstrating good practicality.

Chapter 4 Experimental Results

4.1 Synthetic Data Experiments Results

Our goal in the synthetic data experiments was to evaluate the suggested approach's efficiency, robustness, and scalability in controlled environments. Using a random distribution of regions with different densities and sizes, synthetic environments were created. To examine their effects on exploration efficiency, we changed the number of clusters (K) and the density of regions within each cluster. The percentage of regions explored, exploration coverage, and computational time for path planning and execution were used to measure the effectiveness of exploration.

A benchmark exploration dataset with five distinct scenarios—indoor, campus, garage, tunnel, and forest—is used in the evaluation. These environments differ in that campus and garage environments offer larger spaces, while tunnel, indoor, and forest environments are primarily made up of small passageways. Every environment is vast and intricate. They are not known to any of the methods under evaluation, which are all implemented within the Robot Operating System (ROS) framework, and are simulated in the Gazebo simulator. The "vehicle simulator" ROS package, which simulates robot kinematics and provides precise location data, is part of the evaluation platform. The maximum linear and angular velocities of the simulated robot are 2 m/s and 90 deg/s, respectively.

A minimum threshold is established for successful runs, and the comparison is based on the volumes investigated by the methods. The success rates, exploration trajectories, travel time, distance covered, and exploration efficiency of both approaches are assessed. Because RRT expansion is random, both methods have difficulties navigating spaces with guard rails in long, narrow interior corridors and lobby areas. However, because of its bias-free RRT expansion favoring sparse local candidate goals, the described method shows a slightly higher success rate (90%) than DSVP (80%). The method that is being described completes the exploration more quickly in terms of both distance travelled and time required. By adding a retrying mechanism for RRT expansion, the described method outperforms DSVP (80%) in an undulating campus environment, achieving a higher success rate of 100%. Nevertheless, this method prolongs the process. Both approaches achieve similar exploration efficiency by completing the same amount of time and distance travelled. The best trajectories of the methods are shown in Figures, where the DSVP trajectory outperforms the described method in the campus setting. In narrow indoor environments, the described method generally shows higher success rates and more efficient exploration, while both methods perform similarly, albeit with different trajectory outcomes, in campus environments with undulating terrain.

The exploration task in a complicated network of tunnels is successfully completed by both methods. In terms of exploration efficiency, time, and travelling distance, our method performs better than DSVP. Qualitative trajectory comparisons demonstrate that our approach covers a similar area in less time and space. Because DSVP typically explores the forest environment in a coarse manner, it has a 50% success rate and causes oversights. All runs are completed by our method, although the standard deviation is marginally higher. It does, however, outperform DSVP in terms of exploration efficiency. Qualitative trajectory comparisons demonstrate the higher efficiency of our method.

In most environments, our approach requires shorter runtimes and fewer planning iterations than DSVP. Even with the increased generation of candidate goals in large spaces, the average planning runtime (roughly 0.45 seconds) is still efficient and appropriate for mobile robot deployment in real-world scenarios. Returning home and covering the entire area are not given priority in the FAEL method. It performs poorly in extremely complex environments and frequently changes its navigation targets. FAEL performs poorly when exploring multi-floor environments, such as garages with multiple floors and sloped terrain. Instead, it works best in large, uncomplicated spaces.

4.2 Real World Scenario Experiments Results

The effects of not taking the global home position into account when planning a local tour are examined in this scenario. The process, called "Ours_R1," fixes the cost of travelling to the current robot position (CATSPk,0) from any local candidate goal to zero. The findings show that although travelling distance and exploration coverage are similar, the method needs more time to explore, which results in a lower exploration efficiency when compared to methods that take the home position into account.

In this case, global exploration goals are not clustered during the planning of global tours. More global goals mean more travelling destinations and longer ATSP solving times since they are not clustered if there are more than 40 global goals. As a result of more tour planning iterations or relocations, the total exploration time is prolonged.

A robot with a four-wheel differential drive powers the mobile robot platform. Outfitted with a 6-axis IMU and a Velodyne VLP-16 lidar for perception. Applies real-time lidar-inertial odometry to robot localization and mapping by modifying the LIO-SAM package. The robot's maximum angular and linear velocities are 15 degrees per second and 0.5 meters per second, respectively. An onboard PC with a 4.8GHz i7 CPU and 32GB RAM powers all packages. Held on a university campus in a building with a single floor. The environment consists of dead ends, doors, junctions, and small passageways. Static barriers are erected to bar access to particular spaces (like the lavatory). The process takes 760.2 seconds, covers an area of 1928.3 cubic meters, and travels 230.5 meters. Because of the enclosed corridor and the mobile platform's low speed setting, the exploration efficiency is measured at 2.54 m^3/s. steers the robot through the university campus's underground parking lot. The environment consists of several parking lots with parked cars, obstructions, and building columns. Static obstacles obstruct the parking lot's entrance and exit. In 1243.3 seconds, the robot covers a distance of 567.8 meters. With a volume of explored space of 15077.8 cubic meters, the exploration efficiency is 12.13 m^3/s.

4.3 Overview

The experimental results of using the suggested strategy—which combines spatial clustering and tour planning—for effective full-area exploration are shown in this section. To test the effectiveness of the integrated framework, we ran experiments with both synthetic data and real-world situations. Overall, the experimental findings support the usefulness of the suggested strategy for effectively exploring entire regions by combining tour planning and spatial clustering methods. The method shows promise for multiple exploration applications as it is robust, scalable, and adaptable in both artificial and natural environments.

Chapter 5 Application in Robotics

5.1 Exploration in Robotics

Robots that can travel and explore a variety of environments, including those that are dangerous or inaccessible to humans, are known as exploration robots. These machines can be fully or partially autonomous. These robots can sense their environment and decide on navigation and exploration tasks based on the information they gather from sensors, cameras, and other sensing devices.

Applications such as infrastructure inspection, environmental monitoring, planetary exploration, search and rescue, and environmental monitoring all heavily depend on exploration robots. Exploration robots are able to navigate through complex environments with efficiency by using advanced algorithms and technologies like tour planning and spatial clustering. These tools allow the robots to optimize their paths to cover the entire area of interest while minimizing resource consumption and maximizing exploration coverage. Ruggedized wheels, tracks, or even legs are frequently used in the design of exploration robots in order to enable them to traverse a variety of environments, including rocky surfaces, sandy deserts, and dense forests. To perceive their surroundings in three dimensions, these robots are outfitted with an array of sensors, such as inertial measurement units (IMUs), ultrasonic sensors, cameras, and LIDAR (Light Detection and Ranging) devices. Exploration robots are able to make decisions on their own using preset algorithms and sensor data. Without human assistance, they are able to plan their routes, avoid hazards, and adjust to shifting environmental conditions. Numerous exploration robots are outfitted with communication modules that allow them to receive commands remotely or send data back to a central control station, allowing for real-time monitoring and control [29].

5.2 Case Study

In this case study, we show how to use tour planning and spatial clustering algorithms to allow a robotic platform to explore uncharted territory on its own. The goal is to cover as much ground as possible in as little time and energy as possible while exploring a predetermined area. Imagine that an exploration robot is sent out into a hostile and uncharted area, like a dense forest or a rocky mountain range. The robot has actuators for movement and navigation in addition to sensors for localization, obstacle detection, and terrain mapping.

In order to produce a thorough map of the surroundings, the exploration robot first navigates the terrain and gathers sensor data. Obstacles, terrain elevation, and other pertinent features are all included on this map. Using a clustering algorithm like K-means or density-based clustering, the exploration area is divided into spatial clusters based on the generated terrain map. The terrain points' accessibility and spatial proximity determine how the clusters are formed. The exploration robot uses a tour planning algorithm within each spatial cluster to determine the best exploration path. The goal of this path is to minimize the traversal distance and avoid obstacles while covering every point of interest within the cluster. A comprehensive exploration path encompassing the entire designated area is formed by integrating the individual exploration paths generated for each cluster. The integration process guarantees a smooth transition between neighboring clusters by taking into account their connectivity. The exploration robot follows the designed exploration path on its own, navigating the landscape and modifying its course in response to real-time sensor input. To maximize exploration coverage, it avoids obstacles, modifies its speed, and maximizes energy consumption.

Metrics like traversal time, energy consumption, path optimality, and exploration coverage are used to assess the effectiveness of the suggested method. The results of the autonomous exploration show how well tour planning and spatial clustering help the robotic platform explore uncharted territory effectively. The suggested method demonstrates its applicability and efficiency in real-world exploration scenarios by achieving higher exploration coverage with less traversal time and energy consumption when compared to baseline methods [30].

Chapter 6 Application in Environmental Monitoring

6.1 Sensor Deployment for Environmental Monitoring

The systematic observation and measurement of numerous environmental parameters, including habitat conditions, biodiversity, water and air quality, and air quality, is known as environmental monitoring. Strategic sensor deployment throughout a target area is essential for thorough and efficient monitoring. Sensor deployment in environmental monitoring applications can be done methodically with the help of the suggested technique, which combines spatial clustering and tour planning.

To divide the monitoring area into clusters based on environmental similarities or geographic proximity, we apply spatial clustering techniques like K-Means or Density-Based Clustering. We make sure that every cluster corresponds to a different type of habitat or environmental zone. Based on the environmental parameters that need to be monitored, such as temperature, humidity, pollution levels, and the presence of species, choose the right kinds of sensors. We select sensor locations so that they cover a range of environmental conditions and microhabitats within each cluster. The best route for placing sensors within each cluster can be found by applying tour planning algorithms. To optimize monitoring efficacy, we take into account elements like coverage, accessibility, and sensor range.

For the purpose of developing a thorough sensor deployment plan that covers the whole monitoring region, combine the outcomes of spatial clustering and tour planning. We make sure the sensor locations we selected offer enough coverage and representativeness of the data for an accurate assessment of the environment [31][32].

6.2 Efficient Monitoring of Wildlife Habitats

As dynamic ecosystems, wildlife habitats need to be continuously observed in order to evaluate factors like biodiversity, population trends, and habitat health. Effective monitoring of wildlife habitats is crucial for managing ecosystems and promoting conservation. This case study illustrates how the suggested method may be used to efficiently monitor wildlife habitats. Examine a wildlife reserve to determine the biodiversity and state of the habitats of different animal species. a sizable wildlife reserve made up of various environments, including wetlands, grasslands, forests, and bodies of water. Temperature, humidity, amount of vegetation, water quality, and species presence are examples of environmental parameters. To divide the wildlife reserve into clusters that represent different habitat types (such as forest, grassland, and wetland clusters), use spatial clustering techniques. Make sure that the ecological traits and spatial variability of each type of habitat are captured in each cluster. To keep an eye on important environmental factors that are important for assessing wildlife habitat, choose the right sensors. Install sensors in each cluster in accordance with the suggestions of the tour planning algorithm, taking into account elements like species diversity, habitat complexity, and conservation priorities. We make sure all sensor locations are sufficiently covered in order to capture spatial heterogeneity within each type of habitat. To place sensors in each cluster in the most advantageous deployment path, use tour planning algorithms. Sort sensor locations according to ecological significance, species distribution patterns, and habitat accessibility. Reduce travel times and maximize the location of sensors. We gather environmental data from the sensors that have been deployed over time to keep an eye on species presence and habitat conditions. And examine the gathered information to evaluate ecosystem health, habitat quality, and biodiversity in various habitat clusters. We make decisions about habitat management, conservation tactics, and wildlife protection based on the monitoring data. By capturing both the temporal and spatial variability of environmental conditions, the combined approach guarantees comprehensive coverage of wildlife habitats. Reduced monitoring costs and resource requirements result from optimized sensor deployment paths that minimize travel distances and maximize data collection efficiency. Ecologically Informed. The monitoring data offers insightful information about the dynamics of wildlife habitat, which helps with conservation and management choices [31][32].

Chapter 7 Discussion

7.1 Advantages of the Proposed Approach

The suggested method greatly increases exploration efficiency by dividing the exploration area into clusters based on spatial proximity and organizing optimized tours within each cluster. This minimizes unnecessary movements while enabling comprehensive coverage of the whole region. The method can be adjusted to fit various exploration area sizes and levels of complexity. The clustering and tour planning framework can be adjusted to the task size, regardless of the size of the geographic area being explored. This ensures effective exploration without sacrificing coverage. Spatial clustering and tour planning combined allow for flexibility in responding to different exploration scenarios and goals. The approach can be tailored to meet different needs, such as maximizing area coverage, reducing exploration time, or prioritizing particular regions of interest. The method is resilient to changes in the exploration area's dynamic environment. Through the process of periodically updating the tour plans and clustering based on sensor feedback or real-time data, the exploration system is able to adjust to changing conditions and unforeseen roadblocks. The suggested method's optimal exploration path results in the cost-effective use of resources like equipment, energy, and time. This is especially crucial in situations with limited resources, as effective exploration can save costs and increase mission success. The suggested method can be used in a number of fields, such as robotic exploration, urban planning, environmental monitoring, and disaster response. Because of its adaptability, it can be used for a variety of exploration tasks across several fields [33].

7.2 Limitations and Future Directions

The choice of starting parameters, such as the number of clusters in the spatial clustering step or the population size in the tour planning algorithms, may have an impact on how well the approach performs. The development of reliable methods for adaptive parameter tuning or automatic parameter selection may be the main focus of future research. Although the method can withstand some degree of environmental variability, it might not work well in environments that change quickly or have unpredictable obstacles or topography. Subsequent studies could investigate adaptive techniques for real-time path replanning in order to better manage dynamic environments. Uncertainty about sensor readings, the surrounding environment, and the features of the terrain is common in real-world exploration scenarios. Subsequent studies could look into ways to improve the clustering and tour planning algorithms by incorporating uncertainty. The main focus of the suggested method is on single-robot exploration tasks. Subsequent investigations may delve into expanding the methodology to multi-robot environments, in which numerous robots cooperate to enhance area coverage through synchronization and exchange of information. Although the method has been tested in controlled experiments and simulations, more real-world validation is required to determine its applicability and scalability in a variety of settings and exploration tasks. The suggested method is predicated on a comparatively static exploration setting. Future studies could look into effective methods for exploring dynamic environments, where the exploration area is constantly changing because of weather, human activity, or natural phenomena [34].

Chapter 8 Conclusions

In conclusion, we have presented a dual-stage exploration method that prioritizes simplicity and effectiveness. The spatial clustering and TSP-based tour planning technologies are employed in both stages. In the exploration stage, possible local exploration goals are densely detected from RRT nodes and then spatially clustered. The sparse local navigation goal candidates are extracted from the clusters rather than calculating gains of all possible RRT branches.

The local tour planner is further employed to select optimal navigation goals for efficient exploration. In the relocation stage, the number of global exploration goals is also controlled by clustering and the global tour planning effectively guides the robot to revisit unexplored spaces. Our proposed retrying mechanism significantly enhances the success rate of exploration. To evaluate our approach, we conduct tests in five benchmark simulation environments and two real-world environments. Both the simulation and real-world tests demonstrate the effectiveness and efficiency of our method.

However, both DSVP and our methods have certain limitations. Firstly, the system heavily relies on mapping performance and the accuracy of robot localization. It may not perform as expected in open spaces, such as the outdoor areas on campus. Relying solely on laser scans may prove insufficient for detecting obstacles like road curbs or green belts that could pose a danger to the robot. To enhance terrain travers ability analysis, integrating additional sensors, such as cameras, would be highly beneficial. Secondly, during the exploration stage, the local graph is utilized to plan the path from the current position to the target goal. Due to the sparse sampling of nodes in the local graph, the planned path may not always be optimal. Furthermore, the planned path is not smoothed in accordance with the robot's kinematics model. This can result in the robot making frequent heading adjustments, leading to increased power consumption, longer travel times, and posing additional challenges to the simultaneous localization and mapping process. Future work will be dedicated to addressing these identified shortcomings.

References

[1] M. B. Alatise and G. P. Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access, 8:39830 – 39846, 2020.

[2] Q. Bi, X. Zhang, J. Wen, Z. Pan, S. Zhang, R. Wang, and J. Yuan. Cure: A hierarchical framework for multi-robot autonomous exploration inspired by centroids of unknown regions. IEEE Transactions on Automation Science and Engineering, pages 1–14, 2023.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3d exploration. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1462–1468, 2016.

[4] C. Cao, H. Zhu, F. Yang, Y. Xia, H. Choset, J. Oh, and J. Zhang. Autonomous exploration development environment and the planning algorithms. In 2022 International Conference on Robotics and Automation (ICRA), pages 8921–8928, 2022.

[5] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger. Fast frontier-based information-driven autonomous exploration with an mav. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 9570–9576, 2020.

[6] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. Journal of Field Robotics, 37(8):1363–1388, 2020.

[7] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano, L. M. Gambardella, R. Siegwart, and D. Scaramuzza. The current state and future outlook of rescue robotics. Journal of Field Robotics, 36(7):1171 – 1191, 2019.

[8] R. S. Dileep Muddu, D. Wu, and L. Wu. A frontier based multirobot approach for coverage of unknown environments. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 72–77, 2015.

[9] C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3d. Advanced Robotics, 27(6):459–468, 2013.

[10] K. Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. European Journal of Operational Research, 126(1):106–130, 2000.

[11] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin. Voronoibased multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Transactions on Vehicular Technology, 69(12):14413–14423, 2020.

[12] J. Huang, B. Zhou, Z. Fan, Y. Zhu, Y. Jie, L. Li, and H. Cheng. Fael: Fast autonomous exploration for large-scale environments with a mobile robot. IEEE Robotics and Automation Letters, 8(3):1667– 1674, 2023.

[13] M. A. Khawaldah and A. Nüchter. Enhanced frontier-based exploration for indoor environment with multiple robots. Advanced Robotics, 29(10):657–669, 2015.

[14] Z. Meng, H. Qin, Z. Chen, X. Chen, H. Sun, F. Lin, and M. H. Ang. A two-stage optimized next-view planning framework for 3-d unknown environment exploration, and structural reconstruction. IEEE Robotics and Automation Letters, 2(3):1680–1687, 2017.

[15] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5135–5142. IEEE, 2020.

[16] A. Soni, C. Dasannacharya, A. Gautam, V. S. Shekhawat, and S. Mohan. Multi-robot unknown area exploration using frontier trees. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 9934–9941, 2022.

[17] Z. Sun, B. Wu, C. Xu, and H. Kong. Ada-detector: Adaptive frontier detector for rapid exploration. In 2022 International Conference on Robotics and Automation (ICRA), pages 3706–3712, 2022.

[18] H. Umari and S. Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1396–1402, 2017.

[19] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng. Autonomous robotic exploration by incremental road map construction. IEEE Transactions on Automation Science and Engineering, 16(4):1720–1731, 2019.

[20] Y. Xu, J. Yu, J. Tang, J. Qiu, J. Wang, Y. Shen, Y. Wang, and H. Yang. Explore-bench: Data sets, metrics and evaluations for frontier-based and deep-reinforcement-learning-based autonomous exploration. In 2022 International Conference on Robotics and Automation (ICRA), pages 6225–6231, 2022.

[21] B. Yamauchi. A frontier-based approach for autonomous exploration. In Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation', pages 146– 151, 1997.

[22] H. Zhu, C. Cao, Y. Xia, S. Scherer, J. Zhang, and W. Wang. Dsvp: Dual-stage viewpoint planner for rapid exploration by dynamic expansion. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7623–7630, 2021.

[23] C. C. Aggarwal and C. R. Reddy. Data clustering: Algorithms and applications. In 2014 International Conference on Data Clustering, pages 1-25, 2014.

[24] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 2(3):283-304, 1998.

[25] C. A. Ratanamahatana and D. Gunopulos. Scaling clustering algorithms to large databases. In Data Mining and Knowledge Discovery Handbook, pages 317-343, 2005.

[26] P. Van Oosterom, S. Zlatanova, and F. Penninga. The fuzzy boundaries of spatial objects: approaches from different disciplines. International Journal of Geographical Information Science, 19(9):983-1003, 2005.

[27] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. ACM Comput. Surv. 31, 3 (Sept. 1999), 264–323. 1999.

[28] X. Li and X. Liao. Research on optimization algorithms of traveling salesman problem. In 2015 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), pages 52-55, 2015.

[29] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in Distributed Autonomous Robotic Systems 5, pp. 299-308, Springer, Tokyo, 2002.

[30] W. Du and J. Goldsmith, "A robotic approach to autonomous exploration," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 35, no. 4, pp. 641-652, 2005.

[31] H. Jiang et al., "Efficient sensor deployment strategies for environmental monitoring: a review," Journal of Sensors, vol. 2018, pp. 1-19, 2018.

[32] J. Fischer et al., "Creating a blueprint for ecosystem-based management: a case study of the South Australian Regional NRM Plan," Environmental Science & Policy, vol. 11, no. 5, pp. 441-455, 2008.

[33] L. Gao, C. Yan, and H. Shen, "Efficient coverage path planning for agricultural UAVs based on improved clustering algorithm," Computers and Electronics in Agriculture, vol. 170, p. 105262, 2020.

[34] Q. Chen, S. Ye, and W. Wang, "A Study on the Influence of Initial Parameters on K-means Clustering Algorithm," in International Conference on Intelligent Human-Machine Systems and Cybernetics, pp. 426-429, IEEE, 2017.