DEVELOPMENT OF AN INDOOR TOPOLOGICAL NAVIGATION FRAMEWORK FOR MOBILE ROBOTS USING 360-DEGREE CAMERAS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR OYDU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ROBOTICS

APRIL 2025

Approval of the thesis:

DEVELOPMENT OF AN INDOOR TOPOLOGICAL NAVIGATION FRAMEWORK FOR MOBILE ROBOTS USING 360-DEGREE CAMERAS

submitted by **ONUR OYDU** in partial fulfillment of the requirements for the degree of **Master of Science in Robotics Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Uluç Saranlı
Head of Department, **Robotics** _____

Assoc. Prof. Dr. Ahmet Buğra Koku
Supervisor, **Robotics, METU** _____

Assist. Prof. Dr. Murat Durmaz
Co-supervisor, **Geomatics Engineering, Hacettepe University** _____

Examining Committee Members:

Assoc. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Ahmet Buğra Koku
Mechanical Engineering, METU _____

Assist. Prof. Dr. Akif Hacınecipoğlu
Mechanical Engineering, OSTIM Technical University _____

Date:07.04.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Onur Oydu

Signature :

ABSTRACT

DEVELOPMENT OF AN INDOOR TOPOLOGICAL NAVIGATION FRAMEWORK FOR MOBILE ROBOTS USING 360-DEGREE CAMERAS

Oydu, Onur
M.S., Department of Robotics
Supervisor: Assoc. Prof. Dr. Ahmet Buğra Koku
Co-Supervisor: Assist. Prof. Dr. Murat Durmaz

April 2025, 119 pages

This thesis presents the development of an innovative topological navigation framework for mobile robots operating in indoor environments. A metric-free vision-based navigation system using 360-degree cameras is proposed to overcome the limitations of metric-based navigation systems. Equirectangular images are taken at each node by using a 360-degree camera mounted on the robot head. A topological map with nodes defined by equirectangular images is generated, and connections between nodes are determined using matching features. Feature extraction (ALIKEd) and matching (LightGlue) algorithms identify matching features. The 8-point algorithm with RANSAC estimates the heading vector and relative orientation between nodes. Heading vectors and relative orientations produce scale-free node coordinates, which are then validated with the reference floor plan. A graph search algorithm is utilized in the path-finding stage. Four routes across the map were driven in the normal direction and then in reverse, confirming that the proposed topological navigation algorithm supports reliable bidirectional travel. Also, another topological map consisting of fully connected 9 nodes with known positions was generated. Every node was

used as a start once and as a goal for each of its eight neighbors, giving 72 start–goal trials with a mean position error of 0.32 m and an overall success rate of 88.9 %. A hand-mounted 360-degree camera performs vision-only triangulation of a specific object in the goal node. Two routes were traversed forward and reverse, confirming reliable bidirectional approach to the object. Also, the triangulation algorithm was validated in three indoor layouts, achieving a mean position error of 4.37 cm and a mean orientation error below 2.5 degrees. Overall, the results demonstrate that once the robot reaches its target pose, the manipulator can servo precisely to the object while remaining entirely within its 0.87 m horizontal workspace.

Keywords: Topological Navigation, Equirectagular Image, Feature-Matching, 8-Point Algorithm, Metric-Free Navigation

ÖZ

360 DERECE KAMERA KULLANAN MOBİL ROBOTLAR İÇİN İÇ MEKAN TOPOLOJİK NAVİGASYON ÇERÇEVESİNİN GELİŞTİRİLMESİ

Oydu, Onur
Yüksek Lisans, Robotik Bölümü
Tez Yöneticisi: Doç. Dr. Ahmet Buğra Koku
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Murat Durmaz

Nisan 2025 , 119 sayfa

Bu tez, kapalı ortamlarda çalışan mobil robotlar için yenilikçi bir topolojik navigasyon çerçevesinin geliştirilmesini sunmaktadır. Metrik-sensörlere ihtiyaç duymayan, 360 derece kameralarla çalışan bir görsel navigasyon sistemi önerilmektedir. Robotun başına monte edilen 360 derece kamera ile her düğüm noktasında eşdikdörtgen (equirectangular) görüntüler alınır. Bu görüntülerle düğümler tanımlanarak topolojik bir harita oluşturulur ve düğümler arasındaki bağlantılar, öznitelik çıkarma (ALIKEd) ve eşleştirme (LightGlue) algoritmalarıyla tespit edilen ortak özniteliklere göre belirlenir. 8-nokta algoritması ve RANSAC kullanılarak düğümler arası yön vektörleri ve bağıl yönelimler hesaplanır. Elde edilen bu yön vektörleri ve bağıl yönelimler, ölçeğe bağlı olmayan düğüm koordinatlarını üretir; bu koordinatlar, referans krokiye karşı doğrulanır. Yol bulma aşamasında grafik arama algoritması kullanılır. Harita üzerindeki dört rota, önce normal yönde sonra ters yönde sürülerek önerilen topolojik navigasyon algoritmasının çift yönlü güvenilirliği test edildi. Ayrıca, bilinen pozisyonlara sahip tam bağlı 9 düğümden oluşan başka bir topolojik harita oluşturuldu.

Her düğüm bir kez başlangıç, her komşu düğüm ise bir kez hedef olarak seçilerek 72 başlangıç–hedef denemesi gerçekleştirildi; bu denemelerde ortalama 0.32 m pozisyon hatası ve %88.9 başarı oranı elde edildi. Robotun eline takılı ikinci bir 360 derece kamera ise yalnızca görsel verilerle hedef düğümdeki belirli bir nesnenin üçgenleme yoluyla konumunu belirler. İki rota ileri ve geri yönde sürülerek nesneye çift yönlü yaklaşma güvenilirliği doğrulandı. Ayrıca üç farklı iç mekân düzeninde yapılan testlerde ortalama 4.37 cm pozisyon hatası ve 2.5 derecenin altında yönelim hatası elde edildi. Tüm sonuçlar, robot hedef pozisyona ulaştıktan sonra manipülatörün 0.87 m yatay çalışma alanı içinde kalarak nesneye hassas bir şekilde yaklaşabildiğini göstermektedir.

Anahtar Kelimeler: Topolojik Seyrüsefer, Eşdikdörtgen Görüntü, Öznitelik Eşleştirme, 8-Nokta Algoritması, Metrik Bilgisiz Seyrüsefer

To my family

ACKNOWLEDGMENTS

I would like to thank my professors, Assoc. Prof. Dr. Ahmet Buğra Koku and Asst. Prof. Dr. Murat Durmaz, for their unwavering support and guidance throughout my studies. Their expertise, mentorship, and constructive feedback have played a pivotal role in shaping my academic journey and this thesis. Their encouragement and dedication have inspired me to broaden my knowledge and strive for excellence in my work.

I also wish to thank METU ROMER for providing a vibrant and collaborative environment, as well as an invaluable opportunity to develop my research skills. My friends at ROMER Umut Kurt, Anıl Gülbeden, Rüstem Şevik, Denge Uzel and Çağdaş Güven deserve special thanks for their steadfast support, friendship, and inspiration throughout this process.

Furthermore, I am grateful to Roketsan Missiles Inc. for offering me the chance to work in an innovative setting, contributing to my personal and professional growth. I would like to especially thank my colleague, Arif Mertcan Koca, for his unwavering support and understanding, for which I am sincerely appreciative.

Finally, I want to offer my heartfelt thanks to my father, Nihat Oydu, my mother, Seher Oydu, and my brother, Kıvanç Oydu. Their boundless love, encouragement, and patience have been the driving force behind all my endeavors. Their unwavering belief in me has shaped my journey, and I owe every achievement to their constant guidance and support.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xxiv

CHAPTERS

1 INTRODUCTION 1

1.1 Problem Definition 1

1.2 Motivation 2

1.3 Proposed Methods and Models 3

1.4 Contributions and Novelties 4

1.5 The Outline of the Thesis 5

2 BACKGROUND AND RELATED WORK 7

2.1 Indoor Navigation Paradigms 7

2.1.1 Metric-Based Navigation 8

2.1.1.1 Wheel-Odometry and 2D LiDAR SLAM 8

2.1.1.2 Visual–Inertial and Stereo SLAM 8
2.1.1.3 Dense RGB-D Mapping 9
2.1.1.4 Limitations of Metric Approaches 9
2.1.2 Metric-Free Topological Navigation 10
2.1.2.1 Early Symbolic and Qualitative Methods 10
2.1.2.2 Vision-Based Topological Methods 10
2.1.2.3 360-Degree and Panoramic Vision Approaches 11
2.1.2.4 Limitations of Topological Approaches 11
2.2 360-degree Camera Model and Equirectangular Projection 12
2.3 Feature Extraction and Matching 13
2.3.1 ALIKEd 15
2.3.2 LightGlue 15
2.4 Relative Pose Estimation with the 8-Point Algorithm and RANSAC . 16
2.5 Image Similarity Measure and Visual Place Recognition 17
2.6 Obstacle Avoidance 19
2.7 Robot Control Architecture 19
2.8 Research Gaps 20
3 METHODOLOGY 21
3.1 Proposed Algorithm 21
3.2 Problem Formulation 22
3.3 Camera Modeling and Back-Projection 23
3.4 Heading Vector and Relative Orientation Estimation 24
3.4.1 8-Point Algorithm 24

3.4.2 RANSAC 32
3.5 Similarity Metric 34
3.6 Topological Navigation and Mapping 34
3.6.1 Node Discovery and Uniqueness 35
3.6.2 Edge Construction 36
3.6.3 Pose Embedding and Refinement 36
3.6.4 Route Planning and Online Localization 39
3.7 3D Position Estimation from Triangulation 41
3.8 Implementation Details 44
3.8.1 Obstacle Avoidance and Motion Control 44
3.8.2 ROS Node Architecture 47
3.8.3 State Machine Design 50
4 EXPERIMENTAL SETUP AND PROCEDURE 53
4.1 Robot 53
4.2 Software 54
4.3 Hardware 55
4.4 Test Environment 55
4.5 Experimental Procedure 56
5 RESULTS AND DISCUSSION 59
5.1 Results 59
5.1.1 Performance of the Topological Mapping Algorithm 60
5.1.2 Performance of the Navigation Algorithm 61
5.1.2.1 Reliability Analysis 62

5.1.2.2 Localization Error Analysis 65
5.1.3 Performance of the Triangulation Algorithm 69
5.1.3.1 Reliability Analysis 69
5.1.3.2 Pose Estimation Error Analysis 74
5.2 Discussion 80
6 CONCLUSION 85
REFERENCES 89
APPENDICES 95
A Additional Navigation Scenarios 95
B Final Positions for Each Goal 103
C Additional Triangulation Scenarios 108
D Additional Pose Estimation Error Results 112

LIST OF TABLES

TABLES

Table 3.1 ROS Node Rates 48

Table 4.1 Python Packages 54

Table 5.1 Position Error Statistics by Goal Node 66

Table 5.2 Position Error Statistics of Experiment 1 76

Table 5.3 Orientation Error Statistics of Experiment 1 77

Table 5.4 Duration of each navigation scenario given in Section 5.1.2.1 81

Table D.1 Position Error Statistics of Experiment 2 112

Table D.2 Orientation Error Statistics of Experiment 2 113

Table D.3 Position Error Statistics of Experiment 3 116

Table D.4 Orientation Error Statistics of Experiment 3 117

LIST OF FIGURES

FIGURES

Figure 2.1 Equirectangular projection. Red color indicates the amount of distortion. 12

Figure 2.2 Keypoints of extracted features from an image, features at the bottom half of the image are excluded. 14

Figure 2.3 Architecture of ALIKEd, taken from [1]. 15

Figure 2.4 Architecture of LightGlue, taken from [2]. 16

Figure 2.5 Illustration of the heading vector calculation problem. Purple points represent the correspondences between two images. 16

Figure 2.6 A special case where images are taken from very close positions with high azimuth angle difference. 18

Figure 3.1 Camera coordinate frame. The right-hand rule is used for the directions of axes. 24

Figure 3.2 Visualization of the epipolar geometry. 25

Figure 3.3 Top view of the matched unit vectors. 26

Figure 3.4 Side view of the matched unit vectors. 27

Figure 3.5 The four possible solutions of 8-point algorithm, taken from [3]. 29

Figure 3.6 Flowchart of RANSAC algorithm. 33

Figure 3.7 Illustration of a topological map where each node corresponds to a 360-degree equirectangular image and edges denote traversable connections. 35

Figure 3.8 Topological mapping geometry. 37

Figure 3.9 Flowchart of the topological map construction. 40

Figure 3.10 Visualization of the triangulation process. The arrows represent the heading vectors from the head and gripper cameras to the object's scene. 41

Figure 3.11 Visualization of the control commands and coordinate frames. . . 46

Figure 3.12 Graph of the communication network of implemented ROS nodes. 49

Figure 3.13 State machine of the implemented algorithm. 51

Figure 4.1 TIAGo robot 54

Figure 4.2 Sketch of the METU ROMER laboratories. 55

Figure 4.3 Controlled environment with OptiTrack setup installed. Red arrows show the six cameras that capture the position of the robot. 56

Figure 5.1 Topological map generated from video record. 60

Figure 5.2 Topological map plotted on the sketch of the environment. 61

Figure 5.3 Location of scenario 1 of topological navigation algorithm test in full map. 62

Figure 5.4 Path generated in scenario 1 of topological navigation algorithm test. 63

Figure 5.5 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of topological navigation algorithm test. 63

Figure 5.6 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of topological navigation algorithm test. 63

Figure 5.7 Path generated in scenario 1 (repeated in reverse order). 64

Figure 5.8 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of topological navigation algorithm test (repeated in reverse order). 64

Figure 5.9 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of topological navigation algorithm test (repeated in reverse order). 64

Figure 5.10 Boxplot errors by goal node for successful trials. 66

Figure 5.11 Histogram of the position errors for successful trials. 67

Figure 5.12 Success rate by goal node. 67

Figure 5.13 The ground-truth positions of all nodes with the final positions of successful navigation trials. 68

Figure 5.14 Collected images from the controlled environment. 68

Figure 5.15 Location of scenario 1 of triangulation algorithm test in full map. 70

Figure 5.16 Path generated in scenario 1 of triangulation algorithm test. . . . 70

Figure 5.17 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of triangulation algorithm test. . . . 71

Figure 5.18 View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 1. 71

Figure 5.19 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of triangulation algorithm test. . . 72

Figure 5.20 Path generated in scenario 1 (opposite direction) of triangulation algorithm test 72

Figure 5.21 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 (opposite direction) of triangulation algorithm test. 72

Figure 5.22 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 (opposite direction) of triangulation algorithm test. 73

Figure 5.23 View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 1 (opposite direction). 73

Figure 5.24 Images taken from the predetermined eight points of Experiment 1. 76

Figure 5.25 Position error boxplot of Experiment 1. 77

Figure 5.26 Orientation error boxplot of Experiment 1. 78

Figure 5.27 Position error histogram of Experiment 1. 78

Figure 5.28 Orientation error histogram of Experiment 1. 79

Figure 5.29 Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 1. All positions are given according to robot's base frame (red circle). 79

Figure 5.30 Example of a degenerate configuration where the robot attempts to navigate from node 1 to node 9, illustrating a failure case due to degenerate configurations or insufficient feature correspondence. 81

Figure 5.31 The ground-truth positions of all nodes with the final positions of successful navigation trials. Around each node, a dashed gray circle with a radius of 0.87 m represents the TIAGo arm's horizontal reach limit. 82

Figure A.1 Location of scenario 2 of topological navigation algorithm test in full map. 95

Figure A.2 Path generated in scenario 2 of topological navigation algorithm test. 95

Figure A.3 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of topological navigation algorithm test. 96

Figure A.4 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of topological navigation algorithm test. 96

Figure A.5 Path generated in scenario 2 of topological navigation algorithm test (repeated in reverse order). 96

Figure A.6 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of topological navigation algorithm test (repeated in reverse order). 97

Figure A.7 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of topological navigation algorithm test (repeated in reverse order). 97

Figure A.8 Location of scenario 3 of topological navigation algorithm test in full map. 97

Figure A.9 Path generated in scenario 3 of topological navigation algorithm test. 98

Figure A.10 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 3 of topological navigation algorithm test. 98

Figure A.11 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 3 of topological navigation algorithm test. 98

Figure A.12 Path generated in scenario 3 of topological navigation algorithm test (repeated in reverse order). 99

Figure A.13 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 3 of topological navigation algorithm test (repeated in reverse order). 99

Figure A.14 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 3 of topological navigation algorithm test (repeated in reverse order). 99

Figure A.15 Location of scenario 4 of topological navigation algorithm test in full map. 100

Figure A.16 Path generated in scenario 4 of topological navigation algorithm test. 100

Figure A.17 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 4 of topological navigation algorithm test. 101

Figure A.18 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 4 of topological navigation algorithm test. 101

Figure A.19 Path generated in scenario 4 of topological navigation algorithm test (repeated in reverse order). 101

Figure A.20 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 4 of topological navigation algorithm test (repeated in reverse order). 102

Figure A.21 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 4 of topological navigation algorithm test (repeated in reverse order). 102

Figure B.1 The final positions of successful navigation trials for goal node 1. 103

Figure B.2 The final positions of successful navigation trials for goal node 2. 103

Figure B.3 The final positions of successful navigation trials for goal node 3. 104

Figure B.4 The final positions of successful navigation trials for goal node 4. 104

Figure B.5 The final positions of successful navigation trials for goal node 5. 105

Figure B.6 The final positions of successful navigation trials for goal node 6. 105

Figure B.7 The final positions of successful navigation trials for goal node 7. 106

Figure B.8 The final positions of successful navigation trials for goal node 8. 106

Figure B.9 The final positions of successful navigation trials for goal node 9. 107

Figure C.1 Location of scenario 2 of triangulation algorithm test in full map. 108

Figure C.2 Path generated in scenario 2 of triangulation algorithm test. . . . 108

Figure C.3 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of triangulation algorithm test. . . . 109

Figure C.4 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of triangulation algorithm test. . . 109

Figure C.5 View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 2. 109

Figure C.6 Path generated in scenario 2 (opposite direction) of triangulation algorithm test 110

Figure C.7 View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 (opposite direction) of triangulation algorithm test. 110

Figure C.8 View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 (opposite direction) of triangulation algorithm test. 110

Figure C.9 View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 2 (opposite direction). 111

Figure D.1 Images taken from the predetermined eight points of Experiment 2. 112

Figure D.2 Position error boxplot of Experiment 2. 113

Figure D.3 Orientation error boxplot of Experiment 2. 114

Figure D.4 Position error histogram of Experiment 2. 114

Figure D.5 Orientation error histogram of Experiment 2. 115

Figure D.6 Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 2. All positions are given according to robot's base frame (red circle). 115

Figure D.7 Images taken from the predetermined eight points of Experiment 3. 116

Figure D.8 Position error boxplot of Experiment 3. 117

Figure D.9 Orientation error boxplot of Experiment 3. 118

Figure D.10 Position error histogram of Experiment 3. 118

Figure D.11 Orientation error histogram of Experiment 3. 119

Figure D.12 Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 3. All positions are given according to robot's base frame (red circle). 119

LIST OF ABBREVIATIONS

ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
CNN	Convolutional Neural Network
DLT	Direct Linear Transform
DOF	Degrees of Freedom
GSM	Gold Standard Method
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
RANSAC	Random Sample Consensus
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
RMSE	Root Mean Square Error
SLAM	Simultaneous Localization and Mapping
SSIM	Structural Similarity Index
SVD	Singular Value Decomposition
VIO	Visual–Inertial Odometry

CHAPTER 1

INTRODUCTION

This chapter first defines the shortcomings of metric-based indoor navigation, then motivates a metric-free, vision-only alternative, outlines the proposed topological framework, and previews the thesis organization.

1.1 Problem Definition

With the technological development in robotic hardware and software, robots are increasingly used in critical applications such as healthcare, warehouse automation, agriculture, industry, aerospace, and search-and-rescue missions. Advancements in technology have enabled powerful hardware to fit in small spaces, and this ability has paved the way for the production of small-sized intelligent robots. This way, they are gaining an increasingly significant role in daily human lives. A perfect example is home robotics. With efficient lithium-ion batteries and miniaturization of microprocessors and LiDAR sensors, robot vacuums and mops became very popular due to their time-saving cleaning solutions. The same advances have contributed to the popularity of drones among hobbyists. These developments have increased the demand for robots, making reliable navigation systems even more critical. Traditional navigation methods in robotics mostly rely on metric information such as odometry, LiDAR, GPS, or pre-constructed maps. These approaches have proven successful in static environments. However, they face significant challenges in dynamic and unstructured environments. Unlike outdoor navigation, GPS or line-of-sight sensors may fail since walls and ceilings may block GPS from getting readings, and densely populated areas cause noise in sensor readings. Also, furniture and other objects in in-

door environments may vanish or move elsewhere over time due to human interaction. Metric-based navigation algorithms are unaware of these changes, which may cause the system to make erroneous localization estimates and generate incorrect paths. In summary, metric-based navigation methods have the following challenges:

- **Dependency on Expensive Sensors:** Metric-based navigation often relies on high-cost sensors such as LiDAR, stereo cameras, and IMUs, which significantly increase the cost of the robotic system, making it less accessible for budget-constrained applications. Also, they require frequent maintenance and calibration, increasing the operational expenses even more.
- **Error Accumulation:** Metric-based navigation systems require frequent precise metric measurements, which leads to error accumulation over time in GPS-denied environments.
- **Dynamic Environments:** In environments such as shopping malls, warehouses, and laboratories, metric-based navigation systems may fail because these environments change frequently.

The limitations of metric-based navigation systems indicate the need for a paradigm shift in indoor navigation methods towards more adaptive systems that do not rely on precise metric information.

1.2 Motivation

The human navigation system is based on visually recognizing nearby landmarks rather than measuring exact distances [4]. They can detect and consider environmental changes while navigating without requiring precise measurements or metric data. For example, when navigating a shopping mall, one might use instructions like "Turn right at the coffee shop, then left at the pharmacy, walk until you see restrooms, then take the elevator." In other words, they do not know the exact distances between themselves and the surrounding object; they use landmarks to construct topological maps and use the spatial relationship between these landmarks for localization and navigation. Replicating this strategy in robots offers the following advantages:

- **Robustness to Change:** Landmarks can be re-identified even when shelves, desks, or furniture are moved, whereas metric maps become outdated.
- **Sensor Economy:** A single panoramic camera captures the full 360-degree scene, eliminating the need for range and depth sensors.
- **Drift-Resistant Localization:** Since localization is performed by matching landmarks rather than integrating noisy odometry, long-term cumulative position drift is avoided.

Motivated by these benefits, a vision-based topological navigation system inspired by human navigation is proposed to overcome the limitations of metric-based approaches. The framework captures equirectangular images at each node, links them through robust feature matching, and navigates by traversing this landmark graph, achieving reliable indoor localization and path planning without relying on metric information. Therefore, this study builds on earlier metric-free work that also utilizes 360-degree equirectangular images, specifically the approaches explained in [5] and [6]

1.3 Proposed Methods and Models

This thesis introduces a vision-based topological navigation framework for mobile robots in indoor environments without depending on metric information. The core of the proposed topological navigation framework is divided into the following parts:

- **360-Degree Vision-Based Localization:** A 360-degree camera mounted on the robot's head captures the spherical view of the environment, enabling feature extraction and matching algorithms to find matching features between images, even in narrow spaces.
- **8-Point Algorithm for Relative Pose Estimation:** A robust 8-point algorithm is adopted to estimate the translations and relative rotations between nodes represented by equirectangular images. RANSAC is used together with 8-point algorithm to eliminate outliers in the matches.

- **Topological Mapping Using Visual Features:** A topological map is constructed from a video record. Each node in the graph is defined with equirectangular images. A similarity metric is used to eliminate overlapping nodes. Relative rotations and translations, which are obtained using 8-point algorithm, are used to calculate the relative poses of each node according to a selected node.

- **Graph-Based Path Planning:** Uniform cost search algorithm determines the shortest path between the current node and the goal node. A similarity metric is used to localize the robot while traversing the map.

- **Precise Approach to an Object Using Hand-Mounted 360-Degree Camera:** At the goal node, a procedure for approaching a scene of a specific object is defined. A second 360-degree camera is mounted on the robot's hand to calculate the relative orientation and position of the object's scene precisely from triangulation using matched features of two 360-degree cameras.

1.4 Contributions and Novelties

This work has the following key contributions:

- **Metric-free Topological Navigation Framework:** A complete topological navigation pipeline, implemented as modular ROS nodes, that relies solely on 360-degree vision for mapping, localization, path planning, and object-level approach, eliminating the need for metric sensors is proposed.

- **A Novel Topological Mapping Algorithm:** An adaptation of the 8-point algorithm robustly estimates heading vectors and relative orientations, thereby enabling the construction of a scale-independent map without any metric measurements.

- **Extensive Real-World Evaluation:** Extensive tests in indoor environments, including topological map building, forward/reverse navigation, and vision-only object triangulation, demonstrate the framework's robustness.

1.5 The Outline of the Thesis

Chapter 2 surveys existing indoor navigation paradigms and the core components of the proposed framework. Chapter 3 sets out the problem formulation and explains the algorithm in detail. Chapter 4 describes the hardware–software testbed and the experimental procedure. Experimental evaluation of the proposed topological navigation, mapping, and triangulation algorithms are presented and discussed in Chapter 5. Finally, a summary of the thesis and future research directions are outlined in Chapter 6.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter surveys the foundational concepts and related work that underpin the proposed metric-free navigation framework. It begins by contrasting the two main paradigms of indoor robot navigation, metric-based navigation (Section 2.1.1) and metric-free topological navigation (Section 2.1.2), and highlights their respective strengths and weaknesses. It then introduces 360-degree panoramic vision and the equirectangular camera model (Section 2.2). Section 2.3 reviews recent advances in learned feature extraction and matching, while Section 2.4 covers the robust relative pose estimation method, the eight-point algorithm with RANSAC. In Section 2.5, state-of-the-art image similarity measure and visual place recognition methods are reviewed. Section 2.6 provides a brief overview of the adopted obstacle avoidance technique used for experimental safety. Section 2.7 describes the ROS-based software infrastructure. Finally, Section 2.8 outlines the specific research gaps addressed by this thesis.

2.1 Indoor Navigation Paradigms

Robust navigation is among the key challenges for autonomous robots, especially in GPS-denied indoor environments. In the scope of this thesis, navigation methods are categorized as metric-based navigation and metric-free topological navigation. Metric-based navigation relies on precise distance measurements and dense maps. Conversely, metric-free topological navigation is based on recognizing abstract environmental structures and landmarks.

2.1.1 Metric-Based Navigation

Metric-based navigation systems estimate a robot's six-degree-of-freedom, 6-DOF, pose by fusing measurements from metric sensors, such as LiDAR, GPS, and IMU, into a geometric map [7]. Since the robot state is expressed in metric units, path planning can be handled by graph search algorithms such as A*, D* Lite, or sampling-based planners such as RRT* [8]. This section briefly reviews the main branches of metric navigation, highlights influential algorithms, and summarizes their common limitations.

2.1.1.1 Wheel-Odometry and 2D LiDAR SLAM

Early indoor robots relied on wheel encoders and a planar laser scanner to perform Simultaneous Localization and Mapping (SLAM). GMapping [9] introduced a Rao-Blackwellized particle filter that maintains a set of trajectory hypotheses while updating an occupancy-grid map to achieve real-time map construction in small-scale indoor environments. Hector SLAM [10] replaced the particle filter with real-time scan matching to achieve better accuracy in cluttered or dynamic scenarios by incorporating scan matching with high-frequency LIDAR data. Google's Cartographer [11] extends these ideas by fusing scan matching with loop-closure detection and real-time pose-graph optimization to generate consistent 2D and 3D maps in large-scale indoor environments. Although state-of-the-art 2D LiDAR SLAM can reach centimeter-level accuracy in empty planar corridors, they fail in multi-level environments (stairs, ramps) and accumulate drift whenever the laser plane is occluded [7].

2.1.1.2 Visual–Inertial and Stereo SLAM

Researchers combined cameras with inertial sensors to obtain metric scale maps and improve robustness where wheel odometry or LiDAR are unreliable, giving rise to visual-inertial odometry (VIO) and stereo SLAM methods. ORB-SLAM3 [12] fuses monocular, stereo, and RGB-D image streams with metric measurements received from IMU to achieve state-of-the-art accuracy in hand-held camera sequences and UAV flights. Learning-based SLAM systems such as DeepSLAM [13], DeepVO [14],

and DeepFactors [15] replace traditional hand-crafted descriptors such as SIFT [16] or ORB [17] with features learned by convolutional and recurrent networks, thereby increasing robustness to illumination and texture variations. Nevertheless, scale drift persists when loop closure is infrequent.

2.1.1.3 Dense RGB-D Mapping

RGB-D sensors provide metric depth for every pixel and simplify pose estimation through Iterative Closest Point (ICP) [18]. KinectFusion [19] demonstrated how to integrate ICP on the GPU for real-time fusion of depth frames into a volumetric model, while ElasticFusion [20] added non-rigid loop-closure detection to correct accumulated positional drift over large or revisited areas. Although these dense methods can reconstruct highly detailed 3D meshes, they require substantial computational resources, and their structured light depth measurements degrade under sunlight or on reflective and transparent surfaces.

2.1.1.4 Limitations of Metric Approaches

Despite significant progress, metric navigation systems share three fundamental weaknesses:

- **Drift Accumulation:** Small odometry or scan alignment errors integrate over time. Loop closure, external beacons, or GPS are required to maintain accuracy.
- **Sensor Cost and Calibration:** High-resolution LiDAR sensors, stereo camera systems, and depth sensors significantly increase initial deployment costs and require frequent calibration and maintenance.
- **Sensitivity to Environmental Changes:** Metric maps record precise geometric relationships to fixed landmarks; thus, changes such as moved furniture or objects can invalidate existing maps and trigger costly and time-consuming remapping.

These shortcomings motivate a shift toward topological navigation, where the envi-

ronment is represented as a graph of visually distinctive places rather than a dense metric field. In Section 2.1.2, metric-free topological approaches are reviewed.

2.1.2 Metric-Free Topological Navigation

In metric-free topological navigation methods, the environment is represented as a graph consisting of nodes (distinct locations or landmarks) and edges (traversable paths) rather than relying on exact metric measurements or dense geometric maps. Each node typically corresponds to a visually or perceptually distinctive landmark, with edges capturing spatial connectivity or navigable routes between these landmarks. This section reviews the evolution of topological navigation methods, from early symbolic approaches to recent advances in panoramic vision and deep learning-based landmark recognition, and discusses their limitations.

2.1.2.1 Early Symbolic and Qualitative Methods

Early topological approaches employed symbolic or qualitative representations of the environment. Kuipers and Byun [21] introduced the concept of cognitive maps where places are represented as nodes connected through edges defined by distinct travel actions or paths. Such representations depend on recognizing unique landmarks or other visual signatures rather than maintaining precise metric accuracy; therefore, they are advantageous for dealing with dynamic environments.

2.1.2.2 Vision-Based Topological Methods

Later approaches leveraged visual data captured by conventional cameras for topological navigation. Zivkovic [22] utilized local image descriptors for place recognition, demonstrating the robustness of feature-based approaches in visually repetitive environments. Cummins and Newman [23] introduced FAB-MAP, a probabilistic appearance-based localization method that uses a bag-of-words model for place recognition and became foundational in visual topological SLAM. Recent advances in visual place recognition and learned feature matching have significantly improved

topological navigation. PlaceNav [24] and TopoNav [25] leverage appearance-based localization with learned descriptors to achieve reliable navigation in complex indoor environments.

2.1.2.3 360-Degree and Panoramic Vision Approaches

The use of panoramic or 360-degree cameras significantly improved visual topological navigation methods by providing the entire view of the environment from a single viewpoint. Ulrich and Nourbakhsh [26] were among the first to demonstrate appearance-based topological localization using 360-degree images for robust visual landmark identification. Recent studies [5, 6] applied deep-learning-based feature extraction and matching algorithms such as SuperPoint [27] and SuperGlue [28], improving place recognition and localization performance in indoor environments. These works highlight the benefit of combining 360-degree images with learned visual descriptors.

2.1.2.4 Limitations of Topological Approaches

Despite their robustness to environmental changes, state-of-the-art topological navigation methods have the following limitations:

- **Limited Generalization Capacity:** Deep-learning–based place recognition methods may require retraining (e.g., PlaceNav) on each new environment to maintain accuracy.

- **Lack of Geometric Precision:** Purely topological maps omit metric distances, meaning that the shortest graph-based path may not correspond to the actual spatial minimum.

- **Manipulation Integration Gap:** Topological navigation algorithms typically do not support object-level vision-only interaction.

2.2 360-degree Camera Model and Equirectangular Projection

360-degree cameras can capture the full view of the environment, enabling the robot to perceive its surroundings entirely without leaving any blind spots. The 360-degree camera used in this thesis outputs 2D equirectangular images, a 2D representation of the spherical view. The relation between the spherical coordinates and the equirectangular image is given in Figure 2.1.

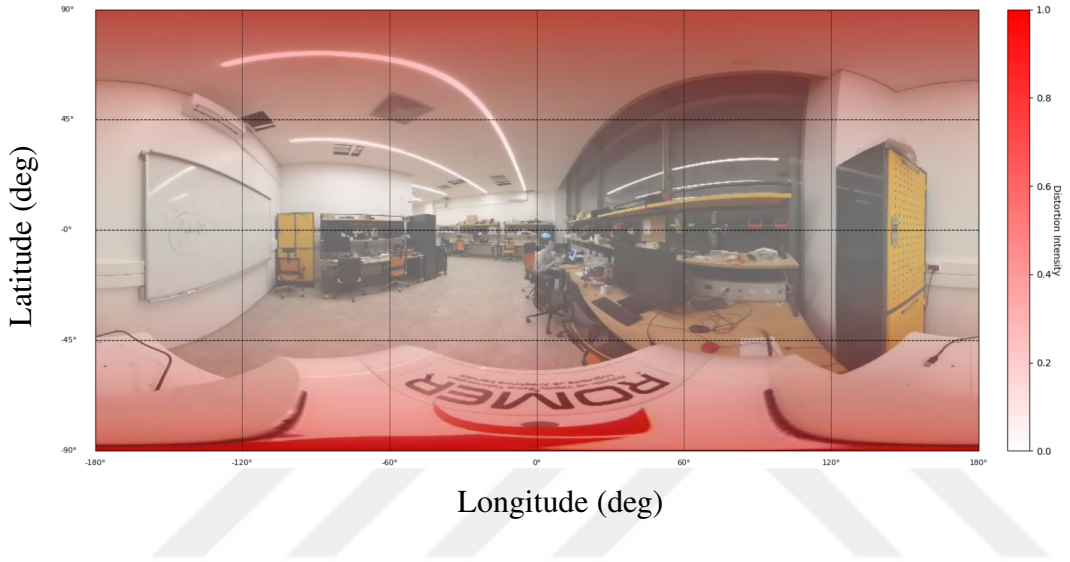


Figure 2.1: Equirectangular projection. Red color indicates the amount of distortion.

In the equirectangular projection, each pixel in the image corresponds to a specific direction in 3D space, parameterized by two angular values: *longitude* and *latitude*. Longitude refers to the horizontal angle (azimuth) around the vertical axis, typically ranging from $-180°$ to $180°$, while latitude refers to the vertical angle (elevation) from the equator to the poles, ranging from $-90°$ to $90°$. These angles are mapped linearly to the image axes, with longitude corresponding to the horizontal axis (columns) and latitude to the vertical axis (rows). While this mapping preserves angular relationships, it introduces distortions that increase near the poles. In Figure 2.1, areas near the poles appear more distorted, as indicated by the stronger red color. Unlike the pinhole camera model with focal length and principal point, the equirectangular model has no additional intrinsic parameters beyond its resolution [29]. Therefore, back-projecting a pixel to a 3D ray is a simple trigonometric conversion. This conversion

is explained in Section 3.3.

2.3 Feature Extraction and Matching

Feature extraction is defined as the process of identifying unique elements (also called as keypoints, features, or interest points) from an image. For each keypoint extracted from an image, a descriptor is assigned which captures the local appearance of the region around the corresponding keypoint. The selected keypoints must remain recognizable despite changes in rotation, translation, scale, or illumination to achieve robust feature matching. Feature matching is the process of pairing features extracted from two images. Correspondences between two different images are used to calculate the relative translation and rotation between the origins of these images.

Feature-matching methods fall into two categories: detector–descriptor pipelines and contextual or dense feature matchers. Detector–descriptor pipelines first select a sparse set of keypoints (e.g., SIFT [16], ORB [17]) and then compute a descriptor for each. Learned variants such as SuperPoint [27], DIScrete Keypoints (DISK) [30] and Adaptive Learned Invariant Keypoint Extractor (ALIKEd) [1] improve repeatability and descriptor distinctiveness by training neural networks end-to-end. Contextual or dense feature matchers skip explicit detection and instead use attention or transformer networks (e.g., SuperGlue [28], LightGlue [2], LoFTR [31], RoMa [32]) to establish correspondences either among all pixels or among a large set of candidate keypoints, yielding higher match density at the cost of increased computation.

In this work, a robust feature extractor and matcher pair is required to utilize 8-point algorithm for precise relative pose estimation between images and for correct topological map construction. ALIKEd and LightGlue are selected over other algorithms due to their advantages. ALIKEd's key strengths compared to other algorithms are:

- **Keypoint Quality:** Traditional detectors such as ORB and SIFT often produce fewer repeatable or distinctive keypoints under challenging conditions. Learned methods like SuperPoint, DISK, and ALIKEd significantly improve repeatability and descriptor uniqueness. In particular, ALIKEd places keypoints along edges and textured regions, yielding more reliable matches than SuperPoint [1].

- **Computational Efficiency:** Although SIFT is robust, it incurs a high computational cost; ORB is faster but less distinctive. Among learned detectors, ALIKEd is faster than SuperPoint and DISK while maintaining comparable quality [1], effectively balancing speed and accuracy.

Key strengths of LightGlue compared to other algorithms are:

- **Matching Accuracy:** LightGlue achieves near–SuperGlue accuracy [28, 2]. On the other hand, newer methods like LOFTR and RoMa may offer marginally higher match rates at the cost of significantly greater runtime.
- **Computational Efficiency:** LightGlue matches 1000 keypoints in 15 ms on a modern GPU, compared to 40 ms for SuperGlue and 80 ms+ for LoFTR/RoMa [2, 32]. This balance makes LightGlue well-suited to real-time applications.

Therefore, ALIKEd-LightGlue pipeline is utilized throughout the thesis. In addition, features found at the bottom half of the images are discarded for the head camera of the robot since features belonging to static objects (ceiling, air conditioner, shelves) are most likely to be in the upper half of the image. In this way, ambiguity caused by potentially dynamic elements is prevented at some point because features obtained from rearranged furniture, moving objects, and people wandering around are excluded. For the hand camera, extracted features of the whole image are kept.

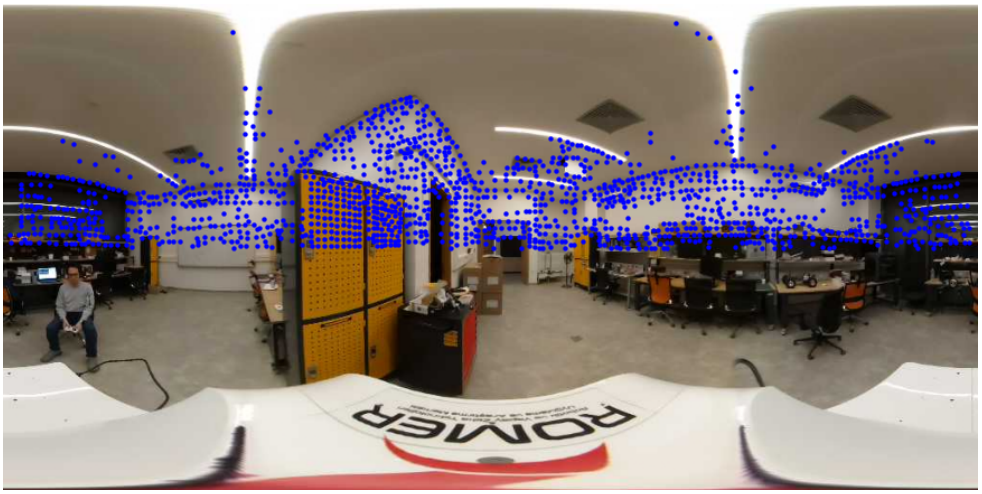


Figure 2.2: Keypoints of extracted features from an image, features at the bottom half of the image are excluded.

2.3.1 ALIKEd

Adaptive Learned Invariant Keypoint Extractor (ALIKEd) is a deep-learning-based feature extraction method. The architecture of the ALIKEd starts with a CNN backbone, which processes the input image and outputs feature maps that contain texture and edge information of the scene (Feature Encoding). Then, outputs are up-sampled and combined into a single multi-scale feature map (Feature Aggregation). Then, the Score Map Head outputs a probability map containing each pixel's likelihood corresponding to a keypoint. Differentiable Keypoint Detection converts the probability map into pixel coordinates of keypoints. SDDH (descriptor head) extracts a local descriptor for each detected keypoint to obtain the descriptors. Figure 2.3 shows the network's visualized architecture.

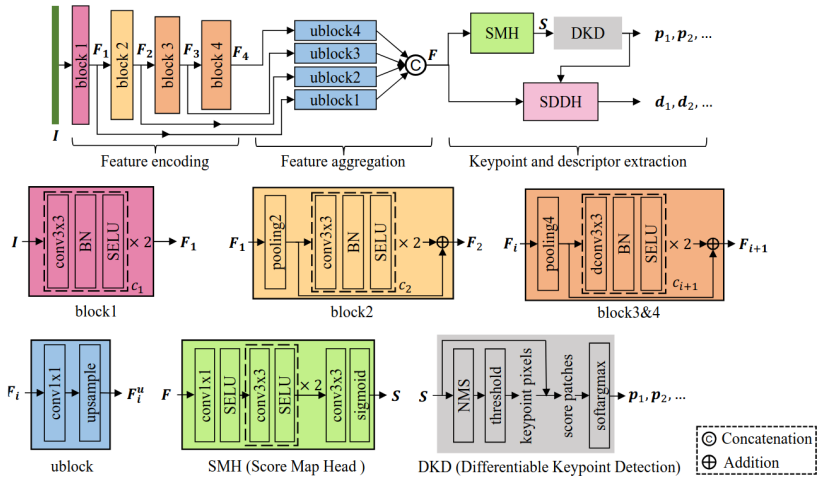


Figure 2.3: Architecture of ALIKEd, taken from [1].

2.3.2 LightGlue

LightGlue is a deep-learning-based feature-matching algorithm that matches the extracted key points of two images using their associated descriptors. LightGlue utilizes a neural attention mechanism to consider the global context of all keypoints in both images rather than treating each descriptor pair independently. LightGlue receives two sets of keypoints and descriptors, each with a variable number of keypoints. Cross-attention is applied to this set of features, and each descriptor is refined by

considering neighboring keypoints and global context. After cross-attention, a match confidence score is generated at the final selection step, where the outlier filtering is done using the confidence scores of the matched features. Visualization of the network is given in Figure 2.4.

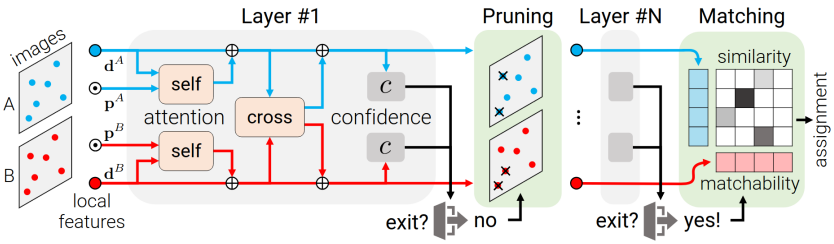


Figure 2.4: Architecture of LightGlue, taken from [2].

2.4 Relative Pose Estimation with the 8-Point Algorithm and RANSAC

Estimating the relative pose between two views is a fundamental step in visual navigation. The *heading vector* $\mathbf{h}$ is defined as the unit vector pointing from the first camera center, O, a toward the second, O'. The heading vector encodes the direction (but not distance) between camera centers. In a topological map, this heading vector tells the robot which way to drive when moving from one node to its neighbor. The heading vector calculation problem is described in Figure 2.5. Correspondences are obtained from feature matching and feature extraction algorithms to calculate the heading vector.

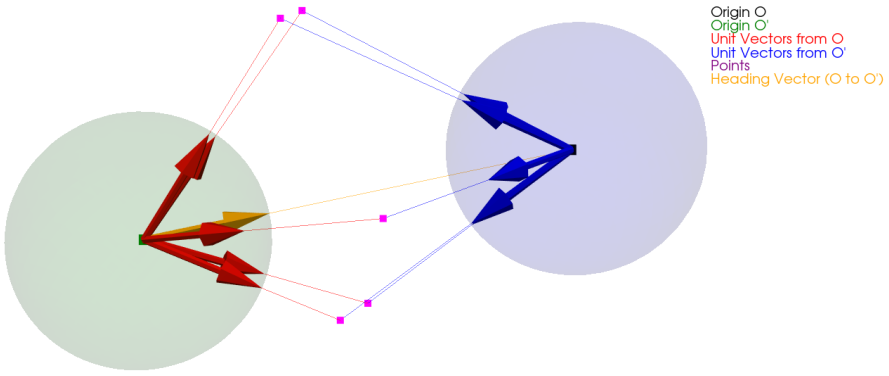


Figure 2.5: Illustration of the heading vector calculation problem. Purple points represent the correspondences between two images.

For heading vector calculation [5], [6], [33] utilize an approach where geometric relations between bisectors of matched points are used, rather than solving it analytically. A precise and robust approach is needed to calculate the heading vector in a real-world application. Therefore, 8-point algorithm [3] is adapted to solve this problem analytically. 8-point algorithm proved stability for large field-of-view images [34], making it a popular choice for 360-degree images [35], [36], [37]. 8-point algorithm recovers the so-called essential matrix from at least eight pairs of matched image points. The essential matrix encodes the rotation between the two camera frames and the translation direction (heading vector). For spherical images, a more robust variant of 8-point algorithm is proposed in [38], which applies spherical normalization to improve stability. Therefore, Robust 360-8PA [38] is adopted in this thesis to estimate the heading vector and relative orientation between two images.

The essential matrix obtained from 8-point algorithm is sensitive to mismatches and noise. To make the process robust, Random Sample Consensus (RANSAC) [39] is employed: repeated minimal subsets of correspondences are used to propose candidate essential matrices, and each proposal is scored by counting how many matches agree with the underlying epipolar geometry. The model with the highest inlier set is refined to yield a final estimate of rotation and translation direction that can tolerate a high fraction of outliers. Therefore, RANSAC is employed in this work to eliminate incorrect feature matches before pose estimation. Detailed mathematical explanation of the 8-point algorithm and RANSAC are given in Section 3.4.

2.5 Image Similarity Measure and Visual Place Recognition

A similarity metric numerically measures how close or alike two images or feature sets are. In a topological map, each node represents a place or viewpoint. It enables localization by matching a live query image to the most similar node in the map. Second, it enforces node uniqueness during map construction by preventing insertion of new nodes whose similarity to an existing node exceeds a threshold.

The commonly used traditional methods for measuring image similarity are Root Mean Square Error (RMSE) and Structural Similarity Index (SSIM) [40]. SSIM

measures image similarity by comparing luminance, contrast, and structure, whereas RMSE is computed as the root mean square of pixel differences. In other words, they may fail to recognize similarity when orientations between target and current images differ. This case is described in Figure 2.6.

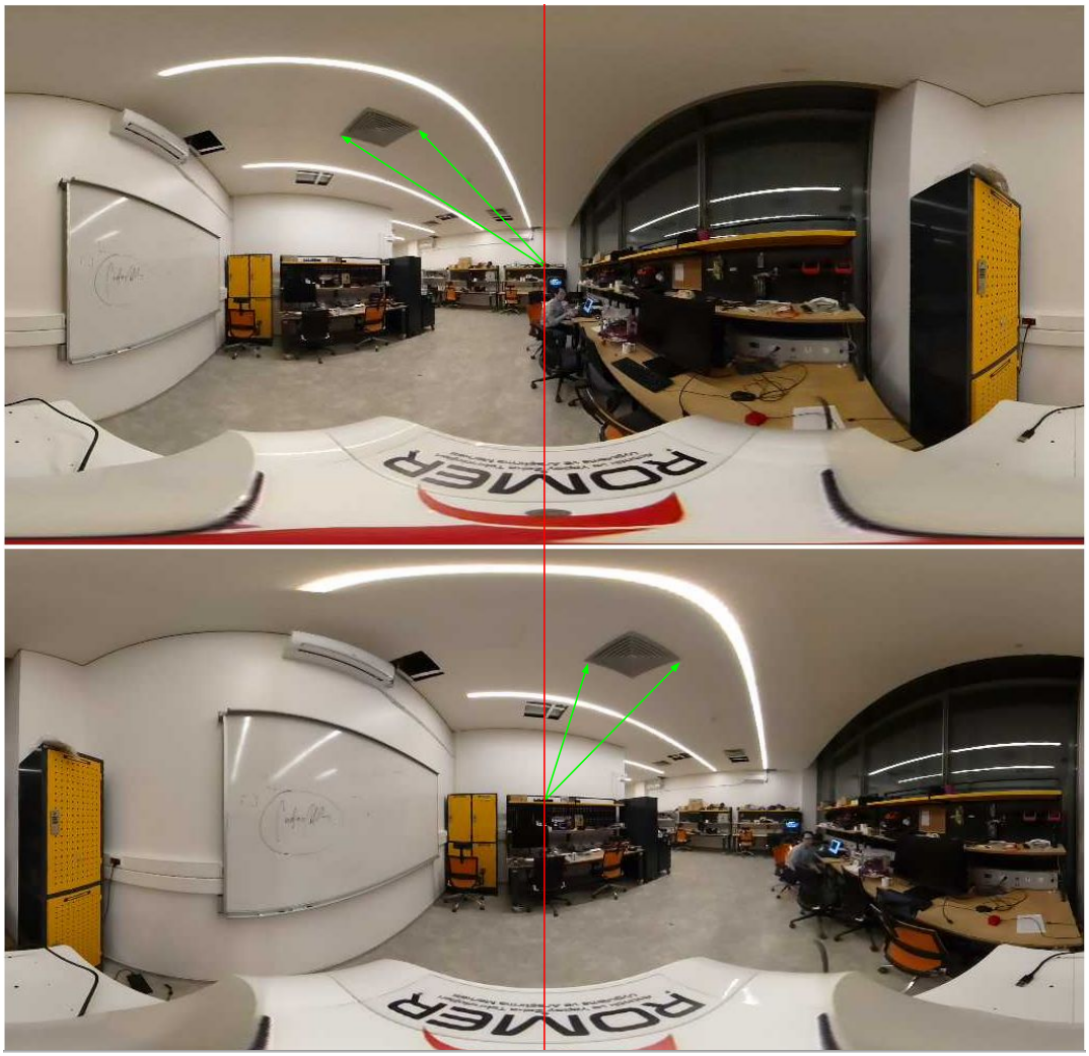


Figure 2.6: A special case where images are taken from very close positions with high azimuth angle difference.

Beyond these traditional techniques, some of the deep-learning-based state-of-the-art approaches like Bag of Queries (BoQ) [41], NetVLAD [42], Variational Autoencoder (VAE) [43] are also used for place recognition. NetVLAD and BoQ are unsuitable for real-time applications due to their high computational cost. VAEs have less computational cost compared to NetVLAD and BoQ. These global descriptors often require

heavy downsampling or large models, making them less suited for full-resolution 360-degree frames. Also, VAEs are not explicitly designed for clustering, meaning clustering-specific loss functions may be required for training. Therefore, inspired by the algorithm given in [33], an image similarity measure algorithm is defined in Section 3.5.

2.6 Obstacle Avoidance

Even though obstacle avoidance is not in the scope of this thesis, a basic safety layer is required for experiments. For this purpose, a simple Artificial Potential Field (APF) [44], where repulsive force is computed from LiDAR point-cloud data, is overlaid on the vision-derived heading vector to prevent collisions during navigation tests (Section 3.8.1). Developing a fully metric-free avoidance planner lies outside the core scope and is left for future work.

2.7 Robot Control Architecture

Robot control architecture is defined as the organization of software and hardware components that enable a robot to sense its environment, make decisions, and perform actions to complete specific tasks. In other words, it is defined as the working principle of a robot's decision-making software. In this work, Robot Operating System (ROS) [45] is adopted. In ROS, the robot's functionality is divided into smaller nodes. Each node performs a specific task and communicates with each other through topics, services, or actions. Topics function as buffers where the publish/subscribe model is adopted. In Section 3.8, each major component of the proposed framework is encapsulated in its own ROS node, enabling clear separation of concerns and easy experimentation in simulation or on real hardware.

2.8 Research Gaps

To address the challenges given in Sections 2.1.1.4 and 2.1.2.4, the proposed framework combines 360-degree view with deep-learning-based feature extraction/matching and 8-point pose estimation with RANSAC, producing a topological graph whose edges encode scale-free spatial relationships between places. The proposed framework addresses several key gaps in the existing indoor navigation research:

- **Elimination of Metric-Sensor Dependence:** By relying on the 360-degree vision and metric-free pose estimation, the system removes the need for LiDAR, depth cameras, wheel odometry, or IMUs, reducing both hardware cost and calibration overhead.
- **Generalization without Environment-Specific Training:** Unlike learned place-recognition methods that require retraining or fine-tuning in each new environment, feature extractors and matchers ensure robust performance in indoor environments without requiring additional training.
- **Integrated Vision-Only Manipulation:** Traditional topological frameworks leave object-level interactions to metric systems. In this work, a dual 360-degree camera triangulation algorithm provides centimeter-level end-effector placement purely from visual data and seamlessly unites navigation and manipulation.
- **Scale-Free and Spatially Informed Mapping:** Traditional topological graphs treat every edge as equal length, meaning the shortest graph path can differ significantly from the actual shortest path. By estimating relative poses (up to an unknown global scale) for every node pair, the proposed framework weights each edge with an approximate distance to closely track the actual shortest path.
- **Robustness to Environmental Changes:** By extracting features only from the upper hemisphere of 360-degree images, where static elements like walls and ceilings dominate, the system ignores transient floor or furniture level changes. Combined with learned feature matching and RANSAC outlier rejection, matches caused by moving obstacles or rearranged furniture are dramatically reduced.

CHAPTER 3

METHODOLOGY

This chapter details the design and implementation of the proposed metric-free, vision-only topological navigation framework. First, the overall architecture is summarized (Section 3.1), and the problem is breakdown into its core components (Section 3.2). Then, each module is formulated and explained in Sections 3.3, 3.4, 3.5, 3.6, 3.7. Finally, obstacle avoidance, motion control, ROS node architecture, and state machine are explained in Section 3.8.

3.1 Proposed Algorithm

As detailed in Chapter 2, this thesis develops a vision-based topological navigation framework for indoor mobile robots without relying on metric information. The robot captures 360-degree equirectangular images at discrete waypoints (nodes). Feature extraction (ALIKEd) and matching (LightGlue) algorithms find correspondences between images. These correspondences are back-projected into 3D unit vectors. A RANSAC-guided robust 8-point algorithm then recovers the relative rotation and heading vector between views, yielding scale-free links. These relative poses are chained, anchored to a chosen origin, and assembled into a topological map whose nodes carry 360-degree equirectangular images and whose edges encode accurate spatial directions. During deployment, uniform-cost graph search navigates this map, while a correspondence-based similarity metric localizes the robot to the nearest node in real-time. Finally, at the goal node, a second 360-degree camera on the manipulator uses matched keypoints from both head and hand views to triangulate a target object's position, allowing the arm to servo precisely to it without any metric sensor.

3.2 Problem Formulation

The problem formulation is divided into five sections, each covering a key aspect of the proposed algorithm.

- **Camera Modeling and Back-Projection:** Given an equirectangular image of resolution $W \times H$, each pixel (u, v) is back-projected to its corresponding 3D unit vector $\mathbf{p} \in \mathbb{R}^3$. This establishes the link between image coordinates and 3D directions; see Section 3.3.
- **Heading Vector and Relative Orientation Estimation:** From two successive 360-degree images, the *heading vector* $\mathbf{h}$ (the unit–length translation direction) and the relative rotation $R \in SO(3)$ between cameras are recovered. Matched feature correspondences are fed into a RANSAC-guided 8-point algorithm to achieve a robust, scale-free relative pose estimation; see Section 3.4.
- **Similarity Metric:** A rotation-invariant similarity measure between two panoramic images is defined so that similar views are not added as new nodes (Node Uniqueness) during mapping. At run–time, a live query image can be matched to its nearest stored node (Localization); see Section 3.5.
- **Topological Map Construction and Path Planning:** Nodes (each carrying an equirectangular image) and their pairwise relative poses are assembled into a scale-free directed graph. Edges inherit accurate spatial directions from the heading vectors, while a graph search algorithm (uniform-cost) finds optimal routes between arbitrary start and goal nodes; see Section 3.6.
- **3D Position Estimation from Triangulation:** At the goal node, a target object's 3D position is triangulated from two 360-degree views (one on the head, one on the end-effector). This allows the manipulator to servo precisely to the object without any metric depth sensor; see Section 3.7.

3.3 Camera Modeling and Back-Projection

As explained in Section 2.2, each pixel in the equirectangular projection corresponds to specific angular values, given in Equation 3.1.

$$u = \frac{\mu + \pi}{2\pi} \cdot W, \qquad v = \frac{\lambda}{\pi} \cdot H + \frac{H}{2} \tag{3.1}$$

where:

- W, H: Width and height of the equirectangular image
- $u \in [0, W]$: Horizontal pixel index.
- $v \in [0, H]$: Vertical pixel index.
- $\mu \in [-\pi, \pi]$: Azimuth angle.
- $\lambda \in [-\pi/2, \pi/2]$: Elevation angle.

For an arbitrary point in the image plane with known pixel indexes, the corresponding 3D unit vector is calculated using Equations 3.2 and 3.3. This relation is also described in Figure 3.1. These unit vectors become the inputs to the 8-point algorithm that is explained in Section 3.4

$$\mu = \frac{u}{W} \cdot 2\pi - \pi, \qquad \lambda = \frac{v}{H} \cdot \pi - \frac{\pi}{2} \tag{3.2}$$

$$\mathbf{p} = \begin{bmatrix} \cos\lambda \cdot \cos\mu \\ \cos\lambda \cdot \sin\mu \\ \sin\lambda \end{bmatrix}. \tag{3.3}$$

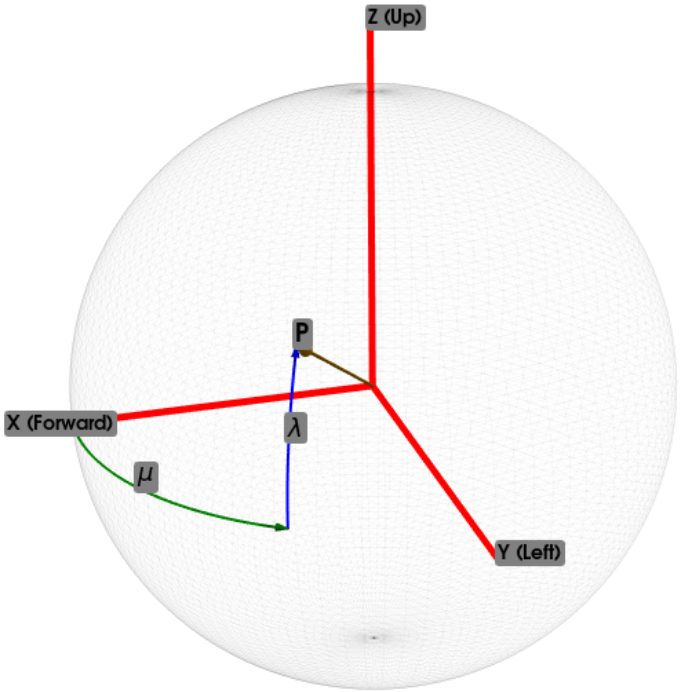


Figure 3.1: Camera coordinate frame. The right-hand rule is used for the directions of axes.

3.4 Heading Vector and Relative Orientation Estimation

This section gives the mathematical formulation of the adopted Robust 360-8PA and RANSAC (Section 2.4). Feature correspondences are obtained using the ALIKEd-LightGlue pipeline (Section 2.3), and then their back-projected unit vectors are fed into the RANSAC-guided 8-point algorithm pipeline.

3.4.1 8-Point Algorithm

8-point algorithm is used to compute the fundamental matrix F by using at least eight corresponding points $x_i, x_i^{'}$, where $x_i = [u_i, v_i, 1]^\top$ and $x_i^{'} = [u_i^{'}, v_i^{'}, 1]^\top$ are homogeneous coordinates (pixel indexes with an extra dimension) in the first and second images respectively. Epipolar constraint is defined in Equation 3.4.

$$(x^{'})^{\top} F x = 0 \tag{3.4}$$

The epipolar constraint ensures that the corresponding point in the second image, $x^{'}$, lies on the epipolar line $l = Fx$.

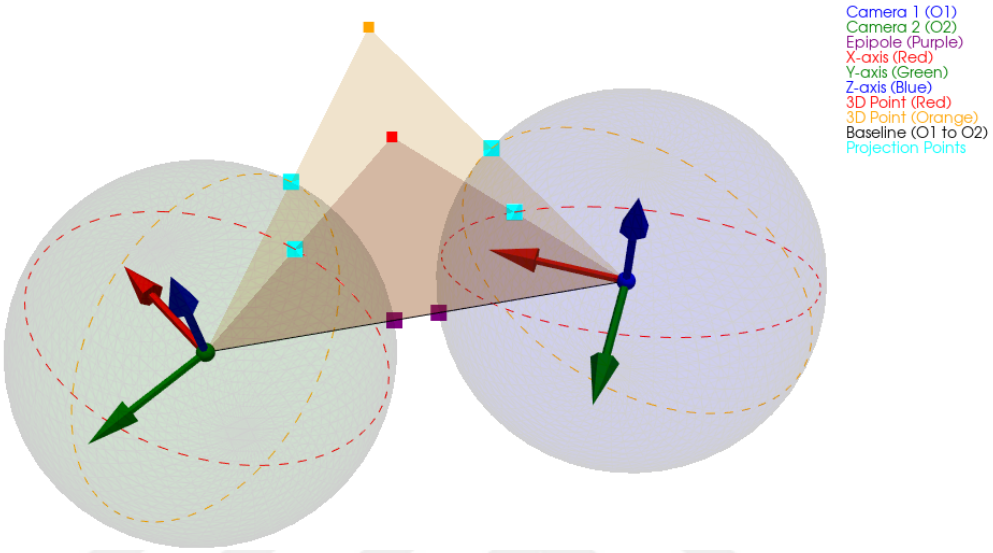


Figure 3.2: Visualization of the epipolar geometry.

F contains the intrinsic and extrinsic parameters of both cameras. By definition, F is rank 2 and has 7 degrees of freedom [3]. It can be decomposed as given in Equation 3.5.

$$F = (K^{'})^{\top} E K^{-1} \tag{3.5}$$

where:

- $E \in \mathbb{R}^{3x3}$: Essential matrix.
- $K \in \mathbb{R}^{3x3}$: Camera matrix.

In Equation 3.5, the camera matrix, K, is the matrix used to convert homogeneous coordinates to cartesian coordinates, and the essential matrix, E, encapsulates rotation

and translation between camera origins. In other words, K contains intrinsic parameters, and E contains extrinsic parameters. Moreover, since the unit vectors of the corresponding features, $\mathbf{P}$, are determined via Equation 3.3, the epipolar constraint equation naturally reduces to its simplified form given by Equation 3.6.

$$(\mathbf{P}^{'})^{\top} E \mathbf{P} = 0 \tag{3.6}$$

where:

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} p_{1_x} \dots p_{n_x} \\ p_{1_y} \dots p_{n_y} \\ p_{1_z} \dots p_{n_z} \end{bmatrix}^{\top}, \quad \mathbf{P}^{'} = \begin{bmatrix} p^{'}_{1_x} \dots p^{'}_{n_x} \\ p^{'}_{1_y} \dots p^{'}_{n_y} \\ p^{'}_{1_z} \dots p^{'}_{n_z} \end{bmatrix}^{\top} \tag{3.7}$$

Visualization of the matched unit vectors, $\mathbf{P_i}$, of the same features, k_i, is given in Figures 3.3 and 3.4.

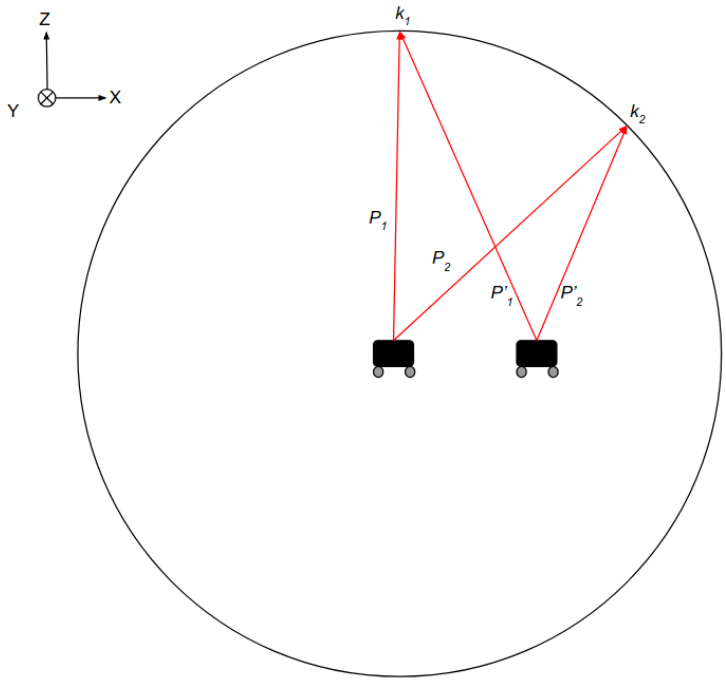


Figure 3.3: Top view of the matched unit vectors.

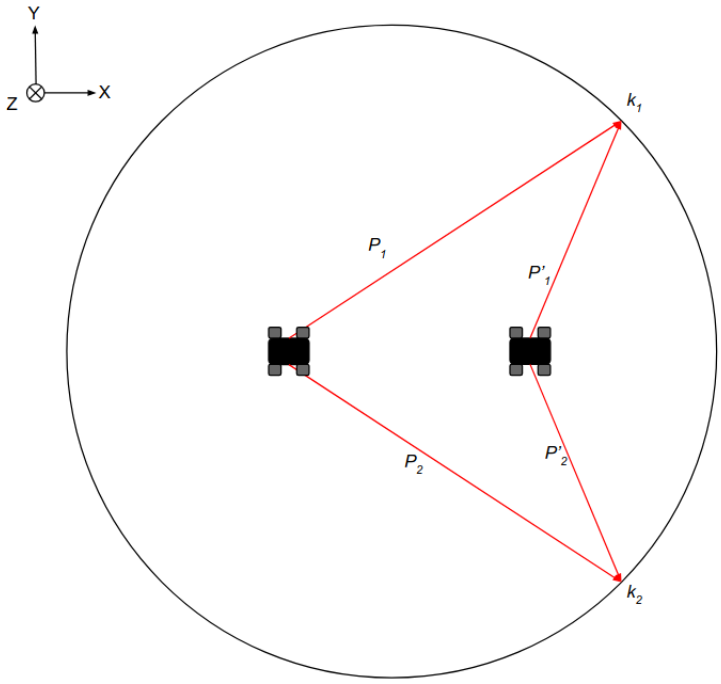


Figure 3.4: Side view of the matched unit vectors.

Similar to the fundamental matrix, the essential matrix is rank 2 and, by definition, has 7 degrees of freedom. Its decomposition gives translation and relative rotation between images. It can be written as given in Equation 3.8 in terms of translation and relative rotation matrices.

$$E = [\mathbf{t}]_X R \tag{3.8}$$

where:

- $R \in R^{3x3}$: Rotation matrix from first image to second image.
- $\mathbf{t} \in R^{3x1}$: Translation vector from first image to second image.
- $[\mathbf{t}]_X \in R^{3x3}$: Skew symmetric of t.

By applying singular value decomposition (SVD) to the linear system derived from the epipolar constraint equation, $A = (\mathbf{P}')^\top E\mathbf{P}$, the essential matrix E can be esti-

mated. This system is represented in matrix form as shown in Equation 3.9, where the matrix A is constructed from the matched unit vectors.

$$A = \begin{bmatrix} p'_{1_x}p_{1_x} & p'_{1_x}p_{1_y} & p'_{1_x}p_{1_z} & p'_{1_y}p_{1_x} & p'_{1_y}p_{1_y} & p'_{1_y}p_{1_z} & p'_{1_z}p_{1_x} & p'_{1_z}p_{1_y} & p'_{1_z}p_{1_z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p'_{n_x}p_{n_x} & p'_{n_x}p_{n_y} & p'_{n_x}p_{n_z} & p'_{n_y}p_{n_x} & p'_{n_y}p_{n_y} & p'_{n_y}p_{n_z} & p'_{n_z}p_{n_x} & p'_{n_z}p_{n_y} & p'_{n_z}p_{n_z} \end{bmatrix}$$

$$b = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33} \end{bmatrix}$$

$$A = U\Sigma V^\top \tag{3.9}$$

Since E must be rank 2, a low-rank approximation is performed by setting its smallest singular value to zero. Given the singular value decomposition (SVD) $E = U\Sigma V^\top$, a new diagonal matrix $\Sigma_{\text{low_rank}}$ is constructed by setting the smallest singular value to zero, and the essential matrix is reconstructed as shown in Equation 3.10. This enforces the rank-2 constraint on E.

$$\Sigma_{low_rank} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_{low_rank} = U\Sigma_{low_rank}V^\top \tag{3.10}$$

The standard 8-point algorithm [3] is used to estimate the essential matrix. For spherical images, a more robust variant is proposed in [38], which applies spherical normalization to improve stability. A normalization matrix N is applied to both sets of unit vectors:

$$N = \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & K \end{bmatrix}, \qquad \hat{\mathbf{P}} = N\mathbf{P}, \qquad \hat{\mathbf{P}}' = N\mathbf{P}' \tag{3.11}$$

S and K are scaling factors obtained by minimizing the epipolar residual using least squares, [38]. The essential matrix is estimated using the normalized vectors and refined by minimizing the loss defined in Equation 3.12.

$$
\begin{aligned}
\hat{E} &= \text{SVD}\left((\hat{\mathbf{P}'}_i)^\top E \hat{\mathbf{P}}_i\right) \\
r_i &= (\hat{\mathbf{P}'}_i)^\top \hat{E} \hat{\mathbf{P}}_i \\
\mathcal{L}_{\text{epipolar}} &= \sum_{i=1}^{N} r_i^2
\end{aligned}
\tag{3.12}
$$

After obtaining the essential matrix from Equation 3.12 it is decomposed by SVD to obtain R and t [3].

$$
\begin{gathered}
E = U\Sigma V^\top, \quad W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{t}_1 = u_3, \quad \mathbf{t}_1 = -u_3, \quad R_1 = UWV^\top, \quad R_2 = UW^\top V^\top
\end{gathered}
\tag{3.13}
$$

From Equation 3.13, four possible solutions are obtained which are $(R_1, \mathbf{t}_1)$, $(R_2, \mathbf{t}_1)$, $(R_1, \mathbf{t}_2)$, $(R_2, \mathbf{t}_2)$. The chirality condition is used for selecting the correct solution [3] in standard camera setups. The chirality condition is based on selecting the solution having its reconstructed 3D points lying in front of both cameras. In other words, it selects the configuration where the reconstructed points have positive depth in both camera coordinate frames. This condition helps choose the correct solution from the four possible solutions.

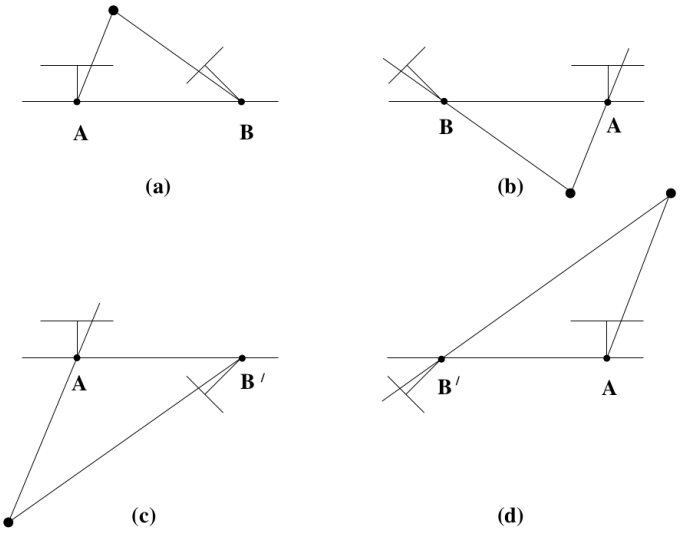


Figure 3.5: The four possible solutions of 8-point algorithm, taken from [3].

For 360-degree cameras, all directions around the camera are visible, making it unreliable to assume that reconstructed 3D points lie in front of the camera. Thus, the chirality condition used in standard cameras is not applicable. Instead, the correct pose is selected by minimizing the angular reprojection error after triangulating 3D points. Direct Linear Transform (DLT) is applied to select the correct solution [3]. DLT is an algebraic method for solving triangulation and pose selection problems in computer vision. The pose of the first camera is taken to be 4×4 identity matrix, and the pose of the second camera is taken as M, which is a 4×4 matrix having $M[0:3, 0:3] = R$ (rotation) and $M[0:3, 3] = t$ (translation). In triangulation, the goal is to find 3D points $X \in R^{4\times n}$ and $X^{'} \in R^{4\times n}$ for corresponding unit vectors P_n and $P_n^{'}$. Each 3D point X_k in homogeneous coordinates must satisfy the projection constraints given in Equation 3.14.

$$P_{k,\times}(IX_k) = 0, \quad P_{k,\times}^{'}(MX_k) = 0, \tag{3.14}$$

$P_{k,\times}$ and $P_{k,\times}^{'}$ are skew-symmetric matrices for the k'th pair of matched unit vectors P_k and $P_k^{'}$, defined as in Equation 3.15.

$$P_{k,\times} = \begin{bmatrix} 0 & -p_{k_z} & p_{k_y} \\ p_{k_z} & 0 & -p_{k_x} \\ -p_{k_y} & p_{k_x} & 0 \end{bmatrix}, \quad P_{k,\times}^{'} = \begin{bmatrix} 0 & -p_{k_z}^{'} & p_{k_y}^{'} \\ p_{k_z}^{'} & 0 & -p_{k_x}^{'} \\ -p_{k_y}^{'} & p_{k_x}^{'} & 0 \end{bmatrix} \tag{3.15}$$

Stacking these constraints for all n pairs of matched unit vectors forms a linear system for each 3D point $\mathbf{X}_k$ is given in Equation 3.16.

$$A_k X_k = \begin{bmatrix} P_{k,\times} I \\ P'_{k,\times} M \end{bmatrix} X_k = 0, \quad A_k \in \mathbb{R}^{6\times 4} \tag{3.16}$$

To solve Equation 3.16 for X_k, SVD is applied.

$$A_k = U\Sigma V^{\top}. \tag{3.17}$$

The solution X_k corresponds to the eigenvector in $V^{\top}$ associated with the smallest

singular value of A_k. Finally, X_k is homogenized by dividing by its fourth coordinate. After triangulating the 3D points, the correct solution among four candidate solutions obtained from the essential matrix is selected based on the angular reprojection error. For each candidate transformation M_i, the triangulated points X_k are reprojected into the first and second cameras as given in Equation 3.18.

$$\tilde{\mathbf{P}}_k = N(IX_k), \quad \tilde{\mathbf{P}'}_k = N(M_i X_k) \tag{3.18}$$

In Equation 3.18, N is the spherical normalization matrix defined in Equation 3.11, $\tilde{P}_k$ and $\tilde{P'}_k$ are the reprojected unit vectors. The angular reprojection error for all 3D points is defined in Equation 3.19.

$$\epsilon_{\text{angular}}(M_i) = \sum_{k=1}^{N} \left(\arccos\left(\mathbf{P}_k \cdot \tilde{\mathbf{P}}_k\right) + \arccos\left(\mathbf{P'}_k \cdot \tilde{\mathbf{P}'}_k\right) \right) \tag{3.19}$$

The pose M_{best} that minimizes the total angular reprojection error is selected as shown in Equation 3.20.

$$M_{\text{best}} = \arg\min_{M_i} \epsilon_{\text{angular}}(M_i). \tag{3.20}$$

After selecting the correct solution, the Gold Standard Method (GSM) with constant weighting factors is applied [3]. standard GSM minimizes the epipolar error [46], which is given in Equation 3.21. Θ denotes the set of model parameters and ϵ is the epipolar error.

$$\epsilon(\Theta) = \frac{\left|(\mathbf{P}')^\top E_\Theta \mathbf{P}\right|}{\|\mathbf{P}'\| \, \|E_\Theta \mathbf{P}\|} \tag{3.21}$$

To use GSM with constant weighting factors, the initial pose, T, is calculated and expressed in Lie algebra representation using Equation 3.13.

$$M = \begin{bmatrix} R & \mathbf{t} \\ 1 & 0 \end{bmatrix}, \quad \xi = \begin{bmatrix} \omega_x & \omega_y & \omega_z & t_x & t_y & t_z \end{bmatrix} \tag{3.22}$$

The loss function for GSM with constant weighting factors, adapted from [38], is given in Equation 3.23. Each residual r_i measures the normalized epipolar error for the i-th matched pair:

$$loss = \sum_{i=1}^{N} w_i \cdot r_i^2, \quad r_i = \frac{\left|(\mathbf{P}_i')^\top E \mathbf{P}_i\right|}{\left\|\mathbf{P}_i'\right\| \left\|E\mathbf{P}_i\right\|} \tag{3.23}$$

After obtaining the final essential matrix from Equation 3.23, four possible camera poses are recovered as in Equation 3.13. The correct pose is then selected using DLT. This concludes the heading vector and relative orientation estimation process.

3.4.2 RANSAC

As explained in Section 2.4, RANSAC is used to estimate the essential matrix. The residual error is a numeric measure of how well a given essential matrix satisfies the epipolar constraint. The residual error for each matched vector pair is calculated from Equation 3.24.

$$residual_error_i = \frac{\left|(\mathbf{P}_i')^\top E \mathbf{P}_i\right|}{\left\|\mathbf{P}_i'\right\| \left\|E\mathbf{P}_i\right\|} \tag{3.24}$$

The goal is to find the pairs of matched unit vectors which yield the lowest residual error. RANSAC algorithm starts by picking 8 random matched vector pairs. Then, it estimates the essential matrix using standard 8-point algorithm. Then residual errors for each vector pair are calculated and vectors having higher residual error than $residual_threshold$ are eliminated. Inlier vectors that have a residual error less than $residual_threshold$ are kept in a list. This process is done until max_trials has been reached. max_trials is calculated from Equation 3.25.

$$max_trials \;\geq\; \frac{\ln\left(1 - p_{\text{success}}\right)}{\ln\left(1 - \left(1 - \text{outlier_ratio}\right)^m\right)} \tag{3.25}$$

Where:

- max_trials: Number of maximum iterations.

- $p_{success}$: Desired probability of success (0.99).
- $outlier_ratio$: Percentage of outliers (0.5).
- m: Minimum number of points needed to estimate essential matrix (8).

Once feature extraction and matching processes are complete, RANSAC is applied to remove outliers. By discarding erroneous correspondences, the accuracy of the 8-point algorithm's essential matrix estimation is significantly improved. The step-by-step breakdown of the RANSAC algorithm utilized in this work is illustrated in Figure 3.6.

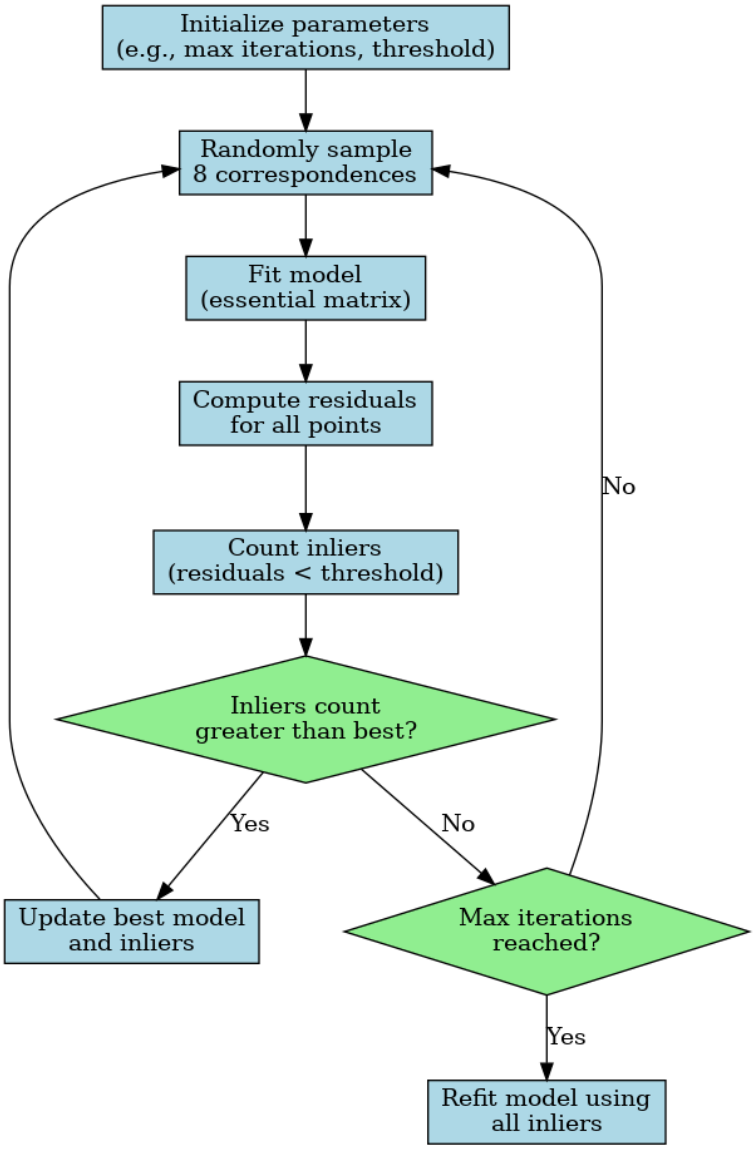


Figure 3.6: Flowchart of RANSAC algorithm.

3.5 Similarity Metric

As explained in Section 2.5, an orientation-independent similarity metric is required. Inspired by the algorithm given in [33], an image similarity measure algorithm is defined. As given in Equation 3.7, vectors in current and target images are $\mathbf{P}$ and $\mathbf{P}^{'}$ respectively. Pairwise angle difference between $\mathbf{P}$ and $\mathbf{P}^{'}$ are calculated as given in Equation 3.26.

$$\begin{aligned} d_{ij} = \mathbf{P}_i \cdot \mathbf{P}_j, \quad d_{ij}^{'} = \mathbf{P}_i^{'} \cdot \mathbf{P}_j^{'} \\ angle_difference = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \left| arccos(d_{ij}) - arccos(d_{ij}^{'}) \right|}{M \cdot N} \end{aligned} \tag{3.26}$$

The pairwise angle difference is independent of the global orientation of the vectors since it is calculated for target and current images. As a result, the similarity remains consistent even if the robot's orientation changes at the target position. When reaching the goal node, the robot's final orientation must be the same as the goal image. Therefore, another similarity measure is used while the robot is reaching the goal node. The yaw angle extracted from the rotation matrix between two images, which is obtained in heading vector and relative rotation estimation, is taken as the angle difference. In Equation 3.27, $R[i, j]$ indicates the element of R in row i and column j in zero-based indices.

$$angle_difference = arctan2(R[1, 0], R[0, 0]) \tag{3.27}$$

3.6 Topological Navigation and Mapping

As mentioned in Chapter 2, the environment is represented as a topological map consisting of nodes with 360-degree equirectangular images. A visualization of the topological map is given in Figure 3.7.

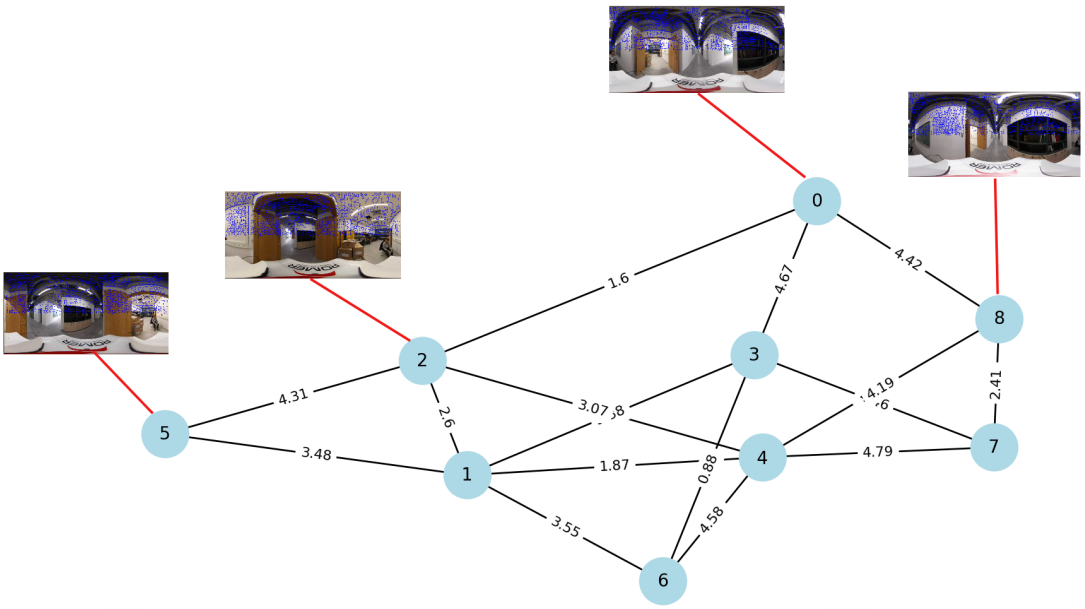


Figure 3.7: Illustration of a topological map where each node corresponds to a 360-degree equirectangular image and edges denote traversable connections.

The topological map is represented as a graph, G.

$$G = (V, E) \tag{3.28}$$

where:

- V: Set of nodes, where each node stores extracted keypoints of the corresponding equirectangular image and the relative position according to a node that is selected as origin.
- E: Set of edges which represents the connection between nodes. Edges are weighted based on a relative distance measure.

3.6.1 Node Discovery and Uniqueness

The topological map generation algorithm uses a video record that is collected manually by simply driving the robot through the environment. After the video recording procedure is completed, unique images are selected according to similarity criteria. Each frame in the video is inspected one by one in a loop. Features are extracted

from images, I, using the feature extractor algorithm F. Then, matching features are obtained using the feature matcher algorithm M.

$$F(I) = \{f_1, f_2, f_3, ..., f_n\} \quad M(I, I^{'}) = \{f_1, f_2, f_3, ..., f_k, f_1^{'}, f_2^{'}, ..., f_k^{'}\} \tag{3.29}$$

New nodes are added to V only if they are sufficiently dissimilar from already existing nodes. Therefore, Equation 3.26 is used for uniqueness check. For each node pair in V, the angular difference is calculated. The lower angular difference means that images are taken from very close positions. Therefore, a threshold of 5 degrees is defined to keep nodes with angular differences higher than the threshold.

3.6.2 Edge Construction

After unique nodes are selected from the video record, edges, E, are created by applying the following steps:

- Step 1: Connect the nodes having a higher or equal number of matches than the threshold (taken as 60), $k \geq min_match_threshold$.
- Step 2: Connect the node with the node having the highest number of matches if the node has less than 3 connected edges. This step is required for the next stage of the mapping.
- Step 3: If the graph is not fully connected, repeat step 2 by increasing the edge threshold by 1. The graph's connectivity is checked using Breath First Search (BFS)[8].

3.6.3 Pose Embedding and Refinement

In the next stage of the mapping, the previously generated topological map is refined by using relative orientations and translations between nodes obtained in Section 3.4. The first node in the graph is taken as the origin. Using BFS in a loop, cumulative rotations and translations are obtained according to the origin node.

$$R_i = R_{path(i)} \tag{3.30}$$

where:

- R_i: The cumulative rotation from i'th node to origin.
- $R_{path(i)}$: The sequential product of rotation matrices along the path.

The position of the origin node is taken as $p_0 = (0, 0, 0)$. In addition, the distance between the origin node and the first neighbor of the origin node is taken as 1 unit, meaning that the positions of the rest of the nodes are calculated relative to the distance between these two nodes. Positions of the other nodes are calculated as given in Equation 3.31.

$$\mathbf{p}_j = \mathbf{p}_i + R_i \mathbf{t}_{ij} \quad \hat{\mathbf{t}}_{ij} = R_i \mathbf{t}_{ij} \tag{3.31}$$

where:

- $\mathbf{p}_i$: Position of the current node.
- $\mathbf{p}_j$: Position of the neighbor of i'th node.
- $\mathbf{t}_{ij}$: Translation from i to j defined in i'th node's coordinate frame.
- $\hat{\mathbf{t}}_{ij}$: Translation from i to j defined in the origin node's coordinate frame.

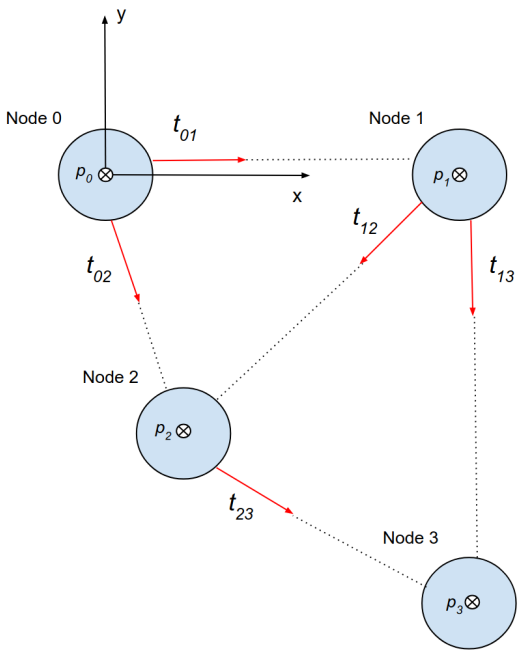


Figure 3.8: Topological mapping geometry.

To solve this for all nodes, a system of linear equations is formed as given in Equation (3.31). Since the camera height remains fixed during the recording, only the x and y components of the relative positions need to be solved.

$$A = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & ... & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & ... & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & ... & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 0 & ... & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & ... & 0 & -1 \end{bmatrix}_{2m \times 2n} \qquad b = \begin{bmatrix} -\hat{t}_{(0,1),x} \\ -\hat{t}_{(0,1),y} \\ -\hat{t}_{(1,2),x} \\ -\hat{t}_{(1,2),y} \\ \vdots \\ \vdots \\ -\hat{t}_{(n-2,n-1),x} \\ -\hat{t}_{(n-2,n-1),y} \end{bmatrix}_{2m \times 1}$$

$$x = \begin{bmatrix} -\hat{p}_{0,x} \\ -\hat{p}_{0,y} \\ -\hat{p}_{1,x} \\ -\hat{p}_{1,y} \\ \vdots \\ \vdots \\ -\hat{p}_{n-1,x} \\ -\hat{p}_{n-1,y} \end{bmatrix}_{2n \times 1} , \quad n : \textbf{Number of nodes}, \quad m : \textbf{Number of edges} \tag{3.32}$$

From the first to the last node, error accumulation occurs due to relative pose estimation error. Therefore, instead of solving Equation 3.32 by using pseudo-inverse, it is solved using least squares optimization. The loss function used during the optimization is given in Equation 3.33. As mentioned above, the translation vectors, t_{ij}, obtained from the 8-point algorithm are unit vectors. Therefore, a constant error always occurs when the given loss function is used. Even though each edge is a unit vector, least-squares optimization ensures that the relative positioning constraints are met in the best possible way given all available data. This distributes the errors evenly across the entire graph instead of accumulating linearly. Therefore, this constant error does not cause any accuracy problems during estimation.

$$\min_{x} \|Ax - b\|^2 \tag{3.33}$$

After obtaining the position of each node, the weights of the edges are taken as the Euclidean distance between the positions of corresponding nodes. In the last stage of the topological mapping, overlapping nodes are detected and removed since their positions are known. A threshold for minimum distance (taken as 0.1 units) is defined to remove nodes very close to each other. Since the positions in this algorithm are calculated in a relative metric scale, not in absolute physical units, this threshold is selected iteratively.

3.6.4 Route Planning and Online Localization

Uniform-Cost Search (UCS) [8] is adopted in the path planning stage. Unlike [5] and [6], distances between nodes are known in a relative metric scale, meaning that UCS is a better choice than BFS. While traversing between nodes in the map by planning the path with UCS, Equation 3.26 is used as the similarity metric. The robot continuously captures a 360-degree equirectangular image of its current view. At each step, it compares this captured image with the stored image of the following target node in the path using the similarity metric in Equation 3.26. This guides the robot to confirm that it is heading toward (and arriving at) the correct node. If the similarity drops under a defined threshold, the target image is changed to the next one on the generated path. Similarity metric given in Equation 3.27 is used to set the orientation of the robot same as the goal node after reaching the goal node. This ensures that the robot not only arrives at the correct location but also faces the desired direction.

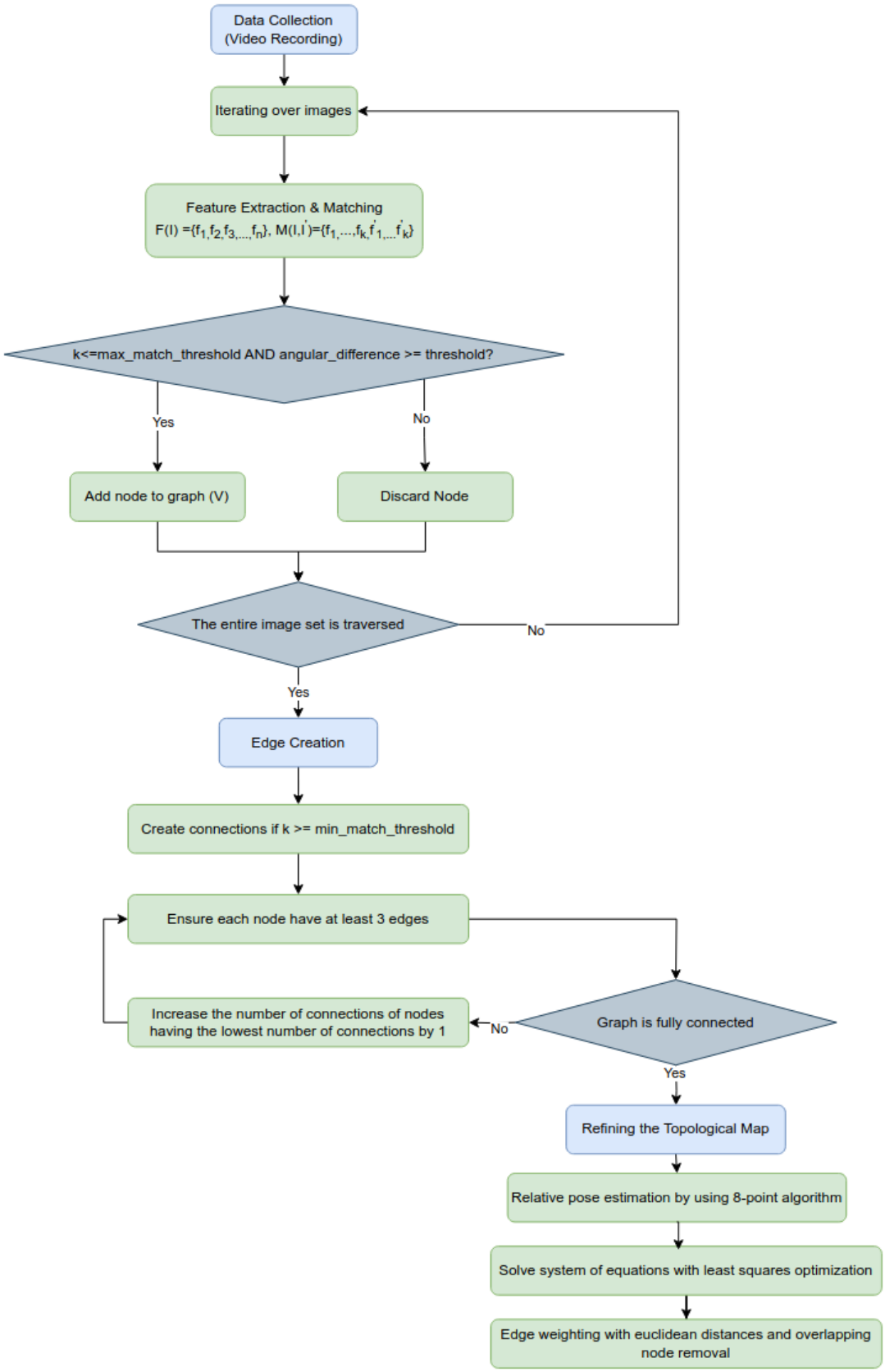


Figure 3.9: Flowchart of the topological map construction.

3.7 3D Position Estimation from Triangulation

A triangulation-based approach is utilized to estimate the 3D position of an object's scene using two 360-degree cameras mounted at the head and gripper (hand) of the robot. The estimation algorithm is based on computing the closest intersection point of two lines, each originating from a camera and directed toward the object. These lines are obtained by matching the 360-degree image of the object's scene with hand and head camera images. After obtaining matched features, the heading vectors of each camera are calculated using the 8-point algorithm, as explained in the previous sections.



Figure 3.10: Visualization of the triangulation process. The arrows represent the heading vectors from the head and gripper cameras to the object's scene.

By using two heading vectors, $\mathbf{h}_1$ and $\mathbf{h}_2$, triangulation is performed to find the 3D location of the object's scene, $\mathbf{X}_{object}$. From the joint states of the robot, the positions of Camera 1 (head) and Camera 2 (hand) are known in the robot's coordinate frames, $\mathbf{X}_1$ and $\mathbf{X}_2$, respectively. To simply the problem, $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{h}_1$ and $\mathbf{h}_2$ are expressed in robot's head frame. In other words, $\mathbf{X}_1 = [0, 0, 0]$ is taken. However, most 360-degree cameras stitch the image such that the direction of gravity always points at the negative z direction in the camera coordinate frame while keeping the yaw angle

the same. Gravity always points to the negative z direction in the head frame of the robot, meaning there is no correction needed for Camera 1. However, it is needed for Camera 2. Therefore, a rotation matrix required for expressing the $\mathbf{h_2}$ vector in the head frame is defined in Equation 3.34.

$$\begin{aligned} \mathbf{g}_{norm}^{hand} &= R_{head \to hand} \, [0, 0, -1]^T \\ R_{camera_2 \to camera_2_neg_g} \, (R_{hand \to camera_2} \, \mathbf{g}_{norm}^{hand}) &= [0, 0, -1]^T \\ R_{camera_2_neg_g \to hand} &= (R_{camera_2 \to camera_2_neg_g} \, R_{hand \to camera_2})^T \\ \mathbf{h}_2 &= R_{hand \to head} \, (R_{camera_2_neg_g \to hand} \, \mathbf{h}_2^{camera_2_neg_g}) \end{aligned} \tag{3.34}$$

where:

- $\mathbf{g}_{norm}^{hand}$: Direction vector of gravity in hand frame.
- $R_{hand \to head}$: Rotation from hand frame to head frame.
- $R_{hand \to camera_2}$: Rotation from hand frame to Camera 2 frame.
- $R_{camera_2_neg_g \to hand}$: Rotation from Camera 2 frame to Camera 2 negative g frame.
- $\mathbf{h}_2^{camera_2_neg_g}$: Heading vector of Camera 2 in Camera 2 negative g frame.

Rotation from hand frame to head frame, rotation from hand frame to Camera 2 frame, and heading vector of Camera 2 in Camera 2 negative g frame are known from joint states of the robot and heading vector of Camera 2 in Camera 2 negative g frame is calculated from heading vector estimation. Only rotation from Camera 2 frame to Camera 2 negative g frame is required to obtain $\mathbf{h}_2$. The required rotation matrix is obtained from Equations 3.35 and 3.36. The rotation order of the selected camera is Y–X.

$$\begin{aligned} \mathbf{g}_{norm}^{hand} &= [g_x, g_y, g_z]^T \\ \theta_y = arctan2(g_x, -g_z), \quad [\hat{g}_x, \hat{g}_y, \hat{g}_z]^T &= R_y(\theta_y) \; [g_x, g_y, g_z]^T \end{aligned} \tag{3.35}$$

$$\begin{aligned} \theta_x &= arctan2(\hat{g}_y, -\hat{g}_z) \\ R_{camera_2 \to camera_2_neg_g} &= R_y(\theta_y) \; R_z(\theta_z) \end{aligned} \tag{3.36}$$

From heading vectors, rays starting from cameras to the object's scene, $\mathbf{r}_1$ and $\mathbf{r}_2$, can be defined as given in Equation 3.37. t and s are the unknown distances along each heading vector. At the true position of the object's scene, these rays are expected to intersect.

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{X}_1 + t\mathbf{h}_1 \\ \mathbf{r}_2 &= \mathbf{X}_2 + s\mathbf{h}_2 \end{aligned} \tag{3.37}$$

In practice, these rays may not intersect exactly because of noise in feature matching, minor calibration offsets, or geometry. Therefore, the midpoint of the closest points between these lines is selected as the intersection point. By using the geometrical relationship between $\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{X}_1$ and $\mathbf{X}_2$, t and s are obtained from line-of-sight intersection formulas as given in Equations 3.38, 3.39 and 3.40.

$$\begin{aligned} a = \mathbf{h}_1 \cdot \mathbf{h}_1, \quad b = \mathbf{h}_1 \cdot \mathbf{h}_2, \quad c = \mathbf{h}_2 \cdot \mathbf{h}_2 \\ d = \mathbf{h}_1 \cdot (\mathbf{X}_1 - \mathbf{X}_2), \quad e = \mathbf{h}_2 \cdot (\mathbf{X}_1 - \mathbf{X}_2) \end{aligned} \tag{3.38}$$

$$\Delta = (\mathbf{h}_1 \cdot \mathbf{h}_1)(\mathbf{h_2} \cdot \mathbf{h_2}) - ((\mathbf{h}_1 \cdot \mathbf{h_2}))^2 \tag{3.39}$$

$$t = \frac{be - cd}{\Delta}, \quad s = \frac{ae - bd}{\Delta} \tag{3.40}$$

Hence, the position of the object's scene is calculated from Equation 3.41.

$$\mathbf{X}_{object} = \frac{(\mathbf{X}_1 + t\mathbf{h_1}) + (\mathbf{X}_2 + s\mathbf{h_2})}{2} \tag{3.41}$$

In the rest of the thesis, the position of the hand is defined in the robot's base frame. Therefore, the following transformation is needed. In Equation 3.42, the transformation between the head frame and base frame is known from the robot's joint states. Hence, the position of the object's scene in the base frame is calculated.

$$\mathbf{X}_{object}^{base} = R_{head \to base} \left(\mathbf{X}_{object} + T_{head \to base}\right) \tag{3.42}$$

where:

- $\mathbf{X}_{object}^{base}$: Position of the object's scene in base frame.
- $R_{head \to base}$: Rotation matrix from head to base.
- $T_{head \to base}$: Translation vector from head to base.

3.8 Implementation Details

This section details the practical software components that transform the previously described algorithms into a working robotic system. First, the obstacle avoidance strategy and motion controller are outlined, showing how an APF layer modulates the heading vector to generate safe velocity commands (Section 3.8.1). Next, the ROS-based node architecture is described, including the image publishers, feature matching, path planning, motion controller, and manipulation nodes, together with their inter-node topics and update rates (Section 3.8.2). Finally, the state machine that coordinates localization, path following, goal alignment, and object-level triangulation is presented (Section 3.8.3).

3.8.1 Obstacle Avoidance and Motion Control

As explained in Section 2.6, APF is adopted which is a well-known method in robotics that is used for obstacle avoidance tasks. APF is based on attractive and repulsive functions; the sum of these forces determines the direction of movement (Equation 3.43).

$$F_{total} = F_{attractive} + F_{repulsive} \tag{3.43}$$

In this thesis, the total force is taken as the sum of the heading vector, $\mathbf{h}$, and the repulsive vector, $\mathbf{r}$, which is calculated using point cloud data received from LiDAR.

$$v(\mathbf{r}) = \begin{cases} v_{\text{max}}, & \|\mathbf{r}\| \leq d_{\text{min}}, \\ 0, & \|\mathbf{r}\| \geq d_{\text{max}}, \quad \text{where} \quad slope = \dfrac{v_{\text{max}} - v_{\text{min}}}{d_{\text{min}} - d_{\text{max}}} \\ slope \cdot (\|\mathbf{r}\| - d_{\text{max}}) + v_{\text{min}}, & \text{otherwise} \end{cases} \tag{3.44}$$

In Equation 3.44, the magnitude of the repulsive vector, $v(\mathbf{r})$, is defined. d_{min} and d_{max} are the minimum and maximum distances at which the repulsive vector reaches its minimum-maximum value (v_{min}, v_{max}), respectively. The final vector, $\hat{h}$, determines the direction of the movement given in Equation 3.45.

$$\hat{\mathbf{h}} = \mathbf{h} + \mathbf{r} \tag{3.45}$$

The proposed algorithm uses the direction vector given in Equation 3.45 to generate a velocity command while the robot moves from one node to another. When it reaches the goal node, it switches the similarity measure from Equation 3.26 to Equation 3.27 and changes the direction vector to Equation 3.46.

$$\hat{\mathbf{h}} = [0, sign(angle_difference), 0]^T + \mathbf{r} \tag{3.46}$$

To follow the direction vector calculated from Equations 3.45 or 3.46, the following motion command calculation method is implemented. Since the motion is in 2D, z component of the direction vector is ignored.

$$\theta = \arctan 2(\hat{h}_y, \hat{h}_x) \tag{3.47}$$

$$v = \begin{cases} 0, & \text{if } |\theta| \geq \theta_{\text{threshold}} \\ k_1 \cdot \text{sign}(\theta), & \text{if } |\theta| < \theta_{\text{threshold}} \end{cases} \tag{3.48}$$

$$\omega = \begin{cases} k_2, & \text{if } |\theta| \geq \theta_{\text{threshold}} \\ k_2 \cdot \theta, & \text{if } |\theta| < \theta_{\text{threshold}} \end{cases} \tag{3.49}$$

where:

- θ: Azimuth angle of the combined heading vector, in radians.
- v: Linear velocity command.
- w: Angular velocity command.
- k_1: Linear velocity constant, taken as 0.25.
- k_2: Angular velocity constant, taken as 0.5.
- $\theta_{\text{threshold}}$: Threshold for prioritizing the angular velocity command, taken as 30 degrees.

LiDAR, located in front of the robot, has a 180-degree field of view. A short-term repulsive vector memory strategy is applied to prevent the local minimum problem of the potential field approach. When an obstacle is detected within the robot's LiDAR range, the repulsive vector is computed and stored for 5 seconds, meaning the algorithm uses the same repulsive vector for 5 seconds to get out of the local minima. Also, this repulsive vector is stored in a fixed coordinate frame, $/odom$, because the repulsive vector is received in a moving coordinate frame, $/base_link$. The required rotation between these frames is obtained from the $/tf$ topic, providing the robot's ongoing transformations for consistent positioning in a global reference.

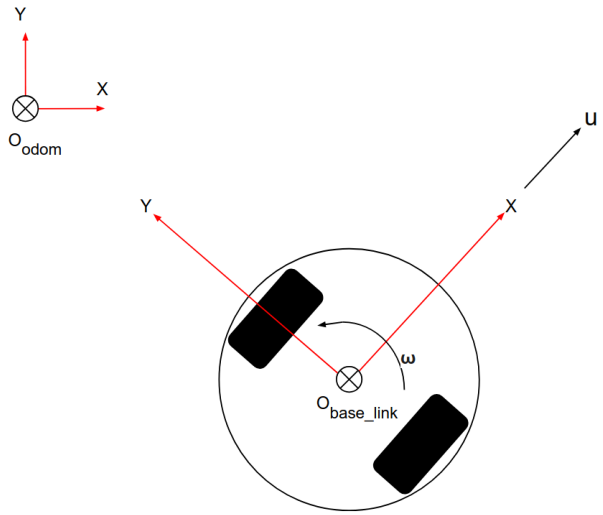


Figure 3.11: Visualization of the control commands and coordinate frames.

3.8.2 ROS Node Architecture

In the proposed framework, the robot's functionality is divided among several ROS nodes, each responsible for a specific task and communicating via topics. The roles of each node and their implementation details are explained:

- Live Image Streaming ($\mathbf{theta_image_publisher}$): These nodes are responsible for handling 360-degree image streams received from a specific IP address. They convert a JPEG image stream into ROS topics, $/theta/image_raw$ and $/theta_hand/image_raw$. The only difference between them is that one receives images from the camera located at the head of the robot while the other one receives from the camera located at the hand of the robot.

- Heading Vector Calculation ($\mathbf{heading_publisher}$): The role of this node is to generate a heading vector for navigation. It reads the index of target images from $/target_image_idx$ and loads the features of the corresponding target node. 360-degree images are received from $/theta/image_raw$ and $/theta_hand/image_raw$ topics from head and hand cameras, respectively, while calculating the heading vector and publishing it to $/heading_vector$ topic. Also, it publishes to $/matching_status$ topic. The number of matches determines the success of the feature matching process; matching status is true if there are more than or equal to 10 matches. The node also monitors the $/nav_mode$ topic to track the robot's navigation state. In addition, it runs the triangulation algorithm, explained in Section 3.7, after reaching the goal node, $/nav_mode \geq 4$, and publishes the estimated position of the object's scene to $/target_position$ topic. If the $/target_position$ is successfully calculated, $/hand_state$ is published as true.

- Path Planning ($\mathbf{planner}$): This node determines the robot's path to reach the goal node, as indicated by the $/goal_image_idx$ topic, using UCS and runs a state machine algorithm explained in Section 3.8.3. The state of the planner is published to $/nav_mode$ topic. Also, it compares the target image with the current image received from the $theta_image_raw$ topic to measure similarity, using Equations 3.26 and 3.27. If the similarity condition is met,

$/similarity$ is published as true and the $/target_image_idx$ is updated accordingly. $/reset_flag$ is used to reset the state machine.

- Motion Control (**go_2_target**): This node implements the motion control strategy described in Section 3.8.1. It receives laser scan data of surrounding objects from $/scan$ topic in the form of a point cloud. Using Equation 3.44, the magnitude of the repulsive vector is calculated. The heading vector is received from $/heading_vector$ topic and sum with repulsive vector as in Equations 3.45 and 3.46. The motion command is generated from Equations 3.48 and 3.49. In addition, it subscribes to $/matching_status$ and $similarity$ topics and moves the robot only when a true matching status and a false similarity are received. Also, it stops sending motion commands to allow for manual control via joystick when $/nav_mode == 5$.
- Manipulator Control (**move_hand_2_target**): This node runs ROS MoveIt motion planning package to plan and execute motions for the robot's hand relative to the base frame. When the $/hand_state$ topic is true, the node moves the robot's hand to the position specified by the $/target_position$ topic. Also, the $/reset_flag$ is used to reset the hand's position to the starting point.

ROS nodes operate at specific rates to ensure synchronized communication between processes. Table 3.1 gives the rates of ROS nodes. Rates of image publisher nodes were set at 15Hz because cameras send images in 15 frames per second. Rates of other nodes are set at 10Hz, which is the approximate time for completing feature extraction and matching processes.

Table 3.1: ROS Node Rates

Node Name	**Rate (Hz)**
planner	10
heading_publisher	10
go_2_target	10
move_hand_2_target	10
theta_image_publisher	15
theta_hand_image_publisher	15

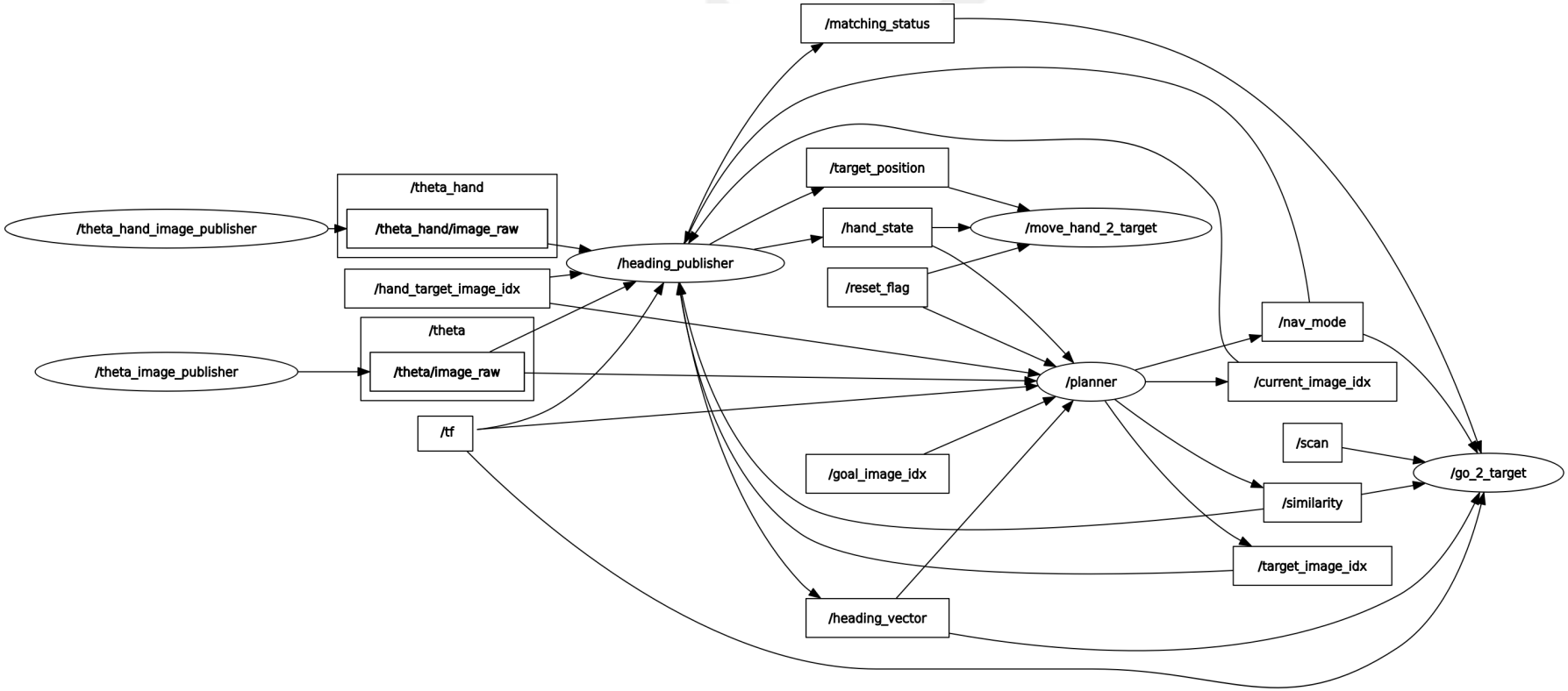


Figure 3.12: Graph of the communication network of implemented ROS nodes.

3.8.3 State Machine Design

The state machine algorithm implemented in **planner** node has 5 modes and defined as the following:

- Mode 0: Idle state, waiting for the index of the goal node to be published on $/goal_image_idx$. Mode is switched to 1 if $/goal_image_idx$ is received. If $/goal_image_idx$ is received and localization is done before, the mode is switched to 2.
- Mode 1: Localization, find the node with the highest number of matches and go to this node until $/similarty$ becomes true. The mode is switched to 2 after localization is complete.
- Mode 2: Path following, the robot follows the planned path until it reaches the desired node. $/target_image_idx$ is published to **heading_publisher**. When $/similarty$ becomes true, the index of the next node in the planned path is published. If the next node is the goal node, then the mode is switched to 3.
- Mode 3: Reaching the goal node, the similarity measure calculation is switched from Equation 3.26 to Equation 3.27 after reaching the goal node. Therefore, the robot aligns its heading to match the orientation of the goal node. After arranging the robot's orientation, the mode is switched to 4.
- Mode 4: Goal reached, the robot stops and searches a given object in the environment. It moves its hand and tracks the given scene of the object. When the $/hand_state$ topic is true, the mode is switched to 5.
- Mode 5: Object tracking, the motion command is received from the joystick instead of receiving from go_2_target node. The robot tries to position its hand to the estimated position of the object's scene while receiving movement commands from the joystick. An external flag, $/reset_flag$, is used to set the robot to idle state (mode 0).

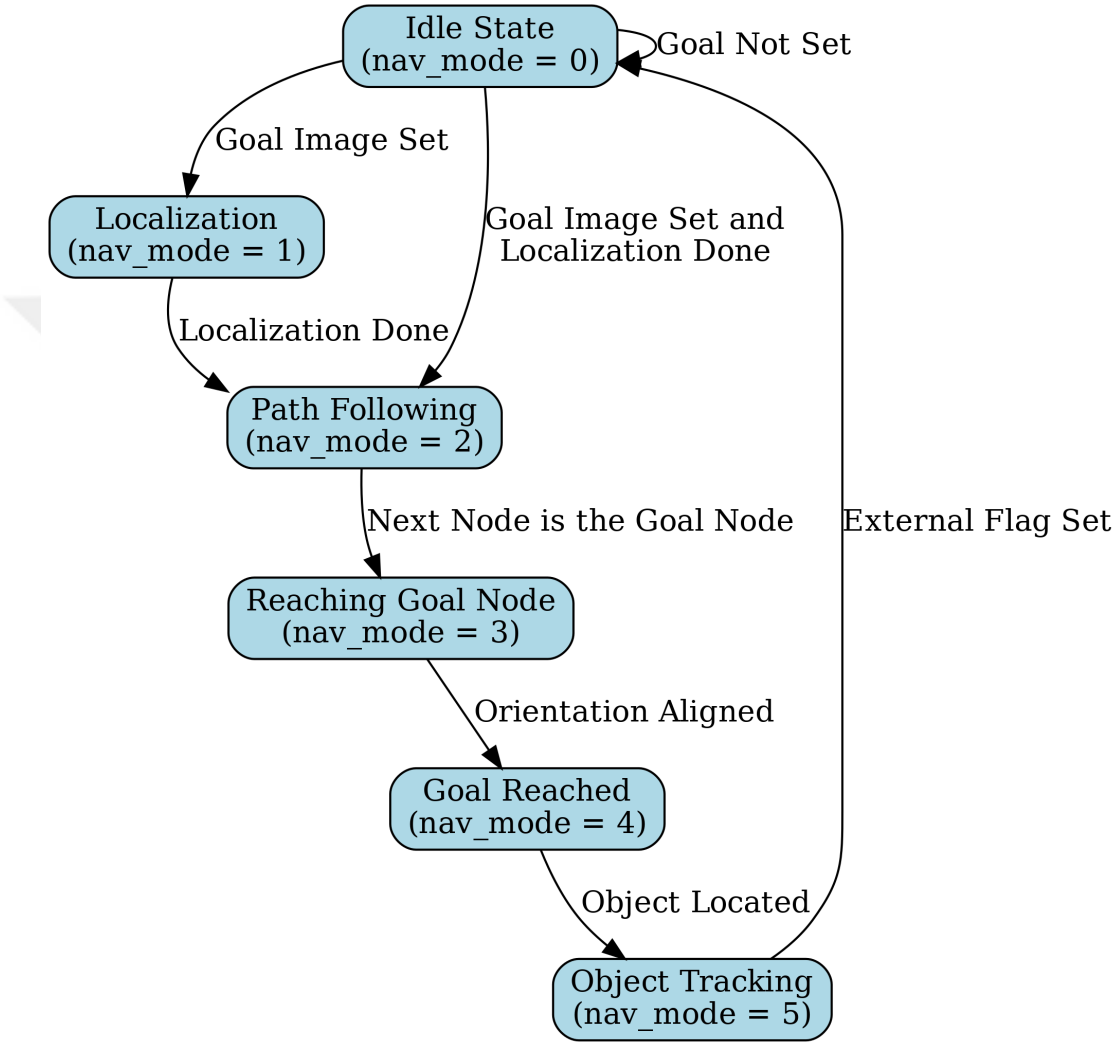


Figure 3.13: State machine of the implemented algorithm.

CHAPTER 4

EXPERIMENTAL SETUP AND PROCEDURE

An overview of the experimental platform, environment, and evaluation procedures is provided in this chapter. In Sections 4.1, 4.2, 4.3, the TIAGo mobile manipulator that served as the testbed is described, together with the on-board sensors, ROS Noetic software stack, and supporting hardware used for real-time computation and data logging. In Section 4.4, the laboratory environments are then shown to establish the conditions under which mapping, navigation, and manipulation trials were conducted. Finally, Section 4.5 enumerates the five experimental protocols that evaluate: (i) large-scale mapping, (ii) bidirectional route reliability, (iii) localization accuracy, (iv) vision-only triangulation during navigation, and (v) pose-estimation accuracy at fixed base locations.

4.1 Robot

The TIAGo robot [47], developed by PAL Robotics, is a mobile platform widely used in research. It features a two-wheel differential drive base for navigation, a 2-DOF head for flexible sensor positioning, and a 7-DOF manipulator arm for handling objects. As shown in Figure 3.11, the TIAGo robot is equipped with two cameras: one on its head to capture a wide field of view and another on its gripper to be used in triangulation. Also, a front-facing 2D LiDAR scanner is available for real-time obstacle detection. Robot's software runs on ROS (Noetic version).

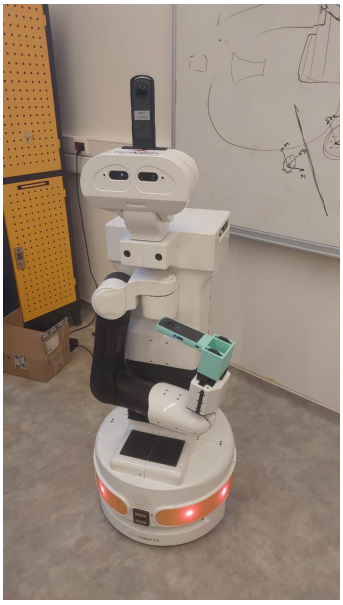


Figure 4.1: TIAGo robot

4.2 Software

The proposed algorithms were implemented using Python 3.10.15 with the following packages. The complete source code is available at the following repository: GitHub - Development of an Indoor Topological Navigation Framework for Mobile Robots Using 360-Degree Cameras.

Table 4.1: Python Packages

Package	**Version**
torch	2.0.1 (for CUDA)
numpy	1.26.4
kornia	0.7.3
scipy	1.14.1
opencv-contrib-python	4.10.0.84

ROS served as middleware. ROS Noetic version was chosen to ensure consistency with the hardware platform, matching the version already deployed on the robot.

4.3 Hardware

ROS nodes explained in Section 3.8.2 are deployed on a laptop with Nvidia RTX4070 GPU. This laptop is used as a base station; the robot's software receives movement commands from this base station. Communication between the robot and the base station is established through a wifi connection. Since the entire environment is connected to the mesh wifi network, data transmission delays or wifi disconnection problems have a very low chance of occurring. Two Ricoh Theta X cameras are mounted on the robot's head and hand. These cameras can send 360-degree equirectangular RGB images with a size of 512×1024 pixels through a wifi connection. Therefore, the base station receives camera images from a specific IP address.

4.4 Test Environment

A video record of METU ROMER laboratories was taken and used to generate the topological map. Figure 4.2 gives the sketch of the environment. Also, a controlled environment with OptiTrack setup installed [48], shown in Figure 4.3, is used.

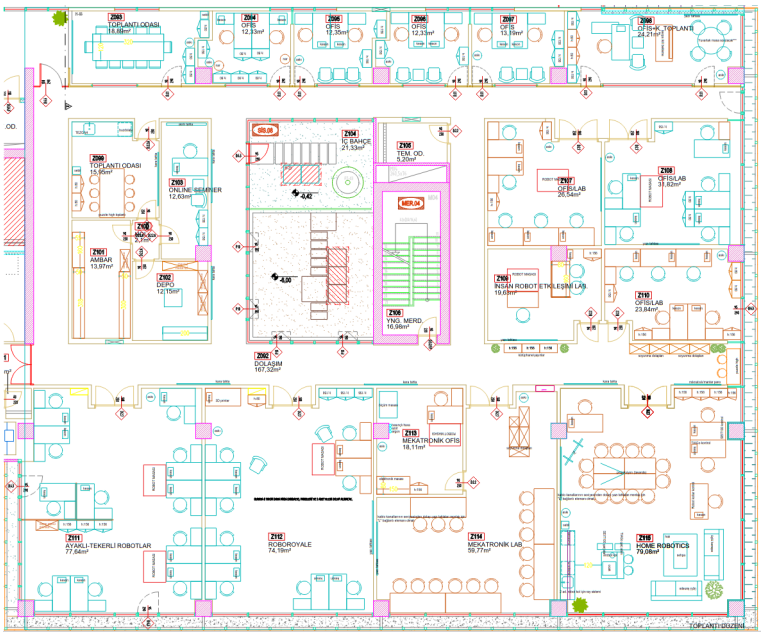


Figure 4.2: Sketch of the METU ROMER laboratories.

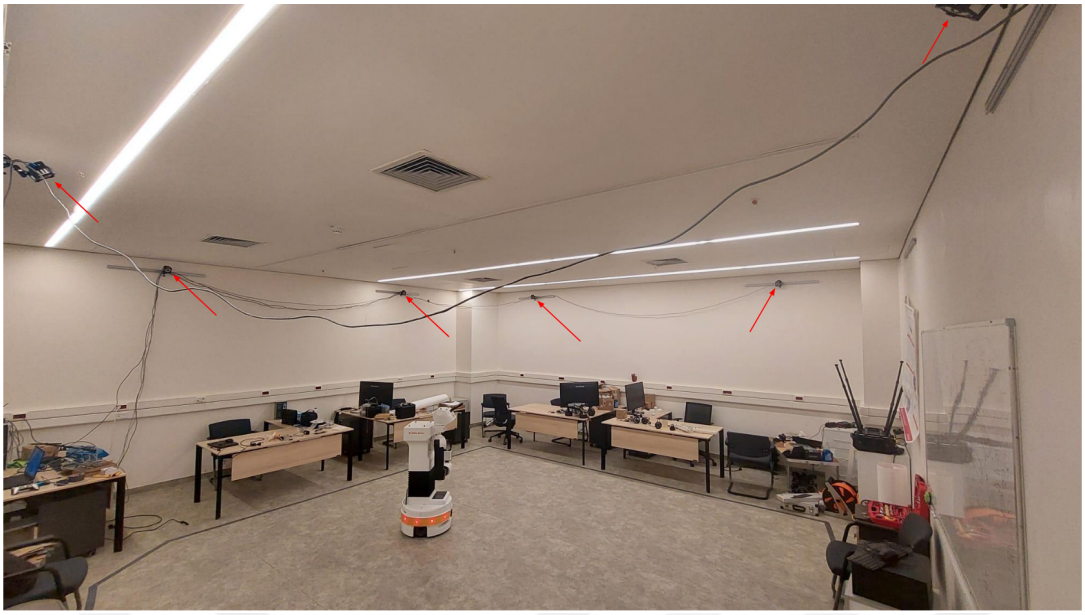


Figure 4.3: Controlled environment with OptiTrack setup installed. Red arrows show the six cameras that capture the position of the robot.

4.5 Experimental Procedure

The experimental procedure for the validation of the proposed algorithms is given in the following:

1. **Mapping Run in ROMER Labs**
    - The robot is hand-driven through the METU ROMER complex (2–3 rooms plus corridors), capturing a continuous head-camera video.
    - Offline processing built the large topological map using a video record of METU ROMER laboratories.
2. **Navigation Trials in ROMER Map**
    - Four distinct start–goal *trajectories* are randomly selected from the previously generated map.
    - Each trajectory is executed *forward* and *reverse* (8 runs total).
    - The success/failure of the bidirectional travel is logged.

3. **Navigation Localization Accuracy in Controlled Room**

    - A separate OptiTrack room is mapped as nine fully connected nodes.
    - All ordered start–goal pairs (9×8 = 72 runs) are commanded in sequence.
    - OptiTrack ground truth is used to compute the final base-pose error for each run.

4. **Triangulation Trials In ROMER Map**

    - Two random goals and two random starts for each goal are chosen, making four trials in forward and reverse.
    - Upon reaching each goal, the robot triangulated a predefined object's scene with its hand-camera.
    - The success/failure of the bidirectional vision-based triangulation is recorded.

5. **Triangulation Pose Estimation Accuracy at Fixed Points:**

    - At three fixed base locations in the ROMER labs, the robot's base is kept stationary.
    - For each base location, the manipulator is moved to eight predefined positions around a test scene.
    - At every pose, the triangulation algorithm is performed, giving $8 \times 7 = 56$ estimated poses per site (168 in total).
    - The estimated 3-D position and rotation angles are compared to the pose computed from the robot's forward kinematics.

CHAPTER 5

RESULTS AND DISCUSSION

This chapter presents the quantitative evaluation of the proposed metric-free, vision-only topological navigation and object-approach framework. Five experimental phases: mapping in the METU ROMER laboratories, bidirectional navigation trials on the large ROMER map, navigation in the controlled environment (OptiTrack), bidirectional triangulation trials on the large ROMER map, and fixed-point hand-pose triangulation were conducted as detailed in Section 4.5. Performance is assessed in three parts: the qualitative fidelity of the generated topological map, reliability and localization accuracy of the topological navigation algorithm, and reliability and pose accuracy of the triangulation algorithm.

5.1 Results

Performance evaluation of the proposed topological navigation framework is breakdown into three sections:

- Performance of the Topological Mapping Algorithm: The topological map generated by the proposed algorithm is compared against a reference sketch of the test environment to show the accuracy of the proposed approach.
- Performance of the Topological Navigation Algorithm: The proposed topological navigation algorithm is evaluated with two metrics: reliability and localization accuracy.
- Performance of the Triangulation Algorithm: The proposed triangulation algorithm is evaluated with two metrics: reliability and pose estimation accuracy.

5.1.1 Performance of the Topological Mapping Algorithm

As explained in Section 3.6, topological map generation has three steps. In the first step, unique nodes are selected. In the second step, connections between nodes (edges) are generated. In the last step, relative positions according to a selected node are defined, and overlapping nodes are deleted using the Euclidean distance between nodes. The following map, Figure 5.1, is obtained after completing all these steps. To compare the accuracy of the map, Figure 4.2 and Figure 5.1 are plotted together in Figure 5.2. It is observed that the proposed topological mapping algorithm can capture the node position on a relative scale without requiring metric information. Therefore, using relative Euclidean distances between nodes as weights is a reasonable choice. Due to the proposed map generation approach, pose estimation errors, which occur during feature matching and essential matrix estimation, cumulatively increase starting from the origin node to the other nodes. Therefore, differences between the sketch of the environment and the generated topological map are expected. These differences can be observed in Figure 5.2.

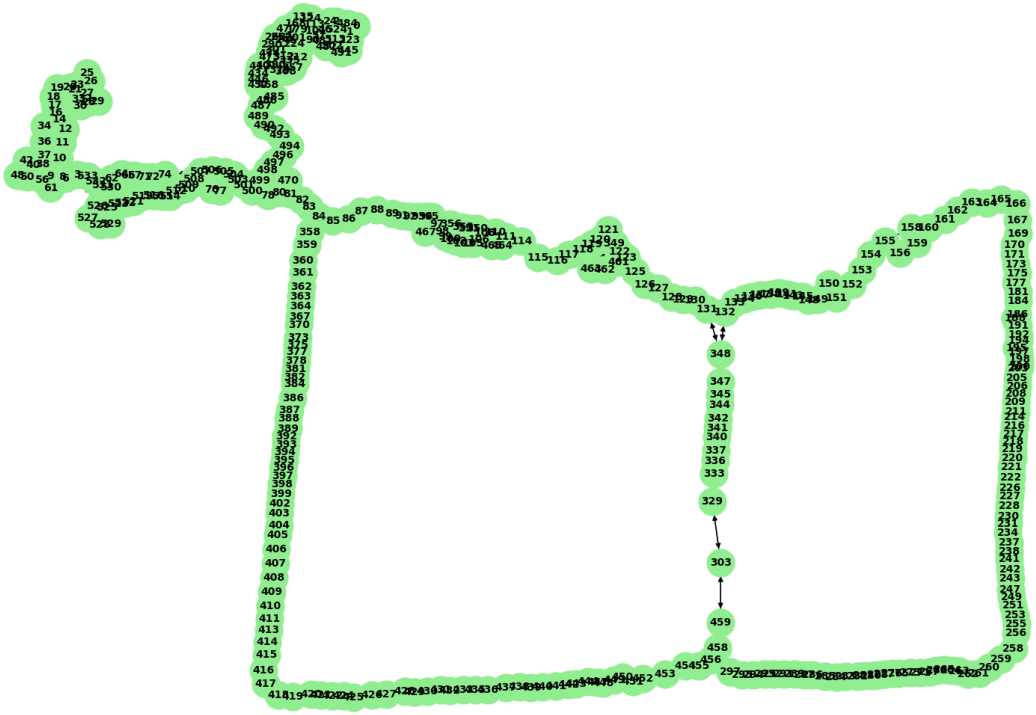


Figure 5.1: Topological map generated from video record.

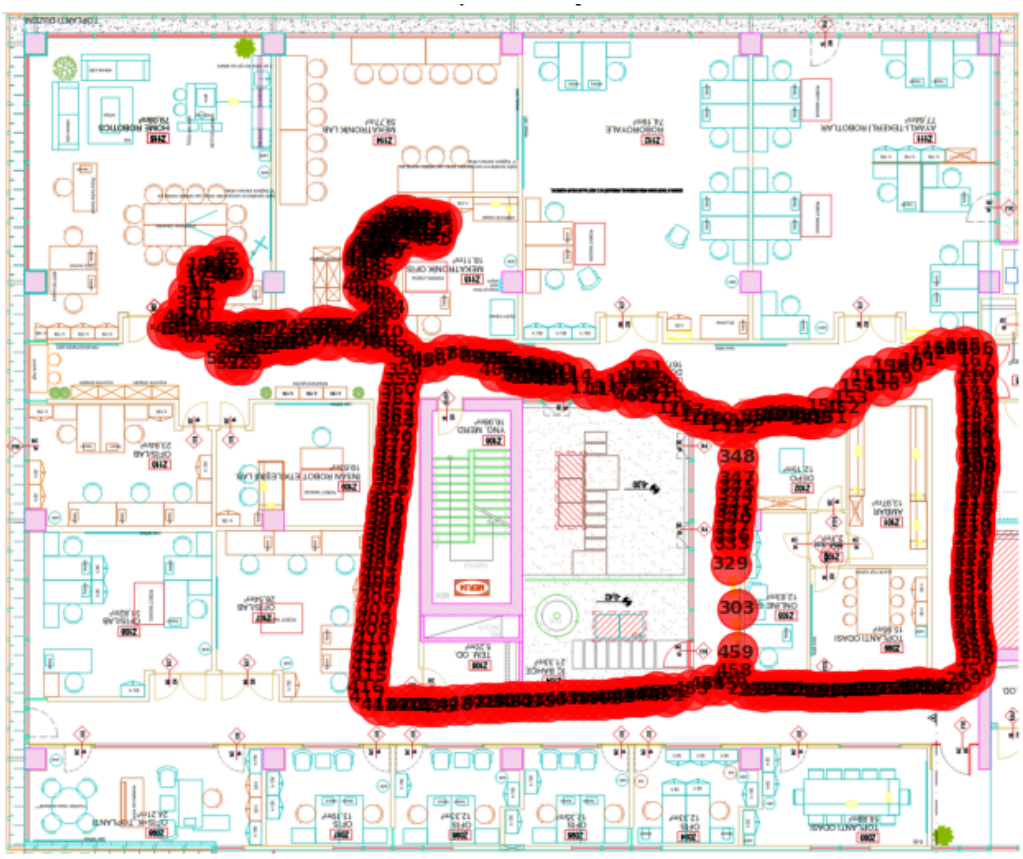


Figure 5.2: Topological map plotted on the sketch of the environment.

5.1.2 Performance of the Navigation Algorithm

An experimental evaluation of the topological navigation algorithm was conducted in two settings. First, its robustness and reliability were tested on the METU ROMER laboratory topological map (Section 5.1.1) by executing four randomly chosen start–goal trajectories both forward and reverse, with success rates recorded. Second, localization precision was quantified in a room installed with an OptiTrack motion-capture system. A fully connected map with nine nodes was built, $8 \times 9 = 72$ start–goal trials were executed, and OptiTrack ground truth was used to compute final position errors.

5.1.2.1 Reliability Analysis

Four scenarios were conducted to test the reliability and consistency of the proposed topological navigation algorithm. In each scenario, the robot starts with localization and then navigates to the given goal nodes. After reaching the goal node, the scenario is repeated in the reverse order to show the reliability of the proposed algorithm. There is no ground truth for comparing results. Therefore, success criteria are defined as reaching the goal node (success rate). In these scenarios, the robot does not arrange the orientation same as the goal node. In other words, the similarity measure given in Equation 3.26 is utilized with a 4-degree threshold. The paths generated during navigation and the view of the scene at the start and the end of the navigation are shown for each scenario. All scenarios were conducted in METU ROMER laboratories, Figure 4.2, and a previously generated topological map, Figure 5.1 was utilized. In all four scenarios, the robot has successfully navigated to and returned from each prescribed goal node without failure. Forward runs were commanded to nodes 457, 130, 17, and 416 (shown in Figs. 5.4, A.2, A.9, and A.16, respectively), and reverse runs back to node 0 were commanded (shown in Figs. 5.7, A.5, A.12, and A.19). In every case, the target was reached successfully in both directions. The figures for the first scenario appear here; the corresponding illustrations for scenarios 2–4 are provided in Appendix A.

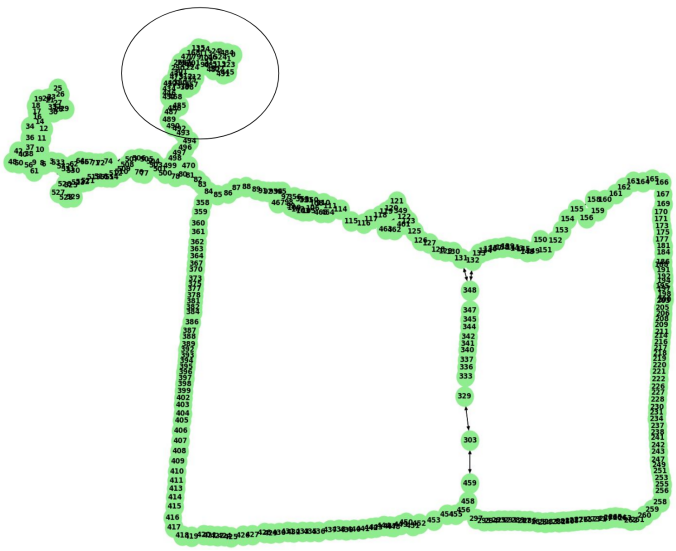


Figure 5.3: Location of scenario 1 of topological navigation algorithm test in full map.

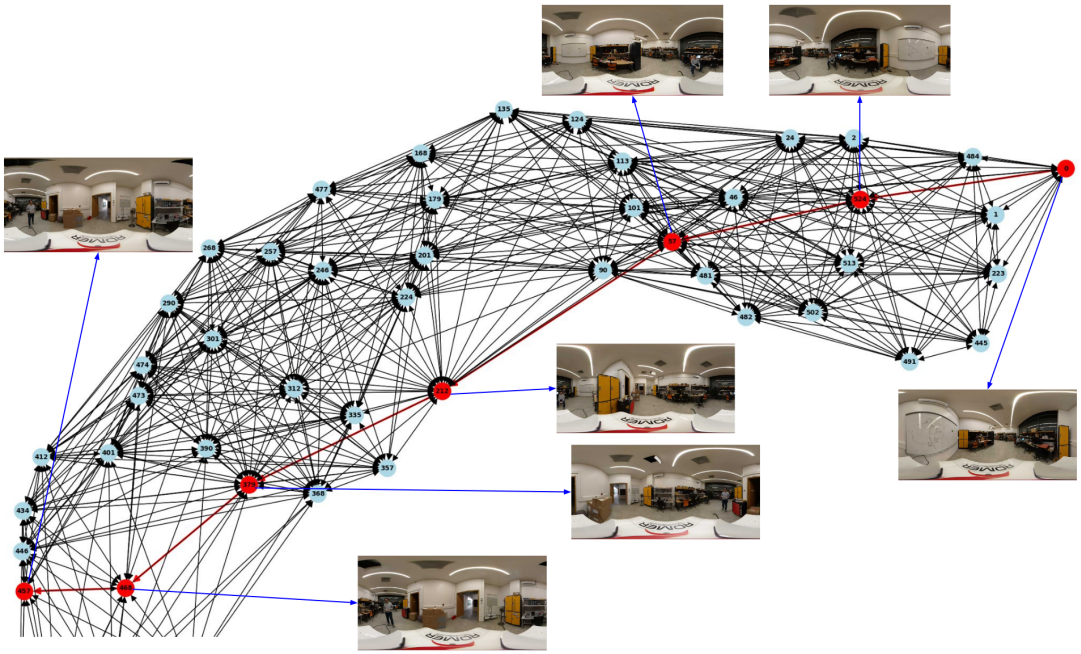


Figure 5.4: Path generated in scenario 1 of topological navigation algorithm test.



Figure 5.5: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of topological navigation algorithm test.

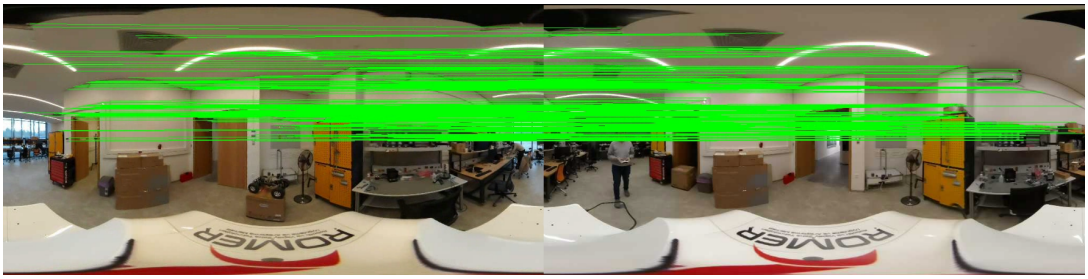


Figure 5.6: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of topological navigation algorithm test.

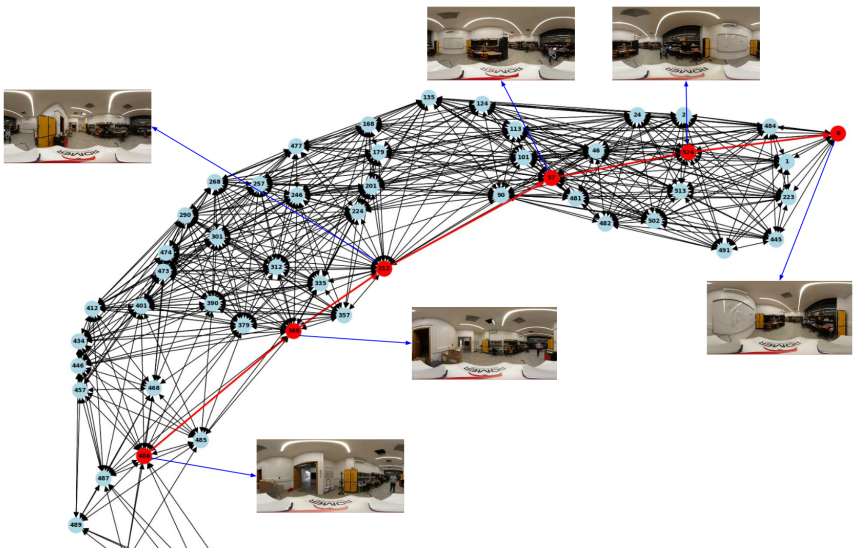


Figure 5.7: Path generated in scenario 1 (repeated in reverse order).

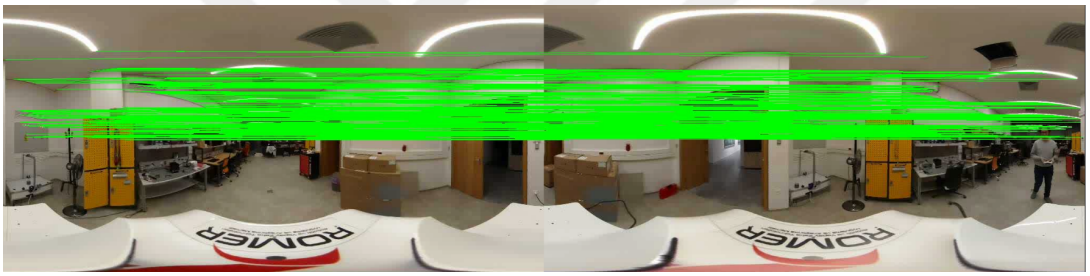


Figure 5.8: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of topological navigation algorithm test (repeated in reverse order).

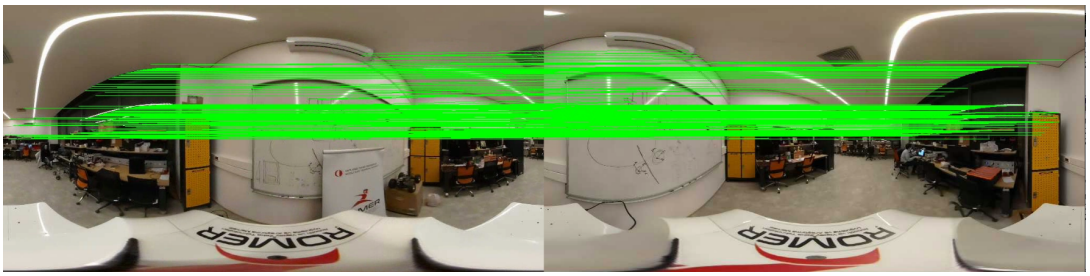


Figure 5.9: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of topological navigation algorithm test (repeated in reverse order).

5.1.2.2 Localization Error Analysis

The experiments were conducted in a controlled environment with OptiTrack setup installed, as shown in Figure 4.3, to assess the accuracy of the proposed topological navigation algorithm. The experiments are structured around the following steps:

- Map Generation: Nine images are collected from nine different points of the environment in random orientations, given in Figure 5.14. A fully connected map is generated from these images.
- Defining the Environment via Ground Truth: A global coordinate system is defined in Optitrack, and ground-truth positions for each key node (image) are in the environment defined according to this coordinate system. These coordinates serve as the benchmark for evaluating the system's performance.
- Conducting Trials: The robot moves from a specified start node to a goal node in each navigation test. Eight nodes are connected to each node, making $9 \times 8 = 72$ trials in total. The robot's final position is recorded in the same global frame as the ground-truth coordinates in 2D during these trials.

In Table 5.1, success rate and position error statistics (mean error, standard deviation, median error) are given for each goal node. For every node, the table reports the success rate (expressed in percentage), the mean position error (in meters), the standard deviation of the error (in meters), and the median position error (in meters). The final row, labeled "All," provides the aggregated statistics across all goal nodes. The distribution of position errors for each goal node is shown in Figure 5.10. The x-axis lists the goal nodes, and the y-axis displays the error values. Each box covers the interquartile range with the median marked inside, while the whiskers extend to the most extreme values that are not outliers, which are shown as individual points. In Figure 5.11, a histogram is given where the x-axis represents the absolute position error (the Euclidean distance between the robot's final position and the goal's ground-truth position) and the y-axis shows the frequency of occurrences for each error range. It provides an overview of how the errors are distributed across all successful runs. In Figure 5.12, a bar plot that shows the percentage of successful navigation runs (those with valid final positions) for each goal node is given. The x-axis indicates the goal

node, and the y-axis represents the success rate percentage. Each bar corresponds to a node, showing the proportion of successful runs. In Figure 5.13, the ground-truth positions of all nodes (shown as black squares with labels) with the final reported positions from successful navigation trials (displayed as colored markers, with colors corresponding to different goal nodes) are given. In Appendix B, separate figures are given for each goal. In these figures, the ground-truth positions of all nodes are marked as black squares, and arrows visually indicate the trajectory taken.

Table 5.1: Position Error Statistics by Goal Node

Goal Node	Success Rate (%)	Mean Error (m)	Std Error (m)	Median Error (m)
1	75%	0.322	0.173	0.310
2	100%	0.347	0.213	0.319
3	100%	0.174	0.107	0.143
4	100%	0.359	0.078	0.366
5	100%	0.428	0.228	0.516
6	100%	0.179	0.135	0.125
7	50%	0.235	0.113	0.239
8	100%	0.504	0.217	0.487
9	75%	0.272	0.138	0.237
All	88.9%	0.319	0.191	0.318

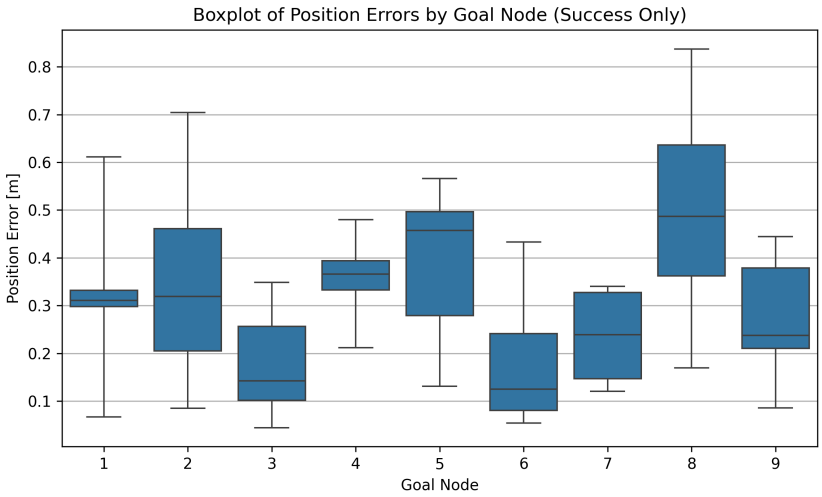


Figure 5.10: Boxplot errors by goal node for successful trials.

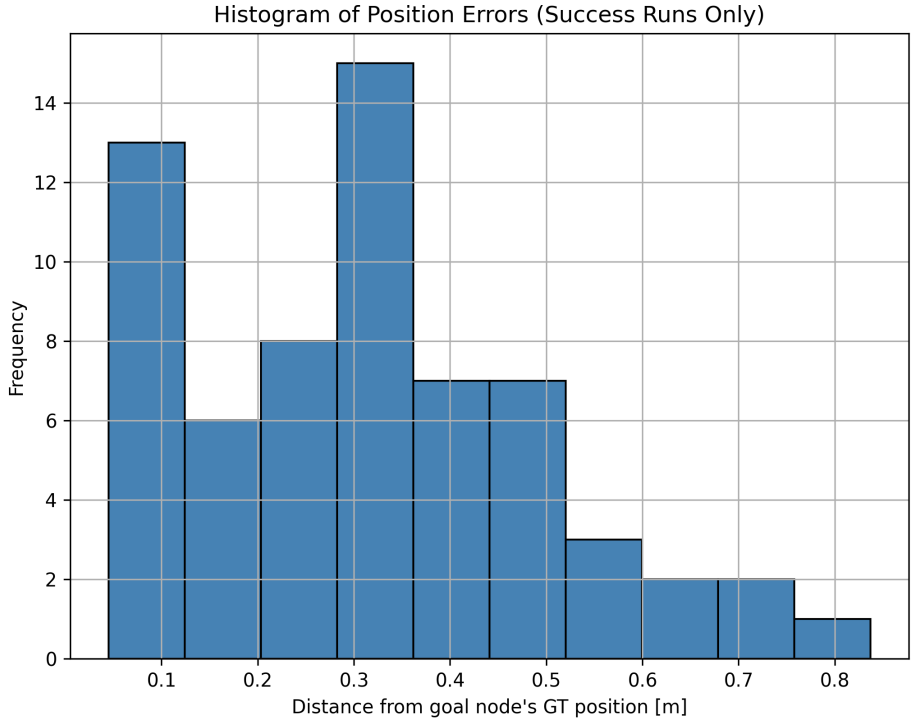


Figure 5.11: Histogram of the position errors for successful trials.

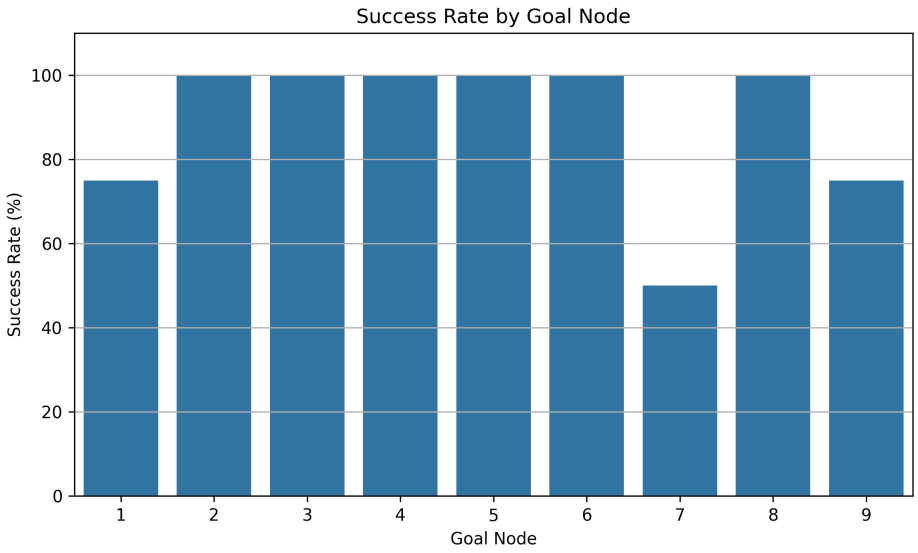


Figure 5.12: Success rate by goal node.

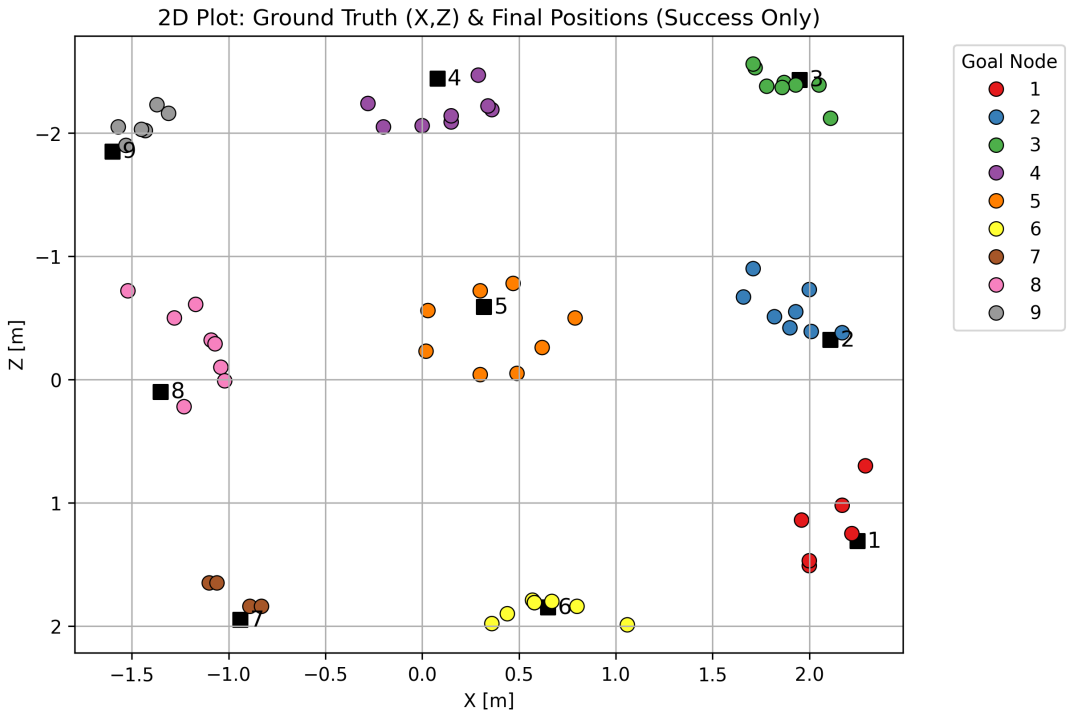


Figure 5.13: The ground-truth positions of all nodes with the final positions of successful navigation trials.

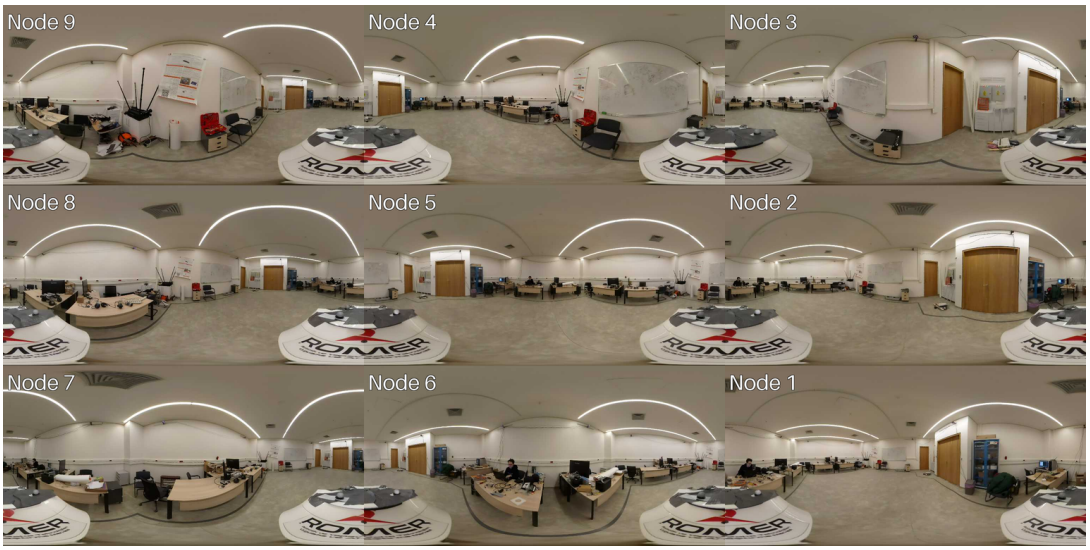


Figure 5.14: Collected images from the controlled environment.

5.1.3 Performance of the Triangulation Algorithm

Two complementary series of tests were conducted to evaluate the performance of the triangulation algorithm. First, the robot was navigated along two randomly selected routes in the large ROMER laboratory map, and upon reaching each goal node (four runs total, forward and reverse), the manipulator extended toward a predefined object scene to perform vision-only triangulation, with each trial's success or failure recorded. Second, with the mobile base held stationary at three different locations, the arm was moved through eight predefined poses around the same test scene (56 reaches per location, 168 trials in total). For each pose, the 3D position and orientation of the triangulated estimate were logged against the robot's internal forward-kinematic position.

5.1.3.1 Reliability Analysis

Two scenarios were conducted to assess the reliability of the triangulation algorithm. In each scenario, the robot performs localization and then navigates toward the specified goal nodes. During navigation, Equation 3.27 is used to compute the similarity measure (15-degree threshold is taken) while reaching the goal node so that the robot adjusts its orientation to match the goal node's orientation. While moving between nodes other than the goal node, Equation 3.26 is used as the similarity measure (4-degree threshold is taken). After reaching the goal node and adjusting the orientation, the robot uses the triangulation algorithm to extend its hand toward the object's scene. The process is repeated starting from the opposite direction, demonstrating the system's reliability through consistent performance across different starting conditions. The generated paths during navigation, the view of the scene at the start and the end of the navigation, and the view of the hand camera after the robot moves its hand to the estimated point are shown for each scenario. Figures for the first scenario are given below. Figures for the second scenario are given in Appendix C

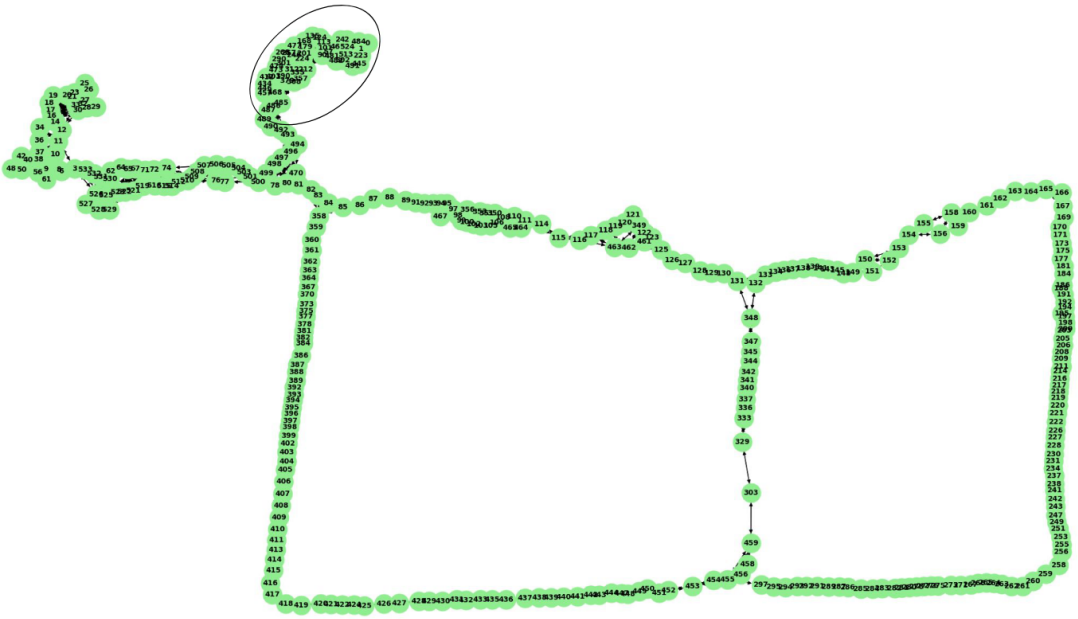


Figure 5.15: Location of scenario 1 of triangulation algorithm test in full map.

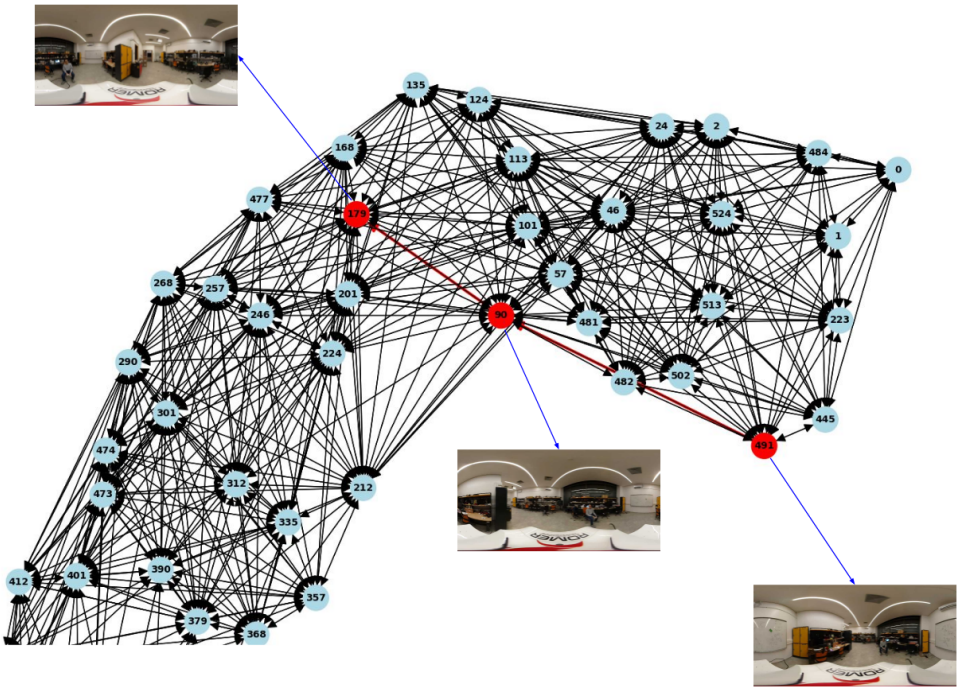


Figure 5.16: Path generated in scenario 1 of triangulation algorithm test.

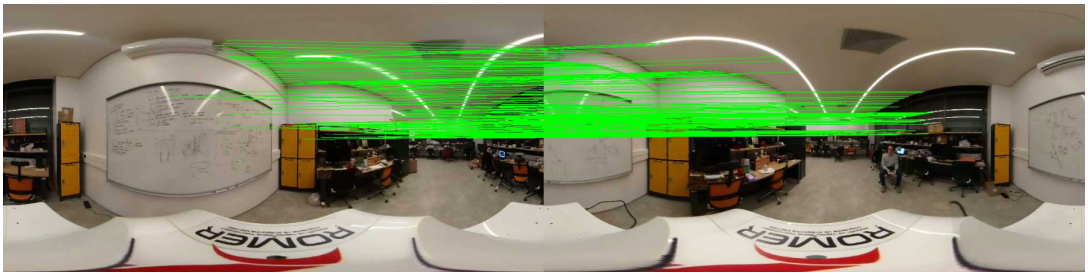


Figure 5.17: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 of triangulation algorithm test.



Figure 5.18: View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 1.

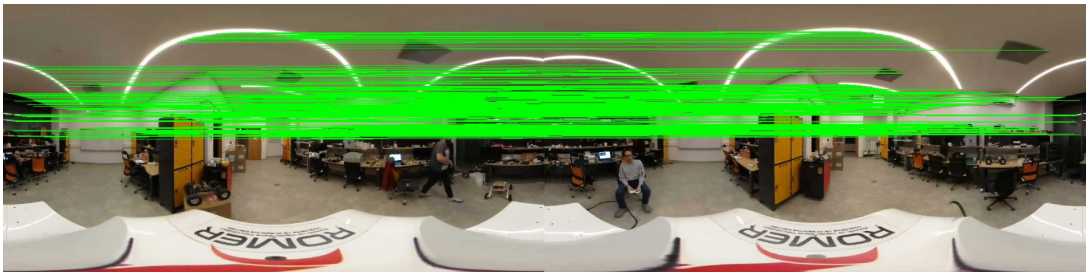


Figure 5.19: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 of triangulation algorithm test.

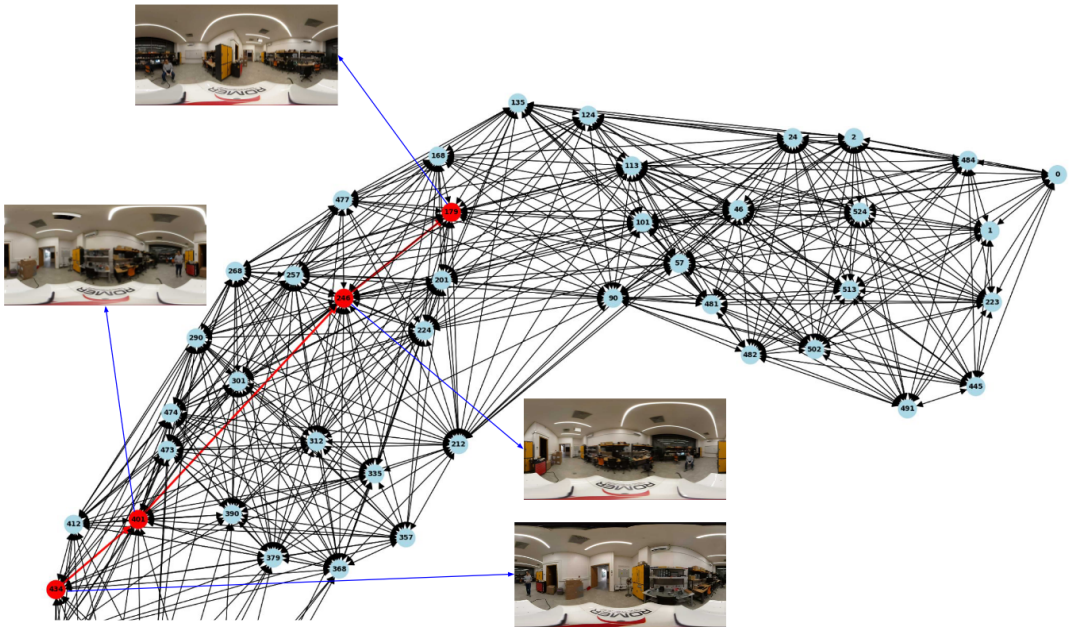


Figure 5.20: Path generated in scenario 1 (opposite direction) of triangulation algorithm test .

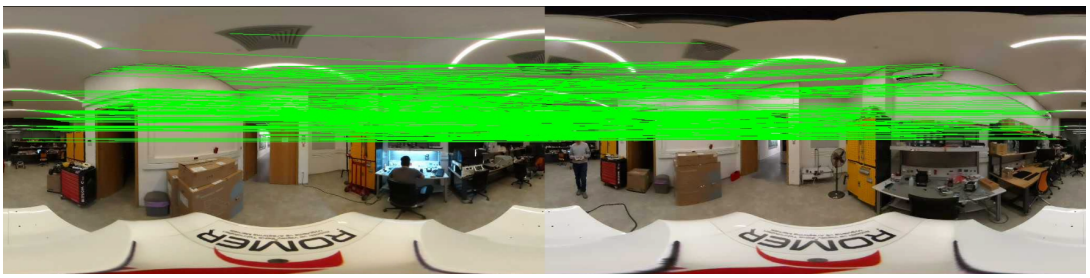


Figure 5.21: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 1 (opposite direction) of triangulation algorithm test.

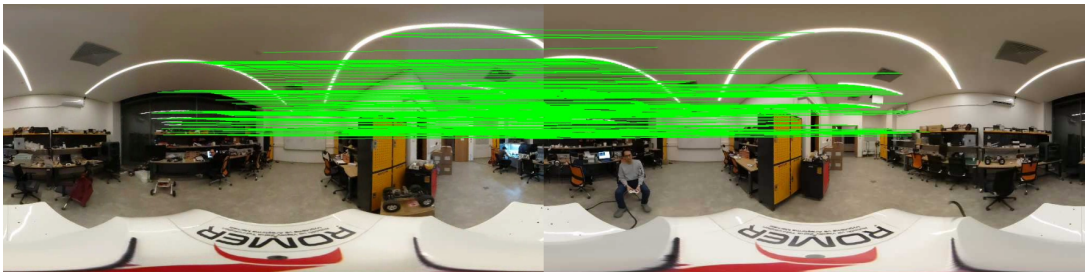


Figure 5.22: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 1 (opposite direction) of triangulation algorithm test.



Figure 5.23: View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 1 (opposite direction).

5.1.3.2 Pose Estimation Error Analysis

In this section, the goal is to evaluate the performance of the triangulation algorithm in terms of pose estimation error. Three experiments were conducted in different locations of METU ROMER laboratories, and the procedure of these experiments are organized as follows:

- Data Preparation: Ground truth data for eight distinct points in (x, y, z) is defined according to the robot's base frame. Images are collected from these eight points using a hand camera without changing the hand's orientation.
- Error Computation: By using the proposed triangulation algorithm, these points are estimated by moving the camera to the predetermined eight points. For instance, the camera is first positioned at point 1, from which the position of point 2 is estimated; then, the camera is moved to point 3 to re-estimate point 2, and this procedure continues through all eight points, making $8 \times 7 = 56$ trials. For each trial, the algorithm computes the Euclidean position error by comparing the predicted positions to the ground truth positions. Similarly, the overall orientation error is calculated using the Euclidean norm of the orientation error components. The rotation matrix between the hand camera and the image collected from the respective point is estimated by the 8-point algorithm, and the Euclidian norm of Euler angles is taken as the orientation error since the camera's orientation is always kept the same.

The 8-point algorithm generally performs best in static environments. However, when using a 360-degree camera, if the robot's manipulator (a dynamic element) appears in the images, it may be mistakenly matched across views, leading to erroneous estimation of the essential matrix. Therefore, only the features from 40% of the 360-degree images (specifically, the rightmost 40 % of pixels for Experiments 1-2 and the leftmost 40 % for Experiment 3) are used during the feature matching process.

The first of these experiments took place in the ROMER Mechatronics Laboratory, which provided a feature-rich environment for testing. Figure 5.24 shows the images used during this experiment. Boxplots are given in Figures 5.25 and 5.26 where the

x-axis represents the point identifiers and the y-axis represents errors (in meters for position error or in degrees for orientation error). Each box represents the interquartile range of errors for that point, with the line inside marking the median. Whiskers extend to the most extreme data points not considered outliers. Two histograms are shown in Figures 5.27 and 5.28 where the x-axis represents errors (in meters for position error or in degrees for orientation error) and the y-axis indicates how many test entries fall into each error bin. Each bar's height reflects the frequency of entries with errors within a specific range. A 3D plot is given in Figure 5.29 to show the distribution of the estimated points (blue circles) over ground truth points (black boxes). Table 5.2 presents the computed position error statistics for each point and overall, including the mean, standard deviation, median, and maximum error in meters. Table 5.3 provides the corresponding orientation error statistics in degrees, with the same descriptive metrics calculated per point and for the overall experiment.

The second experiment was conducted in the corridors of the ROMER laboratories, where the environment has less distinctive visual features than Experiment 1. Figure D.1 shows the images used during this experiment. As in the first experiment, the results are summarized with boxplots, histograms, and a 3D scatter plot, as shown in Figures D.2 to D.6. Also, position and orientation error statistics are given in Tables D.1 and D.2, respectively.

The third experiment was carried out in the ROMER Mechatronics Laboratory, near a board with a lot of writing, resulting in poor feature matches. This case is selected intentionally to show how unsuccessful matching affects the results. Images from this experiment are shown in Figure D.7. As with the previous experiments, the results are summarized in Figures D.8 to D.12 with boxplots, histograms, and a 3D plot. Also, position and orientation error statistics are given in Tables D.3 and D.4, respectively.



Figure 5.24: Images taken from the predetermined eight points of Experiment 1.

Table 5.2: Position Error Statistics of Experiment 1

Point	Mean (m)	Std (m)	Median (m)	Max (m)
1	0.066	0.022	0.069	0.099
2	0.027	0.018	0.019	0.052
3	0.014	0.006	0.015	0.024
4	0.047	0.016	0.047	0.078
5	0.062	0.012	0.064	0.082
6	0.019	0.011	0.020	0.034
7	0.035	0.016	0.027	0.061
8	0.061	0.022	0.058	0.100
All	0.041	0.025	0.039	0.100

Table 5.3: Orientation Error Statistics of Experiment 1

Point	Mean (deg)	Std (deg)	Median (deg)	Max (deg)
1	2.162	1.044	2.167	3.642
2	1.883	1.504	1.284	3.971
3	2.231	1.228	2.010	4.146
4	1.546	0.830	1.409	2.883
5	2.202	1.028	2.116	3.638
6	3.268	1.462	3.411	5.049
7	2.657	1.506	2.496	4.578
8	1.107	0.804	1.441	2.118
All	2.132	1.288	1.982	5.049

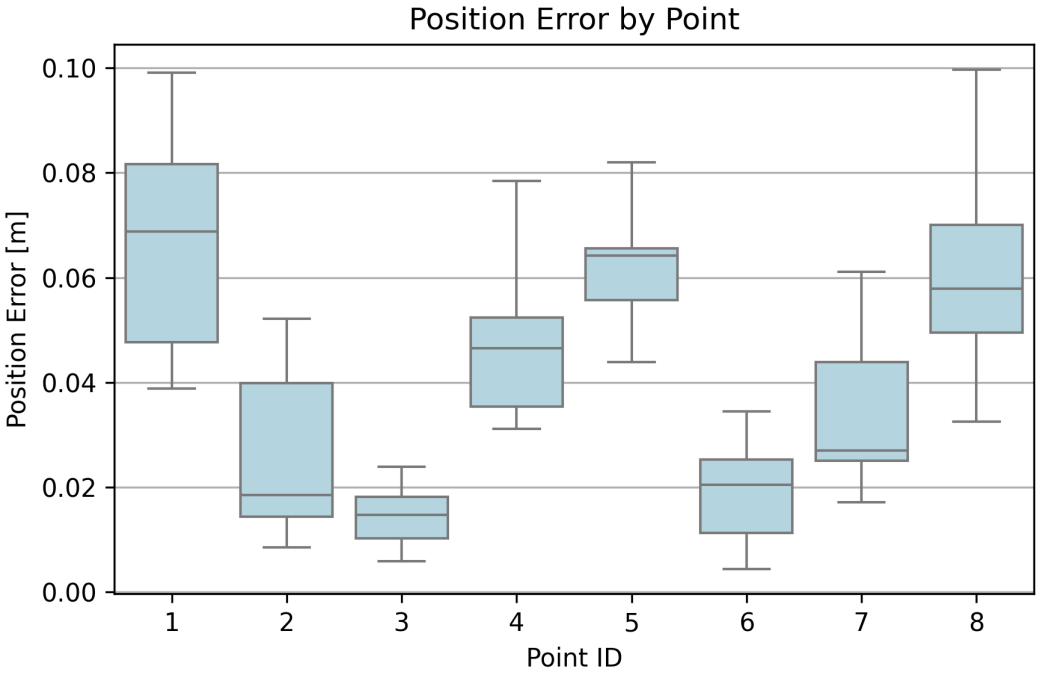


Figure 5.25: Position error boxplot of Experiment 1.

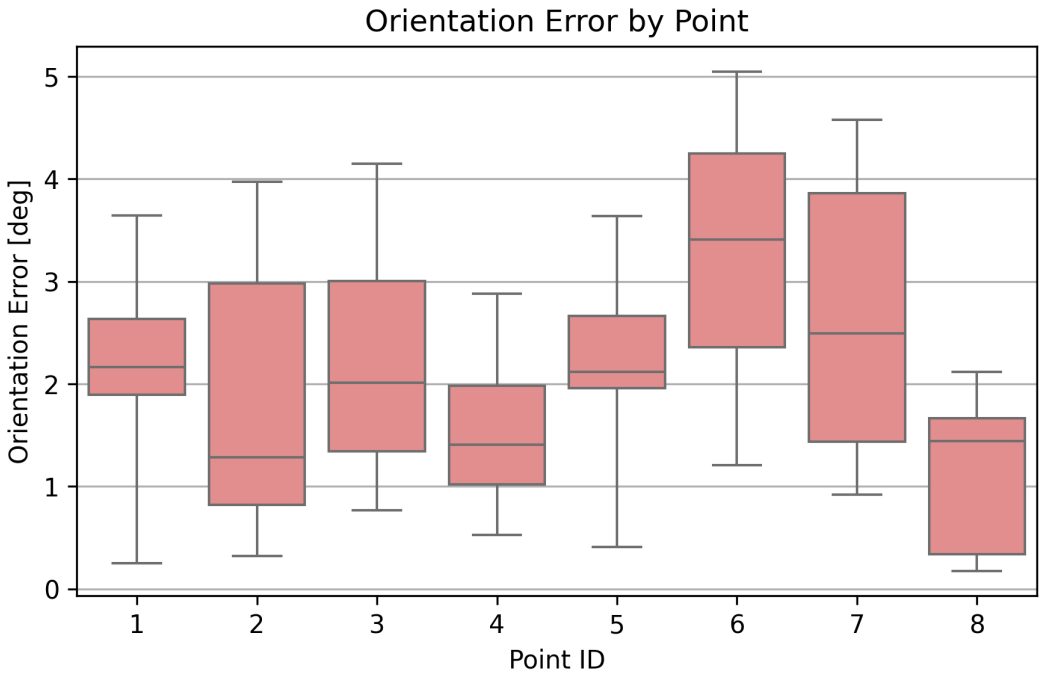


Figure 5.26: Orientation error boxplot of Experiment 1.

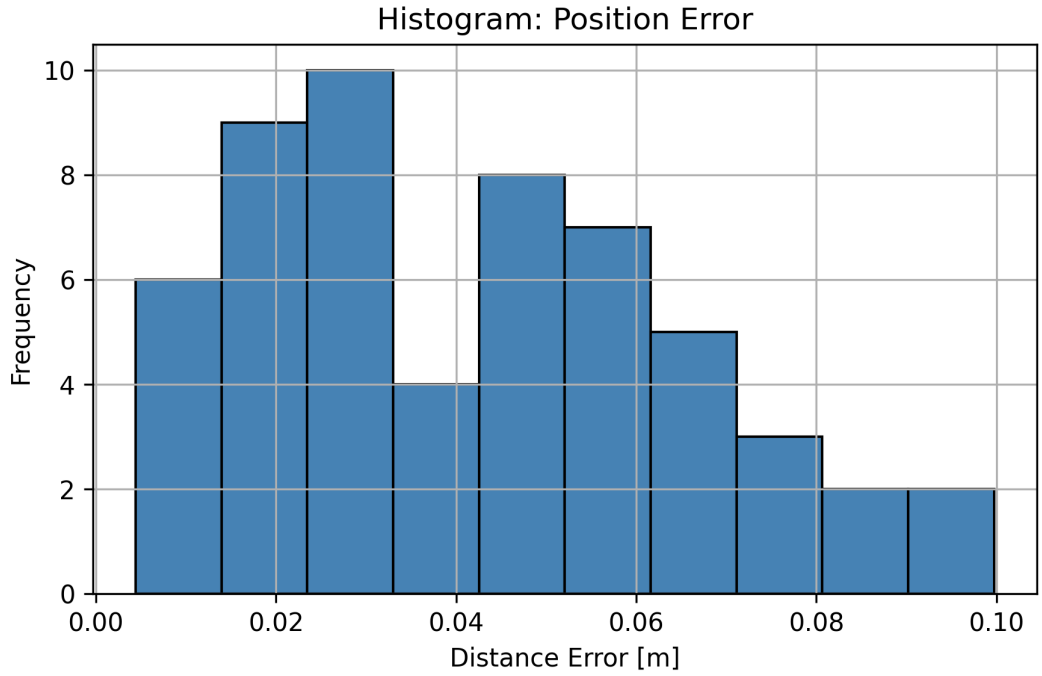


Figure 5.27: Position error histogram of Experiment 1.

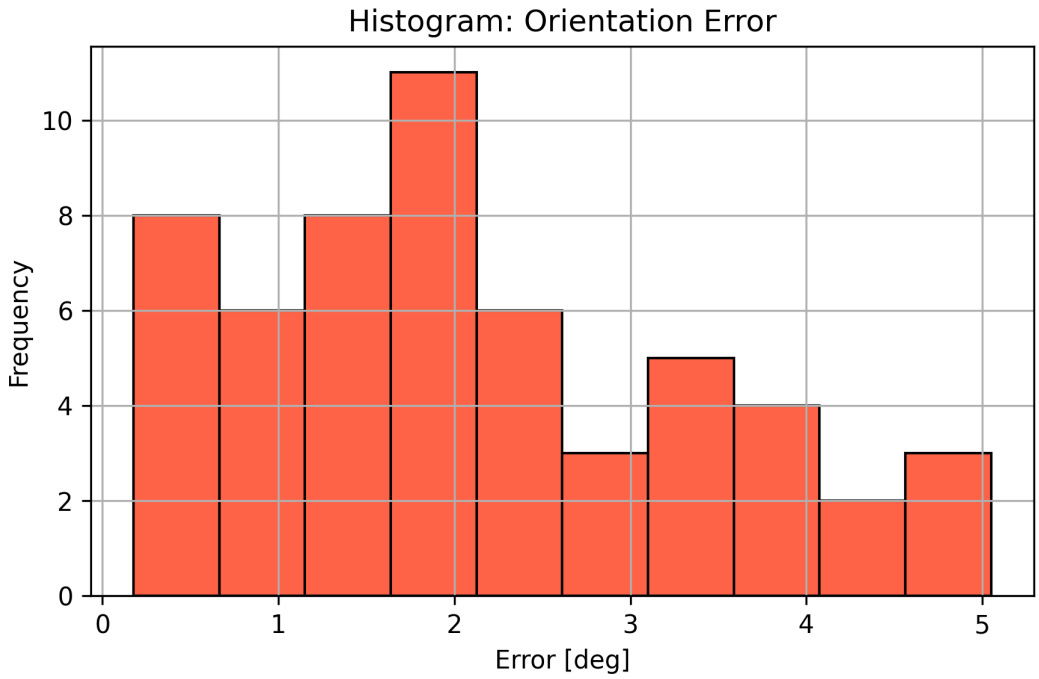


Figure 5.28: Orientation error histogram of Experiment 1.

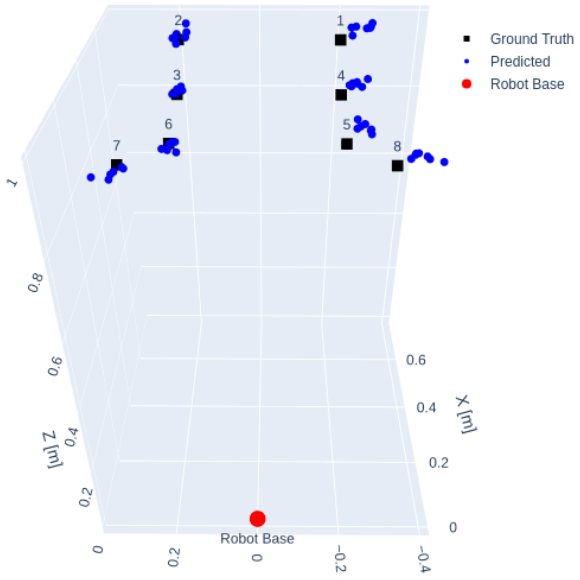


Figure 5.29: Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 1. All positions are given according to robot's base frame (red circle).

5.2 Discussion

Based on the results presented in Section 5.1.1, the proposed topological mapping algorithm successfully generates a map by selecting unique nodes, establishing connections between them, and eliminating overlapping nodes. Figure 5.2 shows that the algorithm captures the overall geometric layout of the environment without relying on metric information. This shows that using relative distances as weights for node connections is a reasonable approach. However, the comparison also reveals inconsistencies between the sketch and the generated map. These differences become clear with increasing distance from the origin, indicating that minor pose estimation errors, introduced during feature matching and essential matrix estimation, accumulate over time. This cumulative error is a limitation of the proposed approach, particularly in large environments with complex or ambiguous visual features. Despite these deviations, the algorithm constructs a topological representation that reflects the environment's structure.

In Section 5.1.2.1, four distinct scenarios were conducted to evaluate the reliability and consistency of the proposed topological navigation algorithm. In each scenario, the robot first localizes itself and then navigates toward a designated goal node; after reaching the goal, the navigation is repeated in reverse order. Since no absolute ground truth is available, success is defined by the robot's ability to reach the target node. The results indicate that in all scenarios, the robot consistently reached the goal, both in the forward and reverse runs. Figures 5.3–5.9 (scenario 1) and A.1–A.21 (scenarios 2–4) present, for each trial, the scenario's location on the complete map, the planned trajectory, and the camera views captured at the beginning and the end of navigation. Moreover, Figures 5.6, 5.9, A.4, A.7, A.11, A.14, A.18, and A.21 show that the robot reached the goal node irrespective of its orientation, which is consistent with the use of Equation 3.26 as the similarity metric. Figure A.11 reveals that even when the environment near the goal node underwent dramatic changes, the algorithm maintained accurate localization, thereby confirming its robustness in dynamically changing settings. Occasional oscillations in the robot's movement, especially at the starting point of scenario 2 (reverse order), were observed due to poor feature matching and the sensitivity of the 8-point algorithm to noise and degenerate configurations

[49] (situations where the feature points are nearly collinear or lie in a single plane, which reduces the reliability of pose estimation). Although the robot occasionally exhibited oscillatory behavior, it kept moving during these oscillations and eventually obtained enough good matches to overcome the issue and reach the target node. In the reverse scenarios, it was also observed that the robot sometimes did not start exactly from the end node; this behavior is expected, as the algorithm resets and localizes itself by selecting the node with the highest number of feature matches.



Figure 5.30: Example of a degenerate configuration where the robot attempts to navigate from node 1 to node 9, illustrating a failure case due to degenerate configurations or insufficient feature correspondence.

Table 5.4 presents the duration of each navigation scenario, first in the forward direction, followed by the reverse direction. The observed differences in traversal times are primarily attributed to occasional oscillations in robot motion, which arise from minor inaccuracies in the heading vector estimation obtained with the 8-point algorithm. These oscillations can cause the robot to deviate from the desired direction, increasing overall navigation time. This behavior is especially apparent in Scenario 3, where the robot took 128 seconds during the forward traversal through a feature-sparse corridor but only 68 seconds on the return leg.

Table 5.4: Duration of each navigation scenario given in Section 5.1.2.1

Scenario	**First run (s)**	**Second run (repeated in the reverse order) (s)**
1	42	34
2	106	126
3	128	68
4	108	102

Section 5.1.2.2 presents the metric evaluation of the proposed topological navigation algorithm. The localization error analysis demonstrates that the proposed topological navigation algorithm achieves reliable performance. The experimental data indicates a mean position error of 0.319 m, with a standard deviation of 0.191 m and a median of 0.318 m, reflecting consistent accuracy in localizing the robot relative to known nodes. Given that the robot's base has a 54 cm diameter, the mean error indicates that the positions are, on average, only about 5 cm away from the robot. Inspection of the boxplot errors, Figure 5.11, shows that no sample exceeded 0.83 m is the true empirical worst case. Since the TIAGo arm's horizontal workspace radius is 0.87 m, every recorded stopping pose lies at least 4 cm inside the arm's maximum reach. Therefore, the manipulator can perform fine servoing without needing the mobile base to inch forward.

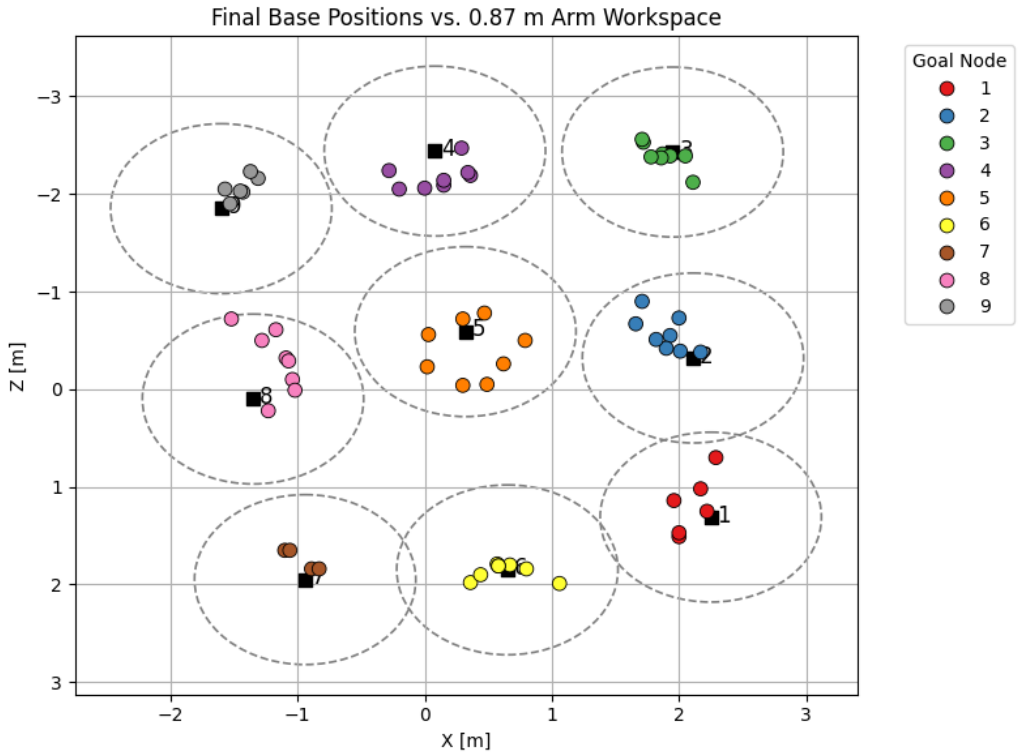


Figure 5.31: The ground-truth positions of all nodes with the final positions of successful navigation trials. Around each node, a dashed gray circle with a radius of 0.87 m represents the TIAGo arm's horizontal reach limit.

Success rates are 100 percent for most nodes, except for nodes 1, 7, and 9, where

the algorithm struggled to determine the correct heading vector due to poor feature matching, low feature density in the environment, and the 8-point algorithm's sensitivity to noise and degenerate configurations.

The proposed similarity metric, Equation 3.26, is based on comparing angular relationships between unit vectors of matched feature keypoints. While this approach provides a rotation-invariant measure of similarity, its sensitivity is influenced by the spatial distribution of features relative to the robot. Specifically, when the features are located close to the robot (such as corners, wall edges, or nearby objects), the angular layout between matched features tends to remain stable, even under small positional or orientational changes. As a result, the similarity metric yields a low error in these cases. In contrast, when features are located at a greater distance, slight variations in the robot's pose can lead to significant distortions in the observed angular relationships. This is primarily due to the increased sensitivity of distant features to small viewpoint shifts, which causes the similarity metric to output larger angle differences. Therefore, the similarity metric favors scenes with close, well-distributed features and exhibits higher error in environments dominated by far-field or sparsely structured visual elements. As a result, the highest position error occurs in goal node 5, Figure B.5, compared to other goal nodes, Figures B.1, B.2, B.3, B.4, B.6, B.7, B.8, B.9.

Two scenarios were conducted to assess the reliability of the proposed triangulation algorithm (Section 5.1.3.1). In each scenario, the robot localized itself and navigated toward a designated goal node using Equation 3.27 as the similarity metric for the goal and Equation 3.26 as the similarity metric for intermediate nodes; upon reaching the goal, the robot adjusted its orientation and extended its hand to the object's scene using the triangulation algorithm. The same procedure was then repeated in reverse order. The generated navigation paths, the views of the scene at the start and end of the navigation, and the hand camera images after the hand was moved to the estimated point are illustrated in Figures 5.15-5.23 and C.1-C.9. From Figures 5.19, 5.22, C.4, and C.8, it can be seen that the robot aligned its orientation with that of the goal node regardless of the approach direction, unlike the scenarios presented in Section 5.1.2.1. Additionally, Figures 5.18, 5.23, C.5, and C.9 show that the triangulation algorithm successfully estimated the position of the object's scene and moved its hand to the

estimated position regardless of the direction of approach.

Section 5.1.3.2 gives a metric evaluation of the proposed triangulation algorithm. The triangulation algorithm's performance in the given experiments was evaluated by comparing estimated positions and orientations to ground truth in three distinct environments. In Experiment 1, conducted in the feature-rich ROMER Mechatronics Laboratory, the algorithm yielded low position error (0.041 m mean error) and low orientation error (2.132-degree mean error), as summarized in Tables 5.2 and 5.3, with corresponding boxplots, histograms, and a 3D plot showing close alignment between estimated and ground truth positions. In Experiment 2, performed in the corridors where visual features were less distinctive, the error statistics, Tables D.1 and D.2, indicate a slight decrease in position error (0.032 m mean error) and a slight increase in orientation error (2.490-degree mean error). Although one might expect richer features to facilitate more accurate localization, the data show a lower position error in Experiment 2. One possible explanation is that only 40% of the 360-degree images were used for matching to avoid mistakenly matching the robot's manipulator. As a result, many potentially useful features may have been excluded, leading to higher errors than Experiment 2. In Experiment 3, the challenging conditions led to false positive matches. As a result, false positive matches caused a slight increase in position error (0.058 m mean error) while maintaining low orientation error (1.969-degree mean error), as shown in Tables D.3 and D.4; however, the given results still demonstrate that the algorithm can estimate positions and target the object's scene despite these difficulties.

CHAPTER 6

CONCLUSION

This thesis proposed a novel indoor topological navigation framework for indoor mobile robots with 360-degree cameras based on feature extraction, matching, and topological mapping. The main goal is to perform reliable navigation without metric information and also to have the ability to create a topological map from a previously recorded environment. The proposed techniques combine state-of-the-art methods in image processing and deep learning to address real-time processing challenges.

A central aspect of this work was the integration of feature extraction and matching techniques with robust pose estimation algorithms. The use of deep learning-based methods, ALIKED combined with LightGlue, enabled the system to identify and match features between spherical equirectangular images with high accuracy and efficiency. With this ability, a topological map was constructed where nodes represent key locations and edges encode heading vectors and relative orientations. The system demonstrated its ability to calculate the required heading command to navigate the robot between nodes.

Another significant contribution of this thesis was the triangulation algorithm utilized once the robot reached a designated goal node. The system effectively pinpointed and approached the object's scene by adjusting the robot's orientation to match that of the goal node and then applying triangulation using 360-degree hand-camera images. Experiments conducted in diverse laboratory environments confirm that the triangulation algorithm can accurately estimate object positions despite occasional errors in feature matching. This additional capability extends the system's usefulness beyond simple navigation tasks, enabling it to interact with specific targets once they are reached.

In conclusion, the proposed topological navigation framework has shown its ability to operate reliably in indoor environments without relying on metric information. The following highlights summarize its main achievements:

- **Topological Mapping & Navigation:** Nodes were automatically selected from 360-degree images, and the system successfully navigated the robot through these nodes using an 8-point algorithm-based heading vector computation.
- **360-Degree Image Processing:** Spherical equirectangular images were handled efficiently with deep learning-based feature detection and matching to accommodate the robot's 360-degree field of view.
- **Triangulation for Object's Scene Estimation:** At the goal node, the robot used the triangulation algorithm to locate the object's position and guide its hand accordingly.
- **Experimental Validation:** The system demonstrated reliability and robustness in experiments conducted in METU ROMER laboratories, indicating that it can cope with variations in feature density and minor environmental changes.

Future work could include:

- **Segmentation for Static Features:** Using segmentation algorithms to detect static objects and primarily rely on their features would reduce the impact of dynamic changes in the environment and enable this approach to handle more complex or rapidly changing scenes.
- **Adaptive Thresholding in Map Construction:** The proposed topological mapping algorithm employs several threshold parameters to prevent false edges in the graph. Future work could explore adaptive or data-driven methods for setting these thresholds, potentially improving map accuracy and reducing the need for manual tuning.
- **Object-Specific Feature Libraries:** Instead of relying on the entire scene of the object, a dedicated feature library could be constructed for the target object. This strategy would exclude the features of dynamic elements in the environment, making the matching process more robust and less prone to false

positives. Additionally, it would enable tracking moving objects or displaced places.

Despite these potential improvements, the current system successfully integrates 360-degree vision-based topological navigation with vision-only triangulation for a precise object approach, offering a practical and reliable solution for indoor robotic applications.

REFERENCES

[1] X. Zhao, X. Wu, W. Chen, P. C. Y. Chen, Q. Xu, and Z. Li, "Aliked: A lighter keypoint and descriptor extraction network via deformable transformation," 2023.

[2] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "Lightglue: Local feature matching at light speed," 2023.

[3] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[4] M. Zhao and W. H. Warren, "Environmental stability modulates the role of path integration in human navigation," *Cognition*, vol. 142, pp. 96–109, 2015.

[5] Y. Şahin, "Development of an indoor reactive navigation method using spherical images," Master's thesis, Middle East Technical University, 2021.

[6] Y. Şahin, "Topological navigation algorithm design and analysis using spherical images," Master's thesis, Middle East Technical University, 2022.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2 ed., 2003.

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[10] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160, 2011.

[11] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[12] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, p. 1874–1890, Dec. 2021.

[13] R. Li, S. Wang, and D. Gu, "Deepslam: A robust monocular slam system with unsupervised deep learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021.

[14] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, p. 2043–2050, IEEE, May 2017.

[15] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, p. 721–728, Apr. 2020.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[18] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, Feb 1992.

[19] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[20] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion," *Int. J. Rob. Res.*, vol. 35, p. 1697–1716, Dec. 2016.

[21] B. Kuipers, "Modeling spatial knowledge," *Cognitive Science*, vol. 2, no. 2, pp. 129–153, 1978.

[22] Z. Zivkovic, B. Bakker, and B. Krose, "Hierarchical map building using visual landmarks and geometric constraints," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2480–2485, 2005.

[23] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[24] L. Suomela, J. Kalliola, H. Edelman, and J.-K. Kämäräinen, "Placenav: Topological navigation through place recognition," 2024.

[25] J. Hossain, A.-Z. Faridee, N. Roy, J. Freeman, T. Gregory, and T. T. Trout, "Toponav: Topological navigation for efficient exploration in sparse reward environments," 2024.

[26] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 1023–1029 vol.2, 2000.

[27] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," 2018.

[28] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," 2020.

[29] S. K. Nayar, "Omnidirectional vision," in *Robotics Research* (Y. Shirai and S. Hirose, eds.), (London), pp. 195–202, Springer London, 1998.

[30] M. J. Tyszkiewicz, P. Fua, and E. Trulls, "Disk: Learning local features with policy gradient," 2020.

[31] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," 2021.

[32] J. Edstedt, Q. Sun, G. Bökman, M. Wadenbäck, and M. Felsberg, "Roma: Robust dense feature matching," 2023.

[33] K. Kawamura, A. Koku, D. Wilkes, R. Ii, and A. Sekmen, "Toward egocentric navigation," *International Journal of Robotics and Automation*, vol. 17, 01 2002.

[34] T. L. T. da Silveira and C. R. Jung, "Perturbation analysis of the 8-point algorithm: A case study for wide fov cameras," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11749–11758, 2019.

[35] H. Guan and W. A. P. Smith, "Structure-from-motion in spherical video using the von mises-fisher distribution," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 711–723, 2017.

[36] J. Fujiki, A. Torii, and S. Akaho, "Epipolar geometry via rectification of spherical images," in *Computer Vision/Computer Graphics Collaboration Techniques* (A. Gagalowicz and W. Philips, eds.), (Berlin, Heidelberg), pp. 461–471, Springer Berlin Heidelberg, 2007.

[37] H. Taira, Y. Inoue, A. Torii, and M. Okutomi, "Robust feature matching for distorted projection by spherical cameras," *IPSJ Transactions on Computer Vision and Applications*, vol. 7, pp. 84–88, 07 2015.

[38] B. Solarte, C.-H. Wu, K.-W. Lu, M. Sun, W.-C. Chiu, and Y.-H. Tsai, "Robust 360-8pa: Redesigning the normalized 8-point algorithm for 360-fov images," 2021.

[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, p. 381–395, June 1981.

[40] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[41] A. Ali-Bey, B. Chaib-draa, and P. Giguère, "Boq: A place is worth a bag of learnable queries," 2024.

[42] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," 2016.

[43] S. Ramachandran, J. Horgan, G. Sistu, and J. McDonald, "Scalable and efficient hierarchical visual topological mapping," in *2023 21st International Conference on Advanced Robotics (ICAR)*, p. 113–120, IEEE, Dec. 2023.

[44] O. Khatib, *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, pp. 396–404. New York, NY: Springer New York, 1990.

[45] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Software*, p. 5, IEEE, 2009.

[46] A. Pagani and D. Stricker, "Structure from motion using full spherical panoramic cameras," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 375–382, 2011.

[47] J. Pagès, L. Marchionni, and F. Ferro, "TIAGo: The modular robot that adapts to different research needs," in *Proceedings of the International Workshop on Robot Modularity (IROS 2016)*, (Daejeon, South Korea), 2016. Workshop paper.

[48] NaturalPoint, Inc., *OptiTrack Motion Capture System*, n.d. Accessed: 17 Mar 2025.

[49] P. Decker, D. Paulus, and T. Feldmann, "Dealing with degeneracy in essential matrix estimation," in *2008 15th IEEE International Conference on Image Processing*, pp. 1964–1967, 2008.

APPENDICES

A Additional Navigation Scenarios

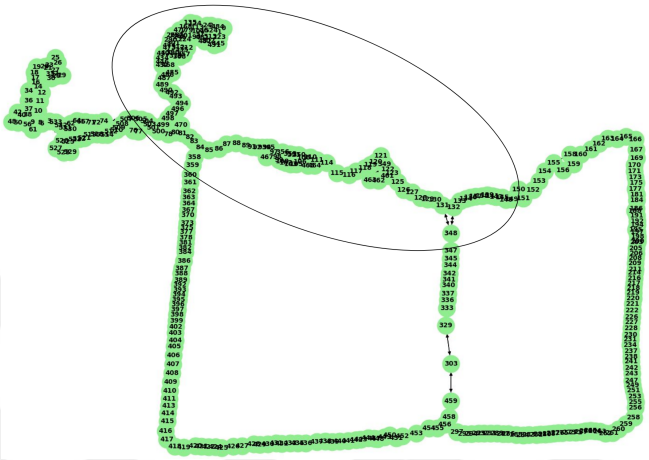


Figure A.1: Location of scenario 2 of topological navigation algorithm test in full map.

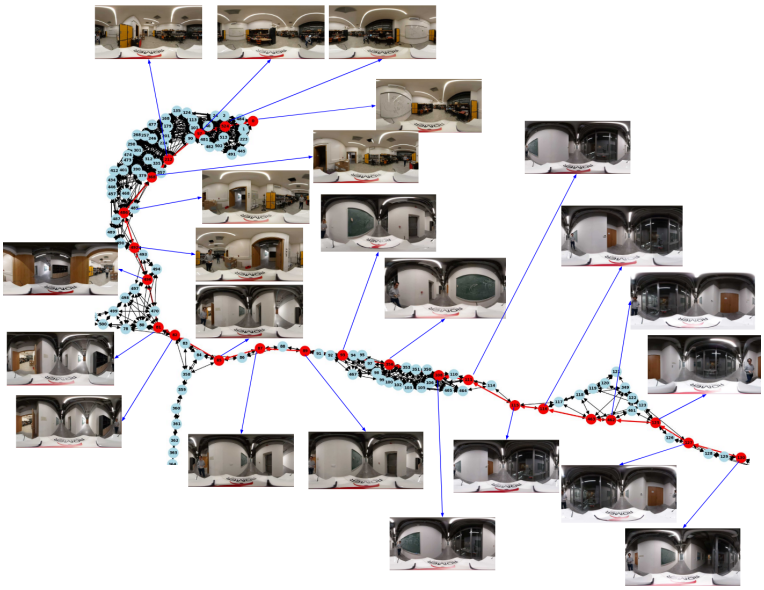


Figure A.2: Path generated in scenario 2 of topological navigation algorithm test.

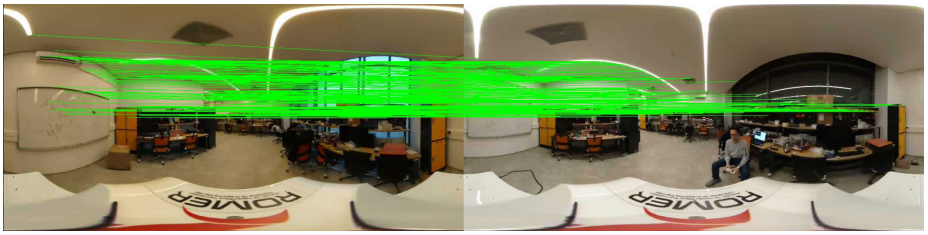


Figure A.3: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of topological navigation algorithm test.



Figure A.4: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of topological navigation algorithm test.

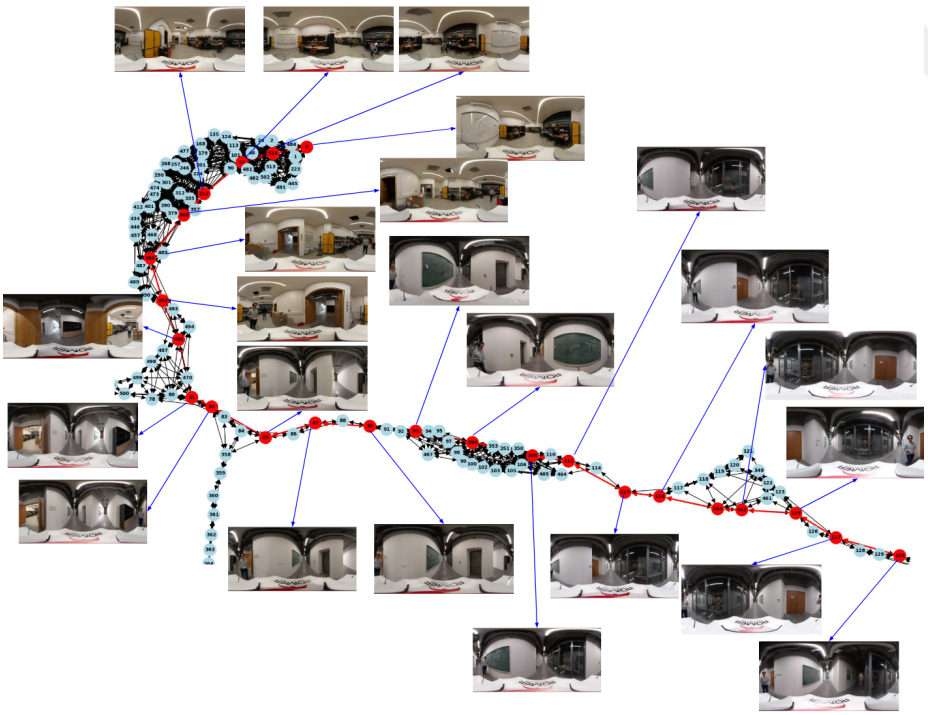


Figure A.5: Path generated in scenario 2 of topological navigation algorithm test (repeated in reverse order).

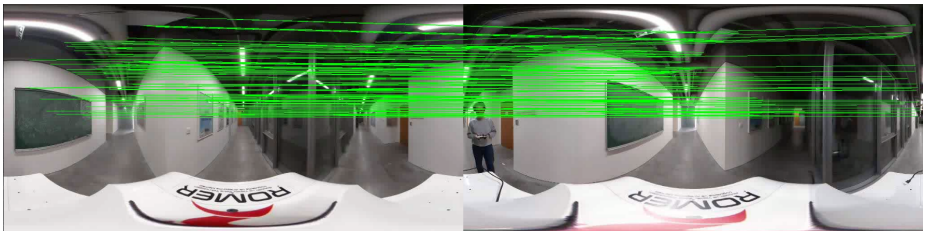


Figure A.6: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of topological navigation algorithm test (repeated in reverse order).



Figure A.7: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of topological navigation algorithm test (repeated in reverse order).

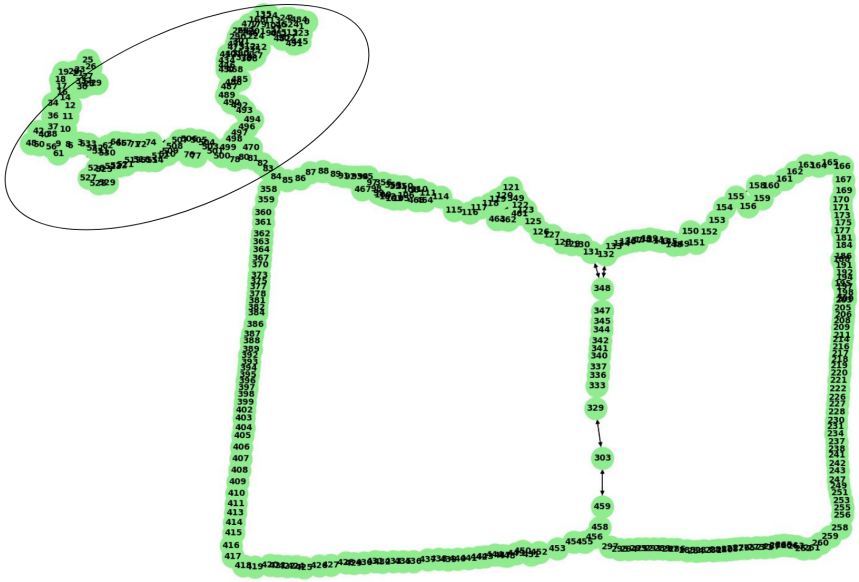


Figure A.8: Location of scenario 3 of topological navigation algorithm test in full map.

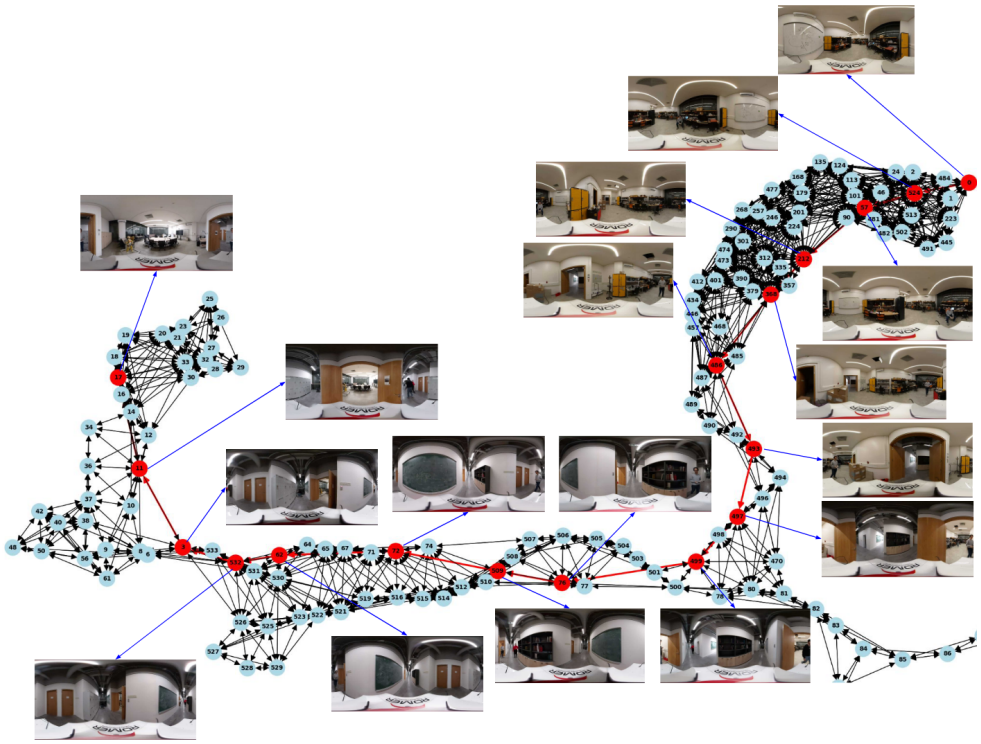


Figure A.9: Path generated in scenario 3 of topological navigation algorithm test.

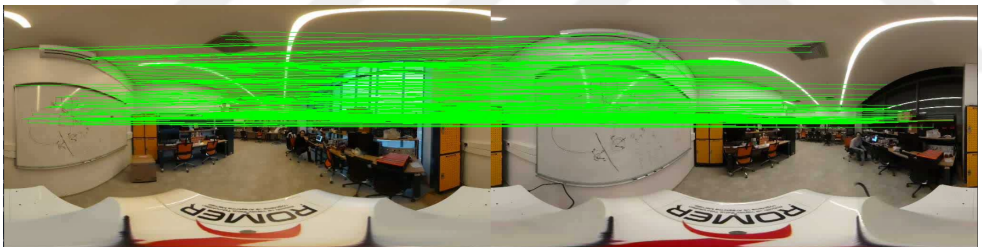


Figure A.10: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 3 of topological navigation algorithm test.



Figure A.11: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 3 of topological navigation algorithm test.

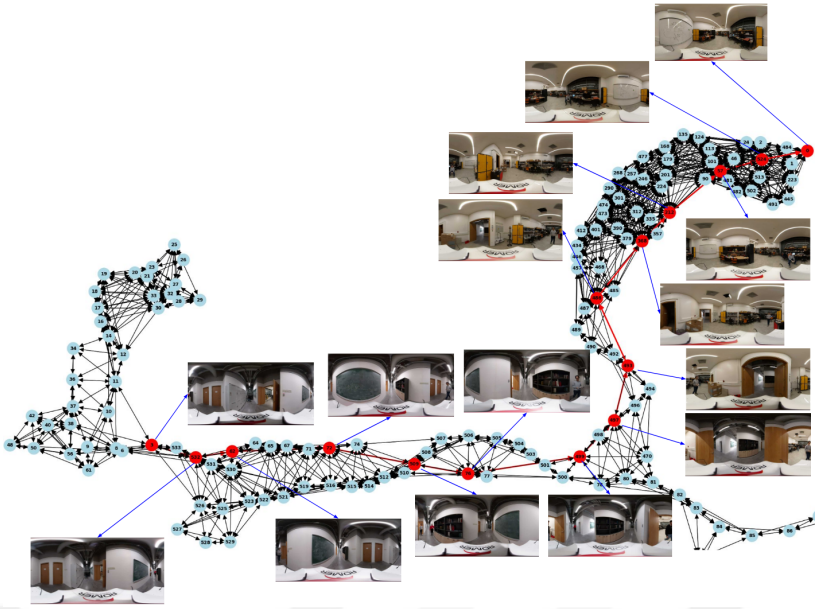


Figure A.12: Path generated in scenario 3 of topological navigation algorithm test (repeated in reverse order).



Figure A.13: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 3 of topological navigation algorithm test (repeated in reverse order).

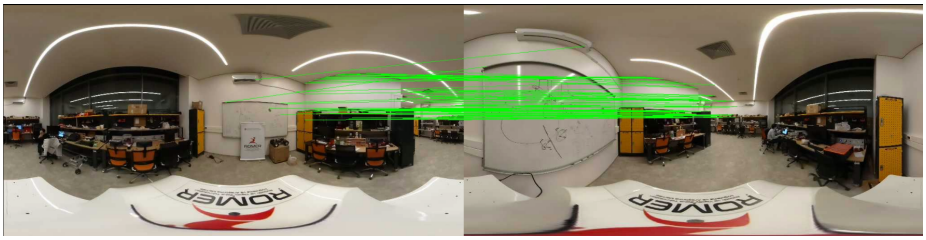


Figure A.14: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 3 of topological navigation algorithm test (repeated in reverse order).

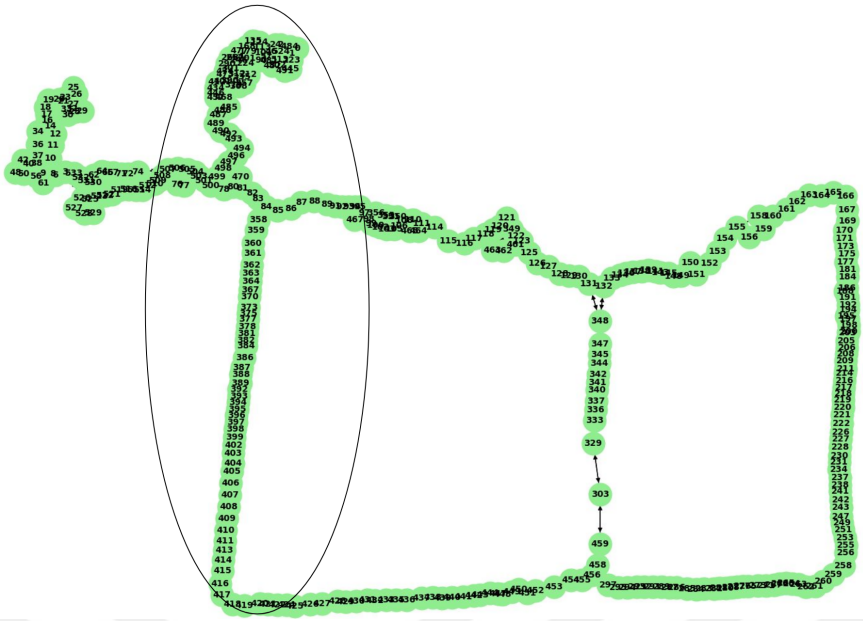


Figure A.15: Location of scenario 4 of topological navigation algorithm test in full map.

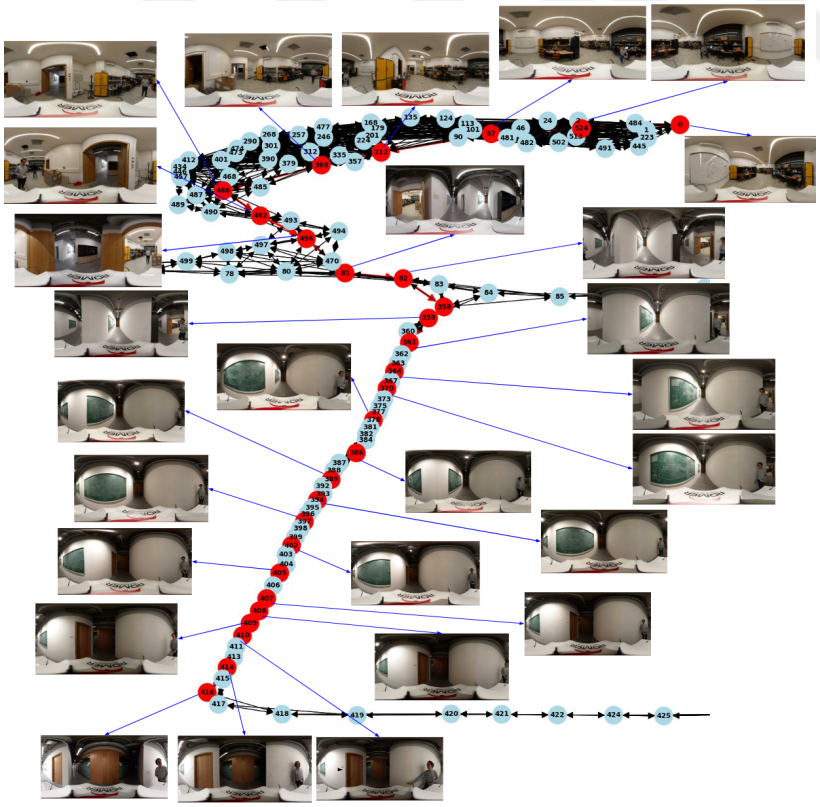


Figure A.16: Path generated in scenario 4 of topological navigation algorithm test.

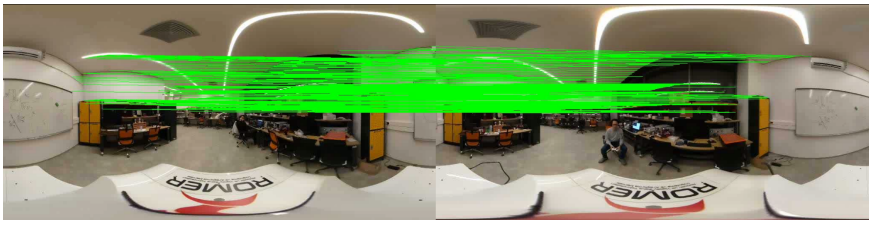


Figure A.17: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 4 of topological navigation algorithm test.



Figure A.18: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 4 of topological navigation algorithm test.

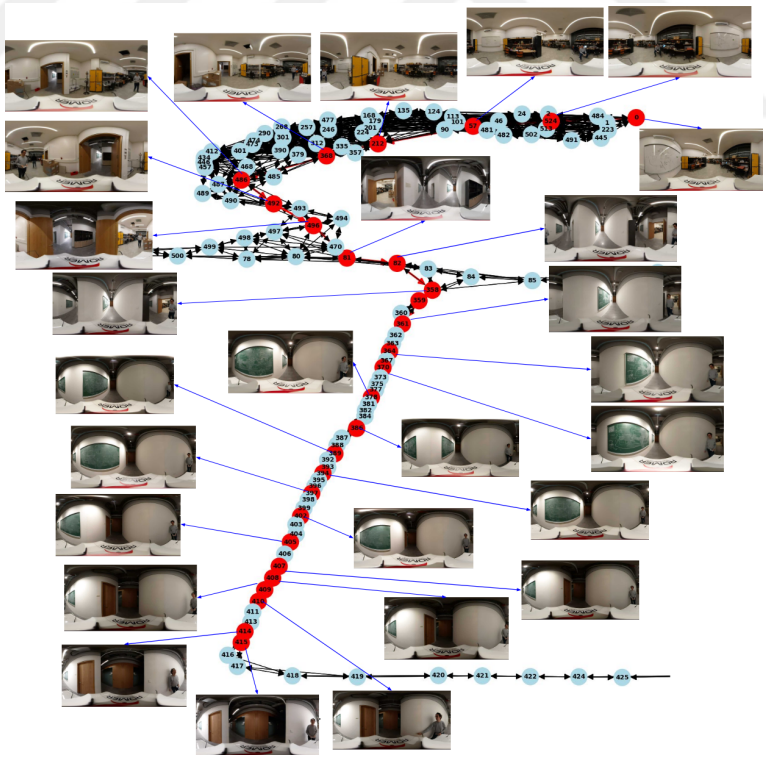


Figure A.19: Path generated in scenario 4 of topological navigation algorithm test (repeated in reverse order).



Figure A.20: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 4 of topological navigation algorithm test (repeated in reverse order).

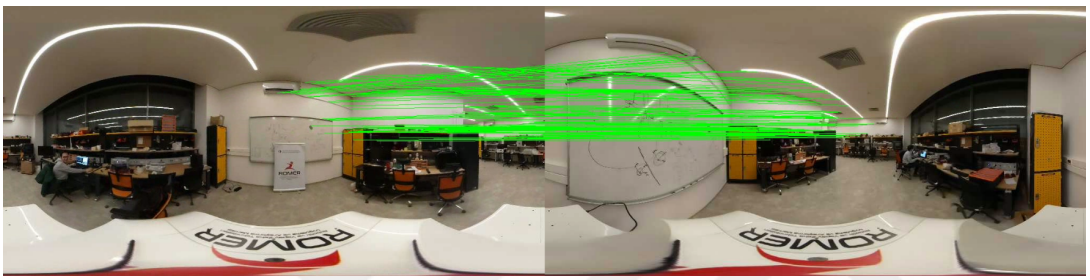


Figure A.21: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 4 of topological navigation algorithm test (repeated in reverse order).

B Final Positions for Each Goal

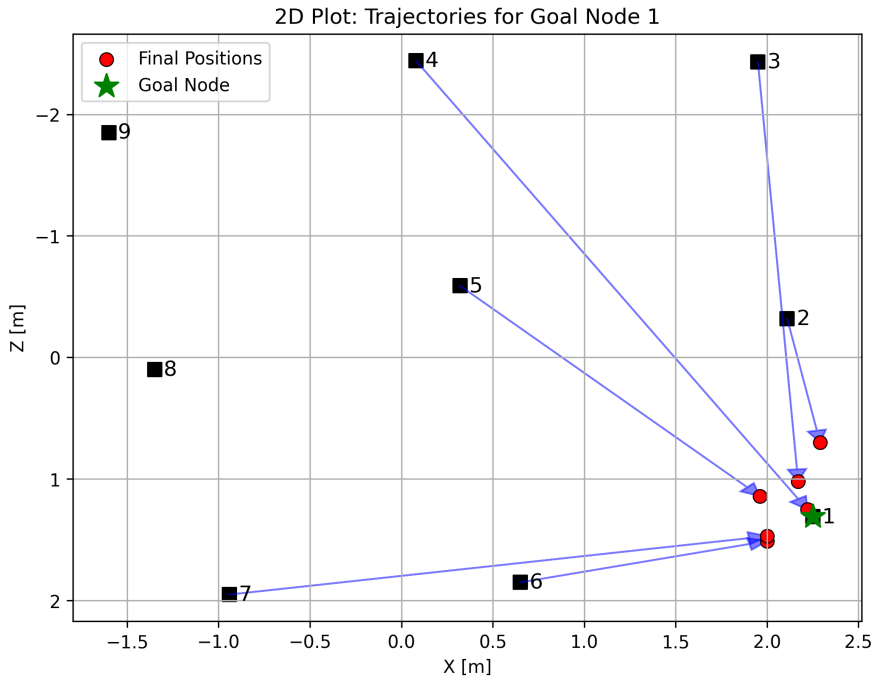


Figure B.1: The final positions of successful navigation trials for goal node 1.

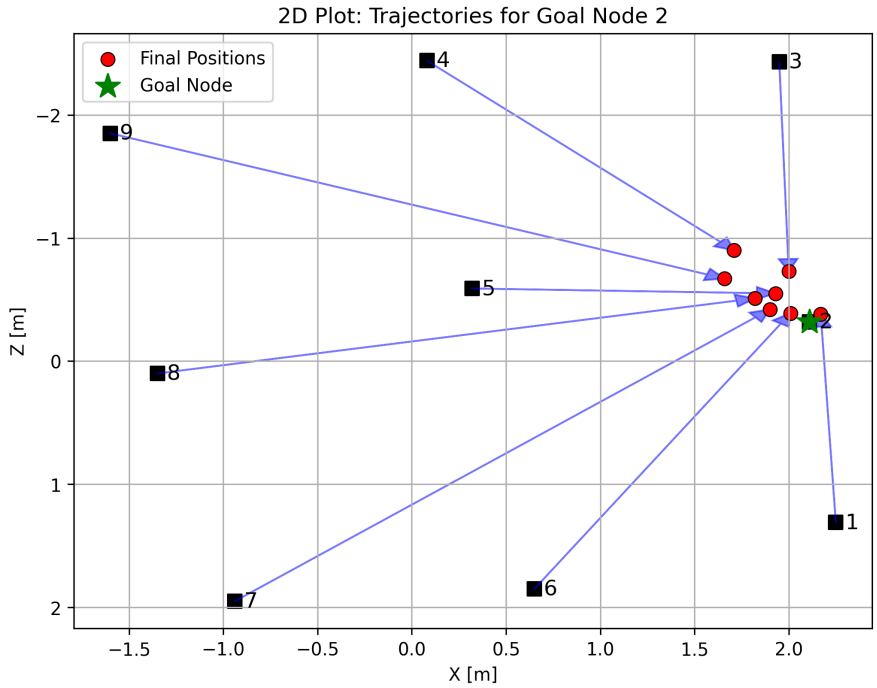


Figure B.2: The final positions of successful navigation trials for goal node 2.

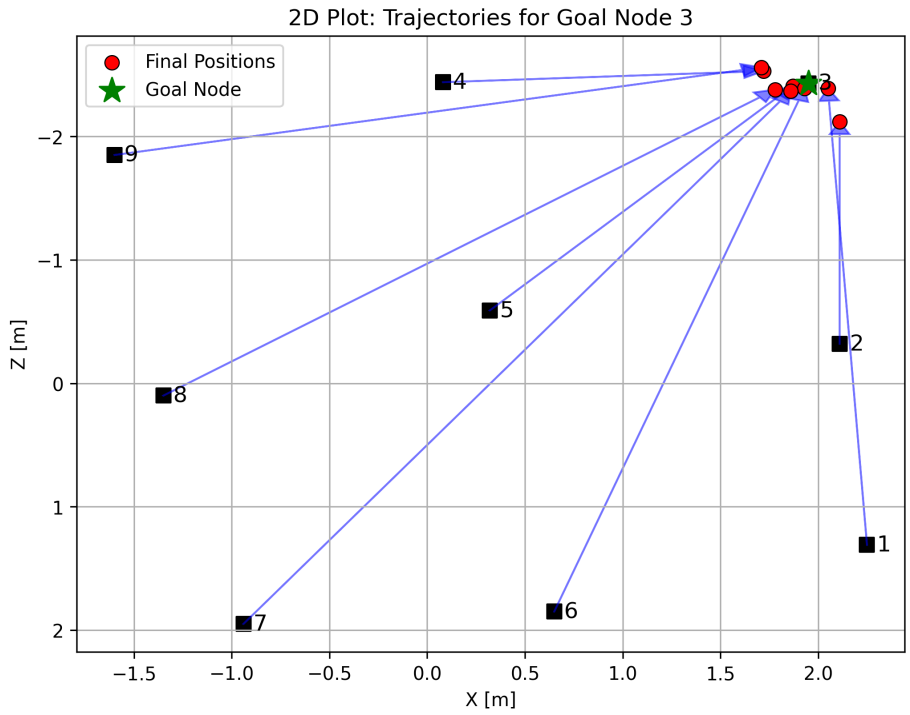


Figure B.3: The final positions of successful navigation trials for goal node 3.

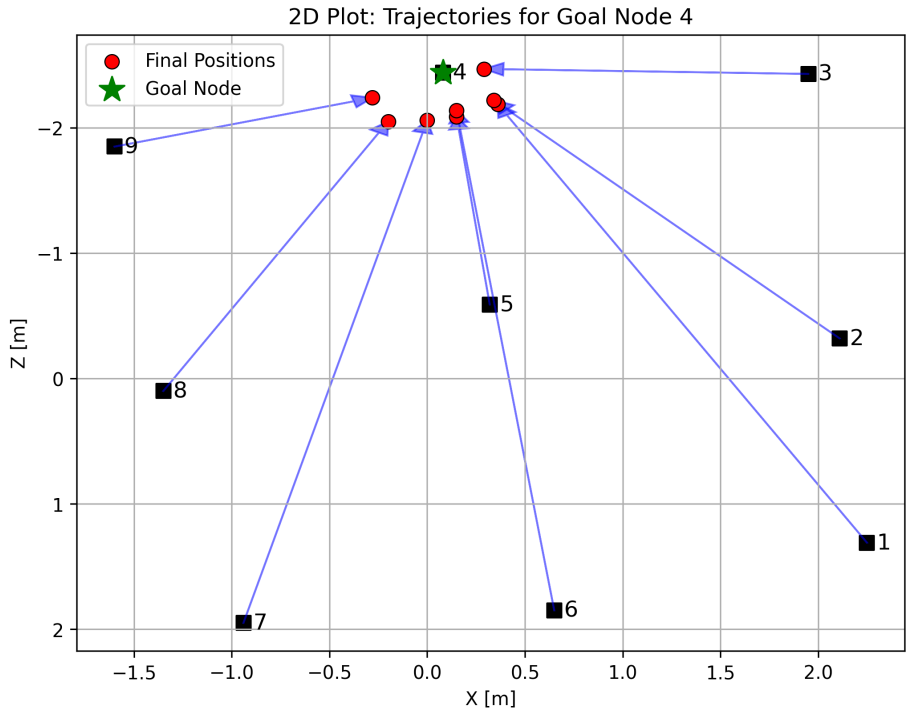


Figure B.4: The final positions of successful navigation trials for goal node 4.

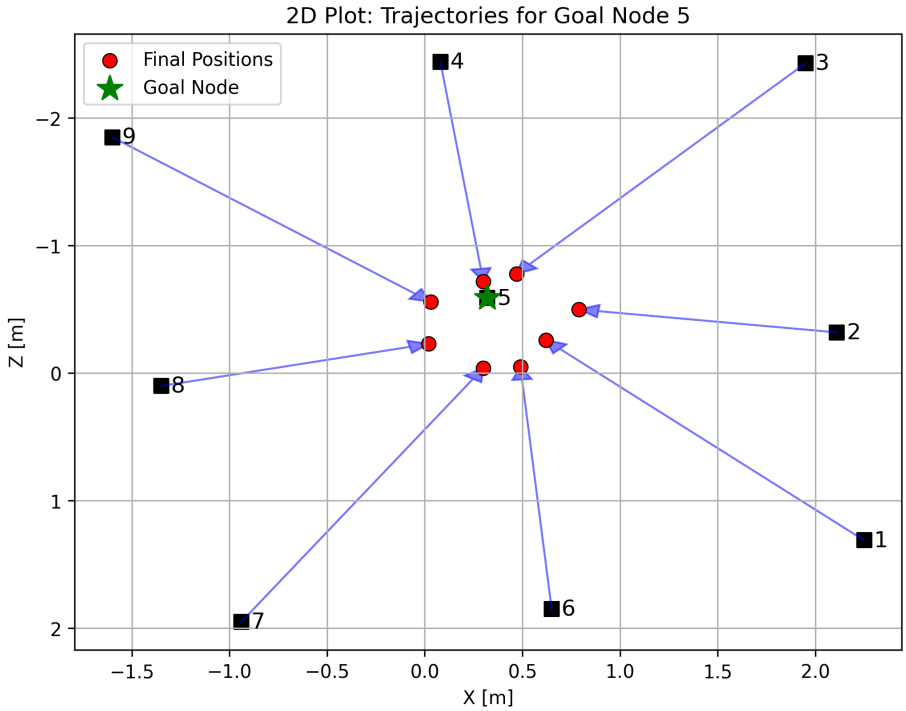


Figure B.5: The final positions of successful navigation trials for goal node 5.

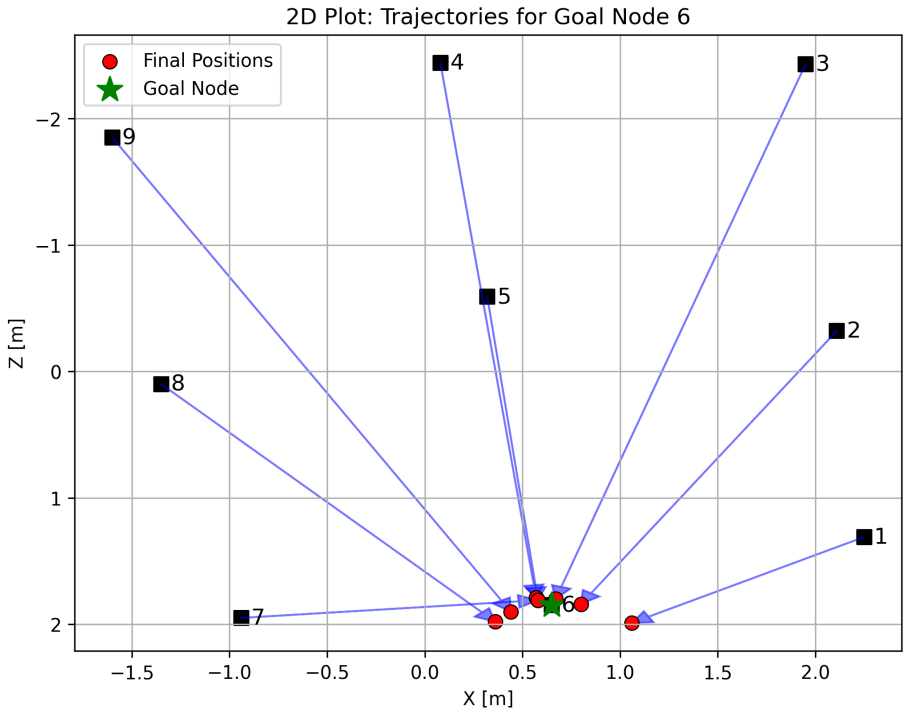


Figure B.6: The final positions of successful navigation trials for goal node 6.

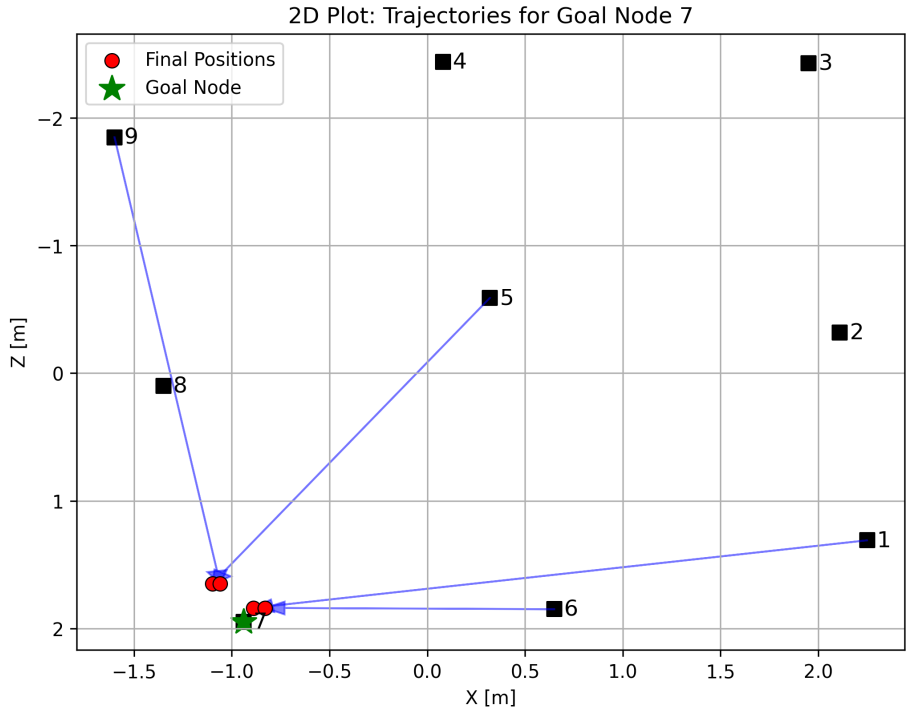


Figure B.7: The final positions of successful navigation trials for goal node 7.

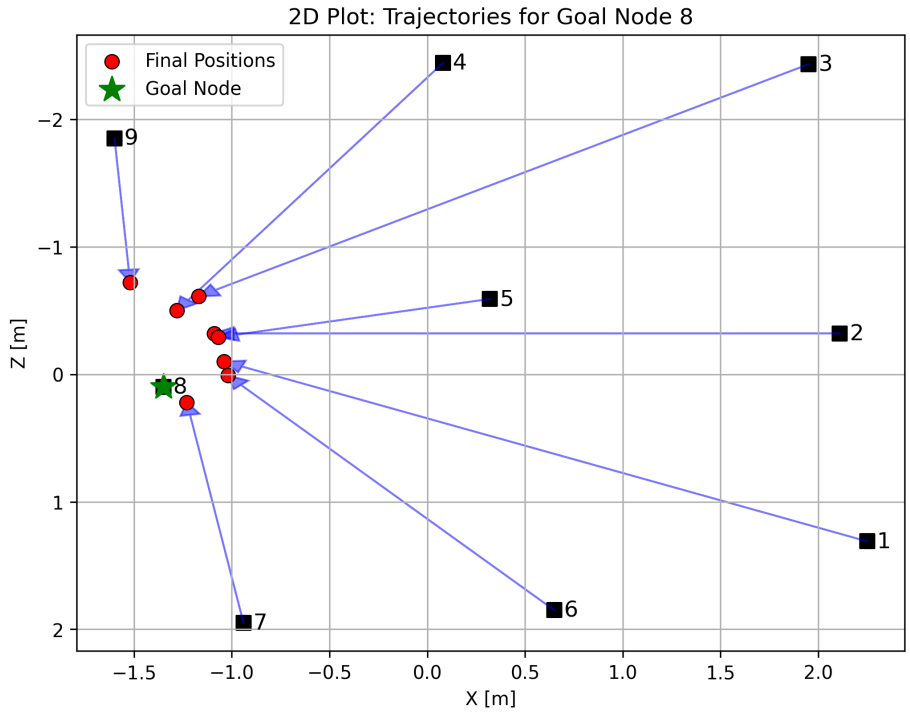


Figure B.8: The final positions of successful navigation trials for goal node 8.

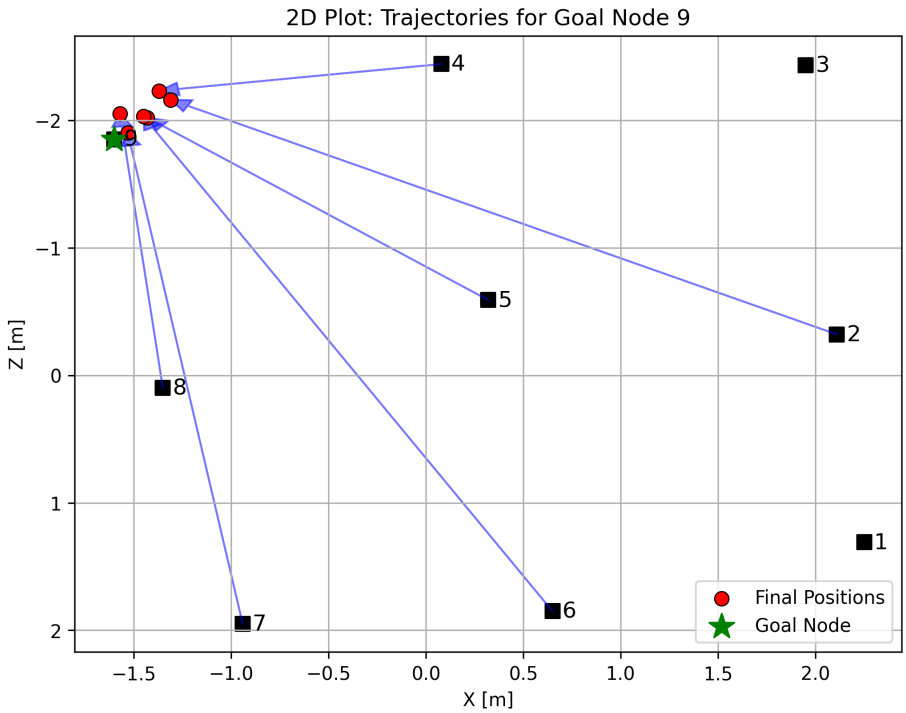


Figure B.9: The final positions of successful navigation trials for goal node 9.

C Additional Triangulation Scenarios

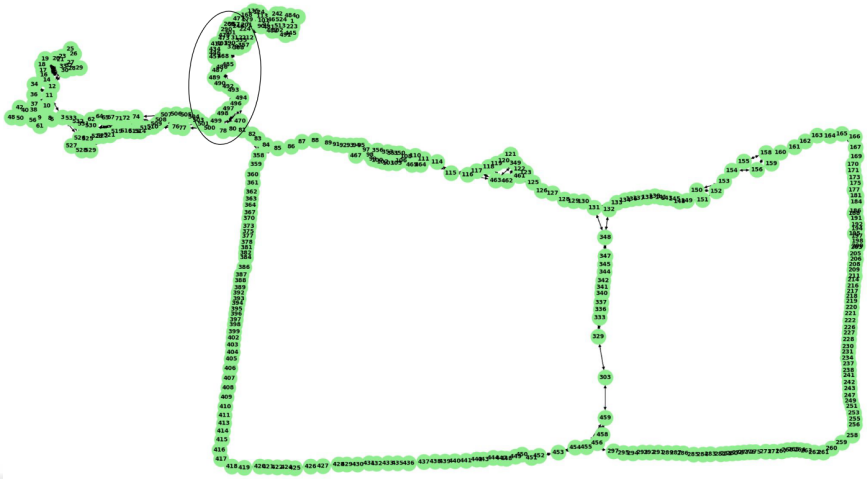


Figure C.1: Location of scenario 2 of triangulation algorithm test in full map.

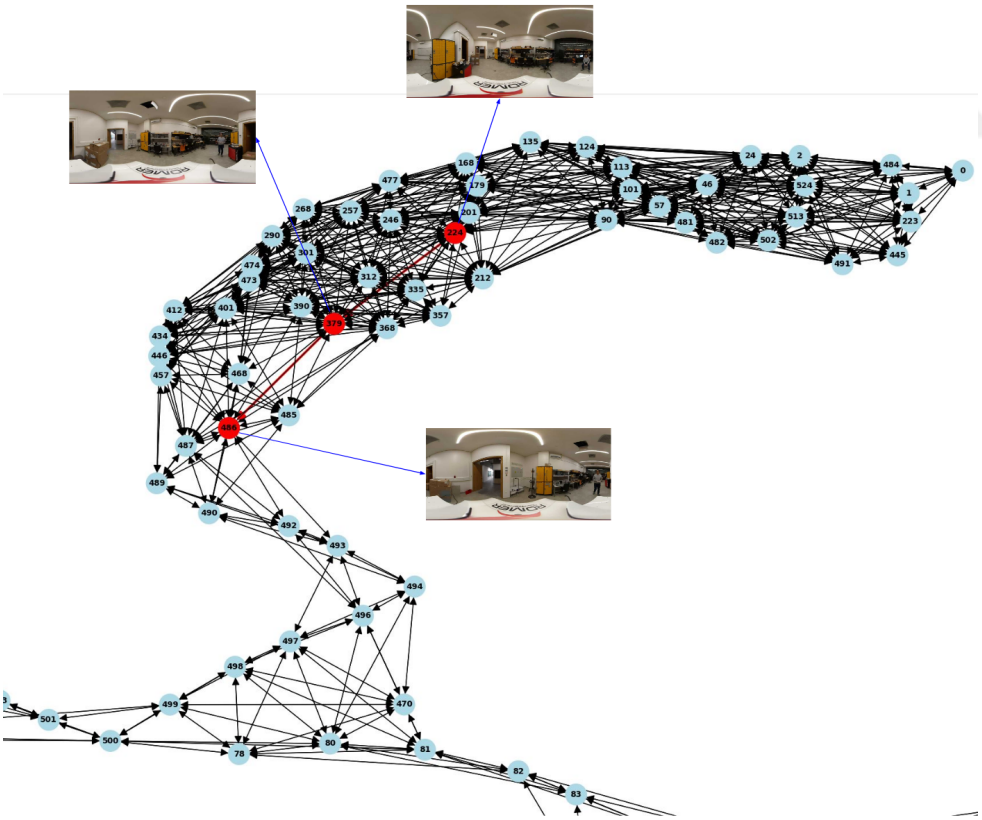


Figure C.2: Path generated in scenario 2 of triangulation algorithm test.

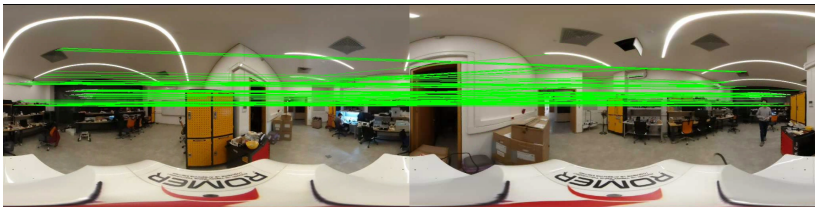


Figure C.3: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 of triangulation algorithm test.

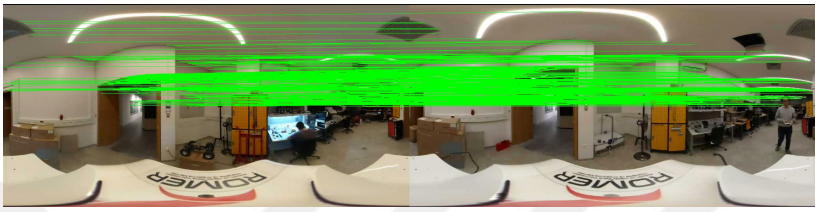


Figure C.4: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 of triangulation algorithm test.



Figure C.5: View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 2.

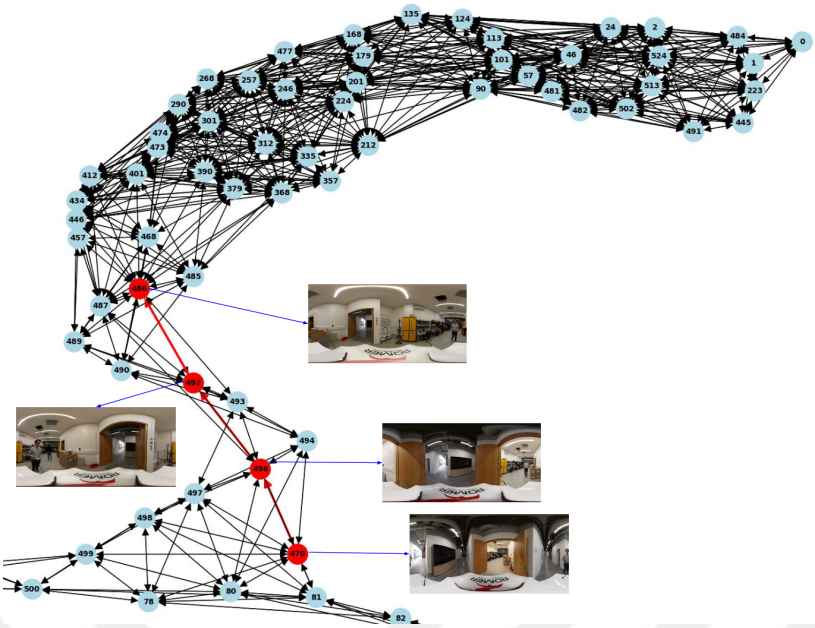


Figure C.6: Path generated in scenario 2 (opposite direction) of triangulation algorithm test .

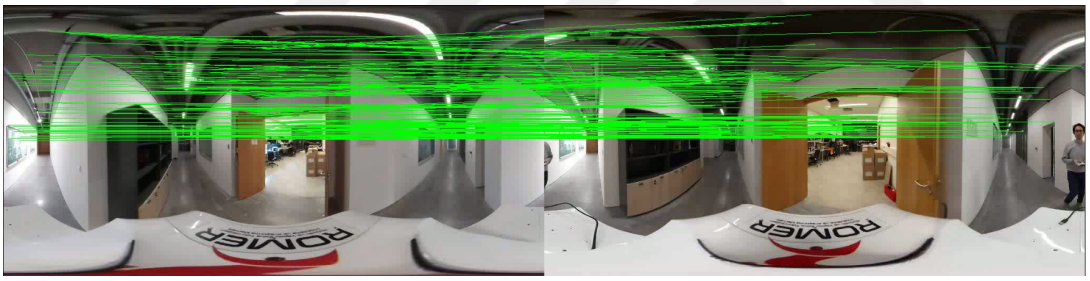


Figure C.7: View of the current scene (left) and target scene (right) when robot starts navigation in scenario 2 (opposite direction) of triangulation algorithm test.

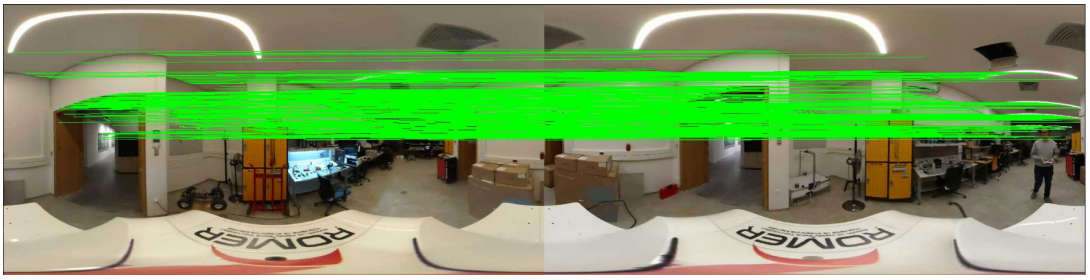


Figure C.8: View of the current scene (left) and target scene (right) when robot reaches to the goal in scenario 2 (opposite direction) of triangulation algorithm test.



Figure C.9: View of the hand camera (bottom) and object's scene (top) after robot executes triangulation algorithm and move its hand to estimated point in scenario 2 (opposite direction).

D Additional Pose Estimation Error Results



Figure D.1: Images taken from the predetermined eight points of Experiment 2.

Table D.1: Position Error Statistics of Experiment 2

Point	Mean (m)	Std (m)	Median (m)	Max (m)
1	0.021	0.012	0.023	0.040
2	0.023	0.011	0.022	0.037
3	0.027	0.011	0.022	0.046
4	0.042	0.016	0.042	0.070
5	0.052	0.013	0.056	0.067
6	0.024	0.011	0.018	0.040
7	0.023	0.006	0.023	0.030
8	0.041	0.009	0.040	0.056
All	0.032	0.015	0.030	0.070

Table D.2: Orientation Error Statistics of Experiment 2

Point	Mean (deg)	Std (deg)	Median (deg)	Max (deg)
1	2.800	1.884	2.107	5.409
2	1.761	0.873	2.016	2.592
3	2.088	0.684	2.228	3.006
4	2.812	1.885	2.066	5.189
5	2.226	1.267	2.141	3.957
6	2.802	2.088	3.514	5.069
7	3.604	1.716	3.781	5.407
8	1.823	1.169	1.748	3.440
All	2.490	1.543	2.173	5.409

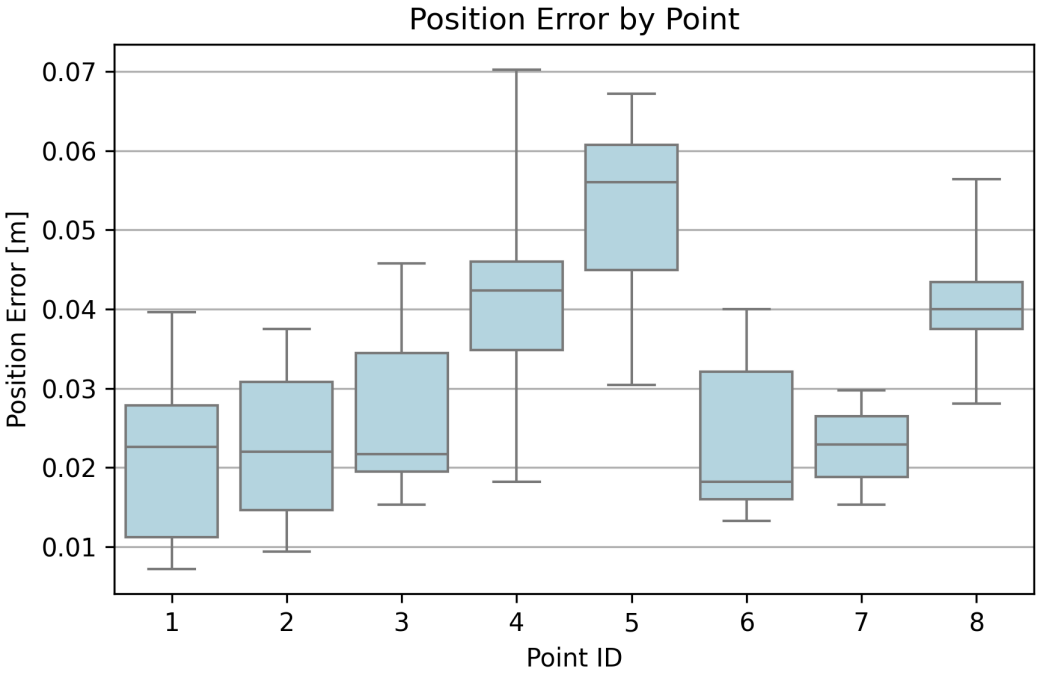


Figure D.2: Position error boxplot of Experiment 2.

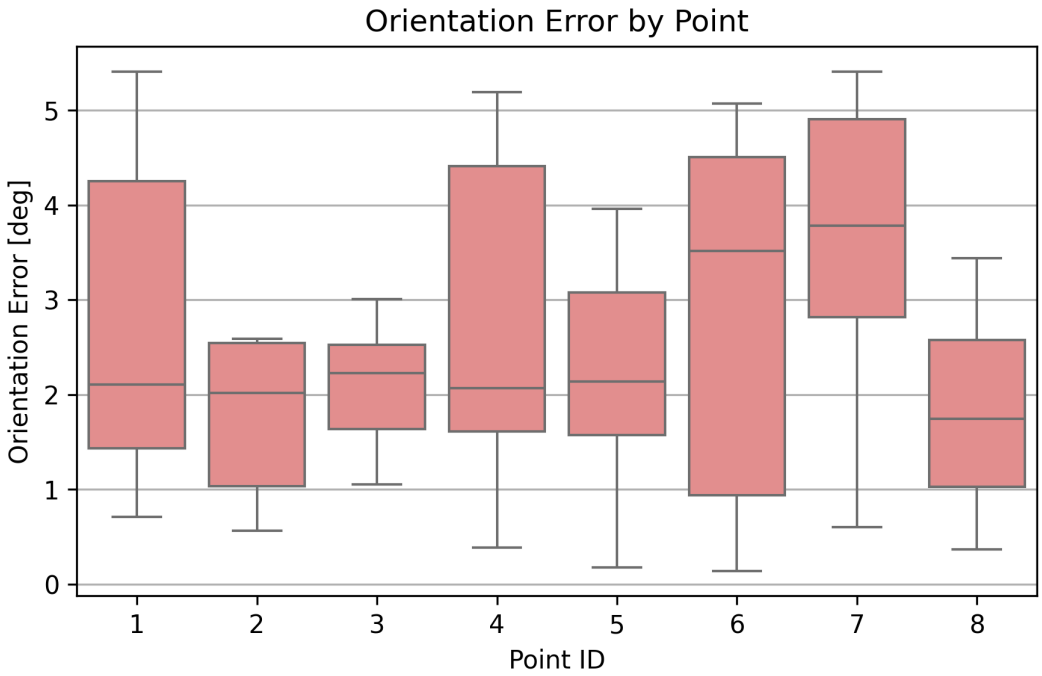


Figure D.3: Orientation error boxplot of Experiment 2.

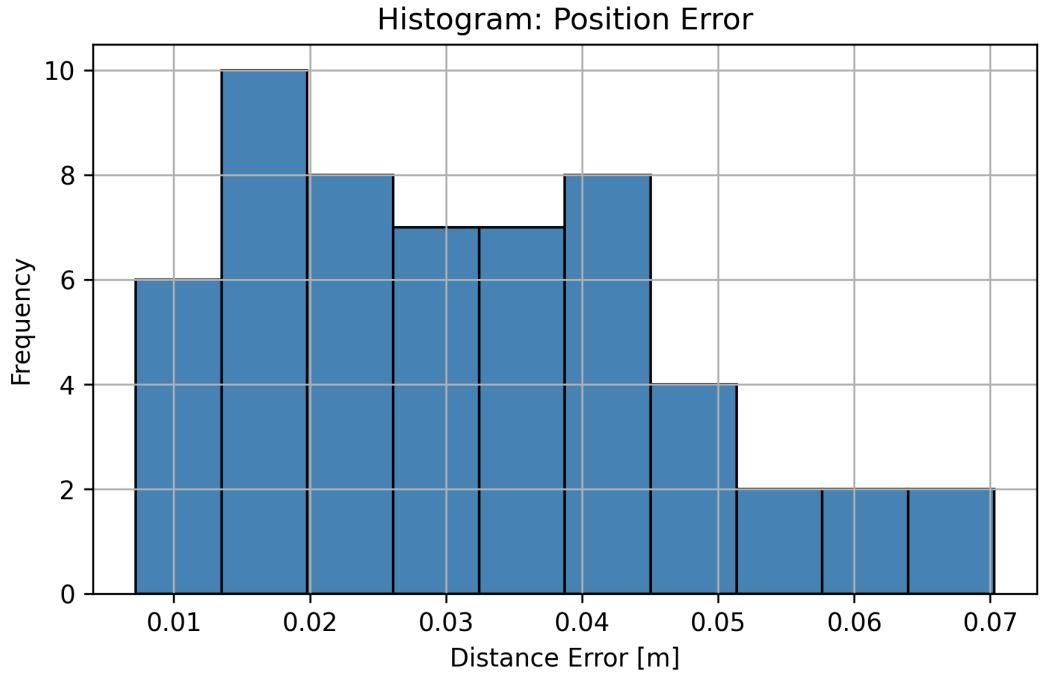


Figure D.4: Position error histogram of Experiment 2.

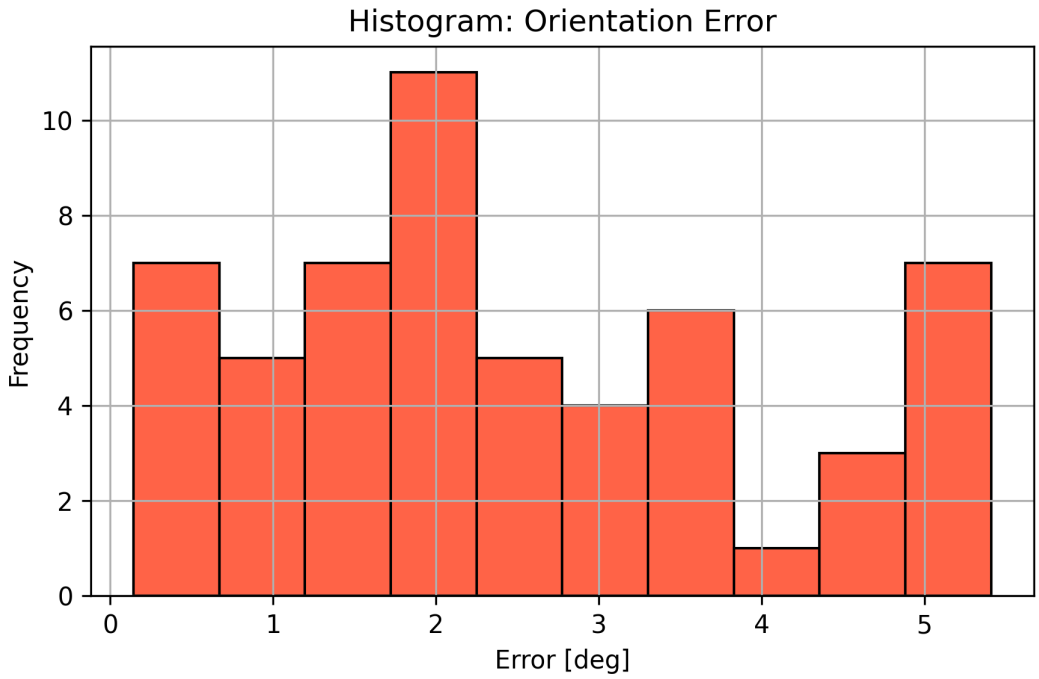


Figure D.5: Orientation error histogram of Experiment 2.

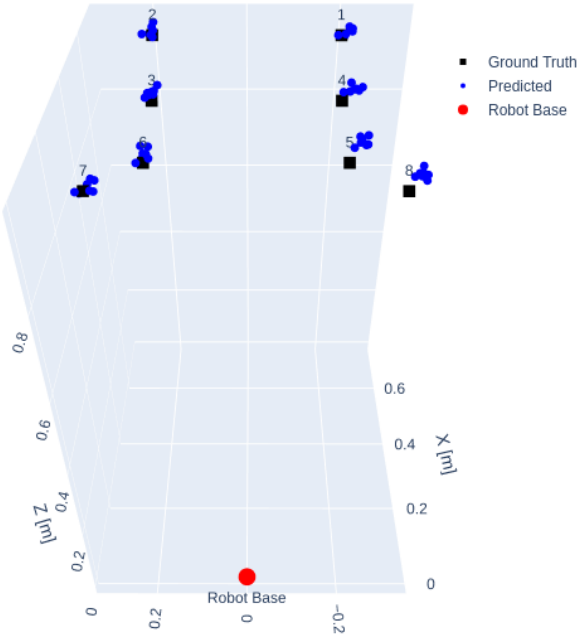


Figure D.6: Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 2. All positions are given according to robot's base frame (red circle).

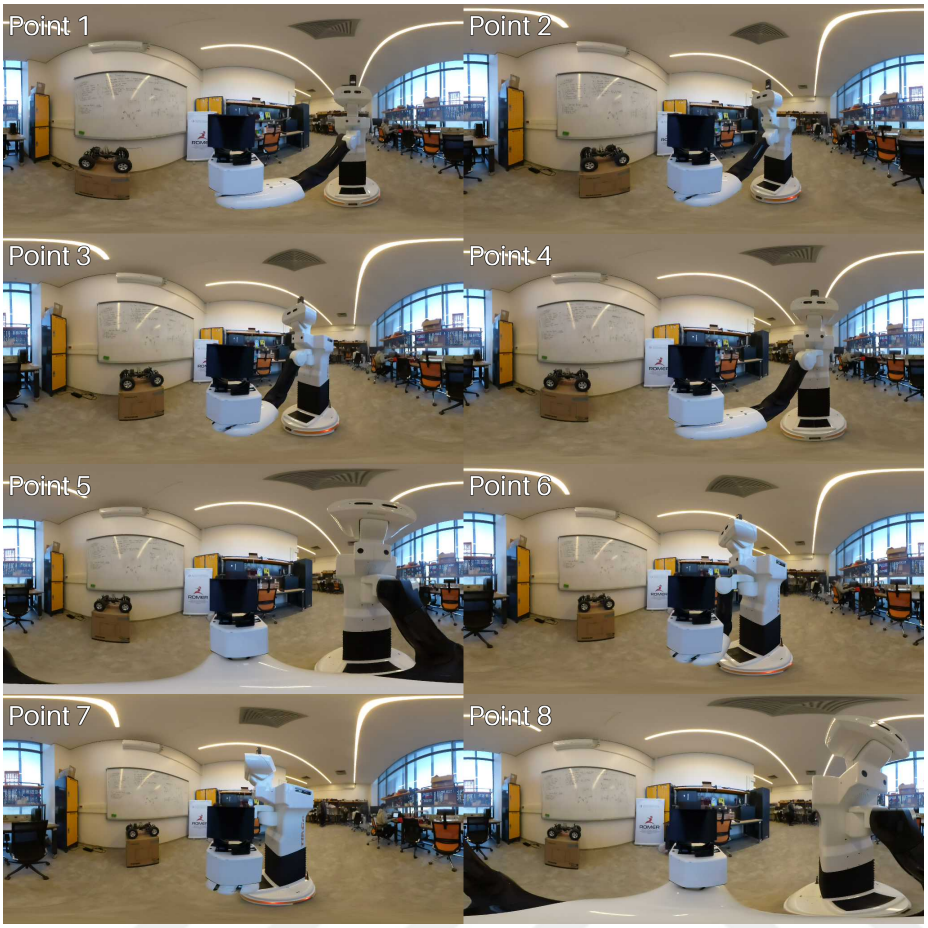


Figure D.7: Images taken from the predetermined eight points of Experiment 3.

Table D.3: Position Error Statistics of Experiment 3

Point	Mean (m)	Std (m)	Median (m)	Max (m)
1	0.065	0.019	0.065	0.085
2	0.052	0.012	0.047	0.069
3	0.049	0.019	0.059	0.074
4	0.043	0.019	0.047	0.069
5	0.069	0.014	0.071	0.087
6	0.065	0.020	0.062	0.097
7	0.054	0.045	0.035	0.137
8	0.069	0.017	0.073	0.087
All	0.058	0.023	0.060	0.137

Table D.4: Orientation Error Statistics of Experiment 3

Point	Mean (deg)	Std (deg)	Median (deg)	Max (deg)
1	2.040	1.506	1.901	4.805
2	1.224	0.814	0.796	2.758
3	1.446	0.892	1.284	2.994
4	1.975	1.812	1.621	5.391
5	3.677	1.282	3.847	4.965
6	2.115	1.619	1.580	4.995
7	1.745	0.861	1.907	2.760
8	1.526	0.788	1.264	2.913
All	1.969	1.375	1.700	5.391

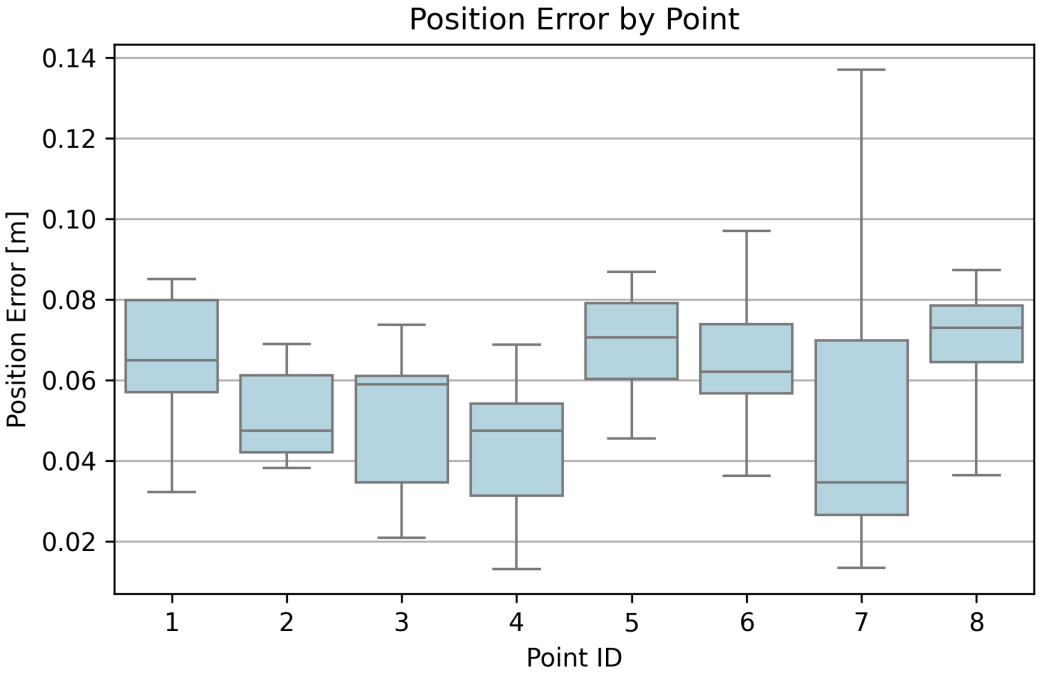


Figure D.8: Position error boxplot of Experiment 3.

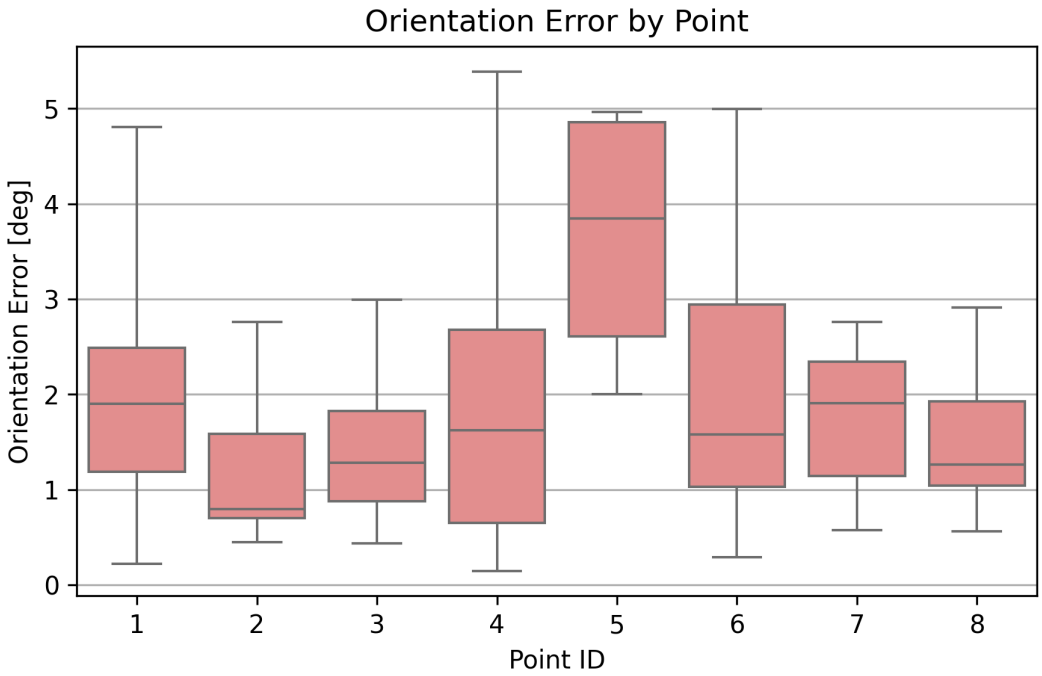


Figure D.9: Orientation error boxplot of Experiment 3.

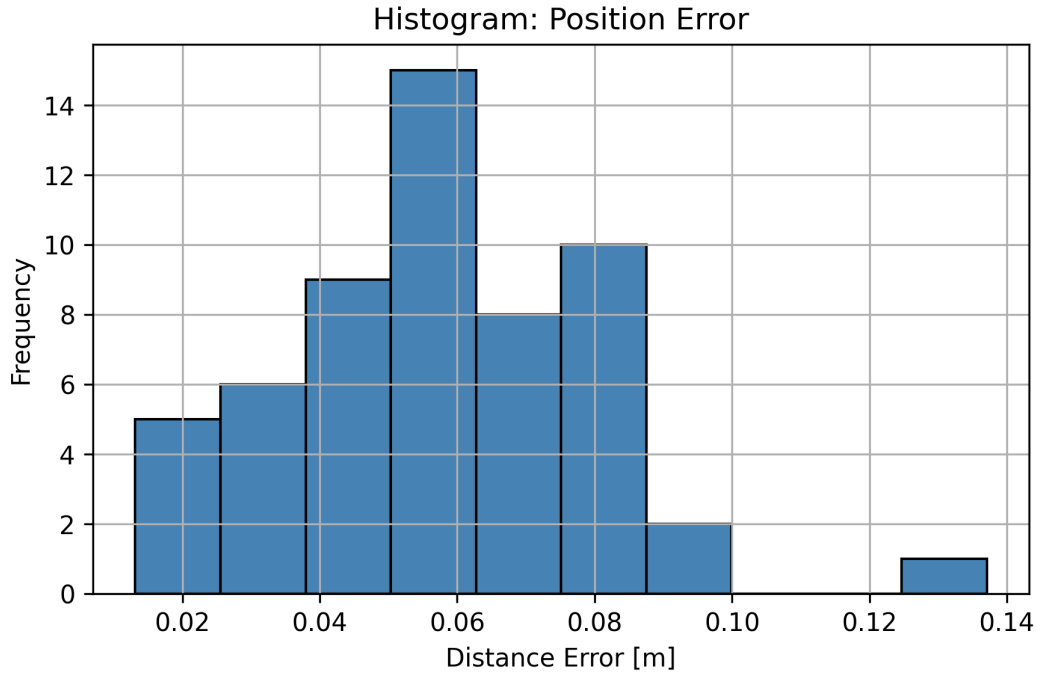


Figure D.10: Position error histogram of Experiment 3.

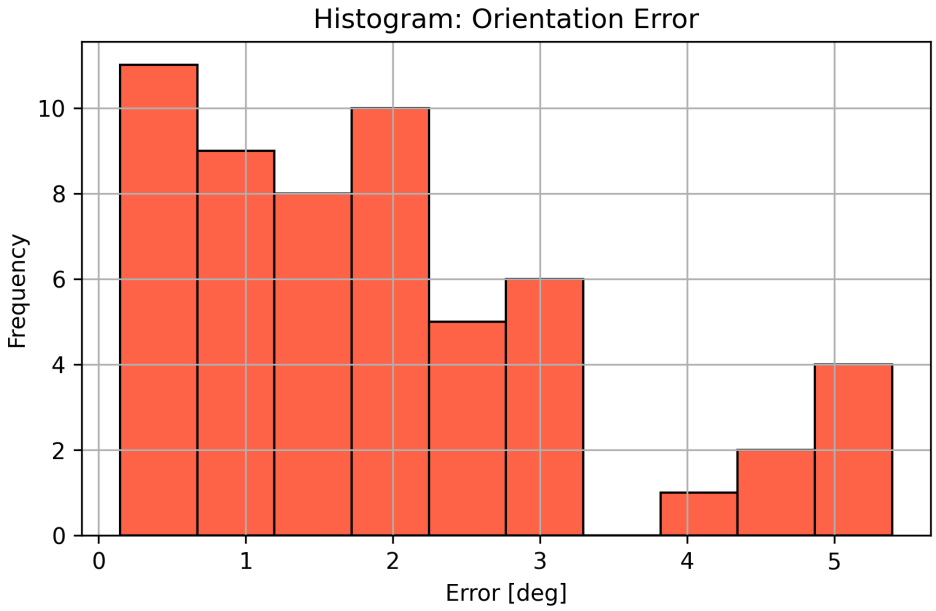


Figure D.11: Orientation error histogram of Experiment 3.

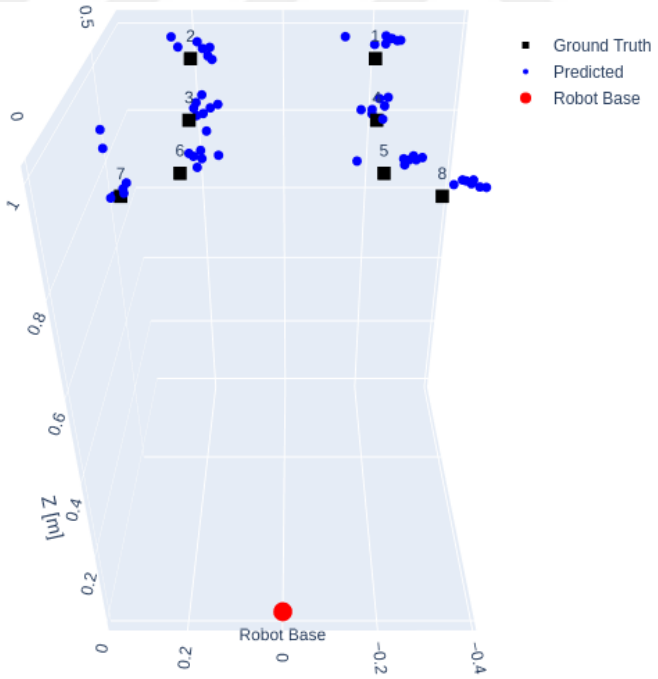


Figure D.12: Ground truth positions (black boxes) and estimated positions (blue circles) of the points of Experiment 3. All positions are given according to robot's base frame (red circle).