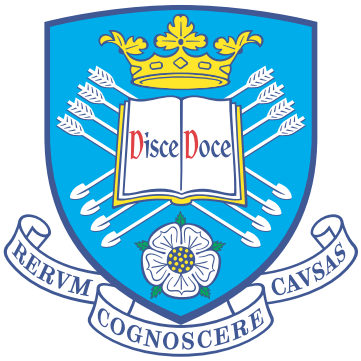
*The University of Sheffield*

*Department of Automatic Control and Systems Engineering*
*Faculty of Engineering*




# Robot Simultaneous Localisation and Mapping in Pipe Networks

**Rui Zhang**

October 2024

Supervisor: Prof. Lyudmila S. Mihaylova
Second Supervisor: Dr Sean Andersson

## Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

# Abstract

This thesis investigates the potential of visual Simultaneous Localisation and Mapping (vSLAM) for robots navigating in sewer pipe networks. SLAM algorithms can estimate a robot's trajectory and build a map simultaneously. Given the complexity and vastness of sewer networks, using multi-robot SLAM is a practical solution for efficient exploration. To accurately estimate the robot's absolute positions, this work also leverages a prior map that details the planar distribution of manholes and pipes with known absolute positions.

However, vSLAM algorithms, suffer from accuracy degradation and scale drift within pipes, leading to rapid error accumulation. This dissertation proposes a vSLAM algorithm which incorporates structural information from the pipe networks to mitigate this problem. The algorithm extracts cylinders and integrates this information into triangulation, local optimisation, and pose graph optimisation. When isodiametric information is available, the algorithm corrects scale drift based on detected cylinders, significantly reducing errors post-correction.

This work introduces a novel, robust, and efficient optimisation-based cylinder extraction method to detect cylinders. It employs a minimal parameter representation, avoiding constraints in optimisation. The thesis also identifies and analyses two numerical instability conditions where point-based cylinder extraction methods can fail. Comparative results with existing methods using both simulated and real data demonstrate the advantages of the proposed method.

This research also proposed a computationally efficient offline map merging method. The method identifies and matches manholes from two maps based on image features and then merges the maps based on overlapping manholes. This approach reduces the computation time to just one-sixth of that of traditional methods.

Finally, the thesis presents a topological fault-tolerant matching method to estimate the absolute pose and scale of the point cloud map using a prior two-dimensional map that details the planar distribution of manholes and pipes. The algorithm accurately determines the absolute pose and scale by aligning the point cloud map with this prior map.

These contributions significantly enhance the accuracy and efficiency of vSLAM in sewer pipe networks, making autonomous exploration and mapping more feasible and reliable than state-of-the-art methods.

# Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof Lyudmila Mihaylova, whose expertise, understanding, and patience, added considerably to my research experience. Without her invaluable advice, continuous support, and encouragement, this journey would not have been possible.

I am also profoundly thankful for the guidance and insightful critiques of my co-supervisor, Dr. Sean Anderson, who has been instrumental in refining my research work.

I am especially grateful to my senior colleague, Dr. Rob Worley, for his mentorship and for dedicating his time to review and provide constructive feedback on my manuscript.

I am blessed with a loving family whose support and understanding have been my strength throughout my studies. Their unwavering belief in my efforts is deeply appreciated.

I would like to extend my heartfelt appreciation to my girlfriend, Xiaoyin, whose companionship ensured that my time at university was filled with joy and never lonely.

To all my friends I've made during my doctoral studies, your friendship means the world to me, and I am grateful for the memories we have created together.

This research was part of the Pipebot project and was made possible by the funding received from the EPSRC. I am sincerely thankful for their support.

Finally, my journey would not have been the same without the academic and social community at the university, and for that, I am forever thankful.

Rui Zhang

# Contents

**Abstract** **v**

**Acknowledgements** **vii**

**List of Symbols** **xv**

**List of Figures** **xvii**

**List of Tables** **xix**

**1 Introduction** **1**

1.1 Background . . . . . 1

1.2 Vision-based Pipe Inspection Methods . . . . . 4

1.3 Key Contributions . . . . . 6

1.4 Publications . . . . . 6

**2 Literature Review** **9**

2.1 Robot Simultaneous Localisation and Mapping . . . . . 9

2.1.1 Preliminaries . . . . . 10

2.1.2 Visual Simultaneous Localisation and Mapping . . . . . 11

2.1.3 Sensor Data Fusion . . . . . 16

2.1.4 Structure-SLAM . . . . . 18

2.2 Structure from Motion . . . . . 20

2.3 Cylinder Detection . . . . . 22

2.3.1 Cylinder Fitting . . . . . 23

2.3.2 Cylinder Extraction . . . . . 24

2.3.3 Cylinder Representation . . . . . 25

2.4 Map Merging and Map Alignment . . . . . 25

2.4.1 Feature Map Integration . . . . . 26

2.4.2 Topological Map Merging . . . . . 27

2.4.3 Graph Matching . . . . . 27

2.5 Summary . . . 29
2.5.1 Research Gaps . . . 29
2.5.2 Aims and Objectives: . . . 30

**3 Cylinder Detection** **31**
3.1 Introduction . . . 31
3.2 New Cylinder Representation . . . 33
3.2.1 Transformation between Different Cylindrical Representations . . . 35
3.3 Cylinder Extraction based on Point Cylinder Distance . . . 37
3.4 Numerical Instability Conditions and Solution . . . 39
3.4.1 Numerical Instability Conditions . . . 39
3.4.2 The Solution to Numerical Instability Conditions . . . 41
3.5 Cylinder Extraction based on Reprojection Error . . . 43
3.5.1 Direct Method . . . 43
3.5.2 RANSAC Algorithm . . . 45
3.5.3 Optimisation . . . 46
3.6 Experiments . . . 48
3.6.1 Synthetic Data 1 . . . 50
3.6.2 Synthetic Data 2 . . . 58
3.6.3 Real Data . . . 60
3.7 Summary . . . 61

**4 Leveraging Cylindrical Regularity in ORB-SLAM** **63**
4.1 Introduction . . . 63
4.2 System Overview . . . 65
4.3 Cylinder Detection . . . 67
4.4 Triangulation . . . 68
4.5 Local Bundle Adjustment with Cylindrical Regularity . . . 71
4.6 Pose Graph Optimisation with Cylindrical Regularity . . . 72
4.7 Pose Graph Optimisation with Isodiametric Information . . . 73
4.8 Experiments and Results . . . 74
4.8.1 Synthetic Data . . . 75
4.8.2 Real Data . . . 78
4.9 Discussion . . . 80
4.10 Summary . . . 80

**5 Map Alignment** **83**

5.1 Introduction . . . 83

5.2 Feature Map Merging . . . 85

5.2.1 System overview . . . 85

5.2.2 Potential Manhole Keyframes Selection . . . 86

5.2.3 Manhole Detection . . . 87

5.2.4 Manhole Matching . . . 88

5.2.5 Map Merging . . . 88

5.3 Topological Map Alignment . . . 89

5.3.1 System Overview . . . 90

5.3.2 Notation . . . 90

5.3.3 Transformation of Feature Maps into Topological Maps . . . 91

5.3.4 Initial Vertex Matching . . . 91

5.4 Extension . . . 95

5.4.1 Cost Function . . . 95

5.4.2 Vertex Matching . . . 96

5.4.3 Neighborhood Matching . . . 98

5.4.4 Penalty For Editing the Map . . . 102

5.4.5 Detailed Algorithm Workflow . . . 104

5.5 Experiments and Results . . . 106

5.5.1 Map Merging Algorithm Evaluation . . . 106

5.5.2 Map Alignment Algorithm Evaluation . . . 109

5.6 Summary . . . 112

**6 Conclusions and Future Work** **113**

6.1 Conclusions . . . 113

6.2 Future Work . . . 115

**Bibliography** **117**

# List of Symbols

**Abbreviations**

| | |
|---|---|
| $2D$ | Two-Dimensional |
| $3D$ | Three-Dimensional |
| $APE$ | Absolute Pose Error |
| $BA$ | Bundle Adjustment |
| $BOW$ | Bag of Words |
| $BRIEF$ | Binary Robust Independent Elementary Features |
| $CCTV$ | Closed-Circuit Television |
| $CRORB$ | A vSLAM algorithm leveraging Cylindrical Regularity based on ORB-SLAM2 proposed in Section 4.8 |
| $CRORBic$ | CRORB with isodiametric correction |
| $DetRPCA$ | RPCA using DetMCD based robust mean and covariance matrix |
| $DSO$ | Direct Sparse Odometry |
| $DSVIO$ | Direct Sparse Visual-Inertial Odometry |
| $FEJ$ | First Estimate Jacobians |
| $ICP$ | Iterative Closest Point |
| $ILS$ | Iterative Least Square proposed in Section 3.3 |
| $IMU$ | Inertial Measurement Unit |
| $LBD$ | Line Band Descriptor |
| $LED$ | Levenshtein Edit Distance |
| $LSD$ | A fast Line Segment Detector |

| | |
|---|---|
| $LSD-SLAM$ | The Large Scale Direct monocular SLAM |
| $MAP$ | Maximum A Posteriori |
| $MCS$ | Maximum Common Subgraph |
| $MSCKF$ | The Multi-State Constraint Kalman Filter |
| $NDI$ | Nondestructive Inspection |
| $NDT$ | Nondestructive Test |
| $ORB-SLAM$ | A future-based method proposed in [1] |
| $ORB-SLAM2$ | A new version of ORB-SLAM proposed in [2] |
| $PCA$ | Principal Component Analysis |
| $PDF$ | Probability Density Function |
| $PnP$ | Perspective-n-Point |
| $PTAM$ | Parallel Tracking And Mapping |
| $RANSAC$ | Random Sample Consensus |
| $RANSAC2$ | RANSAC using reprojection error for evaluation |
| $RANSAC2(O)$ | RANSAC2 results optimised with reprojection error minimisation |
| $RLTS$ | Repeated Least Trimmed Square proposed in [3] |
| $RMSE$ | Root Mean Square Error |
| $ROS$ | Robot Operating System |
| $RPCA$ | Robust Principal Component Analysis |
| $SfM$ | Structure from Motion |
| $SIFT$ | The Scale-Invariant Feature Transform Method |
| $SLAM$ | Simultaneous Localisation And Mapping |
| $SSD$ | Sum of Squared Distance |
| $SVD$ | Singular Value Decomposition |
| $SVD$ | Singular Value Decomposition |
| $SVO$ | Semi-direct monocular Visual Odometry |

| | |
|---|---|
| $UAV$ | Uncrewed Aerial Vehicle |
| $VINS$ | Visual-Inertial Navigation System |
| $VIO$ | Visual Inertial Odometry |
| $VOSLAM$ | Visual-Odometric Simultaneous Localisation And Mapping |
| $vSLAM$ | Visual Simultaneous Localisation And Mapping |

# List of Figures


1.1 Challenging environment: Feature-less pipe wall and varying lighting [4] . . . . . 3

1.2 Challenging environment: Repetitive texture[4] and Results [5] . . . . . 4

2.1 Interface of ORB-SLAM2 [5] running with real data [6] from a sewer pipe in the UK . . . . . 14

2.2 Interface of DSO [7] running with real data [6] from a sewer pipe in the UK . . . 15

2.3 Interface of SVO [8] running with real data [6] from a sewer pipe in the UK . . . 16

3.1 A cylinder coordinate system is shown within a global coordinate system $\mathbb{C}$. The proposed method defines a cylinder by describing the cylinder coordinate system. 34

3.2 A point on it can be described by its angle $\theta$ and height $h$ . . . . . 40

3.3 The numerical instability conditions in cylinder fitting . . . . . 40

3.4 The numerical instability conditions can be detected by the new evaluation method 42

4.1 Scale factor decrease over time using ORB-SLAM2 in a pipe . . . . . 64

4.2 Parallax angles for distant and near landmarks . . . . . 69

4.3 Estimated cylinders in a pipe . . . . . 73

4.4 Simulation results. A: Real-world sewer pipe network. B: Synthetic pipe network created in ROS-Gazebo to emulate the real network in panel A. C-F: Estimated trajectories around the simulated pipe network from the ORB-SLAM2, CRORB and CRORBic algorithms against the ground-truth trajectory. Superscript* means the algorithm runs loop closing without cylindrical regularity. Superscript** means the algorithm runs loop closing leveraging the cylindrical regularity. . . . . 76

4.5 Real-world experimental results. A: Aerial view of the manholes (dots) and two sewer pipe locations (blue lines) corresponding to Experiment 1 and Experiment 2. B: Sewer pipe inspection rover. C: View inside the sewer pipe from the rover camera. D: The top view of the point cloud created by DSO on Data 1. E&G: Comparison of estimated travel distance and ground truth over time for pipes 1 and 2, respectively. F&H: Travel distance error over time for pipes 1 and 2, respectively. I: Cylinder detection using the Benchmark method. J: Cylinder detection using our proposed method in Algorithm 1. . . . . 79

5.1 The point cloud in the manhole . . . . . 87
5.2 Terms and notations . . . . . 91
5.3 Descriptors of Vertex B . . . . . 93
5.4 Three descriptors of estimated Vertex $V_B$ . . . . . 93
5.5 Vertices in the same coordinate system . . . . . 96
5.6 The estimated map and prior map . . . . . 97
5.7 Transformed vertices in the prior map . . . . . 100
5.8 Transformed vertices in the estimated map . . . . . 101
5.9 Simulation environment and the trajectories of the robot . . . . . 106
5.10 Map before Map Merging . . . . . 107
5.11 Map after Map Merging . . . . . 108
5.12 Initial match for Feature map 1 . . . . . 111
5.13 Initial match for Feature map 2 . . . . . 111
5.14 Initial match for Feature map 3 . . . . . 111
5.15 The estimated map after map alignment . . . . . 112

# List of Tables


3.1 Performance of algorithms with data from several incomplete cylinders . . . . . 52
3.2 Performance of algorithms on data from cylinders of different sizes . . . . . 53
3.3 Performance of algorithms on data of different uncertainty . . . . . 55
3.4 Performance of algorithms on data with outliers from different distributions . . . 57
3.5 Performance of algorithms on data with varying numbers of outliers . . . . . 59
3.6 Performance of algorithms on synthetic data 2 . . . . . 60
3.7 Performance of algorithms on real data . . . . . 60

4.1 Comparison between the SLAM algorithms on synthetic data . . . . . 77
4.2 Comparison between the SLAM algorithms on real data . . . . . 78

5.1 candidate matching vertices for the transformed vertices in the prior map . . . . 101
5.2 candidate matching vertices for the transformed vertices in the estimated map . 101
5.3 Vertices and their corresponding potential matches . . . . . 101
5.4 Matching sets for the extension . . . . . 102
5.5 The results for map merging . . . . . 109
5.6 The results for map alignment . . . . . 110

# Chapter 1

# Introduction

## 1.1 Background

Pipe networks play an essential role in transporting essential resources, such as water, oil and natural gas. Due to their low cost and efficiency, pipelines are widely found in the cities and various factories. However, pipe networks may suffer from defects, such as clogging, erosion, and leakage. Those problems can lead to unpredictable economic losses, environmental contamination, and personnel casualties [9].

The buried pipe network for water and wastewater has an approximate combined length of 0.8 million kilometres in the UK [10]. Over £250 billion is invested in UK water infrastructure [11]. Because of failures in pipe networks, leakage of water is estimated to reach upwards of 3 billion litres every day in the UK [12]. Nondestructive inspection (NDI/NDT) is an important and necessary task, and it can provide early detection of defects and avoid undesirable results. It refers to various techniques that can evaluate the properties of a material, component, or system without causing any damage to it. This dissertation focuses on the sewer pipe network, which has the potential to be extended to other pipe networks.

Sewer pipe inspection is essential for the necessary detection of blockages and leakages and is challenging due to inaccessibility and the massive number of pipes. Pipes are always buried or installed at elevated locations. Inspections are commonly performed by inserting a manually operated, tethered probe or rover inside the pipe equipped with a closed-circuit television (CCTV) system for observing faults. However, autonomous, untethered robots have a huge potential to perform long-term, persistent pipe health monitoring. This is because they can navigate complex pipeline networks without the constraints of physical tethers, allowing them to reach areas that are otherwise inaccessible. Additionally, their autonomy enables continuous operation over extended periods, reducing the need for frequent human intervention. Nowadays, fault analysis of piping systems relies on manual detection, which requires many inspectors who do repetitive jobs for long stretches of pipe. The pipeline condition might be interpreted

subjectively without experienced operators [9]. Therefore, a technology that can inspect and localise damages in a large-scale pipe network continuously and autonomously in real-time is required. One of the main challenges is to localise detected damages in an unknown pipe network in real time.

With the rapid development of various range sensing instruments, the three-dimensional (3D) reconstruction method is one of the methods applied to automatic pipe inspection. The obtained model offers an enhanced representation of the environment, which can easily help localise the objects of interest. With the measurement from range sensors: laser scanners, CCTV, pipe sonars, ultrasonic, etc., a 3D model of the inner pipe surface can be obtained [13]. The laser-based methods and camera-based methods are well developed, and lasers usually are more accurate than cameras with extra light illumination. However, active sensors could lead to explosions inside flammable environments. It is notable that there may be a large amount of methane in the sewer pipes. Including structural information of pipes, a camera can also provide detailed textured information of pipes [14]. In terms of cost, a camera is really inexpensive compared to other sensors. Visual 3D reconstruction is commonly used in pipe inspection.

This research focuses on developing methods, approaches, and a system that can localise objects of interest in an unknown pipeline network and build a 3D model simultaneously when the robot goes on patrol in the pipe autonomously. The robot needs to estimate the trajectory within the map as well as its pose, which generally includes three position coordinates and three rotation angles. The input to the algorithm is continuous images, with the output being the camera's relative displacement from the starting point. The algorithm cannot estimate the absolute camera pose and the scale factor. A scale factor is the proportional ratio between the constructed map and the corresponding real-world environment. This process is commonly referred to as Simultaneous Localisation and Mapping (SLAM) [15, 16, 1]. The following will detail these five key features:

1. The robot operates in an unknown pipe network, with limited prior knowledge of the layout and pipe dimensions. The robot must explore and gather this information as it navigates the pipes, often encountering obstacles that complicate its movement.

2. The robot can move freely as long as it captures sufficient image features. Its motion may not always be parallel to the pipe axis, and it could move laterally or follow a zigzag path, which enhances coverage and improves anomaly detection.

3. Relying on a single robot to explore large-scale networks is impractical. The system should be capable of merging multiple maps with overlapping sections.

4. A rough prior map of the pipe network is available, and the algorithm must localise the explored areas within this map.

5. The system should be able to estimate the scale factor to help operators understand the actual dimensions of the model.

Vision-based 3D reconstruction techniques, such as SfM and SLAM, have been widely used for building models of unknown environments and localising cameras [17, 18, 19, 20, 21, 22, 23]. They have similar theories and can be classified into feature-based [1] and direct methods [15]. There are many existing works in this area. However, pipe networks have numerous unique challenges that severely impact the accuracy of the algorithms, rendering them inoperable in some cases.

The pipe is narrow and long, and vSLAM has suffered from severe scale drift in this environment [24]. Detailed information can be found in the experimental results in Section 4.8. The direct method [15] exhibits even more scale drift when using real data than the feature-based method. As the robot goes through the pipeline, the difference between the estimated and actual travel distances increases significantly, and the newly generated point clouds gradually shrink compared to the previously generated ones. After the robot traverses the pipeline, the point cloud generated by vSLAM resembles a cone rather than a cylinder, as shown in Figure 4.1.

A more significant challenge is that the algorithm may fail to extract features within the pipe, as demonstrated in Figure 1.1. This study tested the open-source vSLAM algorithm on data from various pipes [4]. If the algorithm fails to extract features, feature-based methods cannot complete initialisation. Both strong and weak lighting conditions hinder feature extraction, as illustrated in the left figure (see Figure 1.1). Under strong lighting, a noticeable glare spot appears in the image, obscuring the textures on the pipeline walls. Conversely, under dim lighting conditions, the textures in the image become blurred and faint.

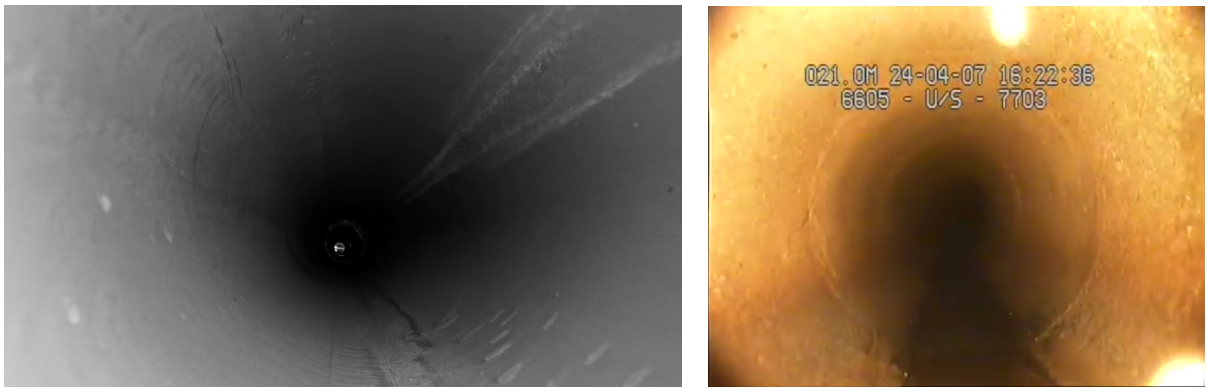



Figure 1.1: Challenging environment: Feature-less pipe wall and varying lighting [4]

Repetitive textures, common in newer pipelines, further complicate vSLAM performance [4], as shown in the left image of Figure 1.2. The camera moves forward at a constant speed inside the pipeline, transitioning from the manhole into the pipe with repetitive textures. While the proposed vSLAM algorithm can operate in the pipe with repetitive textures, the estimated tra-

jectory deviates dramatically from the actual trajectory. The right image of Figure 1.2 shows the interface of ORB-SLAM2. Blue squares represent the camera poses, red dots represent the point cloud, and purple circles indicate detected cylinders. Initially, the camera poses are evenly distributed along its trajectory, suggesting that the estimated speed of the camera is approximately constant. Toward the end, the blue squares become densely clustered, indicating that the estimated speed of the camera approaches zero. The corresponding point cloud also deviates from the cylindrical surface, and the system can no longer detect the cylinder based on the point cloud. The repetitive textures result in significant errors.

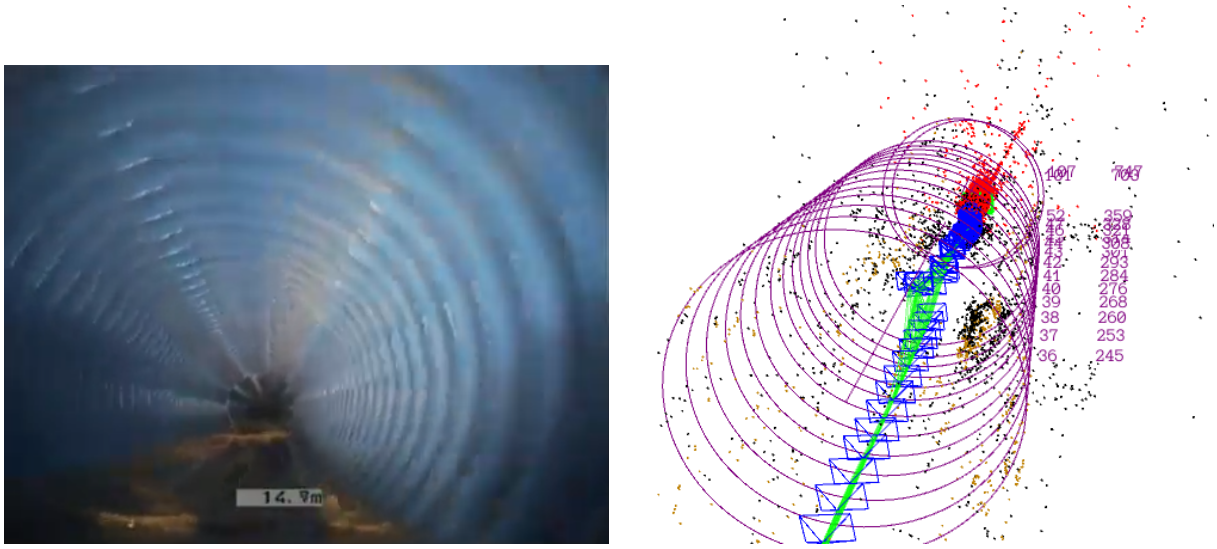



Figure 1.2: Challenging environment: Repetitive texture[4] and Results [5]

In essence, a pipe network is a challenging environment for vision-based algorithms. Factors like lighting conditions, textures, and the inherent structure of the pipeline make accurate modelling and localisation particularly demanding. Hence, this thesis aims to find solutions to overcoming these challenges.

## 1.2 Vision-based Pipe Inspection Methods

A wide variety of pipe inspection methods exist and show some good results. The majority of vision-based methods focus on SfM [25], but those methods cannot perform in real time. While SLAM algorithms can reconstruct pipes in real time, their results are less accurate than some offline methods [26].

Many SfM algorithms designed for pipeline environments leverage the cylindrical shape of the pipes to improve the accuracy of pose estimation and mapping [27, 28, 29, 30, 31, 26]. These algorithms can use the cylindrical information to create point clouds [27, 28], constrain point clouds [30], or calculate scale factors [30, 31]. Some studies assume that the cylinder parameters are known [27, 28, 31], while others estimate them based on the point cloud data [14, 30, 26].

Cylinder extraction methods enable SfM or SLAM systems to utilise cylindrical information flexibly. However, only a few algorithms consider obstacles inside the pipes, and their cylinder extraction method is not robust enough [14, 30, 26].

One of the key challenges for vSLAM leveraging cylindrical information is cylinder detection in a noisy point cloud. Some cylinder detection methods used in SLAM algorithms do not work well in a complex environment [30, 26]. Shakarji et al. [32] and Lukacs et al. [33] fit a cylinder in a point cloud by minimising the point-cylinder distance, but these approaches are sensitive to noise and outliers [34, 3]. Lopez-Escogido et al. [35] used RANdom SAmple Consensus (RANSAC) [36] and a direct method, but this approach may perform poorly when the uncertainty is unknown.

Usually, there will be a layout of the pipe network from infrastructure companies, which provides critical information for understanding its configuration, performance, and maintenance requirements. Specifically, it contains locations of manholes and pipes. It can be regarded as a prior map for SfM or vSLAM algorithms, but research incorporating this knowledge is not well developed.

In this thesis, a robust cylinder fitting method is produced and will be used in a visual SLAM algorithm designed for pipe inspection. The visual method can leverage cylindrical information and a prior map showing the distribution of the pipe network. The scale factor can be recovered by incorporating a prior map.

The research presented in this thesis has several key objectives, which aim to address the challenges and improve the accuracy of vSLAM in pipe environments. These objectives include:

1. To detect cylinders robustly among map points, including outliers and those from the obstacles.

2. To design a visual SLAM algorithm that can leverage cylindrical regularities in pipe networks.

3. To merge maps with overlapping obtained in the pipe networks

4. To leverage a prior map to get the location of explored pipe networks in the real world

5. To test the algorithm on both true data and synthetic data provided by a virtual environment on Robot Operating System (ROS) [37].

The remaining part of this dissertation is structured as follows. Chapter 2 reviews the literature and introduces related research, including SLAM algorithms, SfM algorithms, cylinder detection methods, map merging methods and topological map alignment methods. The key contributions are presented in Chapters 3, 4 and 5. Finally, Chapter 6 discusses the work and concludes the whole thesis.

## 1.3 Key Contributions

The main contributions of this dissertation can be summarised as follows:

**Chapter 3** presents two cylinder extraction methods and a novel evaluation method. The evaluation method can recognise the numerical instability conditions that could mislead the cylinder fitting methods, which rely on point-cylinder distance.

**Chapter 4** proposes a vSLAM approach that leverages cylindrical information in triangulation, bundle adjustment (BA) [38] and pose graph optimisation. The approach can correct the scale factors when the isodiametric information is available.

**Chapter 5** proposes an approach for creating a map of pipe networks. This method can merge the maps created by different robots. Further, it can estimate the scale factor and absolute position of the estimated map by leveraging a prior map.

## 1.4 Publications

The author's publications associated with the thesis are outlined below:

### Peer reviewed journal papers

J1 Zhang, R., Worley, R., Edwards, S., Aitken, J., Anderson, S.R. and Mihaylova, L., 2023. Visual Simultaneous Localisation and Mapping for Sewer Pipe Networks Leveraging Cylindrical Regularity. *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3406-3413, June 2023.

J2 Aitken, J.M., Evans, M.H., Worley, R., Edwards, S., Zhang, R., Dodd, T., Mihaylova, L. and Anderson, S.R., Simultaneous Localisation and Mapping for Inspection Robots in Water and Sewer Pipe Networks: A Review, *IEEE Access*, 9, pp.140173-140198, 2021.

### Peer reviewed conference proceedings

C1. Zhang, R., Evans, M.H., Worley, R., Anderson, S.R. and Mihaylova, L., Improving SLAM in Pipe Networks by Leveraging Cylindrical Regularity, In: Fox, C., Gao, J., Ghalamzan Esfahani, A., Saaj, M., Hanheide, M., Parsons, S. (Eds), *Lecture Notes in Computer Science from Towards Autonomous Robotic Systems (TAROS 2021)*, vol 13054, Springer, Cham., pp. 56-65, 2021, September, Cham: Springer International Publishing.

### Additional peer reviewed journal papers

J3. Edwards, S., Zhang, R., Worley, R., Mihaylova, L., Aitken, J. and Anderson, S.R., 2023. A Robust Method for Approximate Visual Robot Localisation in Feature-Sparse Sewer

Pipes, *Frontiers in Robotics and Artificial Intelligence*, 10, p.1150508.

# Chapter 2

# Literature Review

SLAM in pipe networks is an important task that has yet to be widely studied, in contrast to mobile robot SLAM, which has received significant attention and for which a wealth of methods are available [20]. To fill in this gap, this thesis aims to develop approaches that can cope with the challenges of pipe network environments.

This chapter provides an overview of methods related to the challenges of executing SLAM algorithms in unknown pipeline networks. Section 2.1 introduces popular visual SLAM, visual-inertial SLAM, and structure SLAM algorithms, which estimate trajectories based on sensor data while simultaneously constructing sparse maps of the surrounding environment. Section 2.2 presents vision-based algorithms specifically designed for pipe networks. Given the cylindrical shape of pipelines, most algorithms [30, 31, 26] exploit this cylindrical information to enhance performance. Cylinder detection is a key task for algorithms leveraging cylindrical information. Section 2.3 compares different cylinder fitting and extraction algorithms and discusses various cylinder representation methods. Independently exploring large-scale pipeline networks can be challenging for a single robot, and multi-robot SLAM systems are required to merge maps constructed by different robots. A prior map is often available for a pipe network, typically in the form of a two-dimensional (2D) topological map that includes manholes and pipes. Leveraging the prior map is a logical approach. Section 2.4 outlines map merging techniques and graph matching methods.

## 2.1 Robot Simultaneous Localisation and Mapping

SLAM is a technique that enables an agent equipped with sensors such as cameras, sonar, lasers, or radar to simultaneously estimate its movements and reconstruct the environment in an unknown environment [20]. It is a fundamental technology for autonomous mobile robots, autonomous driving, and virtual or augmented reality, and it has attracted numerous researchers. Sensor data fusion and structure regularity provide more information for estimation, improving

accuracy and offering more observability. This section presents monocular visual SLAM, sensor data fusion and structural SLAM.

This section begins by defining notations, which are used throughout the thesis. The symbols $(\cdot)_w$ and $(\cdot)_{c_k}$are the world frame and camera body frame when taking the k-th image. 3D landmark i and 2D observation in the frame k are defined as $\mathbf{p}^i_{c_k}$ and $f^i_{c_k}$ respectively. $\mathbf{r}^i_{c_k}$ is considered as a ray of light from frame $c_k$ to the 3D landmark i. This thesis uses rotation matrix $\mathbf{R}^i_{c_k c_i}$ and translation $\mathbf{t}^i_{c_k c_i}$ to represent transformation $\mathbf{T}^i_{c_k c_i}$ between frame $i$ to frame $k$. Here $\theta^i_{jk}$ is the parallax angle of feature $i$ between frame j and k.

### 2.1.1 Preliminaries

The typical SLAM problem is formulated by representing the history of robot poses $\mathcal{X}_{1:k}$, up to the current time-step $k$, together with the map $\mathbf{m}$, as the joint probability distribution

$$p\left(\mathcal{X}_{1:k}, \mathbf{m} | \mathcal{Z}_{1:k}, \mathcal{U}_{1:k}, \mathbf{x}_0\right), \tag{2.1}$$

where $\mathbf{x}_0$ is the initial robot pose, $\mathcal{U}_{1:k}$ is the history of control inputs and $\mathcal{Z}_{1:k}$ is the set of all observed map landmarks, respectively

$$\mathcal{X}_{1:k} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}, \tag{2.2}$$

$$\mathcal{Z}_{1:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}, \tag{2.3}$$

$$\mathcal{U}_{1:k} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}. \tag{2.4}$$

The robot pose $\mathbf{x}_k$ can be defined in 3D space as

$$\mathbf{x}_k = \left(X_k, Y_k, Z_k, \theta_k, \phi_k, \psi_k\right)^T, \tag{2.5}$$

where $(X_k, Y_k, Z_k)$ defines the location of the robot in 3D space in the world coordinate frame and $(\theta_k, \phi_k, \psi_k)$ defines the orientation in terms of pitch, yaw, and roll, also in the world coordinate frame.

The map, $\mathbf{m}$, can be specified by the spatial locations of recognisable features or landmarks detected in the world, where

$$\mathbf{m} = \left(\mathbf{l}_1, \ldots, \mathbf{l}_{n_m}\right)^T \tag{2.6}$$

and $\mathbf{l}_i = (l_{i,x}, l_{i,y}, l_{i,z})^T$ is the location of the $i^{th}$ landmark in the world coordinate frame, and $n_m$ is the number of map features. The map can also be represented in a more dense form, e.g. as a grid map. The map is assumed to be static throughout this thesis, therefore $\mathbf{m}$ is not a function of the time-step $k$, in contrast to the robot pose.

The robot state can be updated via

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k) \Longleftrightarrow \mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{z}_k) + \mathbf{w}_k, \tag{2.7}$$

where the $f(.)$ function defines the state transition, $\mathcal{Z}_k$ is the set of current observed map landmarks, and $\mathbf{w}_k$ is state noise and assumed to be Gaussian, zero-mean, white noise, with covariance $\mathbf{Q}_k$, i.e. $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$.

Next, an observation $\mathbf{z}_k$ of a map feature can be related to the state via the measurement function $h(.)$,

$$p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \Longleftrightarrow \mathbf{z}_k = h(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k, \tag{2.8}$$

where $\mathbf{v}_k$ is observation noise that is assumed to be Gaussian, zero-mean, white noise, with covariance $\mathbf{R}_k$, i.e. $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{R}_k)$.

### 2.1.2 Visual Simultaneous Localisation and Mapping

Many researchers have made contributions to vSLAM, with a lot of excellent works [39, 23, 21, 19, 20, 22]. Nister et al. introduced a visual odometry to estimate camera motion [40]. Based on the visual residual model, these methods can be classified as features-based methods and direct methods [16, 41, 15, 2]. Feature-based methods extract and match features in each frame, estimate camera poses, and triangulate features in accordance with multi-view geometry. These estimations are optimised by reducing the reprojection error [1]. Direct methods operate at the pixel level. They detect patches in each frame and track them according to optical flow. The camera poses and inverse depth of patches are provided through the minimisation of the photometric error [15]. Feature-based methods have been broadly adopted in many applications due to their stability, while direct approaches are more robust in texture-less environments [22]. However, direct methods cannot identify previously visited places without features [42].

Generally, feature-based SLAM algorithms consist of four parts: front end, back end, mapping and loop closure. The difference between visual odometry and vSLAM is the loop closure, also known as global optimisation [1].

1. Front end computes the camera poses. The front end aims to find constraints which will be used in the back end.

2. Back end triangulate features and further optimise frame poses and landmarks using Bundle Adjustment [38]. Current methods estimate the robot state using a filter or solving a nonlinear optimisation problem.

3. Mapping constructs landmarks.

4. Loop Closure is used to find long-term data associations. It extracts features in the current frame, trying to associate the same features in previous frames. Once the association is validated, the system incorporates this constraint to pose graph optimisation.

The feature-based methods share a similar structure, which is shown below:

1. Initialisation: The estimation process begins with initialization, which plays a crucial role in overall accuracy. Since depth information is unavailable at this stage, the system estimates camera poses and constructs map points based solely on images. It matches features and recovers relative poses using epipolar constraints [43], followed by triangulating features to serve as references for pose estimation of subsequent frames.

2. Tracking: After initialisation, which builds some map points, the system estimates camera pose based on perspective-n-points (PnP) [44]. If it is selected as a keyframe, it will be further optimised with several keyframes in local optimisation.

3. Loop closure: When the system receives a keyframe, it will check the data association with previous frames. Once a loop is detected, the current keyframe is inserted into the loop and optimised with neighbours in local optimisation. Then, a pose graph optimisation is performed over all the keyframes in this loop. In some algorithms, a global Bundle Adjustment will be used to optimise all related camera poses and landmarks.

This thesis mainly focuses on feature-based methods since they are more robust compared with direct methods in pipe environments, as revealed by testing different algorithms (refer to Section 4.8). Some classic visual SLAM algorithms are introduced below.

Davison et al. proposed the first real-time monocular vSLAM algorithm, MonoSLAM [45], which is a feature-based sparse method, using Shi and Tomasi detectors [46] to find oriented planar textures as features. The system requiring prior knowledge to initialise can also solve the scale factor. After initialisation, the system enters the typical predict-measurement-update tracking sequence. In a time step, the system predicts robot movements with a constant velocity and angular velocity, which will be updated by an extended Kalman filter in the back end. It assumes the position of each feature follows a Gaussian distribution, and after a peaked depth probability density function (PDF) is obtained, the feature is initialised. Although the system has limited landmarks and loses track of features easily, it was still remarkable work at that time.

Parallel Tracking and Mapping (PTAM) is an important work, innovation of which can still be widely found in recent SLAM algorithms. This is the first SLAM algorithm that processes tracking and mapping in parallel [47, 1]. The concepts of front end and back end in SLAM algorithms were developed out of this idea, followed by numerous SLAM algorithms. It is worth noting that PTAM is the first algorithm that optimises estimation by minimising a nonlinear cost function. It optimises the map points with only selected frames, which allows it to spend more computation on keyframes. A FAST detector in the front end is used to extract patches as features. Feature matching is achieved by comparing their zero-mean sum of squared distance (SSD) scores. Since this algorithm is designed for a small environment, it does not have loop

closure.

ORB-SLAM, as a classic feature-based SLAM algorithm, has gained much attention from researchers. This open-source SLAM System is already developed from ORB-SLAM to ORB-SLAM3 [48, 1, 2]. ORB-SLAM3 is a VI-SLAM algorithm, which will be introduced later. Compared with previous SLAM approaches, ORB-SLAM2 had many advantages:

1. ORB-SLAM2 is able to perform SLAM with monocular, stereo, and RGB-D cameras.

2. ORB features are employed throughout all processes. ORB is an efficient and invariant feature and a good trade-off between computation and accuracy. After detecting features, a quadtree is used to discard features crowded in a small area, which can distribute features evenly in the images.

3. ORB-SLAM2 use three threads to perform tracking, local optimisation, and global optimisation. A co-visibility graph is used to select frames in local optimisation, which can limit computation in a large-scale environment. Similarly, an Essential Graph is used in global optimisation.

4. In the algorithm, a generous keyframe selection method and a restrictive keyframe culling method are adopted. The former can avoid losing information, while the latter can keep the system compact.

In addition, the ORB-SLAM2 also has some drawbacks. It extracts features in each image and matches them based on descriptors or Bag of Words (BOW) [49], which is computationally expensive. During initialisation, ORB-SLAM2 fixes the first frame as the reference frame and initialises the system between the reference frame and the current frame. If the number of matches is less than the threshold, then the current frame is set as a reference frame. If the initialisation fails, the system waits for the next frame [2]. It only considers a few possible choices, making the system less robust. Figure 2.1 shows the interface when the ORB-SLAM2 algorithm is running in the real sewer pipe [6]. The details of the data can be found in Subsection 4.8.2. The left part of the figure illustrates frame poses (blue squares) and map points (red dots). The right part shows the feature tracking in the current frame. Some map points are far from the surface of the pipe, which demonstrates the inaccuracy caused by low parallax angles in the pipe environment.

Engel et al. proposed a large-scale direct monocular SLAM method (LSD-SLAM), which marks the success of the monocular direct SLAM method [41]. However, it relies on an appearance-based mapping algorithm [50] that uses a feature-based method to detect loop closure. After that, the author published a direct sparse odometry (DSO) [15]. Compared with LSD, DSO has higher robustness and accuracy [15, 48]. DSO, which does not have an explicit front end

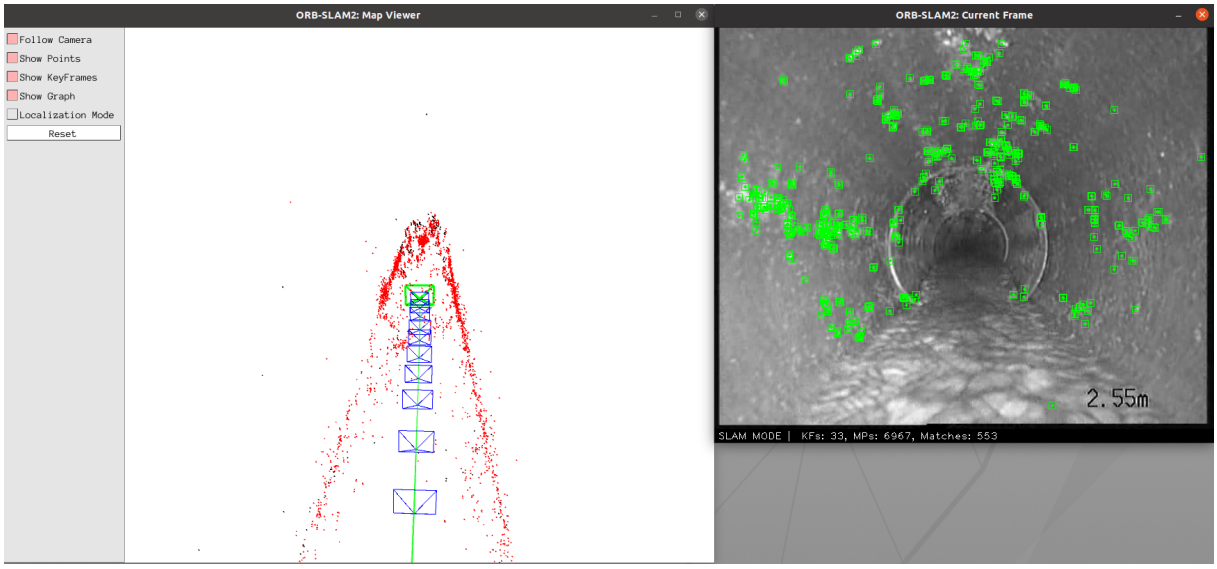



Figure 2.1: Interface of ORB-SLAM2 [5] running with real data [6] from a sewer pipe in the UK

and back end, estimates camera poses and landmarks (inverse depth values) jointly by minimising a non-convex energy function. A sliding window is used to limit the number of frames in optimisation and marginalise the old frames and the corresponding map points. In order to avoid inconsistency caused by the marginalisation, the technique called First Estimate Jacobians (FEJ) is used [51]. FEJ shows a significant advantage in computation without compromising accuracy. It is sensitive to illumination changes and geometric noise, which is also shared by other direct methods. Moreover, the public implementation of DSO lacks readability compared to ORB-SLAM2 and SVO [16], which makes it difficult for other researchers to extend it. The interface of DSO running on the real data of a sewer pipe [6] is shown in Figure 2.2. Similar to interfaces of ORB-SLAM2, the interface displays the point cloud (grey dots) and the frame pose (red square) in real-time.

Forster et al. proposed a semi-direct visual odometry (SVO) [16] that uses photometric errors and geometric constraints to estimate camera poses and uses a Bayesian depth filter to estimate the depth of landmarks. The system extracts FAST corners [52, 53] as features and employs the optical flow to track them. It has an initialisation similar to feature-based SLAM methods. After that, the algorithm has three steps to estimate the camera poses. In the first two steps, the system estimates the initial camera poses and corrects the patch correspondence separately by minimising the photometric error. In the final step, the bundle adjustment is used to optimise the camera poses and position of map points sequentially. The depth filter assumes that the depth of features satisfies a Gaussian distribution. It extracts features from keyframes and uses the information from frames to update the distribution. Once the distribution converges, the corresponding features are added to the map. The main advantage of this algorithm

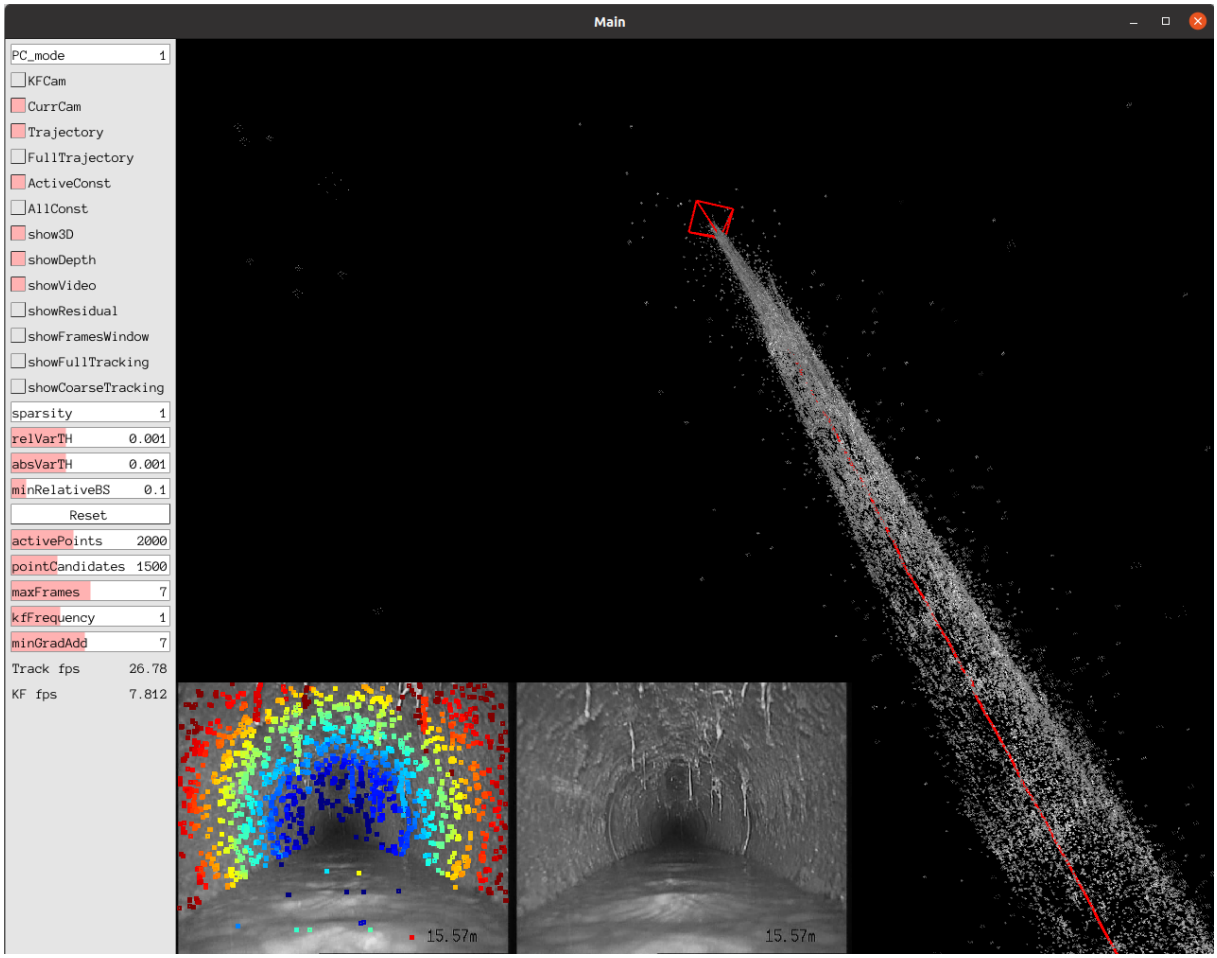



Figure 2.2: Interface of DSO [7] running with real data [6] from a sewer pipe in the UK

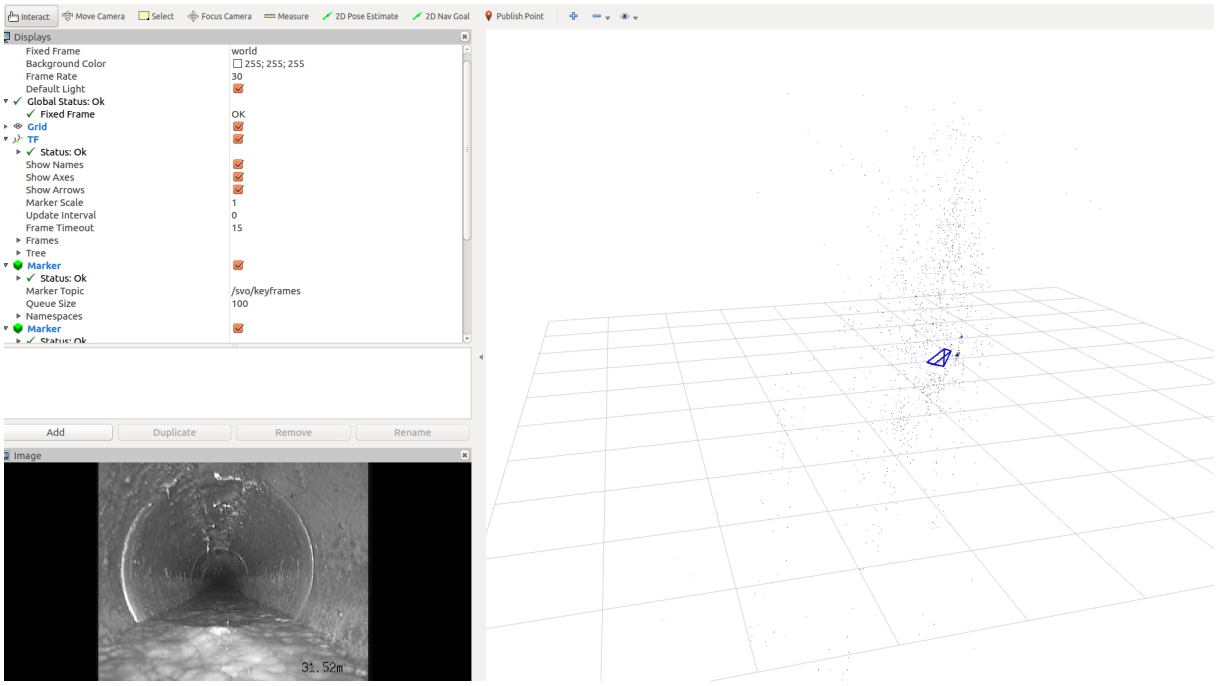



Figure 2.3: Interface of SVO [8] running with real data [6] from a sewer pipe in the UK

is its high efficiency. In terms of drawbacks, it also inherits the disadvantages of other direct methods, such as sensitivity to illumination changes and geometric noise. Moreover, estimating the current camera pose only depends on the last frame, making the system less robust. This algorithm is designed for Uncrewed Aerial Vehicles (UAVs) with a downward camera; some sets are unrealistic for other applications. For instance, instead of a fundamental matrix in initialisation, it computes a homography matrix, which assumes the features are distributed over a plane. When selecting keyframes, the system only considers the translation of the camera, excluding rotation. The SVO algorithm cannot correctly estimate the pose in a pipe environment. Figure 2.3 shows the interface of SVO running on the data [6] from a real sewer pipe in the UK. It presents the current frame pose (blue square) and point cloud (grey dots), which are incorrect due to the unsuccessful initialisation. The point cloud distributes around the blue square and does not look like a pipe.

### 2.1.3 Sensor Data Fusion

#### Visual Inertial Measurement

With the development of different sensors, sensor data fusion technologies have become popular in SLAM, e.g. [54, 55]. Due to the complementarity of cameras and Inertial Measurement Units (IMU), many researchers have developed visual-inertial navigation systems (VINS). With IMU measurement, the scale factor can be estimated. Visual Inertial Odometry (VIO) can be divided into loosely-coupled and tightly-coupled methods in accordance with their sensor data fusion

methods. The former uses camera and IMU measurements to estimate camera poses separately and then fuse their results, resulting in information loss. The tightly-coupled methods, which are more computationally intensive, have more accurate results. They combine visual and inertial measurements to estimate the camera poses, which are applied by more SLAM algorithms.

Mourikis and Roumeliotis proposed a very early feature-based virtual-inertial odometry, multi-state constraint Kalman filter (MSCKF)[56], which has many subsequent variations [57, 58]. It is a tightly-coupled method and uses an extended Kalman filter to estimate the states. Compared with other standard extended Kalman filters, its state vector only includes camera poses and IMU states, excluding features, which reduces the computation. To be specific, the reprojection function is used as the measurement model for updating the state vector. The system removes the features in the measurement model by projecting the reprojection error onto the null space of the feature Jacobian matrix. Another advantage lies in that the system holds several camera poses in a sliding window and optimises them together.

VINS-mono is a very accurate and robust optimisation-based virtual inertial SLAM algorithm [59]. The system tracks features using the Lucas-Kanade algorithm [60], which is slightly more robust than the Binary Robust Independent Elementary Features (BRIEF) descriptor [61]. It has a robust probabilistic optimisation-based online initialisation without prior knowledge of the camera–IMU transformation. Although the system uses a sliding window in optimisation and marginalises previous frames, possibly resulting in inconsistency, FEJ is not used. It detects a loop using BOW [49], and the map can be merged when a loop is detected.

ORB-SLAM3 is the latest version of ORB-SLAM with higher precision than previous methods [2, 62]. The system estimates the robot state with the Maximum A Posteriori (MAP) approach [48], making it more accurate. Besides, it also uses a faster initialisation method. It estimates the camera poses in BA with pure visual information and estimates the robot state by maximising the likelihood with pure inertial measurement. A joint visual-inertial optimisation is performed to refine the solution with results from the previous estimation. This method enables errors to converge more quickly. Another novelty of this method is the multi-map system. When the tracking is lost for 5 seconds, a new visual-inertial map is created and activated. Once the system recognises that it has moved to the location, this map can be merged with a previous map. This algorithm makes the system robust to the large-scale environment. Moreover, it has also improved the place recognition method by involving the neighbour frames in the covisibility graph of the matching frame. With these improvements, the system speeds up its loop detection and improves recall rates.

Based on DSO, an optimisation-based direct sparse visual-inertial odometry (DSVIO) was proposed by Stumberg et al. [63]. The system fuses camera data and IMU data in a tightly-coupled way. In initialisation, the system performs a visual initialisation without IMU data and

then optimises all parameters, including the scale factor in combined energy functional with the selected initial values. It can initialise without waiting until the scale is observable. Another novelty is dynamic marginalisation, which is used in the sliding window to limit the computation. Due to the FEJ [51], the marginalisation prior can only be updated when some variables are removed in the sliding window. However, this could also significantly limit accuracy when the state deviates from the fixed linearisation point. The DSVIO creates three marginalisation priors from the part of the whole factor graph when it removes the old variables. In the optimisation process, once the scale factor changes remarkably, the system will select another suitable marginalisation before melting this trend.

#### Odometry

An odometry, which is a common sensor in a number of ground robots, can measure travelled distance [64, 65, 66, 67]. Wheel odometry performance can degrade due to uncertainty in the interaction with the ground [66]. Eudes et al. used odometer measurement to compute the scale factor for a monocular visual SLAM algorithm [68]. Michot et al. estimated the robot motion by minimising a cost function, including two error terms for two sensors separately [69]. L-curve was used to determine the weights for two error terms [70]. Hidalgo-Carrio et al. trained a Gaussian process model to fit the residual, which is the difference between the ground truth and the odometer estimation. The residual was used to evaluate the performance of the odometer [71]. Nutalapati et al. calibrated the odometer using a Gaussian process model to fit the other estimation, specifically, the result from a visual SLAM algorithm [65]. This algorithm can effectively detect wheel slippage and avoid incorporating the error information into the back end.

Since the odometer measurement exhibits a large drift on the robot rotation estimation [68], it is a natural choice to fuse odometer measurement into the VINS. A VINS performs poorly when the robot is moving at a constant velocity or acceleration. When the IMU does not have enough excitation, the scale could be unobservable[67]. Wu et al. solved this problem by incorporating the odometer measurement and planar-motion constraints. Quan et al. developed a visual-odometric simultaneous localisation and mapping (VOSLAM)

### 2.1.4 Structure-SLAM

Nowadays, many robots work in a specific human-made environment with distinct structural features. In order to improve mapping and localisation in terms of accuracy, some researchers have exploited structural information by building a *Manhattan world* or enforcing structural regularities on landmarks [72, 64].

SLAM algorithms can leverage higher-level features in two main ways. First, the system

discovers specific structures among triangular landmarks and leverages this knowledge at the backend [73, 74, 75]. Second, it abstracts these structural regularities as an underlying world structure, and the new estimated camera poses can be aligned to the world structure through the algorithm [64]. Ideally, the drift of the camera trajectory does not accumulate because the estimated camera poses are no longer dependent on the last frame. It is suitable for environments with consistent structure information, especially human-made environments.

**Structural Regularity**

There are three steps to leverage structural regularities: discovery, incorporation, and measurement [75]. In the first step, some existing triangular landmarks are used to fit specific structures. Gee et al. selected a minimum number of triangular points to fit a line or plane inside a Random Sample Consensus (RANSAC) scheme [36], with only the qualified hypotheses being accepted [75, 76, 77]. These hypothesised plane features can also be obtained by the 3D mesh built by 2D Delaunay triangulation [73, 74, 78].

The detected higher-level features are initialised by linking to existing point or edge features. Gee et al. computed the variance and deviation of inlying features with respect to the objective structure [75]. The features with large variance or deviation are rejected. Rosinol et al. and Li et al. built histograms of the distance of points from the triangular patches or lines and employed a Gaussian filter to select qualified features and corresponding point features in a non-iterative way [73, 74].

Finally, these features, including higher-level features, are associated with measurements of the input image used to build error terms in the reprojection error function. The system imposes constraints on these point and edge features related to the higher-level features. The camera poses and landmarks are optimised by minimising the new cost function. Upon detecting structures, the algorithm introduces a variety of structural constraints. In a broad sense, there are three types, namely, point-to-plane distance [79], reprojection error [80], and clustering constraints [81]. The point-to-plane distance penalises the distance of feature points from the corresponding structures. The algorithm calculates the intersection of feature back-projected rays with the structure and projects it onto other keyframes. The reprojection error is the distance between the projected point and the feature. It is assumed that feature points belonging to a structure should be adjacent to feature points of the same structure [81]. Therefore, the count of adjacent non-structured points is incorporated as part of the loss function. As the landmarks are updated, the algorithm will reselect the point features associated with the higher-level features.

**Structured World**

In human-made environments, many surfaces and edges are aligned with three dominant directions, and the environments can be abstracted as a set of blocks in a Manhattan world [64, 82, 83, 84]. A structure line is defined as a 3D line parallel with one of the three dominant directions. In image space, projections of structure lines parallel to the same direction intersect at vanishing points in the image. Many methods [64, 85, 86, 87, 88, 89, 90, 78, 79, 84, 83, 74, 91, 77] rely on a fast line segment detector (LSD) [92] and LBD descriptor [93] to detect and track line segments in 2D images. Zhou et al. employed vanishing points to initialise the structure lines and added the reprojection error of endpoints of structure lines in the BA [64]. Li et al. took advantage of parallelism, orthogonality and coplanarity to improve accuracy in a Manhattan world [82]. Due to the weakness of the Manhattan world, Zou et al. abstracted the human-made environment as an Atlanta world. In an Atlanta world, several local Manhattan worlds are in different locations with different dominant directions [89]. Li et al. employed a CNN to obtain dense maps and normal estimates. Based on the assumption of the Manhattan world, the system estimates rotation and translation respectively [83].

## 2.2 Structure from Motion

SfM and SLAM use similar methods to build 3D models, and many existing SfM methods are designed for pipe networks. It is helpful to know how problems are solved in the pipe networks. Image-based 3D reconstruction can be divided into three types: sparse reconstruction, dense reconstruction, and surface reconstruction [13]. A sparse model can be obtained by sparse reconstruction using the methods of SfM. The surface reconstruction and dense reconstruction are carried out on the basis of sparse reconstruction.

The corresponding algorithms can be classified into two categories: direct methods and feature-based methods [17]. The former estimates the camera's motion by minimising the photometric error between two frames. These algorithms directly work on the pixels of images [15]. The feature-based methods employ epipolar geometry to recover the camera's relative motion based on some feature correspondences between two frames in initialisation [43]. With triangular features, the pose of later frames can be estimated by PnP [94, 44]. In order to obtain sparse point clouds, these methods mainly include feature extraction, feature matching, motion estimation and feature triangulation [95]. Although direct methods have advantages over feature-based methods in texture-less environments because they do not require feature detection, feature-based methods are employed in most papers on 3D pipe reconstruction [17, 14, 30, 31, 26]. This is because feature-based methods in SfM have sufficient time to refine feature detection and feature matching since it is unnecessary to perform SfM in real-time. A computationally expensive

feature extraction method, the scale-invariant feature transform method (SIFT) [96] is used by many researchers [14, 26, 13, 29, 9, 97]. It is difficult to run an algorithm with a SIFT detector in real time.

Many researchers [17, 29, 26] have applied a variety of imaging sensors to pipe inspection systems, such as standard cameras, fisheye cameras, panoramic cameras, and depth cameras. Kahi et al. used a regular camera in a straight section of a pipe [14]. Due to the narrow structures of the pipe, when the robot moves along the pipe axis, a standard camera with a limited field of view can only observe the far interior surface. Only the peripheral regions of the images show a clear pipe surface, while the image of the distant surface suffers from strong projective deformations. Therefore, in this environment, a camera with a larger field of view is more suitable. Many researchers apply a fisheye camera so as to obtain a larger field of view [28, 98, 30, 31]. Zhang et al. used a panoramic camera to acquire more complete images of urban drainage pipes [13]. In addition, other imaging sensors, including a monocular camera, are also applied. Stereo cameras can improve 3D reconstruction by setting a fixed baseline between two cameras. Hansen et al. used two cameras facing up the surface of the pipe, which can have sufficient stereo image overlap [99]. Gong et al. proposed an axial-stereo vision system in which orientations of cameras coincide [29]. In order To detect obstacles, Chuang et al. used a depth camera to obtain depth information but also suffered from the limited field of view [9]. Besides, auxiliary equipment is also taken into account. To improve the flexibility of the algorithm, two structured lights were used to optimise radius in the optimisation [30]. The laser spot, the intersection between laser rays and the pipe, is computed based on the current pose of frames, the known radius and the fixed poses of lasers relative to the camera. Similar to bundle adjustment [100], the camera poses, and the radius is optimised by minimising the reprojection error. Esquivel et al. controlled the rolling rotation of the camera based on inertial measurement unit data [27]. However, the drift of IMU data is not mentioned, which can make the algorithm difficult to execute in large-scale environments.

Compared with 3D reconstruction for other objectives, a special structural property is the cylindrical shape of a straight pipe. It is reasonable to apply this knowledge to 3D pipe reconstruction, and some works have already benefited from this property [30, 26], along with the known diameter of triangulation and optimisation. The scene points on the pipes can be obtained by the intersection of a known cylinder and the ray from the camera centre to the observation [28, 17]. A feature can be observed in multiple frames, and the system can estimate its position in each frame. It is not clear which position has been selected in the papers [28, 17]. Another example is that each scene point obtained from normal triangulation has the same distance from the cylinder axis if it is from the interior surface of the pipe. This known property was integrated into optimisation by adding constraints to bundle adjustment [30, 26].

Likewise, Künzel also used this prior knowledge in local pose optimisation, which can optimise the camera poses and triangulate features iteratively; however, the mathematics is not clear [31]. In the bundle adjustment, a hyperparameter is involved to control the weight of the cylindrical constraint.

Before applying this prior knowledge to features, it is necessary to distinguish features from the objective pipe. Only a few researchers have considered this problem. Most assume that all features are extracted from the objective pipe. Misclassification can exert negative effects on cylinder fitting, triangulation, or optimisation in bundle adjustment. With restricted camera movement, two methods have been implemented. Kahi et al. implemented a 3D occupancy grid among the point cloud and used a 3D kernel to detect outliers and defects inside the pipe [14]. In some research, an image-based method was used to classify different pipe segments [30]. The proposed method recognises the pipe segments using shadow images. The first method may not be able to work in real time. Moreover, the restriction may not be satisfied in a complicated and unfamiliar pipeline network.

The majority of the works [27, 28, 17, 30, 29, 31] assume that the robot moves forward parallel to the pipe axis when the system benefits from the cylindrical information. Hansen et al. proposed a triangulation method based on the cylindrical shape of a pipe and camera movement parallel to the pipe axis [17]. In order to triangulate the laser spot, the same requirement is proposed for structured light [30]. A back-projection restricts the robot's movement along the pipe axis [27, 31]. In addition, several researchers simplified the pose estimation by assuming the robot moves in a straight line [14, 28, 98]. Kagami et al. lifted the restriction on the camera movement since they used a robust cone detection method among the triangular features [26]. In this method, the RANSAC algorithm was used [36] to extract cones in the point cloud without prior knowledge of the pipe axis [35]. Since cylinder detection is sensitive to intrinsic calibration, Kagami et al. chose to fit a cone instead of a cylinder [35].

## 2.3 Cylinder Detection

Cylinders are a common geometric structure in human-made environments, making cylinder fitting essential for numerous point cloud processing applications. Many researchers have proposed all sorts of methods to estimate cylinders within point clouds [101, 102, 35]. This section provides an overview of cylinder fitting and extraction methods. In a broad sense, these methods can be classified into two categories: cylinder fitting and cylinder extraction [103]. Cylinder fitting entails calculating the best parameters from the point cloud, typically by minimising the point-cylinder distance to determine cylinder parameters. Point-cylinder distance is the distance of points from the cylinder surface. Cylinder extraction involves the identification of cylindrical

points and estimating cylinder parameters within complex point clouds. This task becomes more challenging due to potential issues such as incompleteness, noise, and outliers, with no guarantee of reliable point normals.

### 2.3.1 Cylinder Fitting

Cylinder fitting methods [33, 104] are designed for complete data, operating under the assumption that the cylindrical object is fully captured. Cylinder fitting algorithms calculate cylinder parameters based on the point cloud provided. These methods estimate cylinders by minimising point-cylinder distances using both linear and nonlinear approaches [101, 33, 104]. These methods assume the presence of a single cylinder within the scene and are sensitive to outliers. Most are based on least squares or Singular Value Decomposition (SVD).

Some researchers proposed direct methods based on the geometry [101, 104]. Taubin proposed a cylinder representation and a direct method for estimating cylinder parameters with five points [101]. Beder and Förstner introduced several direct methods requiring only a few points to estimate cylinder parameters. Direct methods have low computational overhead but high sensitivity to noise [104].

Optimisation methods, which have been widely used to estimate the cylinder parameters, can use information from all point clouds [32, 33, 105]. Shakarji and Lukács et al. used an optimisation method to estimate cylinder parameters by reducing point-cylinder distances [33, 32]. Shakarji employed a general representation, imposing constraints on positional parameters during optimisation [32]. The technique defined the position of the cylinder axis using a point closest to the origin. Lukács et al. avoided this issue by adopting a novel representation method [33]. Faber and Fisher employed a constrained Euclidean fitting method [105]. Those methods assume that the noise follows a Gaussian distribution, making it sensitive to outliers. Hypothesising that errors follow a Gaussian mixture distribution, Paláncz et al. adopted a Maximum Likelihood Estimation method for cylinder parameter estimation [102].

A two-step approach can simplify the problem by estimating the direction and the rest of the parameters, respectively. Some researchers have used Principal Component Analysis (PCA) to estimate the axial direction of the cylinder and performed circle fitting on projected points so as to estimate the position and radius of the cylinder [106, 34]. These methods reduce the complexity of cylinder fitting by splitting it into two steps. Nevertheless, PCA is also sensitive to outliers, with a requirement on the length-to-radius ratio of the target cylinder. Some methods [107, 108], which estimate direction according to point normals, can suffer from poor-quality normals. On the whole, error can propagate from the first to the second step [109].

### 2.3.2 Cylinder Extraction

Cylinder extraction is used to detect a cylinder in a much more challenging environment, where the data could be incomplete and noisy and include outliers. The RANSAC method [36] has been employed for cylinder estimation in environments with outliers, where points from the point cloud are randomly selected to estimate cylinders. It repeats multiple times and obtains many estimated cylinders. The method compares the estimated cylinder estimated from different subsets, choosing the one supported by the most points. Lopez-Escogido and Fraga adopted the RANSAC method [36] and direct methods to estimate various geometric structures [35].

RANSAC [36] is a simple and iterative algorithm designed to handle scenarios with a significant number of outliers. RANSAC algorithm initially generates multiple cylinders based on random minimum subsets of points, and the cylinders are then verified within the whole point cloud data. A score will be calculated for each model during the verification by counting the points with point cylinder distances lower than their uncertainties. The RANSAC algorithm runs enough times to ensure that the correct model is produced. It finds the best model in accordance with the number of inlier points. However, due to its loss functions, the direct method yields biased solutions for a cylinder. Since the uncertainties are unknown, the algorithm may not correctly select the inliers, which can lead to false results.

In [110], the authors adopted an iterative single-cylinder fitting method, cyclically optimising cylinder parameters and selecting inliers for the next iteration. Within the current set of inliers, PCA is used to estimate the axis of the cylinder, and a circle fitting method is used for other parameters. Although PCA is sensitive to outliers, Nurunnabi et al. proposed a more robust method [3]. Robust Principal Component Analysis (RPCA) [111, 112] was employed to estimate the direction of the cylinder axis, and the cylinder's position and radius in projected points were then estimated using a robust circle fitting method. The robust circle fitting algorithm adopts the RANSAC method [36] to obtain a good initial estimate prior to refining the cylinder parameters by minimising the point-to-circle distances for a subset of points.

Another class of methods is the Hough-based method, which selects the optimal parameters supported by the most points [107, 113]. The Hough method [114] uses designed voting spaces to extract primitives. Cylinders have five degrees of freedom, making the application of Hough impractical due to computational intensity. Rabbani and van den Heuvel extracted cylinders in two steps to reduce the dimensionality of the voting spaces, estimating the axial direction and the rest of the parameters separately [107]. Patil et al. proposed a prior segmentation and an area-based adaptive Hough transform method [113].

### 2.3.3 Cylinder Representation

An important aspect of vision-based robot SLAM methods in pipes is the representation of the shape of the pipes, e.g. elliptical or cylindrical. Cylinder representation is essential to extract the cylinder from the overall image. Over-parameterised approaches will over-complicate the optimisation process. A cylinder is a surface that revolves around an axis, with all points on the surface equidistant to the axis and all the lines parallel. The distance from every point to the considered axis is the radius. It can be described as an axis in a 3D space with a radius. Its orientation can be fully described by the direction of its axis, which requires two angles to specify in 3D space. The spatial position of the cylinder can be represented by a point on its axis, which is defined by three coordinates. However, because the cylinder is infinite in length, any point along its axis can describe its position equally well, meaning that translation along the axis does not contribute to the degrees of freedom. Thus, the cylinder's spatial position is determined by two translational degrees of freedom. Additionally, the radius of the cylinder provides another independent degree of freedom. Hence, a cylinder only has five degrees of freedom. Typically, a cylinder is described by seven parameters, including a unit vector for the direction, a point on the cylinder axis for its position and a radius [115]. This is overparameterised and will bring constraints to the optimisation or have a negative effect on optimisation.

Some researchers proposed different cylinder representation methods for optimisation. Taubin proposed a cylinder representation with a minimum number in a direct method [101]. Although the first three parameters can be converted to a rotation matrix, the paper does not introduce how to update the cylinder parameters mathematically during optimisation. Shakarji made a complex representation whose corresponding loss function resulted in a biased solution [32]. Hansen et al. came up with a five-parameter representation that can be optimised by reducing the loss function, although the Jacobian matrix of this representation is relatively complicated [30].

## 2.4 Map Merging and Map Alignment

Pipe networks are vast and complex, making the use of multiple robots for exploration practical. When multiple robots create separate maps, each map has its own coordinate system, and the transformation matrices among them are unknown. It is necessary to unify these maps under a single global coordinate system. This task can be accomplished by either merging or aligning the maps. The algorithm identifies the overlapping sections of the maps and calculates the relative transformation matrix accordingly. Finally, map merging or map alignment can be done with the relative transformation matrix.

### 2.4.1 Feature Map Integration

Visual SLAM algorithms mainly generate feature maps that include point, line, and plane features [90, 84]. The algorithms described in the current literature [116, 117, 118] commonly estimate the transformation matrix between two local maps by tracking features using methods like ORB [119], Shi-Tomasi [46], SURF [120], and SIFT [96]. When provided with two maps and their associated videos, pioneer researchers match the features across the videos. These methods employ Iterative Closest Point (ICP) [121] or ImpICP [122] to compute the transformation matrix, starting from two random pairs of correspondences. The most suitable result is then selected using the RANSAC approach [116, 123, 124, 125, 118]. These methods require a lot of computation in a large-scale map.

In [126], the candidate point cloud alignment is achieved by integrating image matching during the point cloud registration process. Specifically, the method builds a similarity matrix between two image sequences and employs a modified version of the Smith-Waterman algorithm [127] to find the matched images. The matrix elements denote the similarity between the corresponding two images. It measures their cosine similarity by the inner dot product of the two image vectors of words. Subsequently, the relative transformation matrix can be solved by Horn [128] among two small local point clouds.

In [117], a similar idea is adopted employing FAB-MAP [129] to find the matched images. FAB-MAP is an appearance-based loop closure detection method and uses a Bayesian framework. Once the matched images are found, Deutsch et al. apply the eight-point method [130] to compute the relative transformation matrix.

Some researchers [131, 55, 132] employ a loop-closure algorithm used by ORB-SLAM2 to determine similar keyframes across maps and use multiple RANSAC iterations of the method of Horn [128] to find the best alignment transform between the two local maps from the matched images. The algorithm calculates the BOW similarity scores among 30 covisible keyframes, and uses the lowest score as a threshold. Then, the similarity scores between 30 keyframes and images from another video are computed. Candidate keyframes are acquired if their scores surpass this threshold. If a candidate keyframe is inadequate, the method performs on other 30 covisible keyframes. Finally, the relative transformation matrix can be obtained by Horn [128]. The result can be further refined by pose graph optimisation and global bundle adjustment.

Bokovoy et al. proposed a method similar to those employed for grid map [133]. The method directly extracts 3D features on the feature-based map or 2D features on the map projected on a plane. Subsequently, these features are matched using descriptors [133]. Then, the RANSAC algorithm [36] is used to find the best transformation matrix.

### 2.4.2 Topological Map Merging

Topological maps are introduced by [134] and demonstrate the connectivity of the environment in a graph structure. Typically, a vertex represents an interesting place in the environment, and an edge map represents specific paths between places. Compared with grid maps and feature maps, topological maps are more abstract. It offers a simple and direct description of the environment. Some topological maps even contain information about the locations of vertices and the lengths of paths. The aforementioned attributes give topological maps a distinct advantage in certain situations. For instance, the maps can be used for path planning, while sparse feature maps cannot. Topological maps encompass Voronoi diagrams, Generalised Voronoi diagrams, annotated Generalised Voronoi diagrams, and extended Voronoi diagrams [135, 136, 137]. In the literature, the data association problem in topological maps known as the maximal common sub-graph problem is an NP-hard problem [138].

### 2.4.3 Graph Matching

Graph matching, which involves calculating the similarity of graphs [138], has attracted many researchers [135, 139, 140, 136]. It includes graph isomorphism, subgraph isomorphism, and Maximum Common Subgraph (MCS) [138]. Two graphs are considered isomorphic if a one-to-one correspondence exists between their vertex and edge sets. Subgraph isomorphism refers to a situation where a subgraph of one graph is isomorphic to another graph. The MCS problem involves identifying the largest subgraph that is shared by both graphs.

The MCS problem is a well-known NP-hard problem, which signifies its computational complexity. But if there is any metric or supplementary information, the problem can be simplified[139]. At first, with metric information, the map has edges encircling corresponding vertices in order, which can filter out some matches against the knowledge. Secondly, the type of vertices can significantly reduce the possible matches among edges. However, the estimated graph may have wrong vertices or edges due to the noise. The graph matching methods need to overcome the problem, which refers to error-tolerant graph matching using graph edit distance [141, 138, 142, 136].

There are many graph matching methods in this area [135, 139, 140, 136, 142]. Dudek et al. explore a topological map with two robots starting from the same starting point and ending at a fixed location [135]. It is assumed that during the process, the robots can record and identify the passing vertices. The algorithms fuse the two maps based on common vertices explored from vertices around the starting point. For some boundary points, the robot can actively deploy a landmark collected by another robot later, thus achieving a match between the two vertices. The assumptions for this method are unrealistic. A similar method uses grid maps to complete the match on the topological map [143]. The robots share grid maps with

each other when rendezvous, and the relative pose can be obtained by maximising the overlap between two local grid maps in optimisation. However, the rendezvous is not guaranteed in reality.

Most graph matching methods rely on the information on vertices [144, 145, 140]. Dedeoglu and Sukhatme built an augmented graph with nodes which are landmarks [144]. According to the types of landmarks, The algorithm first finds candidate vertices matches, which are filtered according to their compass values. The method computes transformation matrices based on candidate matches and selects the best result that has the largest overlap between two maps in the same coordinate. Erinc and Carpin built an appearance based map similar to a pose graph [140]. Given a new map, the method builds a similarity matrix to find the matches. Specifically, it collects all words of features in the first map and checks the features on the target images. If the number of matched features exceeds the threshold, the image correspondence will be verified by the fundamental matrix. The successful image correspondence leads to an increase in the priorities of words of the image, which aims to find more image correspondence [140]. Bonanni et al. match vertices by using a novel point cloud registration on local point clouds. Their method applies NICP [122] on local point clouds from two vertices, which can compute the relative poses and the ratio of inliers to outliers. The match between two vertices is accepted once the ratio exceeds the threshold [145].

Some methods rely on map structure or map geometry [139, 136, 142]. Huang and Beevers find the hypothesised matches among vertices according to the orientation of edges at each vertex and path length for each edge [139]. The single-vertex matches are expanded using the structure of the maps. It checks the compatibility of edges incident to matched vertices and adjacent vertices based on their attributes, path lengths of edges and the angles between edges. This extension stops if no compatible edges or vertices are left. This work assumes no vertices or edges are missing, but this is not true in our study. Wallgrün proposed a method to identify overlapping between two tree-formed annotated Generalised Voronoi graphs [136]. The Generalised Voronoi diagram is obtained from a polygonal 2D environment by using the maximal inscribed circles. Considering the instability of the map, the algorithm uses a batch association approach for tree-formed AGVGs based on inexact graph matching techniques. Specifically, the method attempts to match potential vertices according to their degrees, subsequently expanding the match from matched vertices. The results are assessed using edit distance, with the optimal solution being recorded [136]. Mielle et al. proposed a method to match a sketch-map with a topological map [142]. Due to the inaccurate sketch-map, it is also an efficient error-tolerant graph matching algorithm. The approach finds the candidate matches based on the types of vertices and the order of the edges. The positions of vertices and length of edges in sketch-maps are no longer reliable, while the order of the edges is accepted as a reference. Similarly to the

previous paper, it extends the match from candidate matches, and every result is evaluated by normalised Levenshtein Edit Distance (LED) [142].

## 2.5 Summary

### 2.5.1 Research Gaps

This chapter summarises the mainstream open-source visual SLAM algorithms, including VIO algorithms. These algorithms are known for their robustness and high accuracy. Additionally, this chapter discussed several SLAM and SfM algorithms specifically designed for pipeline environments. These algorithms leverage the structural information of pipelines, enhancing localisation and mapping performance. Similarly, other algorithms that can utilise structural information are also mentioned here. Lastly, this chapter covered map merging and map alignment algorithms. Map merging algorithms can integrate maps generated by multiple robots, thereby improving exploration efficiency in large-scale environments. Map alignment algorithms, on the other hand, align different maps according to the overlapping areas. This review of various algorithms gives a theoretical foundation for the subsequent research and identifies key issues and challenges.

Section 2.1 summarises the main vSLAM algorithms, emphasising the characteristics of different approaches. It introduces open-source algorithms such as ORB-SLAM2 [5], DSO [15], and SVO [16]. Among these methods, ORB-SLAM2 is more robust, and many variants are based on it. This work aims to develop algorithms using ORB-SLAM2 as a foundation. Subsection 2.1.3 also summarises sensor data fusion in SLAM, which can be classified into loosely-coupled methods and tightly-coupled methods [56, 59]. The data from pipe networks[6] contains only images and odometer information. This study has used odometer measurement as the ground truth for algorithm evaluations.

Section 2.3 compares different cylindrical extraction algorithms [35, 110, 3] that can estimate cylinder parameters and select cylindrical points in complex point clouds. Existing algorithms lack robustness in complex point clouds, and current cylindrical representations limit some optimisation-based methods [35, 110]. This research requires a robust cylindrical extraction algorithm that can accurately estimate cylindrical parameters in complex point clouds.

Section 2.2 summarises a variety of vSLAM and SfM algorithms designed for pipeline environments, demonstrating how they leverage cylindrical information. Besides, Section 2.1.4 also presents how vSLAM algorithms leverage structural information. Existing algorithms apply cylindrical information only during triangulation and local optimisation [30, 26]. The accuracy can be further improved by refining the method of utilising cylindrical information.

Subsection 2.4.1 also introduces several map merging algorithms, most of which [116, 117,

118] rely on descriptors or the BoW method for data association. However, when merging large-scale maps, the number of features involved becomes enormous, placing a heavy computational burden on the robot. This research proposes a new map merging method designed for pipeline environments, reducing computational load by minimising the number of images involved.

Finally, Subsection 2.4.3 discusses topological map alignment methods, which can align two maps based on structural and geometric information. Current methods have limitations in information utilisation and matching techniques. They generally did not use metric information [142, 136]. Wallgrün requires the input maps to be tree-form topological maps [136]. Mielle et al. retain only the optimal match according to local information at each expansion step, which can lead to local optima [142]. This research proposes a new method that leverages metric information to align general topological maps and achieve globally optimal results without restrictions on input maps.

### 2.5.2 Aims and Objectives:

The primary aim of this research is to enhance the accuracy, robustness, and computational efficiency of vSLAM algorithms for autonomous robotic exploration in sewer pipe networks. To achieve this aim, the following objectives are set:

1. Develop a robust cylindrical extraction algorithm that can accurately estimate cylindrical parameters in complex point clouds, addressing the limitations of current methods.

2. Enhance vSLAM algorithms by incorporating structural information from pipe networks to mitigate accuracy degradation and scale drift.

3. Design a computationally efficient map merging algorithm for multi-robot SLAM in large-scale pipe networks, reducing the computational burden without compromising accuracy.

4. Propose a novel topological map alignment method that leverages metric information to achieve globally optimal alignment results, overcoming the limitations of existing techniques.

These objectives will guide the development of new methods and algorithms presented in subsequent chapters, addressing the identified research gaps and contributing to more reliable and efficient robotic navigation in sewer pipe networks.

# Chapter 3

# Cylinder Detection

## 3.1 Introduction

One of the key challenges for leveraging cylindrical information is cylinder extraction in a noisy point cloud generated by vSLAM. Pipes can have varying radii, lengths, and textures, and the corresponding point clouds from vSLAM are unorganised, noisy and non-uniformly distributed. Point clouds belonging to pipes and internal objects are extracted, which always represents an incomplete cylinder. Additionally, many outliers do not follow any specific distribution, further complicating the detection process. This complexity makes cylinder extraction particularly challenging. Numerous methods [35, 110, 3] have been developed for extracting cylinders from point clouds, with the choice of method depending on factors such as the precision of the point cloud, its distribution, the availability of point normals, and the number of cylinders present. The proposed approach does not assume any particular structure in the point cloud or require point normals. Instead, it treats the problem as generally as possible, making it adaptable to various practical applications within pipe networks.

Existing cylindrical fitting techniques [101, 33, 104] largely rely on least-squares methods and SVD based on the point-cylinder distance, which measures the distances from point clouds to the cylindrical surface. While Shakarji et al. [32] and Lukacs et al. [33] optimise cylindrical parameters by minimising point-cylinder distance, these approaches are sensitive to noise and outliers [34, 3]. Further, Shakarji et al. do not explain the computation of the point on the axis closest to the origin. Lukacs et al. got biased solutions after optimisation due to its cylinder representation method. Lopez-Escogido et al. [104, 35] used RANSAC and a direct method, but this approach may perform poorly when the uncertainty is unknown.

A two-step approach can be employed [107, 108, 110, 34, 3], starting with estimating the cylinder axis using PCA or point normals, followed by determining the cylinder's position and size through robust circle fitting on points projected onto a plane perpendicular to the cylinder axis [3]. However, estimating the cylinder axis via PCA [110, 34, 3] is highly susceptible to

outliers [3], while point normals require high-quality point clouds. Point normals estimate the axis based on the property that vectors normal to the cylinder surface converge on the axis [107, 108], but this approach is severely impacted by poor point normal quality. Moreover, errors in the first step can propagate to the second [109].

To extract cylinders from complex point clouds while filtering irrelevant points, many methods leverage techniques like the Hough Transform [107, 113] or RANSAC [35, 3]. The Hough method maps the point cloud to a designed cylindrical parameter space, selecting a solution with the highest accumulation as cylindrical parameters [107]. The RANSAC method repeatedly calculates various cylinders using minimum subsets of data and identifies inliers for each cylinder, selecting the best cylinder model based on inlier count. Both techniques are robust, and many implementations [35, 3, 107, 113] have been developed. However, they are computationally demanding, especially for large-scale point clouds, making them unsuitable for real-time applications. Given that a cylinder requires at least five parameters for its representation, the algorithm estimates the axis and radius independently to simplify the estimation process. Furthermore, due to imperfect evaluation metrics and unknown uncertainties in point cloud distributions, outlier detection remains a challenge in RANSAC-based methods [3]. This issue will be elaborated in a subsequent section.

Nurunnabi et al. proposed a robust repeated optimisation technique for cylinder extraction, effectively extracting multiple cylinders in complex environments [3]. This method alternates between optimisation and inlier selection. It uses a two-step approach: first, estimating the cylinder axis direction and position with DetRPCA [3], followed by fitting circles to multiple minimum subsets and refining the circles with a number of points with minimal residuals. The method is robust to outliers and can reliably identify multiple cylinders. However, the limitation of the method is that it requires knowledge of the inlier-to-outlier ratio to determine data quantity for refinement and iteration count. The ratio is unknown in complex scenarios. Additionally, the method inherits the limitations of the two-step approach. And the cylinder aspect ratio must be sufficiently large for DetRPCA to estimate the cylinder axis accurately, which may otherwise hinder successful cylinder extraction.

Thus, this chapter introduces a novel optimisation algorithm to estimate the position, direction of the axis, and radius together. This method reduces the precision requirements for the initial value for the axis, which enables a different method to estimate the axis more robustly. During optimisation, it can exclude outliers and involve inliers gradually.

The cylinder representation is essential to the optimisation, influencing both computational complexity and result accuracy. Cylinders are defined as surfaces consisting of all points on all the lines parallel to a given line, the cylinder axis. The two ends of a cylinder are not involved in estimation, reducing the number of parameters. Estimating the position of two ends

after cylinder extraction is straightforward based on inlier points. A cylinder has 5 degrees of freedom: 4 for the axis (a line in 3D space) and 1 for the radius [101, 32, 33, 30]. The axis in 3D space is usually defined with a unit vector and a point on the axis. Over-parameterisation could introduce additional constraints during optimisation, which makes the problem complex. Minimally, a cylinder can be described by one radius parameter and four for the axis [101, 33, 30]. Most methods used the point on the cylinder axis nearest to the origin to describe the position of the cylinder axis. Taubin did not clarify the transformation between the proposed cylinder representation and transformation matrix [101]. Similarly, Shakarji did not provide a mathematical definition of this point [32]. The expression proposed by Lukács et al. leads to a biased estimation when reducing the corresponding cost function [33]. Hansen et al. presented a cylinder expression converting Roll-Pitch-Yaw angles to rotation matrices, but its Jacobian matrix is complicated [30]. This chapter introduces a novel cylinder representation method suitable for online optimisation with a simple Jacobian matrix.

In summary, this chapter describes cylinder extraction methods. Also, it illustrates two numerical instability conditions widely observed in cylinder fitting methods and a solution to these issues. The main contributions of this chapter include:

1. A novel cylinder extraction method with a new cylinder representation method;
2. An introduction of numerical instability conditions and its solutions;
3. An improvement to the classic RANSAC method;
4. Experiments on simulation and real data reveal that the proposed cylinder extraction methods perform better than others.

The structure of this chapter is as follows: Section 3.2 introduces a new cylinder representation that enables the optimisation algorithm to estimate both the axis and radius simultaneously. The novel cylinder extraction method presented in Section 3.3 avoids the drawbacks of the two-step approach. Using point-cylinder distance as an evaluation metric can lead to erroneous estimations under two specific numerical instability conditions, and Section 3.4 delves into the underlying causes and introduces a solution. Section 3.5 introduces a cylinder extraction method based on reprojection errors suitable for visual SLAM or SfM algorithms. Section 3.6 discusses experiments conducted with both simulated and real data, along with an analysis of the results.

## 3.2 New Cylinder Representation

This section introduces an efficient cylinder representation that can be optimised without adding extra constraints. During optimisation, if the cylinder representation has degrees of freedom ex-

ceeding that of a cylinder, which is five, it may lead to the algorithm not converging. Introducing additional constraints to reduce the degrees of freedom of the representation could complicate the optimisation problem. Typically, a cylinder is parameterised by its axis $\zeta$ and radius $r \in \mathbb{R}$, as illustrated in Figure 3.1. The cylinder axis includes its direction and position. A direction vector, $\mathbf{v}_c \in \mathbb{R}^3$ defines the axis direction, while a point $\mathbf{p}_c \in \mathbb{R}^3$ on the axis denotes its position. The cylinder parameters are collectively denoted by $\mathcal{Y} = \{\mathbf{v}_c, \mathbf{p}_c, r\}$.

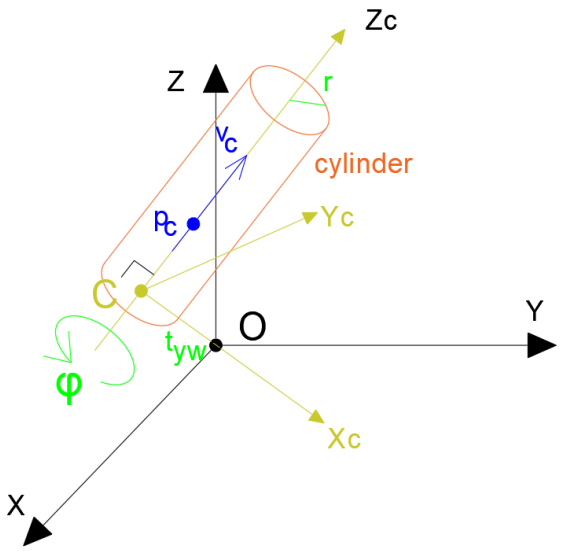



Figure 3.1: A cylinder coordinate system is shown within a global coordinate system $\mathbb{C}$. The proposed method defines a cylinder by describing the cylinder coordinate system.

The cylinder axis can be viewed as a vector aligned with the z-axis of a coordinate system, referred to as the cylinder coordinate system $\mathbb{C}$ in Figure 3.1. The cylinder representation method defines this coordinate system to specify the direction and position of the cylinder axis. It describes the relative pose of the world coordinate system with respect to the cylinder coordinate system, which can be expressed through a rotation matrix and a translation vector, collectively having six degrees of freedom. However, for a given cylinder axis, there are numerous available coordinate systems. Firstly, the origin of the coordinate system can move along its cylinder axis without altering the axis itself. Secondly, rotating this coordinate system around its z-axis does not alter the cylinder axis. With this information, the cylinder representation can be modified to achieve a minimal representation.

Regardless of how the coordinate system moves along the cylinder axis, the distance between the origin $O$ and the cylinder axis remains constant. The origin's position in the cylinder coordinate system is denoted by $\mathbf{t}_{yw}$, and its z-coordinate $\mathbf{t}_{yw_{(z)}}$ can be arbitrary without affecting the cylinder. Hence, the position of the origin $O$ in the cylinder coordinate system can be expressed using only two parameters, the x and y coordinates, while setting the z-coordinate to zero as default.

As the cylinder coordinate system can freely rotate around its z-axis without affecting the cylinder axis in the world coordinate system, this method adjusts the Angle $\varphi$ to minimise the degree of freedom of the cylinder representation method. When Angle $\varphi$ shifts, both the x and y coordinates of the world origin, $\mathbf{t}_{yw_{(x)}}$ and $\mathbf{t}_{yw_{(y)}}$, in the cylinder coordinate are impacted. Therefore, by appropriately rotating Angle $\varphi$ of the cylinder coordinate system, the y-coordinate of the world origin $\mathbf{t}_{yw_{(y)}}$ can be set to zero. Consequently, the world origin is uniquely determined by its x-coordinate $\mathbf{t}_{yw_{(x)}}$, with both y and z coordinates set to zero.

With these modifications, rotation and translation can concisely define the cylinder coordinate system using fewer parameters. Rotation can be depicted using a rotation vector $\phi$, a quaternion, or a rotation matrix $\mathbf{R}_{yw}$. Though these have different numbers of parameters, each maintains three degrees of freedom, with their respective Jacobians being 3D vectors in optimisation. The translation $\mathbf{t}_{yw}$ describes the world origin using a single parameter while setting the y and z coordinates to zero. Hence, the model incorporating minimal cylinder information is defined by the following parameters:

$$\mathcal{Y} = \{\mathbf{R}_{yw}, \mathbf{t}_{yw_{(x)}}, r\}. \tag{3.1}$$

Based on this, a transformation matrix can be obtained as follows:

$$T_{yw} = \begin{bmatrix} & & & \mathbf{t}_{yw_{(x)}} \\ & R_{yw} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

which represents only the cylinder axis, while $r$ represents the radius uniquely. This matrix will be used in subsequent calculations.

The position $\mathbf{p}_{yi}$ of Point $i$ in the world coordinate system can be transformed to the cylinder coordinate system as follows:

$$\mathbf{p}_{yi} = \mathbf{T}_{yw}\mathbf{p}_{wi} \tag{3.3}$$

### 3.2.1 Transformation between Different Cylindrical Representations

This subsection presents the transformation among various representation methods. At the beginning, it will introduce the transformation from a general form to the proposed cylinder representation. In Section 4.6, the cylinder coordinate system will be optimised using a transformation matrix format. However, the resulting coordinate system may not maintain $\mathbf{t}_{yw_{(y)}}$ and $\mathbf{t}_{yw_{(z)}}$ at zero. The modification of a cylinder transformation matrix is also discussed.

Given a cylinder defined by its direction $\mathbf{v}_c$, point $\mathbf{p}_c$, and radius $r$, the algorithm establishes the cylinder coordinate system and computes the transformation matrix $\mathbf{T}_{yw}$. This

method calculates the cylindrical coordinate system within the global coordinate system. It starts this by finding the perpendicular line from the world origin straight down to the axis of the cylinder first. The perpendicular foot $\mathbf{c}_w$ can be described as follows:

$$\mathbf{c}_w = \lambda \mathbf{v}_c + \mathbf{p}_c \tag{3.4}$$

Since line $OC$ is perpendicular to the cylinder axis, the following equation is valid:

$$\begin{aligned} OC * \mathbf{v}_c &= 0 \\ (\lambda \mathbf{v}_c + \mathbf{p}_c) * \mathbf{v}_c &= 0 \\ \lambda &= -\mathbf{p}_c * \mathbf{v}_c \end{aligned} \tag{3.5}$$

The distance between the world origin and the perpendicular foot is the x-coordinate $\mathbf{t}_{yw_{(x)}}$ of the world origin in the cylinder coordinate system, and the perpendicular line defines the x-axis $\mathbf{R}_{yw_{(1,:)}}$ in the coordinate system.

$$\mathbf{R}_{yw_{(1,:)}} = \mathbf{t}_{yw_{(x)}} = |\lambda \mathbf{v}_c + \mathbf{p}_c| \tag{3.6}$$

With the cylinder axis as the z-axis $\mathbf{R}_{yw_{(3,:)}}$, the cross-product of the two gives the y-axis:

$$\mathbf{R}_{yw_{(2,:)}} = \mathbf{v}_c \times \mathbf{c}_w \tag{3.7}$$

After the above calculations, the cylinder matrix is obtained.

Under the new cylinder representation, the cylinder parameters are not unique when the cylinder axis passes through the world origin. In this case, the origin of the cylinder coordinate system coincides with the world origin, and $\mathbf{t}_{yw_{(x)}}$ equals 0. Any rotations $\varphi$ around the z-axis of the cylinder coordinate system will not change the position of world origin in the cylinder coordinate system. As a result, any line perpendicular to the cylinder axis can be regarded as the x-axis. However, the cylinder representation method approximates the cylinder instead of its cylinder coordinate system. Any cylinder coordinate system corresponding to the correct axis is deemed acceptable. This inherent homogeneity does not affect the estimation.

A straightforward method for representing a cylinder under this new method is introduced here when the cylinder axis intersects the world origin. Select any point, $\mathbf{m}$ that does not lie on the cylinder axis, and the cross-product of $\mathbf{m}$ and the direction vector $\mathbf{v}$ of the cylinder axis defines the x-axis coordinate $\mathbf{R}_{yw_{(2,:)}}$. The calculation for the y-axis coordinate remains unchanged.

When the cylinder is optimised as a transformation matrix in the vSLAM algorithm proposed in the next chapter, the x-axis of the cylinder coordinate system might not intersect with the world origin after optimisation. In this case, the cylindrical coordinate system no longer meets the requirements of the new cylinder representation, meaning that the current parameters

describe a different cylinder. It is necessary to update the cylinder parameters after optimisation. The resultant translation of this transformation matrix may not have zero values for the y and z coordinates. Thus, a method is required to compute the new cylinder presentation based on this matrix $T_{y2}$. The third-row vector $R_{y2_{(3,:)}}$ of the transformation matrix can be interpreted as the direction vector $\mathbf{v}_c$ of the cylinder. Similarly, the translation $t_{y2}$ can be seen as the position of a point $\mathbf{p}_c$ on the axis. Based on the direction $\mathbf{v}_c$ and the point $\mathbf{p}_c$, the system can compute the new cylinder representation $\mathbf{T}_{yw}$ as previously introduced.

## 3.3 Cylinder Extraction based on Point Cylinder Distance

This section introduces an iterative cylinder extraction method that is robust against noise and outliers. The algorithm estimates cylinder parameters using the new cylinder representation by minimising the cost function, referred to as Iterative Least Square (ILS). During the optimisation process, the system will select inliers based on the estimated cylinder parameters and use the updated inliers as input for the next iterations.

The system will employ the cylinder extraction algorithm among point clouds generated by the vSLAM algorithm. vSLAM constructs point clouds by triangulating features, resulting in sparse and unorganised point clouds without point normal. There is no prior knowledge about the cylinder in this context. In addition, vSLAM can also offer information on keyframes which are not exploited by existing algorithms.

The cost function (Equation 3.8) that measures the point-cylinder distance is multimodal. Thus, a good choice of the initial value is pivotal to ensure convergence to a global optimum. The system first employs PCA, using local keyframe positions as input, to determine the direction of the cylinder axis. The average of the keyframe positions acts as the position of a point on this axis. By measuring the point-cylinder distances of all map points, the algorithm takes the average to set an initial estimate for the radius. With the capability to optimise the cylinder axis alongside its radius, this method can effectively handle an initially imprecise cylinder axis.

For a precise estimation of cylinder parameters, the algorithm needs the capability to identify inliers during optimisation. While many optimisation algorithms would use kernel functions to suppress outliers, such an approach is not applicable here since the uncertainty of the inliers in 3D space is unknown. While Nurunnabi et al. assumes that the inlier-to-outlier ratio is known, such an assumption does not hold in this dissertation [3]. A more realistic assumption is that the cylindrical points will predominantly be distributed near the cylinder surface, and the point-cylinder distance follows a Gaussian distribution. Thus, 95% of the cylindrical points are expected to fall within a confidence interval close to the cylindrical surface. The confidence interval is designated as 5% of the radius, and its variance of point-cylinder distance is $\frac{25r^2}{38416}$.

Therefore, when the point-cylinder distance is less than 5% of the radius, the system considers the point to be a cylindrical point and uses it as input for the next optimisation.

Given the point cloud and initial cylinder parameters, the system starts to iteratively optimise the cylinder parameters and selects inliers based on the new cylinder parameters until the variance of the point-cylinder distance of inliers is below a specified threshold or the maximum number of iterations is reached. The cost function based on the point-cylinder distance is:

$$f(\mathcal{Y}) = \sum_{n \in \mathcal{I}} \|e_c^n\|^2 \tag{3.8}$$

Where $e_c^n$ is the point-cylinder distance, $\mathcal{I}$ represents the set of the current inliers, and $i$ denotes the index of the map point. The point-cylinder distance is defined as follows:

$$e_c^n = |(\mathbf{T}_{yw}\mathbf{p}_{wi})_{(1:2,:)}| - r \tag{3.9}$$

where $\mathbf{T}_{yw}$ denotes the cylinder in the form of a transformation matrix.

Taking the derivative with respect to the cylinder parameters:

$$J(\mathcal{Y}) = \frac{\partial e_c}{\partial \mathcal{Y}} = \begin{bmatrix} \frac{\partial e_c}{\partial R_{yw}} \\ \frac{\partial e_c}{\partial t_{yw_x}} \\ \frac{\partial e_c}{\partial r} \end{bmatrix}^{\mathrm{T}} \tag{3.10}$$

Every term in Equation 3.10 can be obtained as follows:

$$\begin{aligned} \frac{\partial e_c}{\partial R_{yw}} &= \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(x)}}} \frac{\partial \mathbf{p}_{yi_{(x)}}}{\partial R_{yw}} + \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(y)}}} \frac{\partial \mathbf{p}_{yi_{(y)}}}{\partial R_{yw}} \\ &= \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(x)}}} \frac{\partial \mathbf{p}_{yi_{(x)}}}{\partial R_{yw}} + \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(y)}}} \frac{\partial \mathbf{p}_{yi_{(y)}}}{\partial R_{yw}} \\ &= \frac{\mathbf{p}_{yi_{(x)}} \begin{bmatrix} 0 & \hat{\mathbf{p}_{yi}}_z & -\hat{\mathbf{p}_{yi}}_{(y)} \end{bmatrix}}{\sqrt{{\mathbf{p}_{yi_{(x)}}}^2 + {\mathbf{p}_{yi_{(y)}}}^2}} + \frac{\mathbf{p}_{yi_{(y)}} \begin{bmatrix} -\hat{\mathbf{p}_{yi}}_z & 0 & \hat{\mathbf{p}_{yi}}_{(x)} \end{bmatrix}}{\sqrt{{\mathbf{p}_{yi_{(x)}}}^2 + {\mathbf{p}_{yi_{(y)}}}^2}} \end{aligned} \tag{3.11}$$

$$\begin{aligned} \frac{\partial e_c}{t_{yw_x}} &= \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(x)}}} \frac{\partial \mathbf{p}_{yi_{(x)}}}{\partial t_{yw_x}} \\ &= \frac{\partial e_c}{\partial \mathbf{p}_{yi_{(x)}}} \end{aligned} \tag{3.12}$$

and

$$\frac{\partial e_c}{\partial r} = -1 \tag{3.13}$$

In the optimisation, the cylindrical parameters are updated by the below equation:

$$\mathcal{Y}^{new} = \delta\xi \oplus \mathcal{Y}^{old} = \begin{bmatrix} \exp\left(\delta\xi_{(\mathbf{1:3})}\right) * \mathbf{R}_{yw} \\ \delta\xi_{(4)} + t_{yw_{(x)}} \\ \delta\xi_{(5)} + r \end{bmatrix}^{\mathrm{T}} \tag{3.14}$$

where $\delta\xi$ is the increment obtained by the Gauss-Newton method.

With a set of map points, the system detects the cylinder with Algorithm 1:

**Algorithm 1** Cylinder Detection based on point-cylinder distance

**procedure** ILS(map points **p**)
  Build a empty Set $\mathcal{G}$
  Add all the map points **p** to Set $\mathcal{G}$
  Initialise the cylinder parameters $\mathcal{Y}$
  **while** $\sigma >$threshold $\sigma_t$ **do**
    Fit a cylinder to the map points from Set $\mathcal{G}$ by minimising Equation 3.8
    Compute the distance $d_i$ of all map points from the cylinder axis
    Compute the mean $\mu$ and variance $\sigma$ of the distances $d_j(j \in \mathcal{G})$
    Clean up Set $G$, and add all points whose distance is close to the mean $\mu, |d_j - \mu| \leq 1.96\sigma$
  **end while**
  **if** The proportion of points in Set $\mathcal{G}$ to the total points $> \tau_2$ **then**
    **return** $\mathcal{Y}$
  **else**
    **return** NULL
  **end if**
**end procedure**

Assuming the while loop executes $k_1$ times, and the cylinder fitting algorithm runs $k_2$ iterations each time, with a total of $N$ map points, the overall complexity is $\mathbf{O}(k_1 k_2 N)$.

In most circumstances, the algorithm can optimise the cylinder by reducing the point-cylinder distance with selected points. The algorithm may fail in two numerical instability conditions and produce invalid results. The next section will explain the numerical instability conditions.

## 3.4 Numerical Instability Conditions and Solution

### 3.4.1 Numerical Instability Conditions

Existing algorithms rely on the point-cylinder distance to evaluate the estimated cylinder, which could lead the algorithm to produce an incorrect result in some circumstances. Given a cylinder, a point on it can be described by its angle $\theta$ and height-to-radius ratio $h_r$, as shown in Figure 3.2. Ideally, when evaluating an estimated cylinder model, the error should be between a map point and the corresponding point on the estimated cylinder with the same angle $\theta$ and height-to-

radius ratio $h_r$. Unfortunately, it is impossible to get the angle and ratio of a map point without a real cylinder or point normal. The height-to-radius ratio can be obtained as follows:

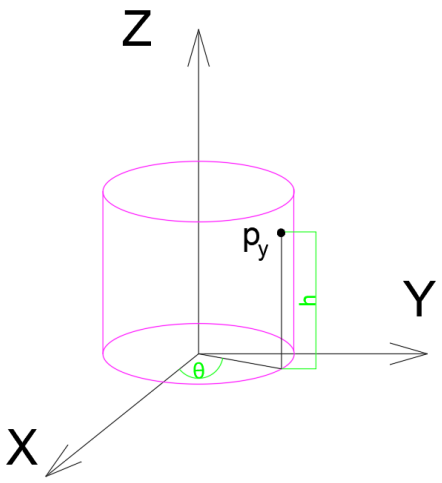



Figure 3.2: A point on it can be described by its angle $\theta$ and height $h$

$$h_r = \frac{h}{r}. \tag{3.15}$$

The point-cylinder distance differs from the ideal error, leading to incorrect cylinder evaluations in certain situations. This problem arises from two main reasons, which will be explained with the following figure. In Figure 3.3 and 3.4, the black circle represents the cross-section of the real cylinder, and the green circle indicates the estimated one. Black dots represent the point cloud, while the green dots denote the corresponding points on the estimated cylinder with the same angle $\theta$ and height-to-radius $h_r$ as black points. Grey dots are outliers. Purple segments illustrate the point-cylinder distance, while dark yellow segments depict the ideal error.

When the estimated cylinder's position significantly deviates from the actual cylinder, the difference between the point-cylinder distance and the ideal error becomes large. As shown in Figure 3.3A, Estimated cylinder 0 is close to the real cylinder, while the centre of Estimated cylinder 1 lies outside the black circle. Both circles have the same radius but differ in position.

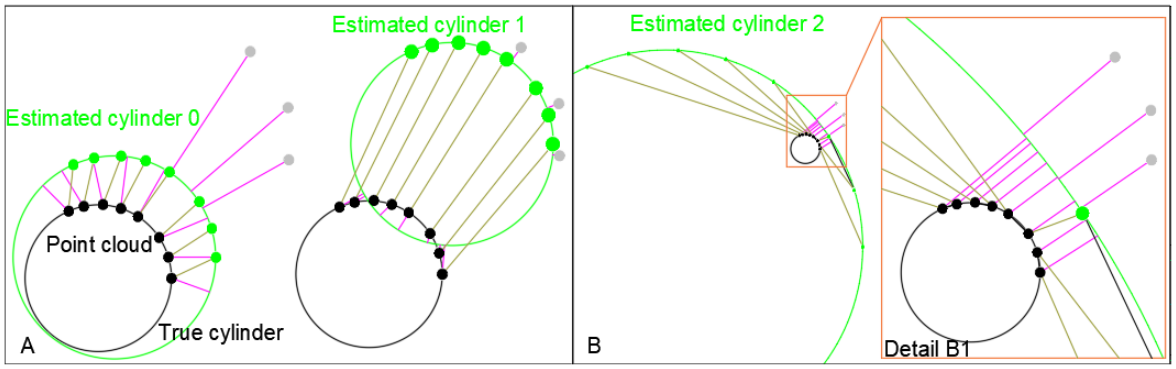



Figure 3.3: The numerical instability conditions in cylinder fitting

In this case, Estimated cylinder 1 has a smaller point-cylinder distance yet a more significant ideal error. The ideal errors of Estimated cylinder 0 approximate its point-cylinder distances. If the system iteratively optimises Estimated cylinder 0 and 1 by reducing the point-cylinder distance, Estimated cylinder 0 may approximate the true value, whereas Estimated cylinder 1 will remain as is.

Another challenging issue affecting cylinder estimation is the uncertainty of outliers used to reject or suppress these outliers. Whether using iterative optimisation algorithms or RANSAC, outliers must be determined based on their cost compared with this uncertainty. Some algorithms define a flexible threshold that varies with the radius of the estimated cylinder, which is defined as a product of the cylinder's radius and a constant. In Figure 3.3B, a massively incorrect estimated cylinder sits beside the real one. Estimated cylinder 2 has a vast radius, resulting in a large threshold for outliers. Consequently, all outliers have point-cylinder distances below this threshold and are erroneously identified as inliers. In this case, iterative optimisation and RANSAC algorithms can not detect these outliers, leading to undesirable outcomes. The result could be a large cylinder with a large radius, and most points could be identified as inliers.

### 3.4.2 The Solution to Numerical Instability Conditions

This subsection introduces a method to identify or avoid ill-conditioned situations and verify the results of cylinder estimation. When a keyframe observes the inside of the corresponding pipeline, vSLAM extracts features from the image and triangulates them to obtain map points on the cylinder. If the estimated cylinder is accurate, the position of a map point can be approximated by the intersection of the backprojected ray and the cylinder surface. The backprojected ray traces back from a feature in an image into the 3D space and represents all the possible positions that the triangulated feature could be in the 3D world. This ray $\mathbf{q}_y$ in the cylinder coordinate system can be obtained as follows:

$$\mathbf{q}_y = \mathbf{R}_{yw}\mathbf{R}_{wi}\kappa^{-1}_{(\mathbf{f}_i)} \tag{3.16}$$

where $\mathbf{R}_{wi}$ is the orientation of keyframe $i$, $\kappa^{-1}_{(\bullet)}$ refers to the reverse process of image formation in which a ray is traced back from feature $\mathbf{f}_i$ in an image into the 3D space.

Compared to evaluating a cylinder in 3D space, this method evaluates the cylinder at the image level. The uncertainty of map points in 3D space is unknown, but the system knows the uncertainty of the corresponding features at the image level. Thus, this method projects the intersection onto other keyframes and calculates the reprojection error, which is the distance between the feature and the projected intersection. The system extracts features at different levels of the pyramid, and ORB-SLAM2 [5] assumes the uncertainty of a feature at the original image is 1 pixel. This uncertainty increases as resolution decreases in a higher level of the pyramid [119].

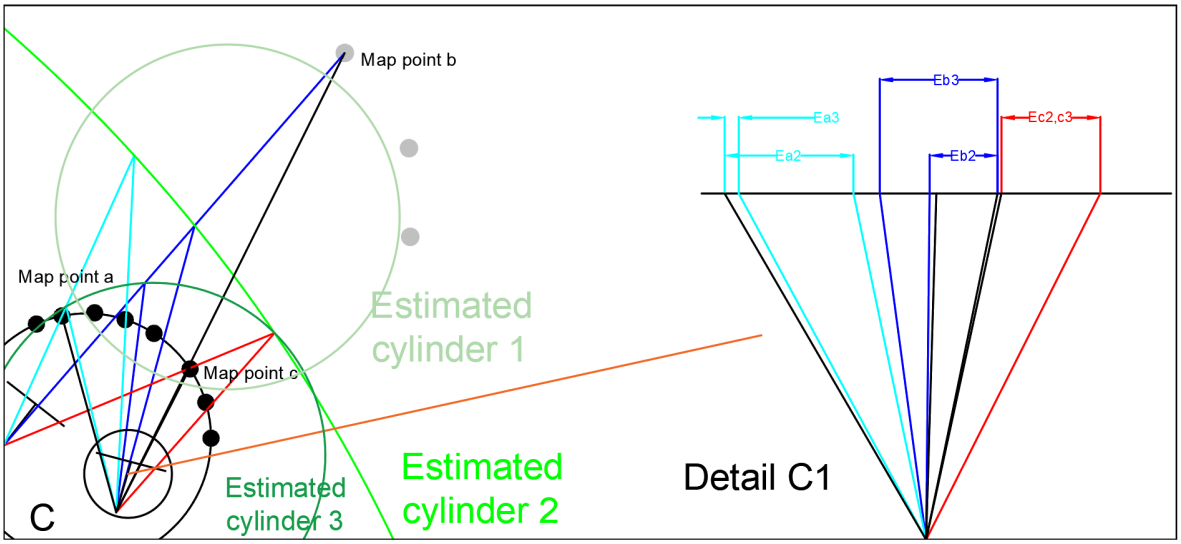



Figure 3.4: The numerical instability conditions can be detected by the new evaluation method

Since the intersection is on the cylinder, its distance to the cylinder axis equals the radius $r$:

$$\begin{aligned} &\mathbf{p}_{y(x)}^2 + \mathbf{p}_{y(y)}^2 = r^2 \\ &(\lambda q_{y(x)} + t_{yi(x)})^2 + (\lambda q_{c(y)} + t_{yi(y)})^2 = r^2 \\ &(q_{c(x)}^2 + q_{c(y)}^2)\lambda^2 + 2(q_{c(x)}t_{ci(x)} + q_{c(y)}t_{ci(y)})\lambda + (t_{ci(x)}^2 + t_{ci(x)}^2 - r^2) = 0 \end{aligned} \tag{3.17}$$

Where $\lambda$ is the depth of the feature. The algorithm estimate $\lambda$ by solving quadratic Equation 3.17, and the value of lambda:

$$\begin{aligned} \lambda &= -\frac{(q_{c(x)}t_{ci(x)} + q_{c(y)}t_{ci(y)})}{(q_{c(x)}^2 + q_{c(y)}^2)} \\ &+ \frac{\sqrt{(q_{c(x)}t_{ci(x)} + q_{c(y)}t_{ci(y)})^2 - (q_{c(x)}^2 + q_{c(y)}^2)(t_{ci(x)}^2 + t_{ci(y)}^2 - r^2)}}{(q_{c(x)}^2 + q_{c(y)}^2)} \\ &= \frac{-(q_{c(x)}t_{ci(x)} + q_{c(y)}t_{ci(y)}) + \sqrt{(q_{c(x)}^2 + q_{c(y)}^2)r^2 - (q_{c(x)}t_{ci(y)} - q_{c(y)}t_{ci(x)})^2}}{(q_{c(x)}^2 + q_{c(y)}^2)} \end{aligned} \tag{3.18}$$

A feature is observed by keyframe $i$ and $j$, and the reprojection error $e_{rp}$ in keyframe $j$ is defined as follows:

$$\begin{aligned} g(\mathcal{Y}) &= \kappa(\mathbf{T}_{yw}\lambda\kappa^{-1}(f_i)) - f_j \\ e_{rp} &= \kappa(\mathbf{T}_{yw}\lambda\kappa^{-1}(f_i)) - f_j \end{aligned} \tag{3.19}$$

Figure 3.4 shows the comparison between the point-cylinder distances and reprojection errors for three map points under two numerical instability conditions. The figure displays three estimated cylinders at different positions. Estimated cylinder 1 has its centre outside the actual cylinder, Estimated cylinder 2 has a large radius as that in Figure 3.3B, and Cylinder 3 closely approximates the true cylinder with a section overlapping the real cylinder. Every backprojected

ray has two intersections with Estimated cylinder 1, indicating that the keyframe is outside the cylinder. This cylinder is discarded since keyframes will not exist outside the cylinder. For each map point on the Estimated cylinders 2 and 3, the reprojection errors are calculated, as shown in Detail C1. Except for reprojection error $E_{a3}$, all other errors are considerably large. When the real cylinder surface near point $a$ coincides with that of the estimated cylinder, the map point has a smaller reprojection error. The rest of the map points deviate from the estimated cylinder surface, resulting in large reproduction errors. Using this new evaluation method, the system can identify cylinders deviating from the true value. In the next section, a cylinder extraction method based on reprojection error will be discussed.

## 3.5 Cylinder Extraction based on Reprojection Error

This section introduces a new cylinder extraction algorithm based on the reprojection error discussed in the last section. The algorithm employs the direct method to fit the cylinder and uses the RANSAC method to reject outliers, followed by optimisation to refine the cylinder parameters.

### 3.5.1 Direct Method

To detect a cylinder in point clouds, this approach adopts the direct method, which is slightly different from the equations mentioned in the literature [104, 35] due to difficulties in realisation.

The distance from cylindrical points to the cylinder axis equals its radius. Once cylinder points are on the cylindrical surface, the below function can be obtained:

$$\mathbf{p}_{yi_{(x)}}^2 + \mathbf{p}_{yi_{(y)}}^2 - r^2 = 0 \tag{3.20}$$

where $\mathbf{p}_{y_{(x)}}$ and $\mathbf{p}_{y_{(y)}}$ represent the positions of the cylindrical point in the cylindrical coordinate system, and $r$ denotes the cylinder radius.

A cylinder, a unique 3D quadratic surface, can be expressed in terms of a symmetric homogeneous matrix $\mathbf{C}$ for cylindrical points with homogeneous coordinates $\mathbf{P}_y$. The equation can be written as:

$$\mathbf{p}_y^{\mathrm{T}} \mathbf{C} \mathbf{p}_y = 0 \tag{3.21}$$

where are the matrices $\mathbf{C}$ and $\mathbf{D}$:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} = \begin{bmatrix} D & 0 \\ 0 & -r^2 \end{bmatrix} \tag{3.22}$$

A notable observation is that $\mathbf{C}$ and $\mathbf{D}$ are singular matrices, and they have three and two eigenvalues, respectively.

Using the point in the world coordinate system denoted by $\mathbf{p}_w$, Equation 3.21 can be transformed to the following function:

$$\mathbf{p}_w{}^{\mathrm{T}}\mathbf{T}_{yw}{}^{\mathrm{T}}\mathbf{C}\mathbf{T}_{yw}\mathbf{p}_w = 0 \tag{3.23}$$

where $\mathbf{T}_{yw}$ is the transformation matrix, and it can convert the point cloud from the world coordinates to the cylindrical coordinate system. Its rotation matrix and translation vector are $\mathbf{R}_{yw}$ and $\mathbf{t}_{yw}$ respectively.

Let simplify Equation 3.23 by introducing a $4 \times 4$ matrix $\mathbf{B}$, defined as:

$$\begin{aligned}\mathbf{B} &= \mathbf{T}_{yw}{}^{\mathrm{T}}\mathbf{C}\mathbf{T}_{yw} \\ &= \begin{bmatrix}\mathbf{R}_{yw}{}^{\mathrm{T}} & 0 \\ \mathbf{t}_{yw}{}^{\mathrm{T}} & 1\end{bmatrix}\begin{bmatrix}D & 0 \\ 0 & -r^2\end{bmatrix}\begin{bmatrix}\mathbf{R}_{yw} & 0 \\ \mathbf{t}_{yw} & 1\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{R}_{yw}{}^{\mathrm{T}}D\mathbf{R}_{yw} & \mathbf{R}_{yw}{}^{\mathrm{T}}D\mathbf{t}_{yw} \\ \mathbf{t}_{yw}{}^{\mathrm{T}}D\mathbf{R}_{yw} & \mathbf{t}_{yw}{}^{\mathrm{T}}D\mathbf{t}_{yw} - r^2\end{bmatrix}\end{aligned} \tag{3.24}$$

Matrix $\mathbf{B}$ is symmetric and has 16 elements, and only ten parameters need estimation due to its symmetric nature. Since the ranks of matrix $\mathbf{C}$ is three, matrix $\mathbf{B}$ has rank three. Therefore, the matrix only has nine degrees of freedom, and the algorithm can estimate the matrix with only nine independent equations. Similar to the estimation of the fundamental matrix, the equation can be cast in the form $\mathbf{A}\mathbf{b} = 0$ and then solved using SVD, where a row of matrix $\mathbf{A}$ is:

$$\mathbf{A}_i = \begin{bmatrix}\mathbf{p}^2_{w_{(x)}} & 2\mathbf{p}_{w_{(x)}}\mathbf{p}_{w_{(y)}} & 2\mathbf{p}_{w_{(x)}}\mathbf{p}_{w_{(z)}} & 2\mathbf{p}_{w_{(x)}} & \mathbf{p}^2_{w_{(y)}} & 2\mathbf{p}_{w_{(y)}}\mathbf{p}_{w_{(z)}} & 2\mathbf{p}_{w_{(y)}} & \mathbf{p}^2_{w_{(z)}} & 2\mathbf{p}_{w_{(z)}} & 1\end{bmatrix} \tag{3.25}$$

and with

$$\mathbf{b} = \begin{bmatrix}\mathbf{b}_{(11)} & \mathbf{b}_{(12)} & \mathbf{b}_{(13)} & \mathbf{b}_{(14)} & \mathbf{b}_{(22)} & \mathbf{b}_{(23)} & \mathbf{b}_{(24)} & \mathbf{b}_{(33)} & \mathbf{b}_{(34)} & \mathbf{b}_{(44)}\end{bmatrix}^{\mathrm{T}} \tag{3.26}$$

After obtaining $\mathbf{b}$ through SVD, the system can compute the matrix $\mathbf{T}_{yw}$ and $r$, and the details will be presented here. Both the positive and negative values of $b$ satisfy $\mathbf{A}b = 0$, but the sign of $b$ can be determined by ensuring $\mathbf{B}(1,1) > 0$ since the elements of the matrix $\mathbf{R}_{yw}{}^{\mathrm{T}}D\mathbf{R}_{yw}$ are non-negative. The system computes the matrix $\mathbf{R}_{yw}$ by applying SVD to $\mathbf{R}_{yw}{}^{\mathrm{T}}D\mathbf{R}_{yw}$ shown as follows:

$$\mathbf{R}_{yw}{}^{\mathrm{T}}\mathbf{D}\mathbf{R}_{yw} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}} \tag{3.27}$$

Matrix $\mathbf{R}_{yw}$ equal to matrix $V^{\mathrm{T}}$. For the equation $\mathbf{A}\mathbf{b} = 0$, multiplying $b$ by any scalar does not affect the validity of the equation. The system can compute the scalar $s$ according to eigenvalues of matrices $\Sigma$ and $D$. The eigenvalues, $\lambda_1$ and $\lambda_2$, of matrix $\mathbf{D}$ are both equal to 1 according to

Equation 3.22, but $\Sigma$ may not have same eigenvalues $\Sigma(1,1), \Sigma(2,2)$. Thus, the system computes the scalar as follows:

$$s = \frac{\lambda_1 + \lambda_2}{\Sigma(1,1) + \Sigma(2,2)} = \frac{2}{\Sigma(1,1) + \Sigma(2,2)} \tag{3.28}$$

The matrix $\mathbf{B}$ can be updated by

$$\mathbf{B} < -s\mathbf{B} \tag{3.29}$$

At this stage, the system calculates $\mathbf{t}_{yw}$ and $r$ based on the known matrices $\mathbf{R}_{yw}$ and $D$. Notably, the third element of $\mathbf{t}_{yw}$ does not affect the position of the cylinder, the reason for which can be found in Section 3.2.

The approach outlined above yields a biased solution due to the nature of the cost function. A map point external to the cylinder has a higher cost in Equation 3.20 than an internal point at the same distance from the cylindrical surface. In practice, it is unlikely to achieve a zero-cost function, so the method adjusts the radius to minimise the overall cost. The system emphasises external points more than internal points, giving greater weight to external points compared to internal ones. The system reduces the cost of external points by overestimating the radius of the cylinder. The method always gives a solution larger than an unbiased estimate. To correct this bias, this work further optimises the cylinder parameters in Subsection 3.5.3.

### 3.5.2 RANSAC Algorithm

This section introduces how to estimate cylinder parameters using the RANSAC method and the direct method, and the result will serve as the initial values for the optimisation algorithm. Compared to existing algorithms [104, 35], this approach employs a different evaluation method to avoid falling into numerical instability conditions.

The proposed method evaluates the cylinder based on the reprojection error of each feature, and the score of the estimated cylinder is derived from the sum of these individual evaluations. After obtaining the keyframes, point cloud, and a candidate cylindrical model, the system calculates the reprojection error and scores the model. Initially, the algorithm sequentially computes the backprojected ray for each feature based on the initial keyframe pose and the pixel position of the feature, as delineated in Equation 3.16. Subsequently, the system computes the intersection of the reprojected ray and the cylindrical surface and projects it onto another keyframe. The reprojection error is the distance between the projected intersection and the corresponding tracked feature. When a feature has a reprojection error less than its uncertainty in the image, the score attributed to the cylinder by the system equals the difference between the uncertainty and the reprojection error. The contribution of a feature with a larger reprojection error is zero. The score of the estimated cylinder is the sum of these evaluations for every feature.

Usually, the system tracks a feature over several consecutive keyframes, but calculating reprojection errors for every single keyframe can be computationally intensive. To reduce this complexity, the algorithm will only select one keyframe. When the feature has a large parallax angle, the reprojection error will be more sensitive to the position of the map point. This sensitivity is desired since this method is used to identify outliers. Consequently, the system simply chooses the keyframe that is furthest from the initial one.

The system will choose the model with the highest score as the optimal model. The main steps of the RANSAC algorithm are outlined in Algorithm 2 below.

At the end of the algorithm, the system will optimise the cylinder parameters with those identified inliers. Further details on this process will be introduced in the next subsection.

### 3.5.3 Optimisation

This work proposes a novel cylinder fitting method that optimises the cylinder parameters by reducing reprojection errors. Unlike previous methods, this approach takes features as its input rather than map points. Given that the proposed algorithm has already identified the inliers in the previous subsection, the system directly optimises the cylinder parameters with inliers without the need for further outlier filtering.

Given the cylinder parameters $\mathcal{Y}$, the inlier features $f_{kj}$ detected in keyframe $j$, and keyframes $i$ and $j$, the cost function based on the reprojection error $e_y$ is formulated as follows:

$$f(\mathcal{Y}) = \sum_{k=1}^{N} \|e_{y_{(k)}}\|^2 \tag{3.30}$$

with

$$e_{y_{(k)}} = \kappa(\mathbf{T}_{jw}\mathbf{T}_{wy}\mathbf{p}'_{yk}) - f_{kj} \tag{3.31}$$

where $f_{kj}$ is observed feature $k$ in keyframe $j$, and point $\mathbf{p}'_{yk}$ is the intersection of feature $k$ in the cylinder coordinate system.

The point $p'^{(k)}$ can be obtained as follow:

$$\mathbf{p}'_{yk} = \lambda_{kj}\mathbf{q}_{yk} + t_{yi} \tag{3.32}$$

where $\mathbf{q}_{yk}$ is the backprojected ray of feature $k$ in the cylinder coordinate system, and $\lambda_{kj}$ can be obtained as Equation 3.18.

The Jacobian matrix describes how minor variations in the cylinder parameters affect the reprojection error. Its Jacobian with respect to the cylinder parameters is given by:

$$J(\mathcal{Y}) = \frac{\partial e_y}{\partial \mathcal{Y}} = \begin{bmatrix} \frac{\partial e_y}{\partial R_{yw}} \\ \frac{\partial e_y}{\partial t_{yw_{(x)}}} \\ \frac{\partial e_y}{\partial r} \end{bmatrix}^{\mathrm{T}} \tag{3.33}$$

**Algorithm 2** A Modified RANSAC algorithm

**procedure** MODIFIEDRANSAC(Point cloud $\mathcal{P}$, related keyframes, number of iterations $N$)
    Initialise best inliers set $\mathcal{I}_{\text{best}}$ to empty
    Initialise the set of input set $\mathcal{S}$ to empty
    **for** $i = 1$ to $N$ **do**
        Randomly select minimal set of points $\mathcal{S}$ from $\mathcal{P}$
        Estimate cylinder $\mathcal{Y}_{\text{temp}}$ from $\mathcal{S}$ using nine-points method with Equation 3.24
        Initialise inliers set $\mathcal{I}$ to empty
        Initialise initial score $s_i$ to 0
        **for** Each point $\mathbf{p}_i$ in $\mathcal{P}$ **do**
            Compute the reprojection error $e_i$ with related keyframes $\mathcal{L}$ as referred to in Equation 3.19
            **if** Reprojection error $e_i <$ critical value $\tau_i$ **then**
                Add $\mathbf{p}_i$ to inliers set $\mathcal{I}$
                Add $\tau_i - e_i$ to score $s_i$
            **end if**
        **end for**
        **if** $s_i > s_{\text{best}}$ **then**
            $\mathcal{I}_{\text{best}} = \mathcal{I}$
            $\mathcal{Y}_{\text{best}} = \mathcal{Y}_{\text{temp}}$
            $s_{\text{best}} = s_i$
        **end if**
    **end for**
    **if** Size of $\mathcal{I}_{\text{best}}$ is sufficient **then**
        Refine $\mathcal{Y}_{\text{best}}$ using all points in $\mathcal{I}_{\text{best}}$
        **return** $\mathcal{Y}_{\text{best}}$
    **else**
        **return** Model not found
    **end if**
**end procedure**

with

$$\begin{aligned}\frac{\partial e_y}{\partial R_{yw}} &= \frac{\partial e_y}{\partial p'}\frac{\partial p'}{\partial \lambda}\frac{\partial \lambda}{\partial R_{yw}} \\ &= \frac{\partial e_y}{\partial p'}\frac{\partial p'}{\partial \lambda}\left(\frac{\partial \lambda}{\partial q_{y_{(x)}}}\frac{\partial q_{y_{(x)}}}{\partial R_{yw}} + \frac{\partial \lambda}{\partial q_{y_{(y)}}}\frac{\partial q_{y_{(y)}}}{\partial R_{yw}} + \frac{\partial \lambda}{\partial t_{yi_{(x)}}}\frac{\partial t_{yi_{(x)}}}{\partial R_{yw}} + \frac{\partial \lambda}{\partial t_{yi_{(y)}}}\frac{\partial t_{yi_{(y)}}}{\partial R_{yw}}\right)\end{aligned} \tag{3.34}$$

$$\begin{aligned}\frac{\partial e_y}{\partial t_{yw_{(x)}}} &= \frac{\partial e_y}{\partial p'}\frac{\partial p'}{\partial \lambda}\frac{\partial \lambda}{\partial t_{yw_{(x)}}} \\ &= \frac{\partial e_y}{\partial p'}\frac{\partial p'}{\partial \lambda}\left(\frac{\partial \lambda}{\partial t_{yi_{(x)}}}\frac{\partial t_{yi_{(x)}}}{\partial t_{yw_{(x)}}} + \frac{\partial \lambda}{\partial t_{yi_{(y)}}}\frac{\partial t_{yi_{(y)}}}{\partial t_{yw_{(x)}}}\right)\end{aligned} \tag{3.35}$$

$$\frac{\partial e_y}{\partial r} = \frac{\partial e_y}{\partial p'}\frac{\partial p'}{\partial \lambda}\frac{\partial \lambda}{\partial r} \tag{3.36}$$

The Gauss-Newton equation is given by:

$$JJ^{\mathrm{T}}\Delta\xi = -Jf(\mathcal{Y}) \tag{3.37}$$

where $\Delta\xi$ is the increment. In every iteration, the increment $\Delta\xi$ can be obtained By solving this equation. The cylinder parameters are updated as follows:

$$\mathcal{Y}^{new} = \Delta\xi \oplus \mathcal{Y}^{old} = \begin{bmatrix} \exp\left(\boldsymbol{\Delta}\xi_{(\mathbf{1:3})}\right) * \mathbf{R}_{yw} \\ \Delta\xi_{(4)} + t_{yw_{(x)}} \\ \Delta\xi_{(5)} + r \end{bmatrix}^{\mathrm{T}}. \tag{3.38}$$

The system employs the Levenberg–Marquardt algorithm [146] to optimise the cylinder parameters in two stages. Given that vSLAM is aware of the uncertainty of features in the image level, the algorithm will utilise a kernel function to suppress outliers in the first stage. After several iterations, the system can identify outliers, specifically map points with errors surpassing the known uncertainty. Finally, the algorithm will discard the kernel function and outliers, optimising the cylinder parameters until the maximum iteration is reached.

## 3.6 Experiments

This section presents results from a series of experiments to evaluate the cylinder extraction methods in terms of precision, robustness to outliers, and computational efficiency. In these experiments, the proposed algorithms were applied to both simulated and real-world data, followed by a comparison of their performance in terms of estimating cylinder parameters, reprojection error $e_{rp}$, and computation time $C_t$. Controlled environments will be established to facilitate the comparison of the proposed method with other approaches. Controlled variables include the completeness of a cylinder, the size of the cylinder, the standard deviation of noise, the

distribution of outliers and the ratio of outliers. Throughout the experiments, all algorithms are executed on the same computer equipped with an Intel Core i7-8700 @ 3.2GHz processor and 32GB of memory.

In this work, various algorithms are tested on different datasets, and their performance is compared using different metrics. After multiple tests, the mean and variance (var in tables) of these metrics are compared. The error in the direction of the cylinder, denoted as $e_v$, is defined as the absolute angular difference between the estimated direction and the true direction. In the new cylinder representation, the parameter $t_{yw_{(x)}}$ can describe the distance from the axis to the origin. The error in the distance can be viewed as an error in the cylinder position. However, the cylinder representation requires both the axis and the $t_{yw_{(x)}}$ to calculate the cylinder position. When the direction of the cylinder is incorrect, a correct $t_{yw_{(x)}}$ does not indicate a correct position. So, the error in distance cannot independently evaluate a cylinder, and it is not used in this experiment.

Another important parameter for cylinders is the radius, and the corresponding error is the difference between the estimated and true radius, denoted as $e_{yr}$. The cylinder extraction algorithm will be applied to visual SLAM (vSLAM), and the proposed vSLAM algorithm evaluates cylinder information by using reprojection error $e_{rp}$, as shown in Equation 3.19. Therefore, the reprojection error is also employed here to assess performance.

Additionally, vSLAM is a real-time algorithm, making the computational cost of the algorithm crucial for this work. Thus, the computation time, denoted as $C_t$, for all methods will also be compared.

It is important to note that the cylinder detection algorithm cannot guarantee an accurate cylinder every time in the data. Therefore, the proportion of satisfactory cylinders is also considered here, denoted as $P_{\text{success}}$. The algorithms are tested on multiple point clouds under each condition. When an algorithm fails, its results deviate significantly from the ground truth, which can greatly affect the calculation of the mean and variance. Hence, only accurate cylinders are considered when calculating the mean. Given an accurate cylinder, the mean reprojection error of inlier features should be lower than a threshold according to the Chi-squared test. The variance for this metric is not reported in the tables.

To demonstrate the reliability, efficiency, and robustness of the proposed algorithms, this work will conduct a comparative analysis of various algorithms. The Repeated Least Trimmed Square (RLTS) algorithm [3] utilises PCA to estimate the orientation of a cylinder and employs a circle fitting method to estimate the position and radius. However, errors in the direction of the cylinder can potentially impact the performance of the circle fitting method. Section 3.3 introduces an ILS algorithm, which optimises all parameters of the cylinder simultaneously, thus mitigating the aforementioned issue. Experiments will be conducted to compare ILS and

RLTS and validate the superiority of ILS. Note that Symbol '—' in the below tables means the statistics are not available or not applicable.

Section 3.4 introduces numerical instability conditions that may mislead the RANSAC method [35]. Subsection 3.5.2 incorporates reprojection errors into the RANSAC method, leading to an improvement known as RANSAC2. Section 3.5.3 further optimises cylinder parameters by reducing the reprojection error, referred to as RANSAC2(O). Experiments will compare RANSAC, RANSAC2, and RANSAC2(O) to demonstrate the effectiveness of the new evaluation method based on the reprojection error.

### 3.6.1 Synthetic Data 1

The algorithm will run on various point cloud datasets generated by triangulation. While other studies may opt for selecting points on a cylinder as inputs [3, 147], this work will employ an approach similar to vSLAM for generating synthetic point clouds. Initially, a cylinder and several keyframes within the cylinder are created. In each experimental run, the position and orientation of the cylinder are randomly generated, with the radius $r$ and length $l$ determined as per experimental requirements. Inlier and outlier landmarks are generated, located on and inside the cylinder, respectively. These landmarks are projected onto keyframes to create features. The algorithm generates point clouds by triangulating these features, resulting in point clouds that resemble real data. It is assumed that the noise on features follows a Gaussian distribution with a default standard deviation $\sigma$ of 1. To limit computational complexity and obtain cylinder information as early as possible, the proposed vSLAM use a small number of map points to estimate the cylinder. Therefore, the number of map points is set to three hundred by default, with the assumption that 10% of them are outliers.

An inlier landmark on the cylinder's surface is defined by its angle $\theta \in (0, \alpha]$ and height-to-radius ratio $h_r \in [0, \frac{l}{r}]$, as shown in Figure 3.2. The angles and height-to-radius ratio are randomly generated within specified ranges as per experimental requirements, following a uniform distribution. Outlier landmarks are within the cylinder but not on its surface. As the camera moves within the pipe, objects or obstacles may be present on the inner wall of the pipe. In this experiment, non-cylindrical points are defined by their angle $\theta \in (0, \beta]$, height-to-radius ratio $h_r \in [0, \frac{l}{r}]$, and distance $d \in [0.25r0.75r]$ to the cylinder axis, as illustrated in Figure 3.2. Their coordinates are given by:

$$\begin{cases} x = d\cos\theta, \\ y = d\sin\theta, \\ z = h_r \cdot r \end{cases} \tag{3.39}$$

where the orientation angle $\theta$, the depth $h_r$, and the distance $d$ to the cylinder axis are all generated from uniform random distributions. In real-world scenarios, a horizontal pipe is often

not fully visible, and only the upper two-thirds of the horizontal pipe can be seen. In simulated data, $b$ is set to a default value of 240 degrees.

Under each condition, 200 sets of point clouds are generated based on the corresponding setting. The system executes all algorithms on the generated point clouds and records the estimation results.

### Complete and Incomplete Data

The experiments will evaluate the performance of algorithms on point clouds from an incomplete cylinder, as vSLAM often encounters such point clouds. Due to factors such as the texture of the pipeline, the camera's field of view, and the camera's orientation, the camera often cannot capture features of the complete pipe. vSLAM generates point clouds by triangulating features, which may not fully represent a cylinder. In the simulation, the point clouds are generated by adjusting the angle values while keeping the number of cylinder points constant. When Angle $\alpha$ is set to 360°, 240°, and 120°, respectively, the point clouds represent a complete cylinder, a two-third cylinder, and a one-third cylinder. Experimental data are given in Table 3.1.

Table 3.1 reveals that ILS achieves the best performance in estimating cylinders, with computation times similar to RLTS. RLTS demonstrates the second-best performance but often fails to estimate cylinders in some data. Both of these methods have similar computation times $C_t$, making them the most computationally efficient methods among those tested. As parameter $b$ decreases, the performance of all algorithms deteriorates. The errors of cylinders estimated by ILS show a slight increase, and, occasionally, the estimation fails when $\alpha = 120°$. RLTS appears to be less affected by variations in the data regarding error values, but its success rate significantly declines. In the case of incomplete cylinders, RLTS exhibits lower errors in estimating cylinder orientation compared to its performance with complete cylinders. This is because RLTS fits a circle to the projected points while being affected by both errors in the direction of the cylinder and the incomplete cylinder. As the cylinder becomes incomplete, circle fitting becomes more sensitive to axial errors. RLTS struggles to estimate an accurate circle to the projected points when axial errors increase. Additionally, the computation time for circle fitting also increases.

On the other hand, RANSAC requires more computational resources but performs poorly due to its challenge in finding accurate cylinders based on point-cylinder distance. RANSAC2 improves robustness by incorporating reprojection errors, though this comes at the expense of longer computation times. RANSAC2(O), which optimises results by minimising reprojection errors for all inliers, further enhances cylinder estimation. However, these methods lack robustness, as they often fail to provide accurate cylinders in most data. The RANSAC method faces more significant challenges in finding correct answers for incomplete cylinders. RANSAC2 and

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| $b_1 = 360$ | | | | | | | | | | |
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 0.72 | 0.40 | 1.96 | 1.01 | 2.33 | 1.47 | 2.02 | 1.27 | 3.28 | 1.24 |
| $e_{yr}$ | 0.04 | 0.04 | 0.20 | 0.23 | 0.72 | 1.02 | 0.83 | 1.19 | 1.95 | 0.84 |
| $e_{rp}$ (pixel) | 1.67 | 0.17 | 2.52 | 0.63 | 3 | 0.55 | 2.68 | 0.57 | 2.74 | 0.23 |
| $C_t$ (ms) | 3.81 | 4.30 | 3.35 | 1.77 | 70.82 | 4.42 | 95.40 | 10.35 | 14.69 | 0.27 |
| $P_{\text{success}}$ (%) | 100 | — | 68.5 | — | 11 | — | 19.5 | — | 1.5 | — |
| $b_1 = 240$ | | | | | | | | | | |
| $e_v$ (°) | 0.88 | 0.46 | 1.89 | 0.91 | 3.32 | 2.04 | 3.33 | 1.84 | 3.93 | 2.82 |
| $e_{yr}$ | 0.05 | 0.05 | 0.27 | 0.31 | 0.65 | 0.67 | 1.14 | 1.27 | 1.61 | 1.39 |
| $e_{rp}$ (pixel) | 1.71 | 0.18 | 2.55 | 0.58 | 2.77 | 0.52 | 2.72 | 0.5 | 2.93 | 0.87 |
| $C_t$ (ms) | 3.83 | 4.45 | 3.82 | 3.42 | 76.33 | 3.28 | 105.13 | 10.38 | 14.55 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 49 | — | 4 | — | 9 | — | 1 | — |
| $b_1 = 120$ | | | | | | | | | | |
| $e_v$ (°) | 0.92 | 0.48 | 1.62 | 1.03 | 3.58 | 0 | 2.82 | 0.70 | — | — |
| $e_{yr}$ | 0.20 | 0.22 | 0.29 | 0.31 | 1.57 | 0 | 2.31 | 1.85 | — | — |
| $e_{rp}$ (pixel) | 2.06 | 0.5 | 2.49 | 0.74 | 3.17 | 0 | 3.02 | 0.40 | — | — |
| $C_t$ (ms) | 4.45 | 4.57 | 6.07 | 6.46 | 72.39 | 0 | 111.66 | 11.65 | — | — |
| $P_{\text{success}}$ (%) | 98 | — | 10.5 | — | 0.5 | — | 2.5 | — | — | — |

Table 3.1: Performance of algorithms with data from several incomplete cylinders

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| $l = 30$ | | | | | | | | | | |
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 0.84 | 0.43 | 1.91 | 0.95 | 4 | 1.65 | 2.68 | 1.43 | 2.15 | 0 |
| $e_{yr}$ | 0.06 | 0.07 | 0.25 | 0.29 | 0.78 | 0.71 | 1.04 | 1.04 | 0.90 | 0 |
| $e_{rp}$ (pixel) | 1.70 | 0.17 | 2.54 | 0.65 | 3.25 | 0.47 | 2.60 | 0.49 | 2.85 | 0 |
| $C_t$ (ms) | 3.88 | 4.11 | 4.42 | 0.89 | 76.72 | 4.52 | 102.96 | 12.11 | 15.06 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 63.5 | — | 3.5 | — | 7 | — | 0.5 | — |
| $l = 20$ | | | | | | | | | | |
| $e_v$ (°) | 0.82 | 0.43 | 1.98 | 1.09 | 3.5 | 1.42 | 3.74 | 3.16 | 3.28 | 1.16 |
| $e_{yr}$ | 0.03 | 0.04 | 0.20 | 0.25 | 1 | 0.99 | 1.12 | 1.04 | 2.98 | 2.49 |
| $e_{rp}$ (pixel) | 1.76 | 0.20 | 2.51 | 0.58 | 3.12 | 0.55 | 2.64 | 0.55 | 3.46 | 0.28 |
| $C_t$ (ms) | 2.54 | 4.35 | 4.80 | 2.66 | 84.60 | 4.88 | 114.35 | 8 | 14.69 | 0.02 |
| $P_{\text{success}}$ (%) | 100 | — | 61.5 | — | 2 | — | 3 | — | 1 | — |
| $l = 10$ | | | | | | | | | | |
| $e_v$ (°) | 1.22 | 0.47 | 2.12 | 0.98 | 4.20 | 1.52 | 3.5 | 1.59 | — | — |
| $e_{yr}$ | 0.02 | 0.02 | 0.14 | 0.17 | 0.70 | 0.64 | 0.91 | 0.95 | — | — |
| $e_{rp}$ (pixel) | 1.84 | 0.17 | 2.5 | 0.57 | 2.98 | 0.70 | 2.66 | 0.66 | — | — |
| $C_t$ (ms) | 2 | 4 | 5.21 | 5.83 | 92.32 | 12.93 | 116.40 | 16.40 | — | — |
| $P_{\text{success}}$ (%) | 100 | — | 51.5 | — | 2.5 | — | 4 | — | — | — |

Table 3.2: Performance of algorithms on data from cylinders of different sizes

RANSAC2(O) perform slightly better than RANSAC.

**Cylinder with Different Size**

This subsection investigates the performance of cylinder extraction algorithms when handling point cloud data from cylinders of different sizes. In practice, pipes can have various sizes, and algorithms should be capable of estimating cylinders in pipes with different length-to-diameter ratios. Therefore, these experiments adjust the length of the cylinders to vary the length-to-diameter ratio. In the experiments, the length of the cylinders varies as per experimental requirements, while the radius and the number of point clouds remain constant. There are three scenarios, with cylinder lengths of 10, 20, and 30 units, while the radius remains at ten units, and the number of point clouds is set at 300. The algorithmic results are presented in Table 3.2.

Table 3.2 shows that ILS delivers the most precise cylinder fitting results with the lowest standard deviation. RLTS performs similarly to ILS but achieves accurate cylinder estimations in about half of the data. As cylinder lengths decrease, cylinder estimation performance deteri-

orates slightly, particularly with regard to radius errors. While ILS exhibits a slight increase in all errors, its computation time decreases. RLTS experiences an increase in errors in direction due to the shorter cylinders, but its reprojection errors show a slight reduction. The circle fitting method performs better when projecting point clouds from shorter cylinders.

Both RANSAC and RANSAC2 exhibit higher reprojection errors, with RANSAC2 displaying greater robustness and success in more cases. RANSAC2(O), an improvement over RANSAC2, yields slightly higher errors compared to RLTS. However, RLTS achieves far more successful cases than RANSAC2(O). RANSAC exhibits the highest reprojection error, particularly with point clouds from shorter cylinders, where the error becomes more pronounced. Additionally, this algorithm struggles to find accurate cylinders in shorter cylinders. While RANSAC2 occasionally identifies cylinders, its reprojection error $e_{rp}$ is similar to that of RANSAC. The reprojection errors of RANSAC2(O) remain almost constant across all three scenarios, with no noticeable trend in other metrics.

**The Level of Noise**

This part compares the performance of algorithms in datasets under different noise levels. In practice, the features extracted by vSLAM have uncertainties based on their pyramid levels. Errors are propagated to map points through triangulation and vary due to viewing angles and depths. Some point clouds with small viewing angles deviate significantly from their true positions.

To evaluate the algorithm's performance under varying levels of noise, the experiments introduce different levels of noise to the observed features. There are five datasets in total, and these result shows in Table 3.3. The noise follows a Gaussian distribution with standard deviations of 1, 2, 3, 4, and 5, respectively. As noise uncertainty increases, the threshold for reprojection errors also changes accordingly. When the noise standard deviation is $\sigma$, the threshold is set to $3.8414\sigma^2$ based on the Chi-squared test.

As noise increases, point clouds deviate further from their true positions, leading to an increase in the error in estimated cylinders from cylinder extraction algorithms. ILS exhibits worsening performance across all metrics due to increasing noise. Its computation load increases slightly because the optimisation algorithm requires more iterations to reject outliers. RLTS shows similar performance degradation, but the trend is less pronounced than ILS. When the standard deviation is 3, RLTS estimates cylinders more accurately than ILS. However, when noise is at its maximum, RLTS can provide accurate cylinders in only approximately 2% of cases, whereas ILS succeeds in estimating a sufficiently accurate cylinder in 75.5% of cases. ILS demonstrates greater robustness against noise and can provide an acceptable cylinder estimation in most situations.

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma = 1$ | | | | | | | | | | |
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 0.83 | 0.43 | 2.10 | 0.99 | 3.19 | 2.48 | 2.93 | 2.15 | 1.65 | 0 |
| $e_{yr}$ | 0.05 | 0.05 | 0.24 | 0.31 | 0.78 | 0.93 | 0.90 | 1.12 | 0.05 | 0 |
| $e_{rp}$ (pixel) | 1.70 | 0.18 | 2.55 | 0.59 | 2.76 | 0.62 | 2.57 | 0.57 | 2.20 | 0 |
| $C_t$ (ms) | 4.04 | 4.97 | 4.94 | 5.80 | 86.92 | 5.92 | 116.26 | 11.39 | 14.93 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 52 | — | 4.5 | — | 9 | — | 0.5 | — |
| $\sigma = 2$ | | | | | | | | | | |
| $e_v$ (°) | 2.12 | 1.09 | 2.20 | 1.19 | 8.28 | 6.13 | 12.35 | 8.96 | 3.88 | 0 |
| $e_{yr}$ | 0.13 | 0.14 | 0.42 | 0.49 | 2.67 | 2.48 | 2.62 | 2.56 | 4.51 | 0 |
| $e_{rp}$ (pixel) | 6.99 | 0.82 | 9.10 | 2.19 | 10.52 | 3 | 10.85 | 2.70 | 12.51 | 0 |
| $C_t$ (ms) | 5.84 | 4.57 | 5.21 | 3.29 | 84.86 | 2.92 | 92.96 | 4.51 | 16.97 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 69 | — | 3.5 | — | 4 | — | 0.5 | — |
| $\sigma = 3$ | | | | | | | | | | |
| $e_v$ (°) | 3.82 | 1.53 | 3.06 | 1.38 | 7.97 | 8.94 | 4.56 | 1.73 | 0.33 | 0 |
| $e_{yr}$ | 0.21 | 0.24 | 0.63 | 0.73 | 3.80 | 3.94 | 3.22 | 3.46 | 0.55 | 0 |
| $e_{rp}$ (pixel) | 16.31 | 2.19 | 20.19 | 5.30 | 25.74 | 5.38 | 25.70 | 4.78 | 28.41 | 0 |
| $C_t$ (ms) | 6.86 | 5.04 | 5.5 | 1.90 | 90.03 | 10.68 | 96.73 | 11.42 | 15.12 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 64.5 | — | 6.5 | — | 6 | — | 0.5 | — |
| $\sigma = 4$ | | | | | | | | | | |
| $e_v$ (°) | 6.19 | 1.83 | 4.47 | 1.67 | 11.78 | 11.37 | 10.97 | 11.18 | 2.70 | 0 |
| $e_{yr}$ | 0.29 | 0.33 | 0.68 | 0.87 | 3.10 | 3.38 | 2.61 | 2.5 | 7.72 | 0 |
| $e_{rp}$ (pixel) | 32.80 | 7.59 | 36.82 | 9.40 | 48.78 | 9.30 | 47.82 | 9.14 | 32.02 | 0 |
| $C_t$ (ms) | 7.66 | 4.72 | 6.74 | 4.37 | 87.61 | 6.96 | 89.64 | 6.59 | 15.94 | 0 |
| $P_{\text{success}}$ (%) | 97.5 | — | 33.5 | — | 10 | — | 8.5 | — | 0.5 | — |
| $\sigma = 5$ | | | | | | | | | | |
| $e_v$ (°) | 8.24 | 1.92 | 5.52 | 1.79 | 17.52 | 14.16 | 16.85 | 14.33 | 4.09 | 0 |
| $e_{yr}$ | 0.41 | 0.43 | 0.82 | 0.68 | 3.46 | 3.80 | 3.07 | 3.15 | 2.29 | 0 |
| $e_{rp}$ (pixel) | 56.99 | 14.92 | 50.13 | 3.53 | 76.94 | 13.71 | 77.43 | 12.78 | 79.56 | 0 |
| $C_t$ (ms) | 9.20 | 5.99 | 6.13 | 0.75 | 85.82 | 5.78 | 88.88 | 5.69 | 15.29 | 0 |
| $P_{\text{success}}$ (%) | 75.5 | — | 2 | — | 14 | — | 14.5 | — | 0.5 | — |

Table 3.3: Performance of algorithms on data of different uncertainty

RANSAC, RANSAC2, and RANSAC2(O) perform poorly when affected by noise, with errors significantly higher than other algorithms. Their computation time remains unaffected by noise. RANSAC successfully estimates a cylinder in only one case in every dataset. RANSAC2, which incorporates reprojection errors, can find an acceptable cylinder in more cases but exhibits slightly higher errors in the direction and radius than RANSAC. RANSAC2(O) optimises the results of RANSAC2 but improves only the errors in cylinder parameters.

**Influence of Outliers**

Outliers may affect the cylinder extraction algorithms, and the experiment will adjust the generation range of outliers to test the algorithms. Various objects may appear inside the pipe, and point clouds from these objects can interfere with cylinder estimation. In addition to the quantity of outliers, the experiment can also choose the location of outliers. Sometimes, outliers may concentrate in a small area, while at other times, they may be distributed in multiple locations. Nurunnabi et al. claimed that clustered outliers have a more significant impact on cylinder estimation than scattered outliers [3]. The experiment created three datasets to test the effect of the type of outliers on the algorithm. The point cloud distribution was controlled by adjusting parameter $\beta$. The $\beta$ values for the datasets are 120°, 60°, and 30°, respectively. As $\beta$ decreased, the outliers gradually clustered. The experimental data is as follows:

Table 3.4 shows that the error in cylinder estimation did not increase due to the concentration of outliers, demonstrating that the distribution of outliers did not affect the performance in this experiment. Although there is some fluctuation in some statistics, it is not significant enough. In the experimental results, ILS performed the best. RLTS was second to ILS but could estimate accurate cylinders in only half of the cases. The RANSAC algorithm can hardly detect cylinders. RANSAC2 had slightly lower errors in cylinder parameters than RANSAC2(O), but the latter could obtain accurate cylinders in more cases.

**Robustness to Outliers**

This experiment evaluates the robustness of the algorithm in the presence of a significant number of outliers. When the ratio of outliers is high, algorithms can be disrupted or even fail. Outliers are common in point clouds generated by vSLAM. Some outliers originate from objects inside the pipeline. Even in clean, ideal pipelines, point clouds can contain outliers. Due to errors in feature matching, map points generated by the triangulation method deviate from ground truth. Additionally, the proportion of outliers in the point cloud is unknown, so vSLAM requires an algorithm that is robust to outliers.

This experiment established five datasets, each with a different proportion of outliers. In each case, the total number of points in the point cloud is 300, with outlier proportions $\gamma$ of

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\beta = 120°$ | | | | | | | | | | |
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 0.89 | 0.48 | 1.86 | 0.96 | 2.78 | 1.80 | 3.29 | 1.78 | 1.01 | 0 |
| $e_{yr}$ | 0.05 | 0.06 | 0.28 | 0.32 | 0.80 | 0.75 | 0.90 | 1.10 | 0.03 | 0 |
| $e_{rp}$ (pixel) | 1.70 | 0.19 | 2.54 | 0.63 | 2.76 | 0.53 | 2.98 | 0.64 | 3.24 | 0 |
| $C_t$ (ms) | 4.42 | 4.58 | 5.14 | 1.06 | 86.83 | 5.21 | 113.80 | 13.84 | 31.95 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 48 | — | 2.5 | — | 10.5 | — | 0.5 | — |
| $\beta = 60°$ | | | | | | | | | | |
| $e_v$ (°) | 0.89 | 0.43 | 1.93 | 1.06 | 3.29 | 1.67 | 3.09 | 2.25 | — | — |
| $e_{yr}$ | 0.05 | 0.06 | 0.31 | 0.34 | 0.95 | 0.95 | 1.36 | 1.95 | — | — |
| $e_{rp}$ (pixel) | 1.74 | 0.17 | 2.65 | 0.60 | 2.94 | 0.26 | 2.71 | 0.41 | — | — |
| $C_t$ (ms) | 4.05 | 4.52 | 4.62 | 2.12 | 81.53 | 3.62 | 108.54 | 10.47 | — | — |
| $P_{\text{success}}$ (%) | 100 | — | 53.5 | — | 3.5 | — | 9.5 | — | — | — |
| $\beta = 30°$ | | | | | | | | | | |
| $e_v$ (°) | 0.85 | 0.43 | 1.86 | 0.99 | 2.24 | 1.41 | 2.55 | 1.66 | 3.19 | 0 |
| $e_{yr}$ | 0.06 | 0.06 | 0.27 | 0.32 | 0.51 | 0.49 | 1.09 | 1.29 | 1.21 | 0 |
| $e_{rp}$ (pixel) | 1.71 | 0.19 | 2.55 | 0.62 | 2.86 | 0.55 | 3 | 0.57 | 2.62 | 0 |
| $C_t$ (ms) | 4.14 | 4.26 | 5.06 | 3.89 | 77.76 | 4.62 | 106.76 | 12.47 | 15.91 | 0 |
| $P_{\text{success}}$ (%) | 100 | — | 56 | — | 2.5 | — | 8.5 | — | 0.5 | — |

Table 3.4: Performance of algorithms on data with outliers from different distributions

10%, 20%, 30%, 40%, and 50%, respectively. The experimental results are shown in Table 3.5.

With an increase in the number of outliers, all algorithms are affected. ILS showed a slight increase in error and standard deviation. As outliers increased, the optimisation algorithm required more iterations to exclude outliers, leading to a slight increase in computation time. However, ILS remained the best-performing algorithm. Similarly, RLTS was also slightly affected, with performance second only to ILS. However, its success rate sharply declined to 21% as the proportion of outliers increased to 50%. ILS demonstrated robustness against the increase in outliers and could handle cases with more outliers.

The RANSAC algorithms are more significantly affected by the increase in outliers and could not even find acceptable cylinders. RANSAC can have inliers to estimate a cylinder if the number of iterations is large enough. RANSAC2 and RANSAC2(O) can successfully estimate cylinders in more cases, indicating that RANSAC failed to identify the optimal method based on point-cylinder distance. RANSAC2, using reprojection errors, can find a cylinder closer to the ground truth in more cases, demonstrating that the reprojection error is a more suitable metric. RANSAC2(O) optimises the results from RANSAC2 but with less pronounced effects.

### 3.6.2 Synthetic Data 2

This subsection compares the performance of cylinder extraction algorithms in point clouds generated from ORB-SLAM2. A synthetic pipe network is constructed in Gazebo, and a synthetic robot equipped with a camera is deployed to navigate within the pipeline. The robot's body is a short and vertical cylinder with two drive wheels and one caster wheel underneath. A monocular camera is mounted on top of the cylinder. The synthetic camera captured the interior walls of the pipe, resulting in a total of three videos. The simulation details can be found in Section 4.8. ORB-SLAM2 runs thirty times using the video data, generating point clouds. The initial 300 points from each point cloud and the first three keyframes are used as input for the cylinder extraction algorithms. In this simulation environment, the actual sizes and orientations of the cylinders are known.

This experiment introduces a different approach to evaluating estimate cylinders due to the uncertainty regarding whether map points are inliers. The synthetic pipe is clean and straight, and all features extracted from the pipe walls are inliers. However, triangulated features may not necessarily be inliers. Map points might deviate significantly from the pipe walls due to errors in feature matching, leading to their classification as outliers. Consequently, it is possible to compute reprojection errors for features, but calculating the average reprojection error for inliers is not feasible. Given that the camera moves within the pipeline, most map points are expected to be inliers. When the reprojection error of most map points falls below their corresponding threshold, the cylinder is considered accurate. This experiment compares different algorithms

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\gamma = 10\%$ | | | | | | | | | | |
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 0.85 | 0.45 | 1.69 | 0.86 | 4 | 2.88 | 4.41 | 2.66 | 2.72 | 1 |
| $e_{yr}$ | 0.06 | 0.08 | 0.24 | 0.29 | 0.66 | 0.74 | 1.78 | 2.53 | 3.48 | 1.74 |
| $e_{rp}$ (pixel) | 1.70 | 0.17 | 2.51 | 0.64 | 2.93 | 0.39 | 3.02 | 0.45 | 2.52 | 0.03 |
| $C_t$ (ms) | 3.95 | 4.57 | 4.89 | 2.40 | 73.66 | 1.21 | 96.93 | 11.21 | 16.73 | 2.74 |
| $P_{\text{success}}$ (%) | 100 | — | 54.5 | — | 2 | — | 6 | — | 1 | — |
| $\gamma = 20\%$ | | | | | | | | | | |
| $e_v$ (°) | 0.98 | 0.48 | 1.89 | 0.96 | 2.17 | 1.52 | 2.84 | 1.07 | — | — |
| $e_{yr}$ | 0.05 | 0.06 | 0.26 | 0.27 | 0.99 | 0.96 | 0.71 | 0.69 | — | — |
| $e_{rp}$ (pixel) | 1.72 | 0.20 | 2.79 | 0.65 | 3.11 | 0.03 | 3.07 | 0.61 | — | — |
| $C_t$ (ms) | 4.39 | 4.78 | 5.13 | 2.03 | 72.52 | 0.57 | 92.82 | 9.25 | — | — |
| $P_{\text{success}}$ (%) | 100 | — | 41.5 | — | 1 | 0 | 4.5 | 0 | — | — |
| $\gamma = 30\%$ | | | | | | | | | | |
| $e_v$ (°) | 0.97 | 0.55 | 2.13 | 1 | — | — | 2.83 | 0 | — | — |
| $e_{yr}$ | 0.06 | 0.07 | 0.32 | 0.35 | — | — | 0.76 | 0 | — | — |
| $e_{rp}$ (pixel) | 1.77 | 0.23 | 2.95 | 0.53 | — | — | 3.18 | 0 | — | — |
| $C_t$ (ms) | 4.81 | 4.41 | 5.74 | 6.84 | — | — | 81.60 | 0 | — | — |
| $P_{\text{success}}$ (%) | 99.5 | — | 29 | — | — | — | 0.5 | 0 | — | — |
| $\gamma = 40\%$ | | | | | | | | | | |
| $e_v$ (°) | 1.08 | 0.58 | 2.86 | 1.5 | 7.85 | 3.65 | 8.13 | 3.25 | — | — |
| $e_{yr}$ | 0.07 | 0.09 | 0.28 | 0.29 | 2.85 | 2.64 | 4.01 | 3.39 | — | — |
| $e_{rp}$ (pixel) | 1.74 | 0.26 | 3.10 | 0.62 | 3.07 | 0.41 | 2.68 | 0.06 | — | — |
| $C_t$ (ms) | 5.07 | 4.74 | 5.18 | 0.76 | 74.46 | 3.41 | 89.73 | 5.48 | — | — |
| $P_{\text{success}}$ (%) | 100 | — | 23.5 | — | 1 | 0 | 1.5 | 0 | — | — |
| $\gamma = 50\%$ | | | | | | | | | | |
| $e_v$ (°) | 1.19 | 0.71 | 2.88 | 1.55 | — | — | — | — | — | — |
| $e_{yr}$ | 0.09 | 0.11 | 0.30 | 0.38 | — | — | — | — | — | — |
| $e_{rp}$ (pixel) | 1.81 | 0.28 | 3 | 0.63 | — | — | — | — | — | — |
| $C_t$ (ms) | 5.71 | 4.61 | 5.66 | 2.89 | — | — | — | — | — | — |
| $P_{\text{success}}$ (%) | 99.5 | — | 21 | — | — | — | — | — | — | — |

Table 3.5: Performance of algorithms on data with varying numbers of outliers

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_v$ (°) | 2.67 | 0.99 | 3.07 | 0.97 | 5.17 | 2.45 | 4.39 | 2.25 | 3.78 | 0.69 |
| $e_{yr}$ | 0.18 | 0.19 | 0.61 | 0.63 | 1.35 | 1.34 | 1.51 | 1.52 | 3.84 | 3.59 |
| $C_t$ (ms) | 5.87 | 4.24 | 5 | 2.66 | 79.10 | 6.04 | 90.77 | 9.02 | 13.98 | 0.11 |
| $P_{\text{success}}$ (%) | 100 | — | 67 | — | 10 | — | 13.30 | — | 3.30 | — |

Table 3.6: Performance of algorithms on synthetic data 2

| Method | ILS | | RLTS | | RANSAC2 | | RANSAC2(O) | | RANSAC | |
|---|---|---|---|---|---|---|---|---|---|---|
| | mean | var | mean | var | mean | var | mean | var | mean | var |
| $e_{rp}$ (pixel) | 18.78 | 3.60 | 21.43 | 5.13 | 36.89 | 5 | 35.28 | 4.66 | — | — |
| $C_t$ (ms) | 9.85 | 4.22 | 8.65 | 1.27 | 73.12 | 5.04 | 76.69 | 4.67 | — | — |
| $P_{\text{success}}$ (%) | 100 | — | 50 | — | 30 | — | 40 | 0 | — | — |

Table 3.7: Performance of algorithms on real data

based on their success rate. The result of the experiment is demonstrated in Table 3.6.

Table 3.6 illustrates that ILS performed the best across all metrics, excluding computation time, and consistently successfully estimated the cylinders in all data. RLTS has slightly lower computation times but higher errors in estimated cylinders. It fails to estimate the cylinder in one-third of the data. RANSAC produced only one success in cylinder extraction, while RANSAC2, utilising the reprojection error, improved robustness and provided precise cylinder estimates in three instances. RANSAC2(O) slightly improve the results of RANSAC2 through optimisation. The results demonstrate that reprojection error is a more suitable metric for evaluating estimated cylinders in visual SLAM algorithms.

### 3.6.3 Real Data

The experiment tests algorithms on real data and compares their success rate and computation time. The real data comprises a set of images captured by a camera mounted on a robot in the pipes [6]. The images have a resolution of 720×576 pixels and are captured at a frame rate of 25fps. Section 4.8 provides specific details about the dataset. Using these images as input, ORB-SLAM2 is executed ten times, resulting in keyframe poses and sparse point clouds. The experiment utilised keyframes and point clouds from the initial portion of the dataset since it suffers from less scale drift. The first 300 points and the first three keyframes are used as input for the cylinder extraction algorithms. The experimental results are shown in Table 3.7.

The experiment introduces a different approach to evaluating the performance of the algorithms on real data. The proposed vSLAM cannot estimate the scale factor and only completes

initialisation when the camera moves. Consequently, the size, orientation, and position of the pipe in the map generated by the vSLAM still need to be discovered. Additionally, whether feature points or map points are inliers is uncertain, making it impossible to compare the algorithm based on the mean reprojection error of inliers. The experiment compares different algorithms based on their success rate in the real dataset.

Table 3.7 shows that ILS performed the best and can accurately estimate the cylinder under complex conditions. RLTS managed to estimate cylinders in five cases with the shortest computation time. The standard RANSAC algorithm did not achieve a good result. However, when RANSAC uses the reprojection error to select the better solution, it successfully estimated cylinders in three cases at the cost of more computation time. RANSAC2(O) further refined the results obtained from RANSAC2, successfully estimating cylinders in four cases. The experiments demonstrate that the algorithm involving all cylinder parameters in optimisation performs better than the one optimising only part of the estimated cylinder's parameters. The reprojection error proves to be a more robust performance metric than the point-cylinder distance.

## 3.7 Summary

Leveraging a novel representation of cylinders, the algorithms proposed in Section 3.3 iteratively optimise all cylinder parameters. This optimisation approach reduces the dependency on the initial value of the direction of the cylinder axis, allowing the use of more robust but less accurate techniques for initial axis estimation. The results demonstrate that the proposed methods offer advantages in terms of robustness, efficiency, and accuracy.

Observing the estimation of cylinder parameters by the optimisation method, this chapter summarised the numerical instability conditions that can mislead algorithms reliant on point-cylinder distances. A novel evaluation method has been introduced in Section 3.4, suitable for use within visual SLAM algorithms. This method can leverage information from keyframes and features, which are typically absent in general point cloud data. The RANSAC method, which depends on point-cylinder distances to find optimal solutions, is often misled by numerical instability conditions. The evaluation method developed in this Section 3.4 has been adapted and applied to the RANSAC method, enhancing its performance. In Section 3.5.3, this work also proposed a new optimisation technique that estimates cylinder parameters by minimising reprojection errors. These methods, although not comparable to IRS, have improvements over traditional RANSAC approaches. Existing algorithms always rely on point clouds for cylinder estimation, whereas the new algorithms proposed use image information. Avoiding the errors introduced through triangulation, these algorithms have the potential to achieve higher precision

estimates.

# Chapter 4

# Leveraging Cylindrical Regularity in ORB-SLAM

## 4.1 Introduction

Visual SLAM (vSLAM) [148] is one appealing approach for SLAM in pipes because cameras are low-cost, provide detailed information about the environment, and are commonly found on pipe inspection robots [17, 149, 150, 151]. Direct method [15] tends to be more fragile. It sometimes has significant drift during initialisation (refer to Subsection 4.8.1), and frequently fails due to changes in illumination. As a result, this work develops a new feature-based SLAM approach based on ORB-SLAM2 [2]. ORB-SLAM2 is a classic algorithm which has numerous variants derived by many researchers [152, 153, 154]. However, vSLAM algorithms can deteriorate in pipe environments because the scale factor decreases along with the forward motion of the robot (refer to Section 4.8). ORB-SLAM2 can run with data [6] from a sewer pipe, and the results are shown in Figure 4.1. Figure 4.1 A and B show the starting and ending images of the pipe inspection along with the corresponding point cloud. Figure 4.1C presents the overall map with two detailed views. The reduction in the scale of the point cloud is evident in the figure. The point cloud output of the vSLAM algorithm resembles a cone, with the diameter ($d_A$&$d_B$) shrinking as the camera moves forward: the accuracy becomes worse the farther the robot moves.

Human-made environments have rich structural information, and many researchers tend to leverage structural information to improve SLAM algorithms in terms of accuracy [89, 83, 74, 78]. Current researchers exploit structural information by building *Manhattan worlds* or enforcing structural regularities on landmarks [72, 64].

A Manhattan world assumes a human-made environment can be abstracted as a block with three dominant directions [155]. Such algorithms can extract lines in images, subsequently reconstructing these 3D lines to find the dominant directions for the Manhattan world. Frames

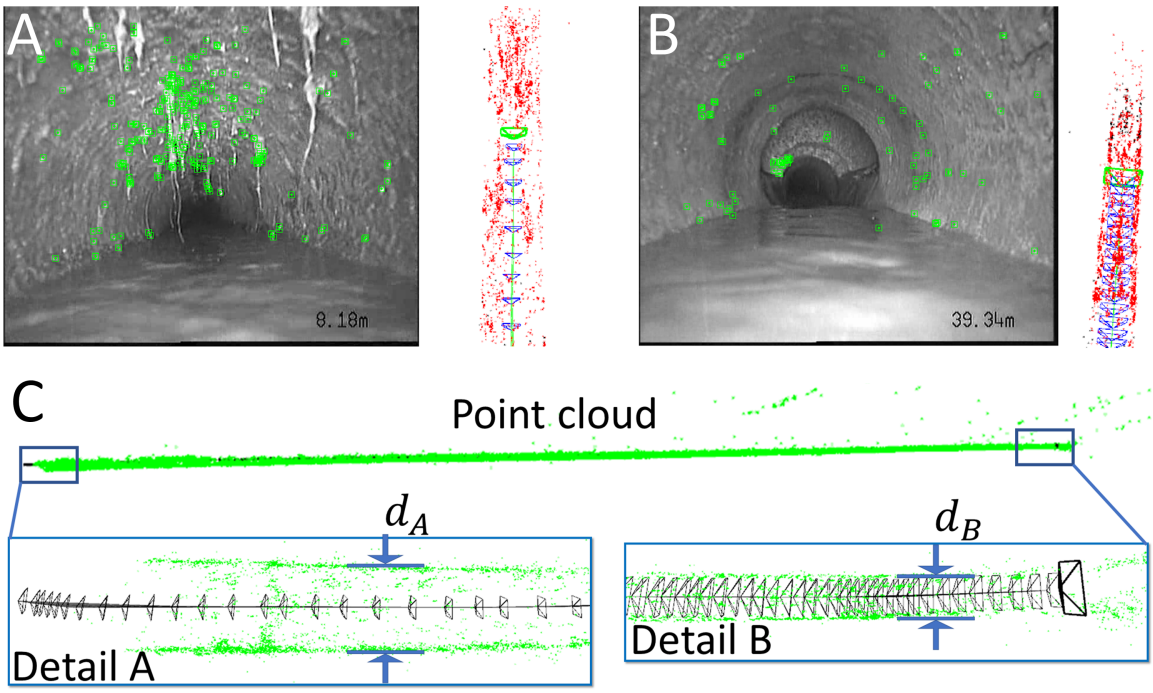



Figure 4.1: Scale factor decrease over time using ORB-SLAM2 in a pipe

within the same Manhattan world can have their relative orientations to the same estimated Manhattan world. Given this, the system can build constraints on the rotation of frames in the estimated Manhattan world. In addition to minimising reprojection errors in the backend, the system also maintains the relative rotation between the keyframes and the Manhattan world.

Some vSLAM algorithms enforce structural constraints on landmarks, which punish the distance or direction of landmarks deviating from the expected value [85, 74]. The algorithms can extract point features, vanishing points, lines and planes on the images, subsequently identifying relations among them. Given the interrelations between points, lines, and planes, the algorithm imposes constraints on point-to-lines, point-to-plane, and line-to-plane distances.

A key structural property of pipe environments is their cylindrical shape, but few SLAM algorithms have been adapted for the pipe environment. Kahi et al. have proposed an offline SfM technique capable of detecting cylinders, but the method seldom uses cylindrical information [14]. A handful of algorithms use the knowledge of cylinders for triangulation or pose estimation [27, 17, 28, 31]. However, these assume that a camera moves along the cylinder axis within the pipeline, maintaining a constant orientation, which is not feasible in this thesis. Hansen et al. proposes an algorithm for gas pipeline visual inspection, simultaneously optimises pose and extracts a cylinder in the backend [30]. But, this method is not robust enough, especially in a complex environment with obstacles in the pipes. Similarly, Kagami et al. also integrates cylindrical constraints to rectify point cloud positions, but the cylinder extraction method does

not function well in complex environments [26]. Moreover, Kagami et al. imposes cylindrical constraints on point clouds, but the method lacks an efficient mechanism to strike a balance between the point-plane distance and the reprojection error [26].

Therefore, to fill the research gap, this chapter proposes a method for effectively detecting and utilising cylindrical information within vSLAM.

In summary, this chapter proposes a new vSLAM algorithm that leverages cylindrical regularity, which is termed CRORB. The approach is an extension of ORB-SLAM2 [2]. The main contributions of this chapter include:

1. a new, more robust cylinder detection algorithm in vSLAM algorithm
2. a vSLAM algorithm that leverages cylindrical information in bundle adjustment and pose graph optimisation
3. an approach to recover the scale factor based on the isodiametric information.

The chapter is organised as follows. Section 4.2 provides an overview of the proposed algorithm. Section 4.3 introduces the cylinder detection algorithm. Section 4.4 presents a method for identifying cylindrical points and extending the detected cylinders to new keyframes. Section 4.5 introduces the vSLAM algorithm, which leverages cylinder information in local optimisation, while Section 4.6 focuses on pose graph optimisation. In Section 4.7, a novel approach is proposed that utilises isodiametric information for scale recovery. The final section evaluates the performance of the proposed algorithm on real and simulated data, validating its effectiveness.

## 4.2 System Overview

This chapter proposes a system that can detect and utilise cylindrical information. Compared to previous research [30, 26], this approach not only uses cylindrical information in local optimisation but also triangulation and pose graph optimisation. The system is built upon ORB-SLAM2 [5], running three parallel threads: tracking, local mapping, and loop closing.

In the front end, the system tracks features and estimates the current frame pose in real time. It extracts ORB features [119] and relies on BRIEF descriptors [61] to track these features between the current and previous frames. During the initialisation, the system calculates the relative pose between two frames and triangulates matched features to obtain initial map points. Subsequently, the system uses a motion-only Bundle Adjustment (BA) to estimate the current frame pose based on existing map points and matched features. If tracking fails, the system runs global relocalisation to recover feature tracking and pose estimation. The module is also responsible for selecting keyframes among frames. If the current frame has enough new features

or the system has not accepted a new keyframe for an extended period, the current frame is chosen as a keyframe. The details of the front end are described in ORB-SLAM2 [1, 2]. To ensure real-time estimation, no additional modifications were made to this work. Both the cylinder extraction method and the algorithm for utilising cylindrical information run on the back end.

After adding a keyframe, the local mapping module processes the new keyframe, creates map points, and performs local optimisation. The system establishes correspondences between the new keyframe and its covisible keyframes using a BOW model [156] and triangulates matched features. Covisible frames refer to keyframes that share features with the current frame. Local optimisation refines the current frame pose, covisible keyframe poses, and all relevant map points by reducing reprojection errors. For cylinder detection, the system runs the cylinder extraction algorithm among the map points during the local mapping, as detailed in Section 4.3. To ensure correctness, the system consistently verifies the cylinder information, mitigating potential estimation errors arising from incorrect cylinders. Once accurate cylinder information is obtained, the system incorporates the cylinder information in both triangulation and local optimisation. Triangulation suffers from low parallax angles in the pipeline, but cylindrical information can mitigate this problem. A new triangulation method proposed here also selects cylindrical points, described in Section 4.4. Cylindrical points refer to points from the cylindrical surface. Section 4.5 details incorporating cylindrical information in local optimisation can rectify the positions of these cylindrical points.

The loop closing thread receives the keyframe from local mapping and uses pose graph optimisation to correct keyframe poses. The system searches for loops with every new keyframe using BOW, verifying the result by point cloud registration. Once loop closure detection is successful, the system performs pose graph optimisation to achieve global consistency. This work also added constraints between the cylinder and the keyframes, allowing for optimisation of cylinder parameters and improvement of pose estimation. Section 4.6 introduces this method in detail.

The work presents a method in Section 4.7 that can use the isodiametric information to correct scale drift. When the camera moves in a pipe with a constant diameter, the point cloud created by the vSLAM algorithm resembles a cone instead of a cylinder. Due to scale drift, vSLAM tends to underestimate the travel distance. The algorithm will estimate the scale of the related keyframes based on the estimated diameters of detected cylinders and then estimate the keyframe poses through pose graph optimisation.

## 4.3 Cylinder Detection

Cylinder detection attempts to estimate a cylinder among the map points and select cylindrical points using the method proposed in Section 3.3. Refer to the Algorithm 1 for detailed steps. The proposed vSLAM extracts the cylinders and then utilises the cylinder information in local optimisation. Given the significant impact of the cylinder on future pose estimation, a robust method for cylinder detection is required. This algorithm formulates a loss function based on the assumption that all cylindrical points maintain an equidistant relation to the axis, as illustrated in Equation 3.8. The cylinder extraction algorithm is a part of cylinder detection, which also includes initial value computation, selection of input point clouds, and verification of the cylinder. The system will run the cylinder detection algorithm after local optimisation and verify the estimated cylinder in the local map points after the estimation. Local optimisation not only improves the position of the local map points but also removes outliers. In this way, the algorithm can test the estimated cylinder with more accurate map points.

Before incorporating cylindrical information, the system employs the optimisation technique detailed in Section 3.3. To mitigate the influence of outliers in cylinder detection, the system selects optimised points observed multiple times as inputs. Optimised points refer to map points that have participated in past local optimisations but are not involved in the current local optimisation. Their positions will remain the same in the future local optimisation. Local optimisation alters the positions of local map points, so a cylinder estimated using these points might not fit the local map points after optimisation. Consequently, the system only uses previously optimised points that remain static. Additionally, as outliers are less likely to be observed repeatedly, the system requires the points observed by multiple keyframes.

The system continually accumulates optimised points as input for cylinder detection in the backend. When the number of accumulated optimised points reaches a threshold, the system initiates the optimisation algorithm for cylinder extraction. Once a cylinder is successfully estimated, all inlier optimised points are confirmed as cylindrical points. Otherwise, half of the optimised points are discarded, and the system continues to collect new optimised points for future cylinder detection.

The cylinder extraction algorithm is an iterative optimisation algorithm where the initial values significantly impact the algorithm's convergence. The proposed algorithm employs PCA to estimate the cylinder's orientation. It takes the positions of covisible keyframes of the current keyframe as input. As the robot moves along the pipeline, the camera's trajectory approximates the pipe axial. The mean position of the covisible keyframes is used as the cylinder's position. The algorithm transforms the aforementioned values into the new cylindrical representation as the initial values for the cylinder extraction algorithm. This method is more robust than PCA, which uses point clouds as input. Compared to point clouds, frame poses have almost no outliers.

When the system produces a seriously wrong frame pose, the vSLAM algorithm is likely to fail.

After the cylinder extraction, the system verifies the estimated cylinder by comparing the proportion of inliers to optimised points and inliers to local map points in the local optimisation. During estimation, the iterative optimisation algorithm gradually excludes outliers, and the remaining map points are considered inliers. The system estimates the cylinder to leverage its information in future estimations, and the estimated cylinder should be able to fit most of the local point cloud. Thus, the system calculates the proportions of inliers in optimised points and local map points. The camera is not within the pipe if the inlier proportion is below a threshold. However, if the inlier proportion exceeds 99%, the system might encounter a numerical instability condition introduced in Section 3.4. Even in a simulation, the vSLAM algorithm has outliers of more than 1%. Hence, this result likely is an erroneous large cylinder, and the system will discard that estimation result.

In a pipeline network, vSLAM can detect cylinders in multiple pipelines, marking the cylinder from the current pipe where the camera is located as activated. After obtaining the cylinder using the method mentioned above, the system will utilise the activated cylinder in the back end and extend it to new map points created when a new keyframe is inserted. As the camera moves, it will explore new environments. It is unknown if the camera is still within the same pipe. Therefore, the system will check the cylinder information whenever a keyframe is inserted into the mapping. If the cylinder fails verification, the system will deactivate this cylinder. The system will then accumulate new optimised points and prepare for a new cylinder estimation. The extension of the cylinder will be detailed in the next section.

## 4.4 Triangulation

This section explains how the algorithm incorporates cylindrical information in triangulation and selects cylindrical points. The new triangulation method helps mitigate issues arising from low parallax angles, which can affect the estimation of map points. The new method also identifies cylindrical points according to their reprojection errors.

The system uses cylindrical information to improve the original triangulation method in ORB-SLAM2 [2]. The original algorithm employs BOW [156] to establish correspondence between the current keyframe and covisible keyframes and create new map points by triangulating the tracked features. The algorithm calculates the backprojected rays of the tracked features and uses their intersection as positions of the map points.

When the parallax angle, specifically the angle between the rays, is small, there is more significant uncertainty in the position of the map point. As shown in Figure 4.2, there are two frames (Frame 0 and Frame 1) next to the pipe wall. They are able to observe two landmarks: one

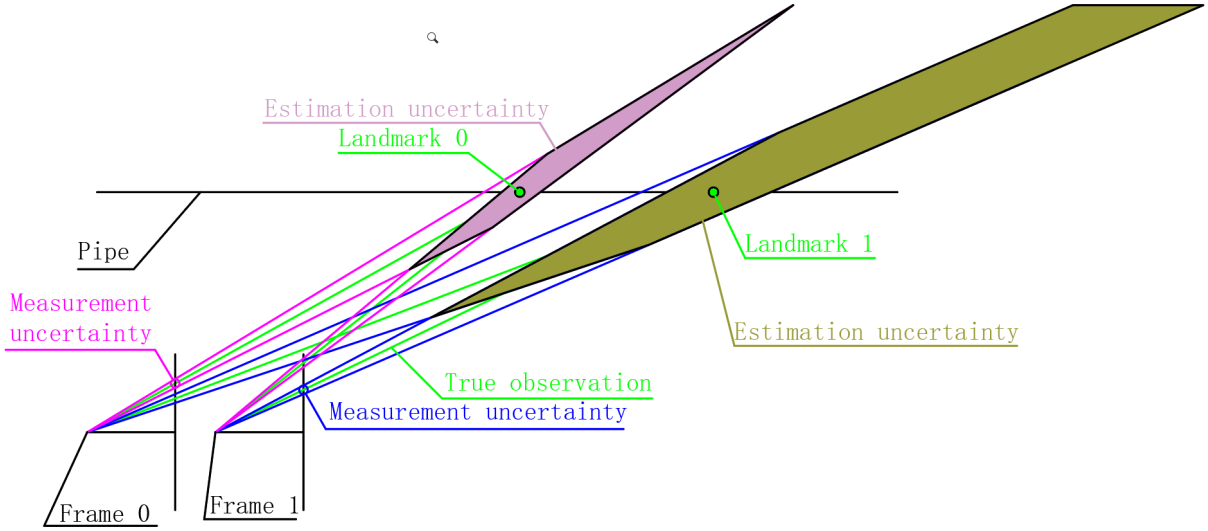



Figure 4.2: Parallax angles for distant and near landmarks

closer (Landmark 0) and the other more distant (Landmark 1). The algorithm can estimate these landmarks' positions by computing the corresponding rays' intersection. Due to observation uncertainties, the camera observes the features around their true values, and the direction of the corresponding backprojected rays is very close to that of the actual backprojected rays. However, the filled polygons in the figure represent the potential areas where the intersections of the rays might occur. The yellow area is more significant even though it is not fully displayed due to the limitations in image size. So, distant landmarks in pipes tend to have greater positional uncertainty.

The distant landmarks exhibit low parallax angles when the camera navigates within a pipe. These small parallax angles render triangulation more sensitive to noise, leading to reduced accuracy in mapping distant points compared to their closer points. Moreover, as the camera moves forward, these front landmarks get closer until the camera eventually passes them. Consequently, distant landmarks tend to remain within the camera's field of view longer than near landmarks. Their influence on the accuracy of the trajectory is more significant. This work uses the intersection between the ray and the cylindrical surface over that of the rays to estimate the positions of cylindrical points, given that the former yields a larger included angle.

Before applying the cylindrical information, the system verifies the reliability of the cylinder. When the cylindrical detection method yields an imprecise cylinder or the camera moves to a different pipe, the estimated cylinder cannot fit the current map points. Incorrect cylindrical information can mislead both estimations of pose and map point. The system evaluates a cylinder by calculating its reprojection error $e_i$, as described in Section 3.4. When the algorithm computes the reprojection error for a feature, it first calculates the intersection of the backprojected ray and the cylindrical surface and projects the intersection on the current keyframe.

The reprojection error is the distance between the projected intersection and the feature. If Reprojection error $e_i$ is below a threshold $\tau_1$, this intersection point is considered a potential cylindrical point. Otherwise, the system triangulates the point using the original method. When the proportion of potential cylindrical points exceeds a threshold $\tau_2$, say 65%, the system accepts the estimated cylinder, indicating the camera is inside a pipe. The system continues to utilise this cylindrical information and accepts the potential cylindrical points as cylindrical points. The intersections are regarded as their positions. Otherwise, the system abandons the estimated cylinder and recalculates the positions of the potential cylindrical points using the original method. The system then deactivates the cylinder, accumulating new optimised points for future cylinder detection.

The whole process is illustrated in Algorithm 3:

**Algorithm 3** Cylinder evaluation

**procedure** CylinderEvaluation(Tacked features $\mathcal{F}$, Estimated cylinder $\mathcal{Y}$)
    **for** each pair of features in $\mathcal{F}$ **do**
        Build Set $\mathcal{P}$ to include triangulated map points
        Compute the intersection of the backprojected ray and the cylinder surface using Equation 3.17
        Compute Reprojection error $e_i$ using Equation 3.19
        **if** Reprojection error $e_i < \tau_1$ **then**
            Mark Point $i$ as a candidate cylindrical point $P_{c_i}$
        **else**
            Triangulate Feature $i$ with the original method
            Add Map point $i$ to Set $\mathcal{P}$
        **end if**
    **end for**
    **if** The ratio of candidate cylindrical point $> \tau_2$ **then**
        Accept the cylindrical assumption
        Add all candidate cylindrical points to Set $\mathcal{P}$
    **else**
        Reject the cylindrical assumption
        Deactivate the cylinder
        Triangulate all candidate cylindrical point $P_c$ with the original method
    **end if**
    **return** Triangulated map points in Set $\mathcal{P}$
**end procedure**

## 4.5 Local Bundle Adjustment with Cylindrical Regularity

This section introduces local BA leveraging the cylindrical regularities. The optimisation refines keyframes, map points and cylinders by reducing the reprojection error and point-cylinder distance.

### Formulation

The local BA includes cylinders, map points and keyframes $\mathcal{L}_k$ according to the covisibility graph. The full state vector contains the current keyframe $k$, its covisible keyframes $\mathcal{V}_k$ and map points $\mathcal{P}_k$ seen by those keyframes. The covisible keyframes are defined as keyframes which share common features with the current keyframe. The optimisation also includes cylinders $\mathcal{I}_k$ whose cylindrical points join this optimisation. The states at time $\mathbf{t}$ are defined as follows:

$$\mathcal{X}_t = \{\mathbf{T}_{iw}, \mathbf{p}^j, \mathcal{Y}^m\}_{i\in\mathcal{V}_k, j\in\mathcal{P}_k, m\in\mathcal{I}_k} \tag{4.1}$$

where $\mathbf{T}_{iw} \in \mathrm{SE}(3)$ denotes the frame poses, and can be written as

$$\mathbf{T}_{iw} = \begin{bmatrix} \mathbf{R}_{iw} & \mathbf{t}_{iw} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.2}$$

with rotation matrix $\mathbf{R}_{iw} \in \mathrm{SO}(3)$ and translation vector $\mathbf{t}_{iw} \in \mathbb{R}^3$.

The cost function is

$$\begin{aligned} J(\mathcal{X}_t) = &\sum_{j\in\mathcal{V}_k}\sum_{i\in\mathcal{P}_k} \rho\left(||\mathbf{e}_v^{ij}||^2_{\mathbf{\Sigma}_v}\right) \\ &+ \sum_{m\in\mathcal{I}_k}\sum_{n\in\mathcal{Q}_m\cap\mathcal{P}_k} \rho\left(||e_c^{mn}||^2_{\mathbf{\Sigma}_c}\right) \end{aligned} \tag{4.3}$$

where $\mathbf{e}_v^{ij} \in \mathbb{R}^2$ is the reprojection error of feature $j$ observed by keyframe $i$, and $e_c^{mn} \in \mathbb{R}$ is the point-cylinder distance of cylindrical point $n$ on cylinder $m$. $\mathcal{Q}_m$ is a set of the corresponding cylindrical points on cylinder $m$. The Huber kernel function $\rho$ is used to robustify the optimisation by suppressing outliers. $\Sigma_v$ is the covariance matrix of a feature observation, and $\Sigma_c$ is the variance of the point-to-surface constraint. This study models the point-cylinder distance with Gaussian distribution. In the following experiments, it is assumed that the distances of cylindrical points are within the interval $[-0.05r, 0.05r]$ from the cylindrical surface. The uncertainty of the regularity $\Sigma_c$ is set to $\frac{25r^2}{38416}$. Also, a Chi-squared test is used to reject outliers, and the threshold is 3.814. This optimisation problem is solved using the g2o solver [157].

### Reprojection error

The reprojection error is the distance between the projected map point and the observed feature in the image frame. For frame $i$ and feature $j$, this is given by

$$e_v^{ij} = z_{ij} - \kappa(\mathbf{T}_{iw} p_w^j) \tag{4.4}$$

where $\kappa(\cdot)$ is used to map a 3D point $P_{ij}$ to a 2D pixel $f_{ij}$ on the image, $T_{iw}$ is the transformation matrix from the world coordinate to the camera pose.

#### Cylindrical regularity

Given a cylindrical point $\mathbf{p}^n$ and the cylinder parameters $\{\mathbf{c}_m, \vec{\mathbf{v}}_m, r_m\}$, the cylindrical regularity term $e_c^{mn}$ is given by equation 3.9. The following subsection introduces pose graph optimisation leveraging a new regularity.

## 4.6 Pose Graph Optimisation with Cylindrical Regularity

This part introduces a new cylindrical regularity in pose graph optimisation according to the structure of pipes. Pose graph optimisation is used to correct keyframe poses in the detected loop. It enforces a Sim(3) constraint between two keyframes, mainly according to the covisibility graph. Thus, the keyframes that share few features with others have few constraints. They may change more than expected after the loop closing. The system builds Sim(3) constraints between estimated cylinders and the corresponding cylindrical keyframes, which can help maintain the relative poses. It can prevent keyframes from moving away from the corresponding pipe.

#### Formulation

The full state vector in the pose graph optimisation contains all keyframe poses $\mathcal{F}$ and cylinders $\mathcal{C}$. The set of states is defined as:

$$\mathcal{X} = \{\mathbf{S}_{wi}, \mathbf{S}_{wm}\}_{i\in\mathcal{F}, m\in\mathcal{C}}, \tag{4.5}$$

where $\mathbf{S}_{wi} \in \text{Sim}(3)$ is the similarity transformation matrix of keyframe $i$, which has 7 degrees of freedom. Cylinders $m$ join the pose graph optimisation in the form of a similarity transformation matrix $\mathbf{S}_{mw} \in \text{Sim}(3)$ also. The transformation matrix of the cylinder can be obtained by converting the cylinder parameters, described in Section 3.2. The scale factor is set to 1.

The system minimises the following cost function with respect to all parameters $\mathcal{X}$

$$J(\mathcal{X}) = \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{G}_i} ||\boldsymbol{\epsilon}_p^{ij}||^2_{\boldsymbol{\Sigma}_p} + \sum_{m\in\mathcal{C}} \sum_{n\in\mathcal{H}_m} ||\boldsymbol{\epsilon}_c^{mn}||^2_{\boldsymbol{\Sigma}_p}, \tag{4.6}$$

where $\boldsymbol{\epsilon}_p$ is the relative pose error between keyframe $i$ and related keyframes in set $\mathcal{G}_i$ [1]. Cylindrical relative pose error $\boldsymbol{\epsilon}_c \in \mathfrak{sim}(3)$ is given between cylinder $m$ and its cylindrical keyframes in set $\mathcal{H}_m$. $\boldsymbol{\Sigma}_\mathbf{p}$ is assumed to be an identity matrix in this case.

#### Cylindrical Relative Pose Error

Cylindrical relative pose error is the difference between the original relative pose $\mathbf{S}'_{im} \in \text{Sim}(3)$ and the updated relative pose between the cylinder and the corresponding cylindrical keyframes.

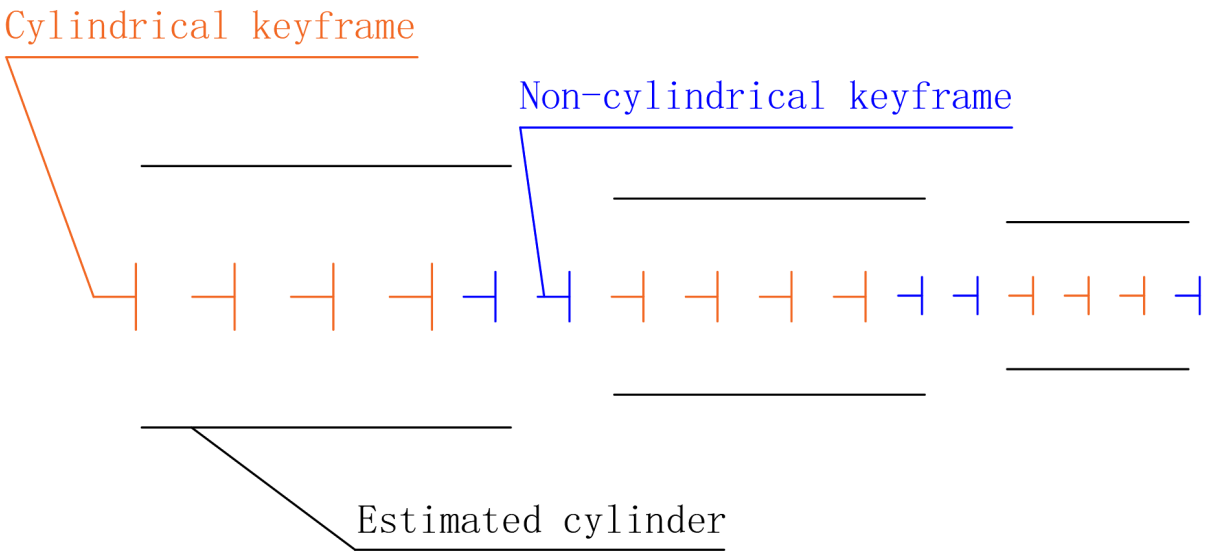



Figure 4.3: Estimated cylinders in a pipe

It is defined as follows:

$$\epsilon_c^{mn} = \log_{Sim(3)} (\mathbf{S}_{mw}\mathbf{S}_{wi}\mathbf{S}'_{im}), \tag{4.7}$$

After pose-graph optimisation, the system optimises all related keyframes, map points and cylinders in global bundle adjustment.

## 4.7 Pose Graph Optimisation with Isodiametric Information

This section introduces a novel approach based on pose graph optimisation to correct scale drift using isometric information. The point cloud generated by the vSLAM algorithm as the camera moves within a pipe always forms a cone shape, and the corresponding movement distance progressively decreases. This is particularly a problem in pipes, as has been frequently observed here empirically and in the literature [26]. Both monocular feature-based and direct methods exhibit this behavior when operated in a pipe. This is because the scale factor decreases as the camera moves. Loop closing can address scale drift, but in practice, loop closures are not always present. A more typical situation in sewer networks is that the pipes have a fixed diameter between manholes. Ideally, all the cylinders detected by vSLAM should have the same radius. However, due to scale drift, the radius of a new cylinder as the vehicle moves will be smaller than that of the previous cylinder, as shown in Figure 4.3. This scale drift is evident as a decrease in the cylinders' radii, and the ratio of cylinders' radii can be regarded as the ratio of scale factors between cylindrical keyframes.

The algorithm leverages the estimated cylinder radii to compute the relative scale factors for cylindrical keyframes. Without additional information, monocular vSLAM cannot determine

scale. The relative scale factor for a cylindrical keyframe is the ratio of its corresponding cylinder's radius to that of the first cylinder. It is assumed here that the first cylinder is least affected by scale drift, with its radius serving as a reference, denoted as $r_i$. The relative scale factor for cylindrical keyframe $i$ is calculated as follows:

$$\mathbf{s}_i^* = \frac{r_i}{r_r}, \tag{4.8}$$

where $\mathbf{s}_i^*$ is the relative scale factor, and $r_i$ is the radius of the corresponding cylinder.

Given the relative scale factor, the algorithm will utilise pose graph optimisation to correct keyframe poses. Similar to that of the loop closing, the loss function has all keyframes and constraints based on the covisibility graph. The algorithm uses the relative scale as the scale factor for cylindrical keyframes, while the scale factors for other keyframes is set to 1 by default. During optimisation, the pose of the first keyframe and the scale factors of the cylindrical keyframes are fixed. The algorithm optimises the keyframe poses and the scale factor of non-cylindrical keyframes by minimising the relative pose error. The variable definitions are as follows:

$$\mathcal{X} = \{\mathbf{S}_{wi}, \mathbf{T}_{wm}\}_{i\in\mathcal{F}, m\in\mathcal{H}}, \tag{4.9}$$

where $\mathcal{H}$ is the set of all cylindrical keyframes, and $\mathbf{T}_{wm}$ is their transformation matrix. The set of non-cylindrical keyframes is denoted by $\mathcal{F}$, and their similarity transformation matrix is $\mathbf{S}_{wi}$.

The cost function is:

$$J(\mathcal{X}) = \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{G}_i} ||\boldsymbol{\epsilon}_p^{ij}||_{\boldsymbol{\Sigma}_p}^2 + \sum_{m\in\mathcal{C}} \sum_{n\in\mathcal{H}_m} ||\boldsymbol{\epsilon}_y^{mn}||_{\boldsymbol{\Sigma}_p}^2, \tag{4.10}$$

where $\boldsymbol{\epsilon}_y$ is relative pose error for cylindrical keyframes. The algorithm optimises the rotation and translation of cylindrical keyframes, excluding its scale factor. This version of the algorithm is referred to as CRORB with isodiametric correction (CRORBic).

## 4.8 Experiments and Results

To evaluate the proposed algorithms CRORB and CRORBic, experiments were conducted on:

1. Synthetic data from a 3D pipe network simulation modelled on a real sewer network constructed using robot operating system [37] and Gazebo [158].

2. Real-world data, from live CCTV inspection data of a sewer pipe.

To demonstrate the advantages of CRORB/CRORBic, this study benchmarked and compared it to ORB-SLAM2, DSO [15], and a *Benchmark* vSLAM method with cylindrical regularity where cylinder detection using point-cylinder distance as used in [30] is integrated with ORB-SLAM2

implementation. In these experiments, the algorithms were implemented on a computer with Intel Core i7-8700 @ 3.2GHz, 32GB memory.

The algorithms are evaluated and compared in terms of trajectory accuracy and running time. The open-source software `evo_traj` was used to align the estimated trajectories with ground truth [159]. The absolute pose error (APE) is the difference between the estimated poses and the ground truth. The main metric used was the root mean square error (RMSE) of the APE, $e_{RMSE} = \sqrt{\sum_{i=1}^{n} e_i^2/n}$, where $e_i$ is the APE for Pose $i$ and there are $n$ estimated poses in total. The scale is estimated as the ratio of the true travel distance to the estimated distance between the first two keyframes created in the initialisation.

### 4.8.1 Synthetic Data

A synthetic 3D pipe network environment based on a real-world sewer pipe network in the UK (Figure 4.4A) was created in Gazebo [158] (Figure 4.4B) and simulated with ROS [37]. Manholes connect several straight pipe sections with a constant 1 metre diameter to form a branched pipe network similar in layout to the real sewer pipe network. A virtual robot moves through the pipes, and the front-facing camera observes the inner pipe surface. The inner pipe surface had colour variation emulating the inside of a sewer pipe. The virtual camera collects images of $800 \times 800$ pixels at 30 frames per second, with added Gaussian noise.

In Table 4.1, ORB2 is an abbreviation for ORB-SLAM2. Superscript* means the algorithm runs loop closing without cylindrical regularity. Superscript** means the algorithm runs loop closing leveraging the cylindrical regularity. Symbol '—' means the statistics are not available or not applicable.

These experiments tested the vSLAM algorithms for three different trajectories. Table 4.1 shows the results of different algorithms on synthetic data. The last column shows the results obtained from algorithms without and with loop closing separately. The corresponding trajectories are shown in Figure 4.4C-F. The orientations of the frame poses are similar to the true value, and the errors in translation mainly stem from the drift of the scale factor. The cylindrical regularity slows down the decrease of the scale factor, and the trajectories from CRORB are all more accurate than those from other algorithms. With the isodiametric information (CROR-Bic), the algorithm can recover the scale factor even if the system detected cylinders only in some keyframes. The results show that cylindrical regularities improve estimation at a small computation cost. Compared to ORB-SLAM2, Benchmark has a more significant error due to poor cylinder fitting among noisy point clouds (Figure 4.4 and Table 4.1). DSO was also tested on the synthetic data. The results show that DSO has substantially worse accuracy than both ORB-SLAM2 and our own proposed methods since the inaccurate initialisation (Figure 4.4 and Table 4.1). The scale factor also decreases after the initial response.

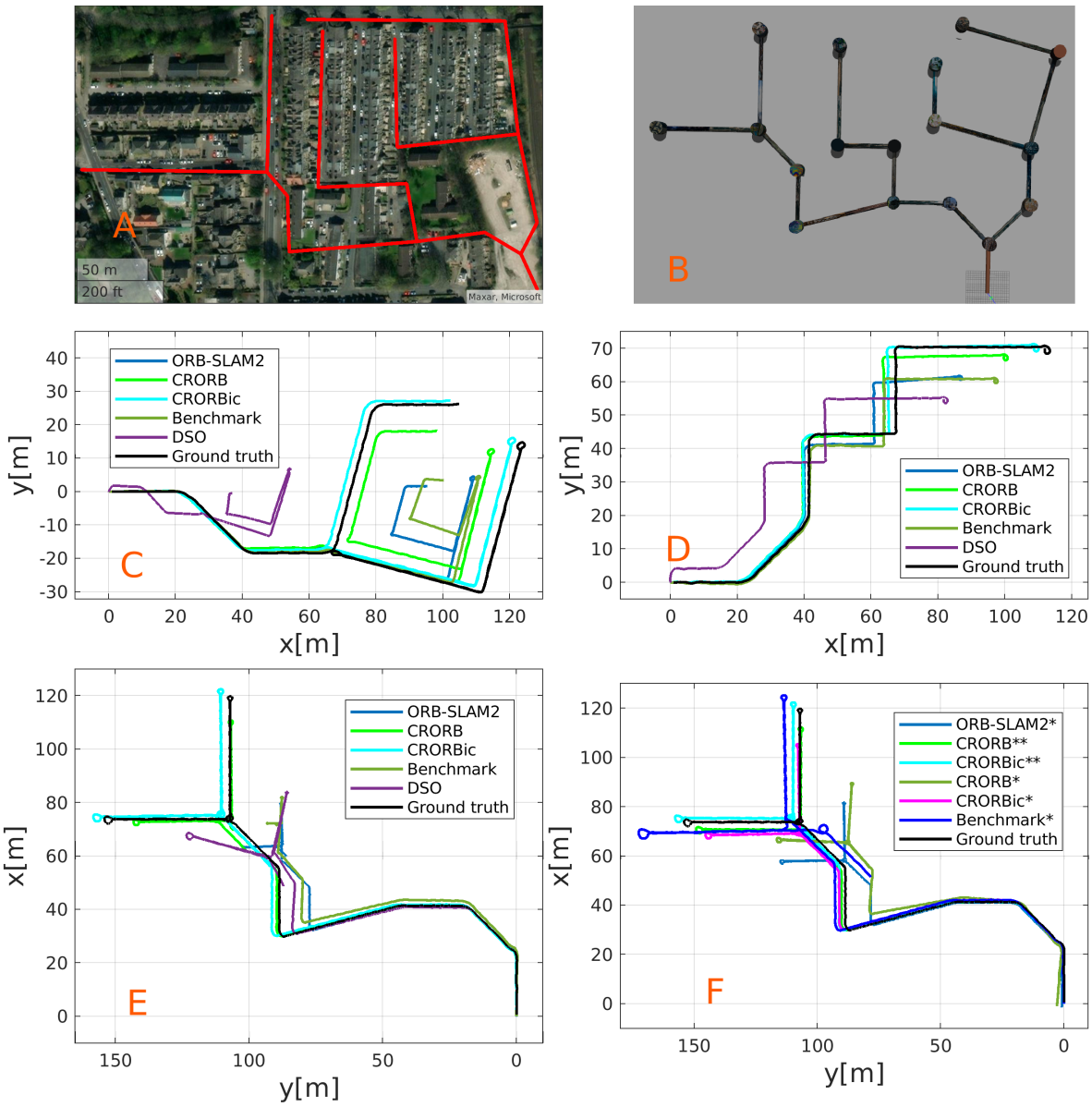



Figure 4.4: Simulation results. A: Real-world sewer pipe network. B: Synthetic pipe network created in ROS-Gazebo to emulate the real network in panel A. C-F: Estimated trajectories around the simulated pipe network from the ORB-SLAM2, CRORB and CRORBic algorithms against the ground-truth trajectory. Superscript* means the algorithm runs loop closing without cylindrical regularity. Superscript** means the algorithm runs loop closing leveraging the cylindrical regularity.

| Dataset | Synthetic Data 1 | | Synthetic Data 2 | | |
|---|---|---|---|---|---|
| Travel Distance (m) | 333.6 | | 175.5 | | |
| Method | ORB | CRORB(ic) | ORB | CRORB(ic) | |
| RMSE(m) | 13.79 | 7.84(3.29) | 12.01 | 6.82(2.84) | |
| Triangulation(ms) | 57.6 | 56.2 | 48.2 | 47.0 | |
| Local optimisation(ms) | 150.3 | 169.4 | 132.9 | 153.7 | |
| Cylinder Detection(ms) | — | 31.0 | — | 28.5 | |
| Method | DSO | Benchmark | DSO | Benchmark | |
| RMSE(m) | 43.7 | 23.88 | 27.5 | 25.54 | |
| Local optimisation(ms) | — | 145.8 | — | 133.8 | |
| Dataset | Synthetic Data 3 | | | | |
| Travel Distance (m) | 395.6 | | | | |
| Method | ORB | CRORB(ic) | ORB* | CRORB(ic)* | CRORB(ic)** |
| RMSE(m) | 26.70 | 7.59(4.34) | 24.92 | 6.21(7.78) | 5.63(4.66) |
| Triangulation(ms) | 53.0 | 52.1 | 54.0 | 52.4 | 54.8 |
| Local optimisation(ms) | 170.5 | 194.4 | 166.8 | 190.8 | 205.1 |
| Cylinder Detection(ms) | — | 35.7 | — | 39.6 | 45.9 |
| Method | DSO | Benchmark | Benchmark* | | |
| RMSE(m) | 43.5 | 51.96 | 47.43 | | |
| Local optimisation(ms) | — | 167.9 | 174.3 | | |

Table 4.1: Comparison between the SLAM algorithms on synthetic data

| Dataset | Experimental Data 1 | | | |
|---|---|---|---|---|
| Travel distance (m) | 39.5 | | | |
| Approach | ORB-SLAM2 | CRORB(-ic) | Benchmark | DSO |
| RMSE (m) | 6.41 | 3.30(1.62) | 10.29 | 18.55 |
| Triangulation (ms) | 68.6 | 60.5 | – | – |
| Local BA (ms) | 76.5 | 98.4 | 82.2 | – |
| Dataset | Experimental Data 2 | | | |
| Travel distance (m) | 20.3 | | | |
| Approach | ORB-SLAM2 | CRORB(-ic) | Benchmark | |
| RMSE (m) | 3.33 | 2.71(0.44) | 5.04 | |
| Triangulation (ms) | 64.9 | 54 | – | |
| Local BA (ms) | 75.8 | 83.3 | 76.5 | |

Table 4.2: Comparison between the SLAM algorithms on real data

Note that the RMSE across the data sets does not correlate with distance travelled as might be expected - this is because, although errors do accumulate, the *rate* of error accumulation tends to be different in separate experiments due to a number of other factors, such as lighting conditions, texture variability and robot motion - a systematic study of these effects is beyond the scope of this research.

### 4.8.2 Real Data

Real data was acquired from a CCTV rover inspection of two straight sewer pipes, 40 and 50 metres long, respectively, connected by a manhole (Figure 4.5A). The inspection of the first pipe segment is referred to here as experiment 1 (Expt 1) and the experiment in the second pipe segment as experiment 2 (Expt 2). The robot (Figure 4.5B) was equipped with a forward pinhole camera collecting images of $720 \times 576$ resolution at 25 FPS (Figure 4.5C). The robot moved slowly at an average rate of 0.18 m/s down the centre of the pipe. In Expt 2, due to disturbances (camera shake and severe flowing water), ORB-SLAM2 only works on the final 20 metres.

A tether is attached to the robot to measure this one-dimensional distance to establish ground truth for localisation. Note that the ground truth for the full 6DOF pose is not available in the real sewer pipe. The metric evaluated was the root mean square of the error of travel distance. The camera intrinsic parameters were obtained by calibration with a checkerboard. The algorithm used the default parameters from ORB-SLAM2, except the number of features per image, which was set to 3,000 [5].

The results demonstrate the effectiveness of the proposed approach in estimating the

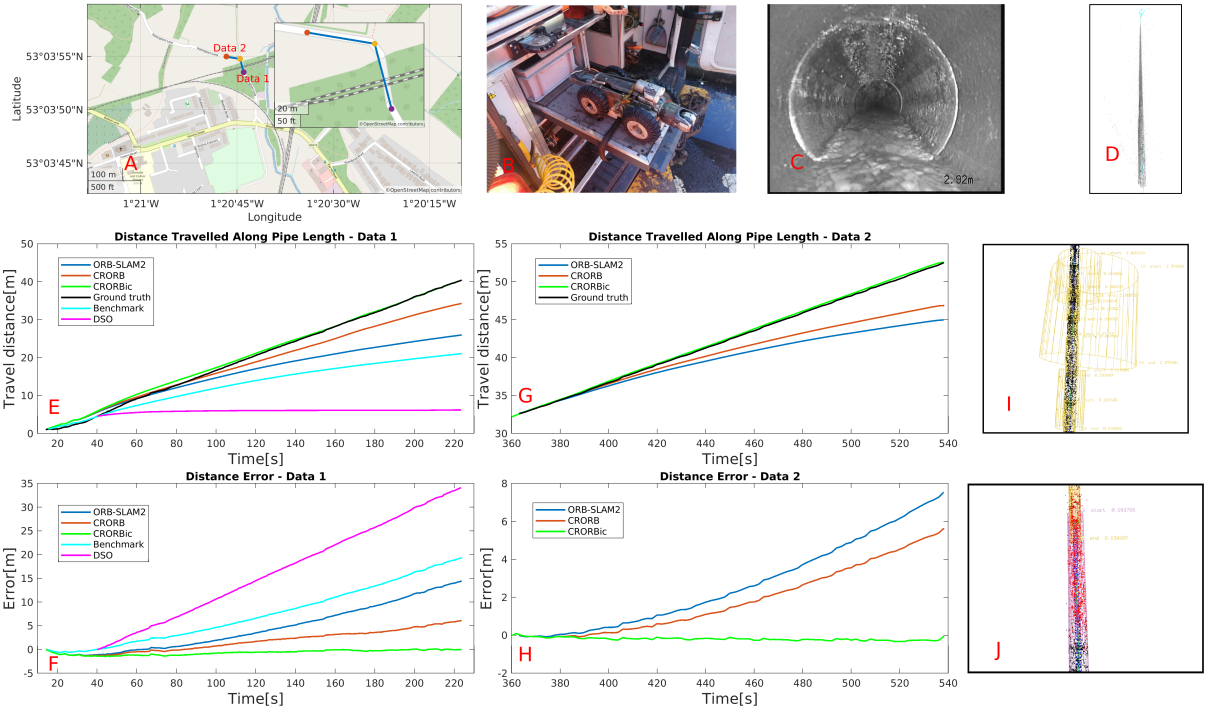



Figure 4.5: Real-world experimental results. A: Aerial view of the manholes (dots) and two sewer pipe locations (blue lines) corresponding to Experiment 1 and Experiment 2. B: Sewer pipe inspection rover. C: View inside the sewer pipe from the rover camera. D: The top view of the point cloud created by DSO on Data 1. E&G: Comparison of estimated travel distance and ground truth over time for pipes 1 and 2, respectively. F&H: Travel distance error over time for pipes 1 and 2, respectively. I: Cylinder detection using the Benchmark method. J: Cylinder detection using our proposed method in Algorithm 1.

robot's travel distance. Table 4.2 shows that CRORB obtains better accuracy than ORB-SLAM2, reducing the RMSE by 48% with experimental data set 1 and an 18% reduction in RMSE in data set 2. Symbol '—' means the statistics are not available or not applicable. CRORB with isodiametric correction (CRORBic) can use intermittent estimated cylinders to correct the scale drift, and Table 4.2 shows that it works much more accurately, giving an 86% reduction in RMSE compared to ORB-SLAM2 in data set 2. Performance results for all algorithms are compared in Figure 4.5E-H, and the problems with cylinder detection from [30] compared to our method are illustrated in Figure 4.5I-J. The point cloud in Figure 4.5D is conical, which means that the scale factor shrinks rapidly on Data 1. The execution time for triangulation, local optimisation, and cylinder detection is shown in Table 4.2 averaged over ten runs.

## 4.9 Discussion

This chapter presents a vSLAM algorithm that exploits the structural regularity in the pipes to address the scale drift problem and achieve accurate mapping and localisation. The results show substantial improvements over ORB-SLAM2 in both simulations and real data.

There is still scope for future work. In certain regions of sewer pipes, there may be insufficient features for the vSLAM algorithm to match between keyframes, which could cause the algorithm to fail. In practice, multisensor SLAM is needed to deal with occasions with insufficient visual features. There are times during operation when there are significant amounts of flowing water in the pipe, which will cover the lower part of the pipe and disrupt the vSLAM algorithm. Future work in path planning algorithms will consider flow conditions and schedule the robot to navigate through the pipe at periods of low flow and rest or recharge during periods of high flow. Analysis of the impact of different uncertainties on the algorithm performance in changeable lighting environments and variable pipe textures is another area of future work.

## 4.10 Summary

This chapter proposes a visual SLAM algorithm specifically designed for pipe environments capable of leveraging the structural information within the pipe. The algorithm robustly extracts cylindrical information from complex point clouds using the ILS algorithm and verifies the estimated cylinder during the triangulation of map points. Given an available cylinder, the system incorporates corresponding cylindrical regularities, both local optimisation and pose graph optimisation. This approach enhances pose estimation and reduces scale drift. In local optimisation, the cylinder constraints improve pose estimation by correcting the positions of cylindrical points. In pose graph optimisation, the newly introduced cylinder constraints main-

tain the relative poses between the cylinders and related keyframes. The proposed algorithm innovatively utilises cylindrical information to eliminate scale drift, significantly improving pose estimation.

# Chapter 5

# Map Alignment

## 5.1 Introduction

Large-scale mapping necessitates a multi-robot SLAM algorithm capable of integrating maps created by multiple robots. This work aims to establish a comprehensive map model of an extensive pipeline network, providing a clear and intuitive understanding of the network. It is impractical to perform SLAM with a single robot in a massive pipeline network. When a robot explores a large-scale environment, it will store substantial data and spend extensive time on it. Numerous studies have demonstrated that multi-robot mapping is more efficient and accurate than using a single robot [124, 160].

Multi-robot SLAM is a significant research topic that attracts widespread scholarly attention [161, 162, 163]. It not only expands the exploration area but also enhances efficiency [164]. During the exploration, it is assumed the robots will enter the pipe network from any manholes and explore individually and autonomously. Every robot builds a local map, and each has its own coordinate system.

When two robots explore a new environment, they construct their maps individually. Due to the absence of a common reference between the two maps, merging them into one map is difficult. Efficiently and accurately merging maps is critical for quickly constructing large-scale maps in new environments.

To merge maps, algorithms either search for overlapping areas between maps or calculate the relative pose of robots at rendezvous points [116, 132, 165], determining the relative transformation between the local maps. Although robot trajectories may overlap, robots may not necessarily meet in practice. Additionally, communication within pipeline environments is challenging, rendering most online fusion methods ineffective. Hence, an offline map merging method based on overlapping areas is required.

A place recognition algorithm is utilised to identify overlapping areas, where image-to-image matching demonstrates superior performance in large environments compared to map-

to-map or image-to-map matching [166]. Some researchers employ a BOW technique to match features from different images, where each feature is treated as a word in the dictionary [2, 62, 59]. The dictionary is built by training with a lot of images [49]. The textures in pipe networks are various, and it is certain that some images from the pipe networks are not represented in the training dataset [140, 132]. Additionally, the BOW approach struggles to recognise loop closures in simulation when vSLAM re-enters a pipe from the opposite direction. Therefore, this work uses descriptors to match features from different images. The large-scale environment maps make feature matching computationally expensive. Consequently, an appropriate method to reduce computational overhead is required.

Furthermore, monocular SLAM algorithms typically cannot estimate absolute poses and the scale factor [5], which will hinder people from correctly understanding the pipe network through the estimated maps. The coordinate system of every map is built in initialisation, but the scale factor and relative transformation to the world coordinate system remain unknown, even after merging multiple SLAM maps. To address this issue, the proposed method incorporates a prior map of the pipeline network. By aligning the estimated map with the prior map, the system can obtain the size and absolute poses of the estimated map. The challenge here lies in integrating a 2D topological map with point cloud maps. A straightforward approach involves converting the point cloud map into a topological map, thereby simplifying the process of aligning heterogeneous maps to that of aligning topological maps.

Topological map alignment, which involves identifying and matching overlapping sections of two topological maps, has attracted significant attention in the research community [135, 136, 139, 142]. Huang and Beevers addressed topological map alignment by matching vertices and edges based on structural and geometric information [139]. However, maps built by vSLAM may not always be accurate, potentially missing or containing extra vertices. Wallgrün uses inexact graph matching techniques to modify maps and achieve alignment between two maps [136]. Similarly, Mielle et al. aligned sketch-maps and Voronoi graphs [142]. This chapter will also propose an inexact graph matching techniques to accomplish map alignment.

A map merging method mainly consists of three components: data association, matching extension, and transformation computation. The data association of point cloud maps is generally resolved through image matching. However, the vastness of the pipeline network complicates local map matching. Repeated detection of massive image data requires significant computational resources. Thus, this study proposes a feasible method to reduce the computational load. Moreover, 2D prior maps and point cloud maps are different types of maps, and there is no existing method for their matching and merging, necessitating the development of a new approach.

Specifically, this chapter introduces a map merging method and a map alignment method.

Section 5.2 proposes a technique to identify manholes in the point cloud maps and locate the corresponding keyframes in the maps. The map merging method will only examine the manhole keyframes in the local maps, thereby avoiding the processing of a large number of images. Sections 5.3 and 5.4 explain the topological map alignment method in detail. For the matching and aligning of different maps, this work converts the feature maps into a topological map and then proceeds with the matching and aligning. Pipelines act as edges, and manholes serve as vertices. The proposed method can leverage their structural information to complete the matching. This chapter proposes a scheme that can effectively merge local maps of pipeline networks. The specific contributions are as follows:

1. This chapter proposes a simple manhole detection method based on point clouds.
2. This study introduces a straightforward map merging technique.
3. This work incorporates prior maps and accomplishes the matching and merging of two types of maps.
4. The proposed methods are validated with simulation experiments.

## 5.2 Feature Map Merging

Given two feature maps (point clouds), it is essential to identify the overlapping regions for map merging. Most algorithms [116, 117, 118] solve the problem by matching features between keyframes from two maps, and these can be computationally expensive in a large-scale environment. The proposed method significantly narrows down the number of keyframes used for feature matching to reduce computational demands, with the algorithm solely utilising features extracted in the manhole. This necessitates the algorithm to robustly detect manholes. In unknown pipe networks, the shape of a manhole is more consistent than its appearance, leading this work to use a method for manhole detection based on point clouds.

Some researchers use BOW to match features [117, 131, 55, 132]. The dictionary for the BOW model is generated offline, and its training data only includes images from limited environments. If the test data comes from environments significantly different from the training data, the BOW model may not perform effectively [140]. Thus, this method uses descriptors to match features between two maps. This work proposes a method to reduce the number of keyframes in feature matching.

### 5.2.1 System overview

The proposed system aims to merge maps for pipe networks constructed by different robots, transforming them into a unified world coordinate system. It accepts as input two point cloud

maps generated by ORB-SLAM2 [2], which include point clouds, keyframes, features and descriptors. The system comprises several modules: manhole detection, manhole matching, and map transformation. It uniquely merges matched manholes rather than features across different maps, which makes the system efficient.

The system can recognise manholes to reduce the number of keyframes in feature matching. The robots can only access and exit the pipe network through manholes, and vSLAM builds the map from one manhole to another. Additionally, the pipes within the network are all straight, and manholes connect pipes in different directions. Consequently, overlapping areas between different feature maps must include manholes. Only a small number of keyframes are within the manholes. The system performs feature matching between keyframes from different manholes and establishes manhole correspondence between different feature maps.

The system conducts feature matching between keyframes from different manholes and computes relative similarity transformation matrices to verify the matches. The system can transform multiple maps into the same coordinate system, ensuring they share the same scale factor. However, the absolute positions and scale factors of the maps remain unavailable. This issue is addressed by topological map alignment proposed in Section 5.3 and 5.4.

### 5.2.2 Potential Manhole Keyframes Selection

This subsection proposes a method for identifying manhole keyframes based on point clouds, which can be used to compute initial values for these manholes. Typically, a pipeline network consists of pipes and manholes, which can be perceived as horizontal and vertical cylinders, respectively. Two challenges must be addressed for successful manhole detection. First, given the vast number of point clouds from the whole map, where a majority are irrelevant to the manhole, a method is needed to filter potential cylindrical points originating from manholes, thereby reducing interference from these irrelevant points. Secondly, an appropriate initial estimate is crucial for accurately extracting the cylinder.

This work uses an intuitive approach to identify potential manhole keyframes and record their timestamps, features and descriptors. As the robot passes through the manhole, features on its inner walls are triangulated. This method can roughly infer whether the robot is inside a manhole according to the distribution of local map points. Figure 5.1 illustrates the camera motion in the pipe and manhole. When the robot moves in the pipe, the point clouds (indicated by orange circles) originate from the inner pipe walls. When the camera passes over a manhole, the point clouds (indicated by blue circles) stem from both the inner surface of the manhole and the pipe. The point clouds from the manhole are noticeably higher compared to those from the pipe. The blue circles significantly deviate from the primary direction of robot motion, and this deviation aligns with the axis of the manhole. In that case, the keyframes are regarded as

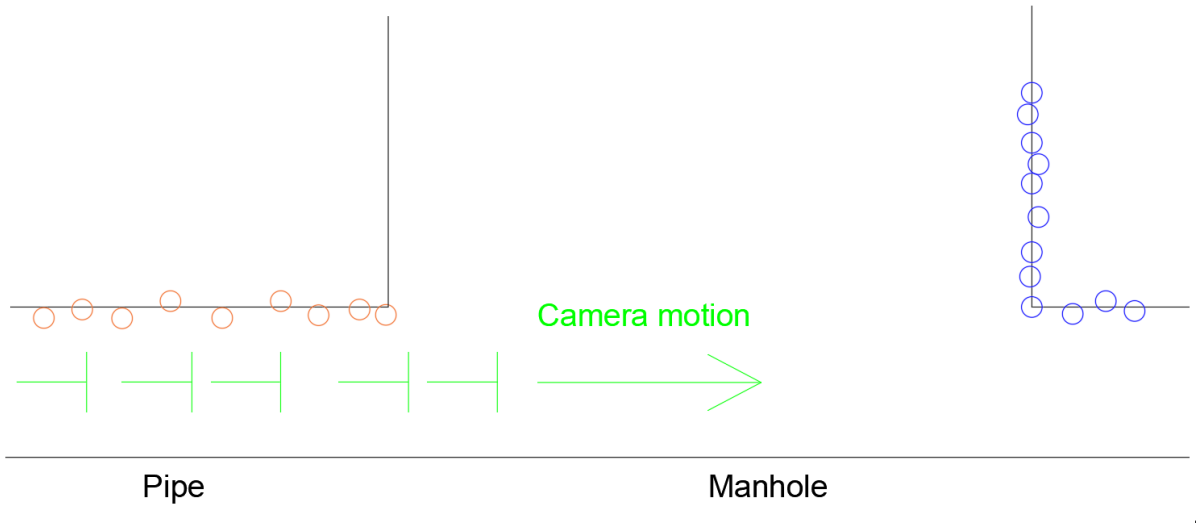



Figure 5.1: The point cloud in the manhole

potential manhole keyframes. The system also records the descriptors of the observed features of potential keyframes. Keyframes that can observe the inner walls of the manhole and are located within the manhole are considered manhole keyframes.

The method employs PCA to estimate the primary direction of the robot motion and make a rough estimation of the cylinder's orientation. Using the positions of covisible keyframes as input, PCA provides the principal components, and the first component represents the primary direction. Since the robot is assumed to move horizontally, vertical movement is expected to be minimal. The direction of a manhole is vertical, and the third component is close to the direction of the manhole, denoted as vector $v_1$. When the mean of the point cloud $p_m$ is moving away from the robot in the direction $v_3$ of the smallest component, it is considered that the robot may go through the manhole. This is mathematically expressed as:

$$v_3 p_m > \tau \tag{5.1}$$

where $\tau$ is a threshold. In practice, the algorithm will select point clouds to reduce the interference from outliers, similar to the approach discussed in Section 4.3 for cylinder detection. The system chooses point clouds that have been observed by multiple keyframes.

### 5.2.3 Manhole Detection

Given the manhole keyframes and related point clouds, the system uses an iterative optimisation approach to estimate the position and radius of the cylinder. Initially, the system selects manhole keyframes in the same manhole and calculates the mean position as the initial position of the cylinder. Subsequently, using the keyframe positions as input, the system utilises PCA to compute the initial estimate for the direction of the cylinder. The system selects the map

points to avoid interference from outliers. The input map points must be observed by multiple keyframes and have parallax angles greater than a threshold. When the camera enters the manhole, the features extracted from the inner wall by vSLAM have very small parallax angles. These points deviate significantly from their actual positions and can adversely affect cylinder estimation.

Given the initial values for the cylinder parameters and the selected map points, the cylinder fitting algorithm mentioned in Section 3.3 can accurately estimate the cylinder. To ensure the correctness of the estimation, the algorithm checks the reprojection error of the map points proposed in Section 3.4. If the manhole extraction is successful, the keyframes are considered manhole keyframes. In the next subsection, the system will establish manhole correspondence based on manhole keyframes.

### 5.2.4 Manhole Matching

After successfully detecting the manholes, the algorithm establishes manhole correspondence among manholes from different maps by matching their features from manhole keyframes. Given the manhole keyframes, the system can find the number of shared features between any two keyframes within the same manhole. This number is used as a threshold for identifying a keyframe from another map within this manhole. It is reasonable to assume that any keyframes share features with those manhole keyframes beyond the threshold belonging to the same manhole.

The system performs feature matching among the manhole keyframes that have the most cylindrical points in every manhole. When the count of matched features exceeds the threshold for both manholes, the system verifies the matches through the ICP method [121]. The system employs the ICP algorithm [121] and RANSAC method [36] to attempt to determine the relative similarity transformation matrix between the corresponding map points. If the majority of matched point clouds are considered inliers, the match is accepted.

To further verify this match, the system involves more manhole keyframes from the two manholes in feature matching. The system performs a brute-force feature matching between keyframes from the two manholes. If any pair of keyframes share features more than the threshold, the system employs the point cloud registration method to confirm the feature matching. The relative scale factor should be similar to the ratio of the radii of two manholes. Otherwise, the match is incorrect.

### 5.2.5 Map Merging

After obtaining manhole correspondence, the system computes the relative similarity transformation matrix between the two maps. There may be multiple matched manholes between the

two maps, and the algorithm can compute relative similarity transformation matrices based on the matched features in the matched manholes. Due to the cumulative error in vSLAM, these several similarity transformation matrices may not be identical. Usually, the algorithm will use pose graph optimisation and bundle adjustment to optimise the camera trajectory and point cloud, which requires an initial value of the relative transformation matrix between the maps.

In this study, a simplified approach is employed to calculate the similarity transformation matrices between the maps as the initial value. The algorithm takes the positions of manholes as input and utilises the ICP [121] to estimate relative poses. Suppose there are more than three pairs of matched manholes. In that case, the algorithm utilises the RANSAC method [36] to select the optimal transformation matrix with the most inlier matches, similar to point cloud registration. Subsequently, the algorithm employs pose graph optimisation and global bundle adjustment to refine the entire map and the robot's trajectory.

## 5.3 Topological Map Alignment

It is a natural choice to leverage a prior map to estimate the absolute positions and the scale factor of the estimated map when a prior map is available. The prior map is a 2D map of a pipe network, including manholes and pipes. It illustrates the distribution of pipes and manholes and can be considered a topological map. In the prior map, vertices represent manholes, and edges represent pipes. The feature map consists of point clouds from pipes and manholes and does not contain any semantic information. Due to the different types of two maps, this constitutes a heterogeneous map merging problem [165]. To align the two, this work transforms the feature map into a topological map named the estimated map, using the results of manhole detection. After the transformation, the issue can be viewed as a topological map alignment problem. Since the robots usually only explore limited areas in the pipe network, an estimated map represents only a portion of the pipe network. This study assumes that all explored areas can be found in the prior map, which makes the problem turn into subgraph isomorphism [138].

Errors and inaccuracies in the estimated map complicate and challenge map alignment in this dissertation. vSLAM suffers from cumulative errors, causing point cloud maps and the estimated map to deviate from the truth. This necessitates a robust algorithm. Moreover, errors in manhole detection results can cause the estimated map to diverge from the real topological map. As manhole detection is not guaranteed, the estimated topological map might lack vertices. Therefore, the algorithm needs to add or remove vertices in the estimated map during map alignment. A fault-tolerant graph matching method is employed [142, 136].

### 5.3.1 System Overview

This section introduces a novel approach to align feature maps with 2D prior maps to estimate the absolute positions and scale factors of the estimated map. In this dissertation, it is assumed that the prior map is accurate and complete. The feature map is built by using ORB-SLAM2 [2]. The two maps are of different types and have different coordinate systems and scale factors. The system proceeds in two steps: initial vertex matching and extension.

The system builds vertex matches between two maps through initial vertex matching and extension and selects the best one with minimum cost. The system detects manholes in the feature map as vertices and connects them according to timestamps to form a topological map. The feature map is transformed into a topological map, which is detailed in Subsection 5.3.3. In Subsection 5.3.4, the algorithm starts by using structural and geometric information to identify potential vertex matches, serving as seeds. This method considers only the individual vertices and their connected edges, resulting in matches that include both correct and incorrect matched vertices. In the next step, the system uses all vertices from the estimated map to complete the matching and select the right solutions. The system incrementally extends vertex matching from every seed to all vertices in the estimated map during this process, considering different matching possibilities.

Meanwhile, the estimated map might lack vertices or have erroneous vertices due to false manhole detection. The algorithm can add or delete the vertices in the estimated map based on the prior map. Subsection 5.4.4 introduces a new loss function to measure the matching quality. Upon completing the exploration, the system selects the result with the lowest cost.

### 5.3.2 Notation

The remainder of this text will use the following terms and notations: A sewer pipe network is tree-formed. $M$ stand for the estimated topological map, and $M'$ stand for the prior topological map. The Letters $V_A$, $V_B$, $V_C$ and so on denote the vertices of the 2D estimated map $M$, while Letters $V'_A$, $V'_B$, $V'_C$ and so on denote those of the prior map $M'$. Define a root vertex as the starting vertex of the tree structure without any parent vertices. The level or depth $\phi_A$ of Vertex $V_A$ is the minimum number of edges from the root vertex, which is 2 in Figure 5.2. The parent vertex $V_{(p)A}$ of Vertex $V_A$ is a vertex connected to it and resides one level higher, while the child vertices $V_{(c)A}$ are one level lower and connect to Vertex $V_A$. Neighbour vertices include both its child vertices and parent vertices, and the degree $\phi_A$ of Vertex $V_A$ equals the number of neighbour vertices. A leaf vertex is a vertex without any children. Ancestor vertices are vertices on the path from the root vertex to a certain vertex, excluding the itself. Descendant vertices are All vertices under Vertex $V_A$.

These vertices in the prior map are referred to as prior vertices for convenience. Similarly,

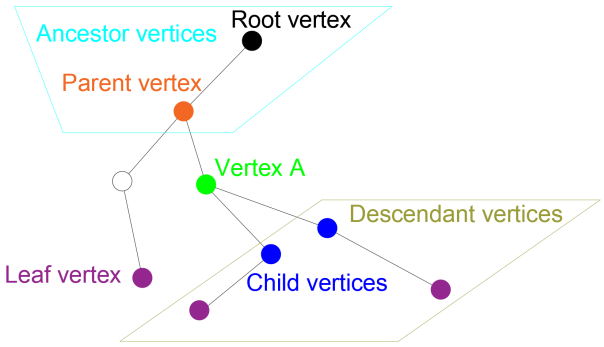



Figure 5.2: Terms and notations

these vertices in the estimated map are called estimated vertices.

### 5.3.3 Transformation of Feature Maps into Topological Maps

To align the two different types of maps, the system transforms the feature map into a topological map. The vSLAM algorithm creates point clouds when the robots explore manholes, and the detected manholes are viewed as vertices in the estimated topological map. The system detects manholes in the point cloud map using the method mentioned in Section 3.3. To save computational resources, the system does not detect pipes to locate the edges.

After extracting the manholes, the algorithm constructs a topological map with metric information. In the estimated map, vertices represent the detected manholes. Based on the timestamps of the manhole keyframes, the algorithm connects the vertices in chronological order. If multiple edges connect between two vertices, only one is retained.

It is important to note that manhole detection may miss a manhole, and the estimated topological map may deviate from the true map. A topological map with errors can complicate map alignment, which will be addressed in the following sections.

### 5.3.4 Initial Vertex Matching

Initially, the algorithm relies on the structure and geometric information to identify the matched vertices as seeds. It compares vertices according to their length and includes angles of connected edges, which are described with specific descriptors. The results include correct and incorrect matches, which will be verified in Section 5.4.

Considering the tree-formed structure of the sewer pipe network, the algorithm initially filters out unlikely matches based on the degrees of the vertices. Unlike the method in [139], it cannot ensure the vertex degrees in the estimated map since edges are only generated when the related vertices are successfully detected. The explored area is only a portion of the whole pipe network. Thus, the degree of estimated vertices will not exceed that of their matched vertices in the prior map. Leveraging this information, the algorithm significantly reduces the number

of candidate vertices. After that, the system will match the vertices according to descriptors, as illustrated in Algorithm 4:

**Algorithm 4** The Initial Vertex Matching

```
1: procedure InitialMatching(Topological map M, Estimated topological map M')
2:     Build a set C ← ∅ for matched vertices
3:     Compute degree φ and descriptors ψ for vertices in both maps
4:     for Vertex V_i in M do
5:         for every Vertex V'_j in M' do
6:             if φ_i > φ'_j then
7:                 continue
8:             end if
9:             for every descriptor ψ_{i(k)} of Vertex V_i do
10:                M = computeSimilarity(ψ_{i(k)}, ψ'_j, φ_i, φ'_j) (referred to as Algorithm 5)
11:                C ← C ∪ M
12:            end for
13:        end for
14:    end for
15: end procedure
```

This method designs a descriptor for each vertex to match manholes, capturing the angles between edges and their length ratios. The two maps have metric information, and the algorithm can compute angles and length ratios between edges. This method uses length ratios instead of absolute lengths due to the scale uncertainty and scale drift (refer to Section 4.1) in the estimated map. Scale uncertainty implies that edge lengths in the estimated map lack a fixed unit, while the prior map has actual length units. Additionally, scale drift within the pipeline introduces errors in edge lengths over time. By focusing on the length ratios of adjacent edges, the method effectively mitigates the impact of scale drift.

Given prior Vertex $V'_B$ in Figure 5.3, the algorithm calculates the angles and length ratios between the edge connecting the parent vertex and the other edges. The algorithm selects the parent edge as a reference and computes the angles and length ratios with all other edges. However, in the estimated map, the parent vertices may be unknown, and each connected vertex could potentially be a parent vertex. Therefore, the algorithm considers all possible cases, computing multiple descriptors equal in number to the degree of the vertex. Also, every descriptor of an estimated vertex will be compared to that of a prior vertex. The dimensionality of the descriptor depends on its degree. The size of a descriptor for a vertex with $n$ edges is $(n-1) \times 2$. The descriptors of Vertex $V_B$ are shown in Figure 5.4.

where $l_{AB}$ is the length of the Edge$AB$, and $\angle ABC$ denote the angle between Edge$AB$

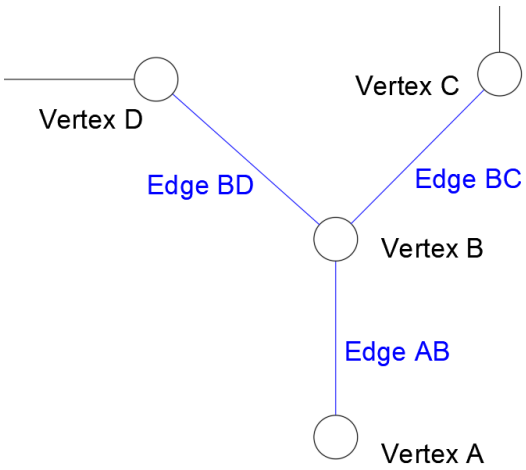



Figure 5.3: Descriptors of Vertex B

$$\begin{bmatrix} \frac{l_{AB}}{l_{BC}} & \angle ABC \\ \frac{l_{AB}}{l_{BD}} & 2\pi - \angle ABD \end{bmatrix} \begin{bmatrix} \frac{l_{BC}}{l_{BD}} & \angle CBD \\ \frac{l_{BC}}{l_{AB}} & 2\pi - \angle CBA \end{bmatrix} \begin{bmatrix} \frac{l_{BD}}{l_{AB}} & \angle DBA \\ \frac{l_{BD}}{l_{BC}} & 2\pi - \angle DBC \end{bmatrix}$$

Figure 5.4: Three descriptors of estimated Vertex $V_B$

and Edge$BC$.

The algorithm compares the descriptors row by row and calculates the difference between them. If the difference in every row of the descriptors for the estimated vertex is below a threshold, the two descriptors are considered similar. Each row of the descriptor represents the angle and length ratio of an edge relative to the reference edge. If the number of rows in the two descriptors is the same, the algorithm compares the corresponding rows from the two descriptors using Equations 5.2 and 5.3 below. The descriptors are not considered similar if any row's dissimilarity exceeds the threshold. The algorithm can determine the matching relationship of the seed and its neighbours based on similar descriptors.

The algorithm uses the following equations to calculate the difference between two rows of descriptors:

$$e_\theta = |\sin(\frac{\theta_1 - \theta_2}{2})| \tag{5.2}$$

$$e_r = |\frac{lr_1}{lr_2} - 1| \tag{5.3}$$

where $e_\theta$ and $e_r$ represent the differences in angle and length ratio, respectively.

Missing edges result in fewer rows in the descriptors of the estimated vertices, complicating the similarity calculation. The estimated map may be incomplete, with some estimated vertices having fewer edges than their corresponding prior vertices. Consequently, the descriptors for these estimated vertices have fewer rows than those for the prior vertices. When comparing descriptors, the algorithm must account for the missing rows. If the number of rows in the two descriptors is $n$ and $N$ respectively ($n < N$), the descriptor of the estimated vertex lacks $N - n$ rows. The system compensates by adding blank rows to the descriptor $\psi^\star$ of the estimated

vertex. These blank rows are placeholders and do not participate in the calculation. It is assumed that blank rows are considered similar to any row from the descriptor of the prior vertex.

The algorithm inserts blank rows at any position in the estimated vertex's descriptor, resulting in $C_{N-n}^{2N-n-1}$ possible configurations. The system then compares the descriptors of the estimated and prior vertices row by row, and all matches that meet the criteria are recorded. The detailed description is provided in Algorithm 5:

**Algorithm 5** The similarity between two vertices

**procedure** COMPUTESIMILARITY(Descriptor $\psi$ and $\psi'$, degrees $n$ and $N$ )
    Build Set $\mathcal{D}$ to record $C_{N-n}^{2N-n-1}$ completed descriptors in the comparison.
    Build Set $\mathcal{M} \leftarrow \emptyset$ to include matched descriptors of the estimated vertex
    **for** Every completed descriptor $\psi^{\star(i)}$ in Set $\mathcal{D}$ **do**
        $status = 1$ ▷ The status of the descriptor
        **for** Every row $\psi_{(j,:)}^{\star(i)}$ in $\psi^{\star(i)}$ **do**
            Compute similarity between Descriptors $\psi_{(j,:)}^{\star(i)}$ and $\psi'_{(j,:)}$ refer to Equations 5.2 and 5.3
            **if** The similarity exceeds the threshold **then**
                $status = 0$ ▷ The two descriptors are not similar
                **break** ▷ Exit loop
            **end if**
        **end for**
        **if** $status == 1$ **then**
            $\mathcal{M} \leftarrow \mathcal{M} \cup \left\{\psi^{\star(i)}\right\}$
        **end if**
    **end for**
    **return** $\mathcal{M}$
**end procedure**

The matching results inevitably include both correct and incorrect matches based on the strict requirements of structural and geometric information matching. This method only considers the information of individual vertices, and some incorrect matches may coincidentally meet the criteria. This approach can only achieve partial vertex matching, and some vertices still need to be matched. If the algorithm chooses a more relaxed threshold, it will introduce more incorrect matches.

In the next step, the algorithm extends the match to all vertices on the 2D prior map, and the specific details of this algorithm will be presented later. Considering the potential failure of manhole detection, the system needs to either add or reduce vertices based on the prior map to

ensure a more accurate match.

## 5.4 Extension

This section introduces an algorithm to extend the match to all vertices regarding a 2D prior map [167]. During the process, it starts from a seed vertex and manages to establish matches for all vertices in the estimated map. The algorithm builds a variety of matching results and selects the best one. Given the potential inaccuracies in the estimated map, it may edit the map to create additional matching possibilities. After filtering out unsuitable results based on existing information, all potential matches are evaluated according to the distance between matched vertices transformed in the prior map. The one with the minimal cost will be selected as the optimal solution.

After identifying seed matches using descriptors, the algorithm extends the matches from these seeds. The matching process starts from already matched vertices towards adjacent unmatched ones, extending to the unmatched neighbour vertices at each step. During every iteration, the algorithm explores candidates for unmatched neighbour vertices of the previously matched vertices. This extension procedure repeats until all estimated vertices are matched or the solution is deemed incorrect. The algorithm then calculates the cost of the complete solutions. The final solution is the one with the minimum cost. It is crucial to define an appropriate cost function.

### 5.4.1 Cost Function

The method employs the distance between vertices in the prior map as an evaluation criterion since the maps have metric information. Matching the vertices of the topological map can be seen as a point cloud registration problem. After transformation, vertices from two maps are aligned within the same coordinate system. A good match is achieved when the distance between all matched vertices is minimised in the same coordinate system. Since the two maps have different scale factors, a similarity transformation matrix is needed here. The algorithm can calculate the relative similarity transformation matrix based on the existing matched vertices and then determine the vertex correspondence. In theory, when the vertices are transformed into the same coordinate system, they should coincide with their corresponding vertices. However, due to noise and accumulated errors, in practice, the vertices and their matched counterparts may not overlap exactly. After obtaining the similarity transformation matrix, the algorithm selects the appropriate matched vertices to minimise the distances.

To perform the transformation, the algorithm computes a similarity transformation matrix. The relative transformation matrix is derived from the seeds or other matched parent

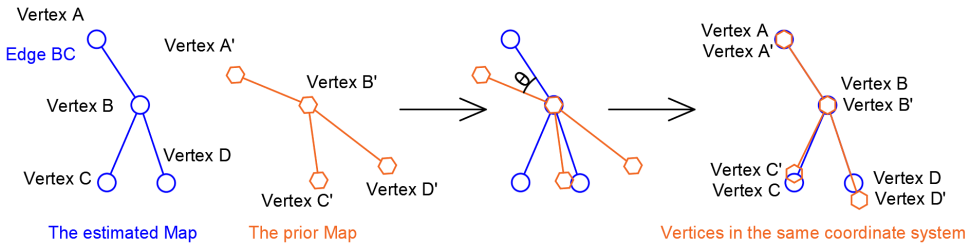



Figure 5.5: Vertices in the same coordinate system

vertices. However, the vSLAM is subject to cumulative errors, and point clouds with later timestamps have more errors than those with earlier timestamps. As a result, the estimated manholes may deviate from their true positions, especially those detected later. Therefore, the map alignment is affected by cumulative drifts, particularly for the vertices far from the matched vertices used to compute the transformation matrix. To reduce the impact of cumulative errors, the method relies on two close ancestor vertices (Vertex A and B) shown in Figure 5.5, and defines the error as follows:

$$e_m(v) = \sum_{k=1}^{N} v' - S_{pe}v \tag{5.4}$$

where $S_{pe}$ is the similarity transformation matrix calculated based on Vertices $V_A$ and $V_B$ and their corresponding matched vertices. There are $N$ pairs of matched vertices. Then, the system can compare the positions of matched vertices in the same coordinate system. As shown in Figure 5.5, the algorithm computes the similarity transformation matrix according to the angle $\theta$ and positions of matched vertices.

$$S_{pe} = \begin{bmatrix} sR_{pe} & t_{pe} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & v' - sR_{pe}v \\ s\sin\theta & s\cos\theta & \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

where $\theta$ and $s$ represent the angle and length ratio between matched edges, respectively. $v$ and $v'$ denote the positions of the matched vertices in their original maps.

### 5.4.2 Vertex Matching

When the child vertices of matched vertices are transformed into the same coordinate system, the algorithm can match vertices from two different maps according to the distances between them. Due to various errors in the estimated map, the vertices will not coincide with their matched vertices. The algorithm searches for nearby vertices of the transformed vertex as potential matching candidates. There may be more than one candidate, and the algorithm considers all nearby vertices as potential candidates.

A single vertex may have multiple potential matched vertices, and the system will extend each possible match into a complete solution. In each solution, every vertex in the estimated

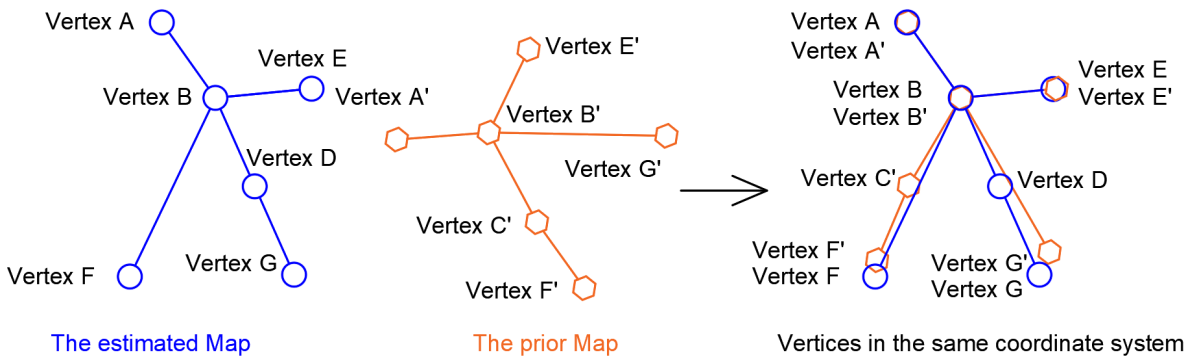



Figure 5.6: The estimated map and prior map

map has a matched vertex except for the deleted estimated vertices. The system finds seeds in the initial vertex matching, but the information from a single vertex may not be sufficient for the system to find the correct solution. As the system gains more information by extending existing solutions, it can discard some incorrect solutions progressively. The system compares the losses of the remaining solutions and selects the best one as the result.

A key challenge during this extension process is handling missing or false positive vertices in the estimated map. The manhole detection method may fail for various reasons, leading to deviations in the estimated topological map. In a challenging pipeline environment, it is possible to miss manholes. Such omissions can result in upstream and downstream vertices bypassing the missed vertices and connecting directly. The lack of manholes will change the angles and length of the edges that should connect to the missed vertex, affecting the descriptors of related vertices. If a terminal vertex is missing, the degree of its parent vertex will decrease by one. Otherwise, the degree of vertices connected to the missing vertex will not change.

Compared to missing a manhole, it is less likely to detect a manhole incorrectly for the algorithm. However, this method still takes that into account. As the manhole detection method only accepts manholes with keyframes in them, an erroneous manhole can only appear on the existing edge. When the system detects a manhole incorrectly, a vertex is added on the edge mistakenly. The incorrect manhole breaks the edge, which causes shortness of related edges without increasing degrees of associated vertices.

This subsection proposes a method for exploring possible matches to address potential errors in manhole detection. As illustrated in Figure 5.6, Vertices $V_A$ and $V_B$ match with Vertices $V'_A$ and $V'_B$. The system needs to find matches for the remaining vertices. The algorithm calculates a similarity transformation matrix based on the matched vertices, Vertices $V'_A$ and $V'_B$, and transforms the remaining vertices into the same coordinate system. There are roughly three scenarios, each of which will be discussed here.

*Match* As illustrated in Figure 5.6, Vertex $V_E$ matches Vertex $V'_E$. In this situation, the

estimated map requires no modifications, and the registration error is the distance between two vertices within the same coordinate, referred to as Equation 5.4. Here, the degrees of the Vertex $V_E$ should not exceed that of the corresponding vertices. Otherwise, this match is considered invalid.

*Add* When an actual manhole is missed, no estimated vertex is close to transformed Vertex $V'_C$. One of the possible ways is to add a new vertex corresponding to Vertex $V_C$ in the estimated map. The addition of a new manhole may influence the direction and position of the edges, which necessitates verification with keyframes. The system will estimate the location of the missing manhole based on the prior map. If there are some keyframes located in or next to the missing manhole, then this hypothesis is deemed plausible. The penalty for adding a manhole is the distance between the generated manhole and the actual manhole. The method of generating a manhole will be detailed later.

*Delete* In this case, it is assumed that the system incorrectly detects a manhole that does not exist in the prior map. In the prior map, no vertex is close to transformed Vertex $V_D$ after transformation. The system could delete this vertex as one of the feasible ways. This hypothesis requires verification by inspecting the modified edges. If a manhole is detected falsely, it should only be connected to its parent and child vertex and lie within the edge between them. After removing this vertex, the two connected edges will merge to form a single edge. Thus, if a manhole is erroneous, its degree should be two, and the two related edges are collinear. The penalty for removing a manhole is analogous to a registration error, which subsequent sections will elaborate upon.

### 5.4.3 Neighborhood Matching

The system iteratively performs neighbourhood matching until all estimated vertices have been matched. It seeks candidate vertices for unmatched vertices that are connected to matched vertices. The system selects candidate vertices based on structural and geometric information, extending the existing solution based on these obtained candidate vertices. During this process, this work denotes the set of unmatched vertices and their corresponding candidate matching vertices as the matching set $\mathcal{E}$.

Neighbourhood matching does not necessarily produce only one matching set. The system can eliminate infeasible matching sets based on local information, accepting the remaining sets. If no valid set is available, the extension of the solution is considered unsuccessful, and the corresponding solution is discarded. When multiple matching sets are available, the system duplicates the solution for each matching set and extends them separately. Once multiple solutions are fully extended and all their estimated vertices are matched, the system evaluates each completed solution by calculating the loss function.

Figure 5.6 shows an estimated map and a prior map, where Vertices $V_A$ and $V_B$ match Vertices $V'_A$ and $V'_B$, respectively. The algorithm performs neighbourhood matching to find matched vertices for other vertices in the figure. The system completes neighbourhood matching by searching for vertices adjacent to the transformed vertices. This method is introduced in Algorithm 6. Procedure SearchAdjacentVertices is used to find candidate vertices for the input unmatched vertices and is called by NeighbourhoodMatching.

**Algorithm 6** Neighbourhood matching

**procedure** NeighbourhoodMatching(Estimated topological map $M$, Prior topological map $M'$, Matched Vertices $V_A$, $V_B$, $V'_A$, and $V'_B$)
    Build a matching set $\mathcal{E} \leftarrow \emptyset$ for matched vertices
    Compute similarity transformation matrix $\mathbf{S}_{pe}^{AB}$ with Vertices $V_A$, $V_B$, $V'_A$, and $V'_B$ referred to as Equation 5.5
    Build Set $\mathcal{F}$ to include neighbour vertices of Vertex $V_B$ without matched vertices
    $\mathcal{E}$ = SearchAdjacentVertices($M'$, $\mathcal{E}$, $\mathcal{F}$, $\mathbf{S}_{ep}^{AB}$)
    Build Set $\mathcal{F}'$ to include neighbour vertices of Vertex $V'_B$ without matched vertices
    $\mathcal{E}$ = SearchAdjacentVertices($M$, $\mathcal{E}$, $\mathcal{F}'$, $\mathbf{S}_{pe}^{AB}$)
    Filter out infeasible matches from Match Set $\mathcal{E}$
    **return** $\mathcal{E}$
**end procedure**
**procedure** SearchAdjacentVertices($M$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $S$)
    **for** every vertex $V_i$ in Set $\mathcal{F}$ **do**
        Transform vertices $V_i$ in Set $\mathcal{E}$ to the prior map $M$
        Search for vertices adjacent to transformed vertices
        Compute maximum distance $d_3$ according to adjacent vertices
        Build Set $\mathcal{G}$ for vertices whose distance to $V_i$ within $d_3$
        **for** every vertex $V_j$ in Set $\mathcal{G}$ **do**
            Set $\mathcal{E} \leftarrow \mathcal{E} \cup (V_i, V_j)$
        **end for**
    **end for**
    **return** Set $\mathcal{E}$
**end procedure**

The algorithm transforms unmatched neighbour vertices (refer to $\mathcal{F}$ in Algorithm 6) of Vertex $V_B$ to the prior map, and transforms unmatched neighbour vertices (refer to $\mathcal{F}'$ in Algorithm 6) of $V'_B$ to the estimated map, searching for candidate matching vertices for the transformed vertices. Even if the estimated map contains some inaccuracies, vertices are considered close when they are aligned in the same coordinate system. Therefore, the algorithm searches

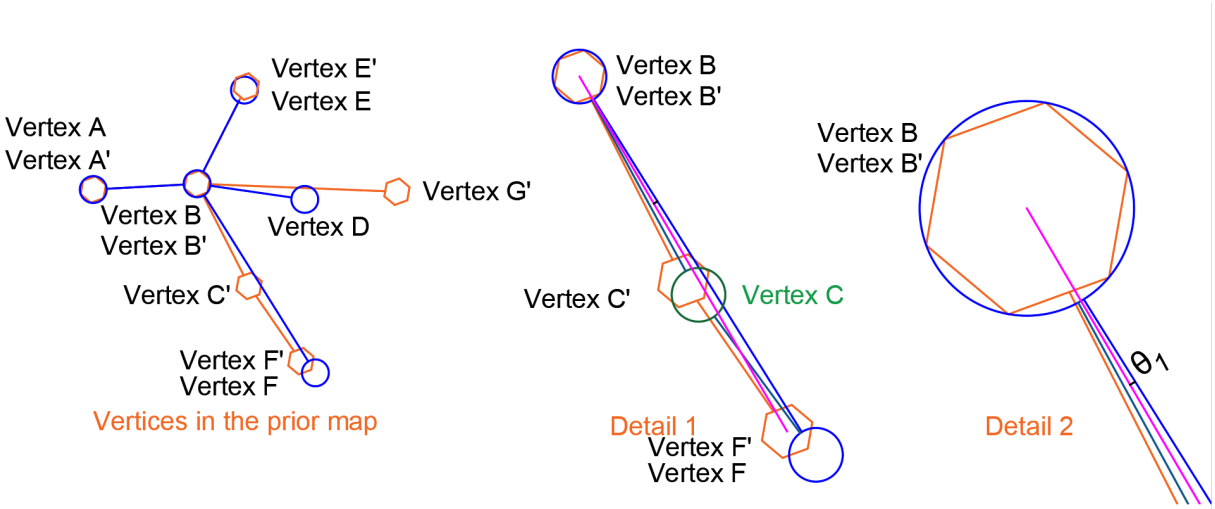



Figure 5.7: Transformed vertices in the prior map

for candidate vertices near the transformed vertex. Due to cumulative error in the estimated map, the farther a vertex is from the two matched vertices used to calculate the transformation matrix, the greater its deviation from the true position. For accuracy, the system should check vertices near the matching vertex. Additionally, simultaneously searching for candidate vertices for all vertices complicates the system. Therefore, the system only focuses on adjacent vertices.

As shown in Figure 5.7, the system transforms the neighbouring vertices of Vertex $V_B$ to the prior map and searches for candidate vertices in a broader range than just the immediate neighbourhood. This wider search is necessary because the matched vertices for the transformed vertices may be out of the neighbourhood of Vertex $V'_B$. The estimated map might lack real child vertices of Vertex $V_B$, whose grandchild vertex $V_F$ is directly connected to Vertex $V_B$. The matched Vertex for $V_B$'s grandchild vertex $V_F$ is not in the neighbourhood of Vertex $V'_B$. To find such possibilities, the algorithm transforms the neighbour vertices to the prior map and searches for candidate vertices within a larger range (refer to Line 5 in Algorithm 6). Due to the vast number of prior map vertices, the neighbour matching is limited to the vertices connected to Vertex $V'_B$ within a distance $d_3$.

The maximum distance $d_3$ from potential matching vertices to the transformed vertex may vary depending on the distribution of vertices. The distance from the closest prior vertex to the transformed vertex is denoted as $d_1$, and the distance from the transformed vertex to its parent Vertex $V_B$ is denoted as $d_2$. The minimum value between $2d_1$ and $d_2$ is denoted as $d_3$. If there are no vertices within this range, then the closest vertex is chosen as the only candidate.

In Figure 5.7, Vertices $V'_C$ and $V'_F$ is adjacent to Vertex $V_F$. In this scenario, there are two candidates. Vertex $V_F$ can be matched with either Vertex $V'_C$ or $V'_F$. Table 5.1 below illustrates the candidate vertices for the transformed vertices.

| Vertex | D | E | F |
|---|---|---|---|
| Candidate | $V'_C$,$V'_G$ | $V'_E$ | $V'_F$ |

Table 5.1: candidate matching vertices for the transformed vertices in the prior map

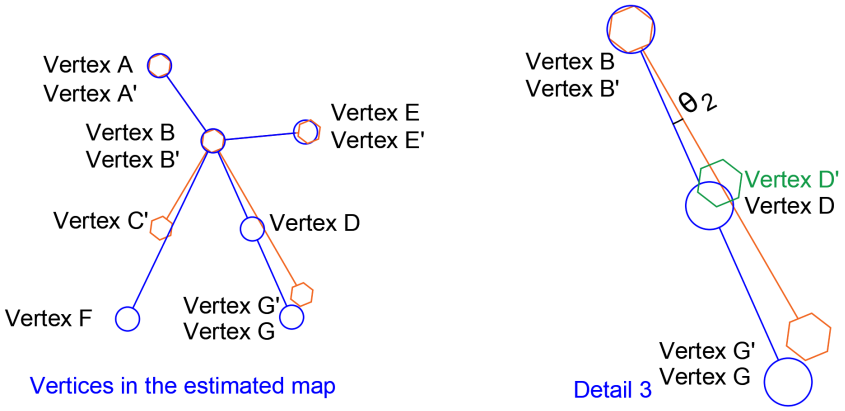



Figure 5.8: Transformed vertices in the estimated map

To discover more potentially matching vertices, the system will transform the neighbouring vertices of Vertex $V'_B$ to the estimated map, as shown in Figure 5.8. Similarly, the system will collect the estimated vertices near the transformed vertices (refer to Line 7 in Algorithm 6). Vertex $V'_G$ has only one matching candidate, Vertex $V_G$, which is a grandchild vertex of Vertex $V_B$. Therefore, Vertex $V_D$ matches no estimated vertex, and Vertex $V_G$ matches Vertex $V'_G$. The results are presented in Table 5.2:

| Vertex | D | E | F | G |
|---|---|---|---|---|
| Candidate | $\emptyset$ | $V'_E$ | $V'_C$ | $V'_G$ |

Table 5.2: candidate matching vertices for the transformed vertices in the estimated map

The two tables have been merged into Table 5.3.

| Vertex | D | E | F | G |
|---|---|---|---|---|
| Candidate | $\emptyset, C', G'$ | $V'_E$ | $F', C'$ | $V'_G$ |

Table 5.3: Vertices and their corresponding potential matches

With the results obtained above, the system generates all possible matching sets and eliminates duplicate or invalid ones, as shown in Table 5.4. In Matching sets 3, Vertex $V_D$ incorrectly corresponds to a vertex that is the parent of Vertex $V_F$, which is not consistent with reality. In Matching sets 4, 5, and 6, different estimated vertices are matched to the same prior vertex, which is not permissible. These inconsistencies are identified and removed from consideration.

A vertex has different losses due to different matches with itself and its parent vertices,

| Vertex | D | E | F | G | Feasibility |
|---|---|---|---|---|---|
| Matching set 1 | $\emptyset$ | $V'_E$ | $V'_F$ | $V'_G$ | Positive |
| Matching set 2 | $\emptyset$ | $V'_E$ | $V'_C$ | $V'_G$ | Positive |
| Matching set 3 | $V'_C$ | $V'_E$ | $V'_F$ | $V'_G$ | Negative |
| Matching set 4 | $V'_C$ | $V'_E$ | $V'_C$ | $V'_G$ | Negative |
| Matching set 5 | $V'_G$ | $V'_E$ | $V'_F$ | $V'_G$ | Negative |
| Matching set 6 | $V'_G$ | $V'_E$ | $V'_C$ | $V'_G$ | Negative |

Table 5.4: Matching sets for the extension

contributing to part of the cost of the solution. The algorithm does not evaluate solutions immediately after each neighbourhood matching but only once the entire extension process is complete. Local optima do not necessarily lead to global optima. Meanwhile, some solutions may also be discarded during extension due to being infeasible. Therefore, the algorithm compares all complete solutions after the extensions are finished. The next subsection illustrates the losses of a vertex when the system edits the map.

### 5.4.4 Penalty For Editing the Map

The estimated map may deviate from the actual map, and the system needs to edit the map to achieve alignment. Based on the matching sets obtained in the last section, the system may add or remove estimated vertices. The system calculates the corresponding loss functions for all operations, which will be discussed here.

In Matching set 1, Vertex $V'_C$ has no corresponding candidate matching vertex, which means that the estimated map is missing Vertex $V_C$ for Vertex $V'_C$, as shown in Figure 5.7. To resolve this, the algorithm will add a new vertex between Vertices $V_F$ and $V_B$ in the estimated map based on the relevant vertices. Theoretically, Vertices $V'_B$, $V'_F$, and $V'_C$ can theoretically form a triangle, similar to the triangle formed by Vertices $V_B$, $V_F$, and added Vertex $V_C$ in the estimated map. Similar triangles are triangles that have the same shape but differ in size. Based on this, the algorithm computes the position of Vertex $V_C$ using the following formula:

$$v_c = v_b + \mathbf{R}_{ep}^{BF}(v'_c - v'_b)\frac{\|v_f - v_b\|}{\|v'_f - v'_b\|} \tag{5.6}$$

where the rotation matrix $\mathbf{R}_{ep}^{BF}$ describes the rotation between Edges $BF$ and $B'F'$, and it can be obtained based on Equation 5.5 with $\theta$ as input. The positions of Vertices $V_C$, $V_B$, and $V_F$ are denoted by $v_c$, $v_b$, and $v_f$. Similarly, $v'_c$, $v'_b$, and $v'_f$ denote the positions of Vertices $V'_B$, $V'_C$, and $V'_F$.

The cost function $f^{add}$ for adding a new manhole is defined as the distance between the prior vertex and the newly added vertex in the estimated map, similar to the matching error

defined in Equation 5.4. By calculating the position of Vertex $V_C$ using the properties of similar triangles, the distance between Vertices $V_C$ and $V'_C$ becomes proportional to the distance between Vertices $V_F$ and $V'_F$. The algorithm determines the presence of the newly added vertex based on the relationship between Vertices $V_F$ and $V'_F$. Additionally, the penalty dynamically changes based on the matching error of Vertex $V_F$. The cost $f^{add}$ for adding a new manhole is shown as follows:

$$f^{add}(C) = \|v_c - \mathbf{S}_{ep}^{AB} v'_c\| \tag{5.7}$$

where $\mathbf{S}_{ep}^{BF}$ represents the similarity transformation matrix that transforms the vertices from the estimated map to the prior map. $S_{ep}$ performs the inverse transformation.

In Matching sets 1 and 2, the prior map lacks a corresponding Vertex $V'_D$ for Vertex $V_D$, indicating that the algorithm erroneously detected a manhole, as shown in Figure 5.8. The system will delete Vertex $V_D$ to complete topological map alignment. When the system deletes a vertex in the estimated map, it calculates the cost, denoted as $f^{delete}$. The algorithm determines the deletion of vertex $V_D$ based on the matching of Vertex $V'_G$, and has a loss function influenced by the matching of Vertex $V_G$. The system will create a pseudo prior Vertex $V'_D$ to compute the penalty.

The manhole detection method assumes that the manhole keyframes are within the estimated manhole. Hence, the false positive manhole should lie on the straight pipe between the upstream and downstream manholes. Ideally, the connected edges in the estimated map should be collinear. Therefore, the algorithm estimates the position of the pseudo prior vertex on the straight line segment between the parent and child vertices. The specific location is determined based on the ratio of the length of the connected edges in the estimated map. Theoretically, Edges $B'G'$ and $BG$ are straight lines. Vertex $V'_D$ is on Edge $B'G'$, and the ratio of the lengths of its adjacent edges should be the same as that of Vertex $V_D$. Therefore, the position of Vertex $V'_D$ is calculated as follows:

$$v'_d = v'_b + (v'_g - v'_b)\frac{\|v_d - v_b\|}{\|v_g - v_b\|} \tag{5.8}$$

where $v_b$, $v_d$, and $v_g$ are the positions of Vertices $V_B$, $V_D$, and $V_G$, and $v'_b$ and $v'_g$ are the positions of Vertices $V'_B$ and $V'_G$.

Similarly, the loss function $f^{delete}$ for deleting a vertex is defined as the distance between the pseudo prior vertex and the false positive vertex in the estimated map. This penalty is also adaptive. When Vertices $V_G$ and $V'_G$ are very close, and Edges $BD$ and $DG$ are collinear, the penalty in the loss function will be small.

$$f^{delete}(D) = \|v_d - S_{ep}^{AB} v'_d\| \tag{5.9}$$

where $S_{ep}^{AB}$ is the similarity transformation matrix between the two maps.

Due to the non-differentiable nature of the cost function, the algorithm compares every feasible solution and selects the optimal one with minimum cost. Each vertex may have multiple corresponding candidates, leading to many solutions. The system must consider as many feasible solutions as possible to achieve a global optimum. The algorithm does not discard a solution with a high local cost. Instead, it will extend every solution until it is deemed implausible or has no unmatched vertex left. Once the system completes the extension for all viable solutions, the cost will be calculated. Finally, by comparing the costs of different solutions, the algorithm selects the globally optimal. The following will provide a detailed introduction to the algorithmic process.

### 5.4.5 Detailed Algorithm Workflow

This subsection outlines how the algorithm extends from initial seed matches to all estimated vertices, as demonstrated in Algorithm 7. The work perceives the topological graph matching leveraging metric information as point cloud registration. The algorithm transforms vertices from both two maps into the same coordinate system and selects potentially matching vertices based on the distances to transformed vertices. The system filters matching sets based on known information and ultimately accepts the feasible matching sets.

The algorithm first uses the structural and geometric information of the vertices to initially identify several matched vertices as seeds, as discussed in Subsection 5.3.4. In some cases, engineers may know the starting position in the pipe network, which can also be directly used as a seed without correspondence of its child vertices. Typically, the initial matching not only found matched vertices as seeds but also obtained the matches for its child vertices based on their descriptors. Thus, the algorithm still needs initial matching to check matches of child vertices.

If the lengths of the edges of the seed vertices are the same and the angles are equal, the descriptors of the estimated vertices are the same or similar. In this case, the number of matching candidates of their child vertices is equal to the degree of the seed. The algorithm will accept all matched vertices as seeds and extend them accordingly.

The algorithm starts with seeds obtained by the initial matching, with each seed considered as an independent solution. The system extends these solutions iteratively by matching neighbouring vertices of the matched vertices. If multiple matching sets are found during the extension, the algorithm replicates the solution for each matching set, and incorporates a different set into each solution. If the extension produces no matching set, the solution is deemed unfeasible and subsequently discarded. The procedure is detailed in Algorithm 7.

**Algorithm 7** Extension

**procedure** EXTENSION(Estimated topological map $M$, Prior topological map $M'$, Set $\mathcal{C}$ for seeds)
    Build Set $\mathcal{H} \leftarrow \emptyset$ for all solutions
    **for** Every seed $(V_i, V_i')$ obtained in the initial matching **do**
        Build a solution $X_i$ with Seed $(V_i, V_i')$
        **while** Solution $X_i$ is not completed or discarded **do**
            Establish Set $\mathcal{E} \leftarrow \emptyset$ that contains the vertices and their corresponding candidate matches
            **for** every leaf vertices in $V_j$ and $V_j'$ in Solution $X_i$ **do**
                Local Set $\mathcal{E}_j$ for candidate matches $\mathcal{E}_j$ = NeighbourhoodMatching($M$, $M'$, $V_j$, $V_j'$)
                $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}_j$
            **end for**
            Create Set $\mathcal{L}$ that contains matching sets according to Set $\mathcal{E}$
            Filter out infeasible matching sets in Set $\mathcal{L}$
            **if** if Matching set $\mathcal{L}$ is empty **then**
                Discard Solution $X_i$
            **end if**
            **for** every Matching set $\mathcal{L}_k$ from Set $\mathcal{L}$ **do**
                Replicates Solution $X_i$ to $X_ik$
                Enrich Solution $X_ik$ by including matched vertices in Match set $\mathbf{L}_k$
            **end for**
        **end while**
    **end for**
    Evaluates every solution by computing the cost function
    **return** Solution with the minimum cost
**end procedure**

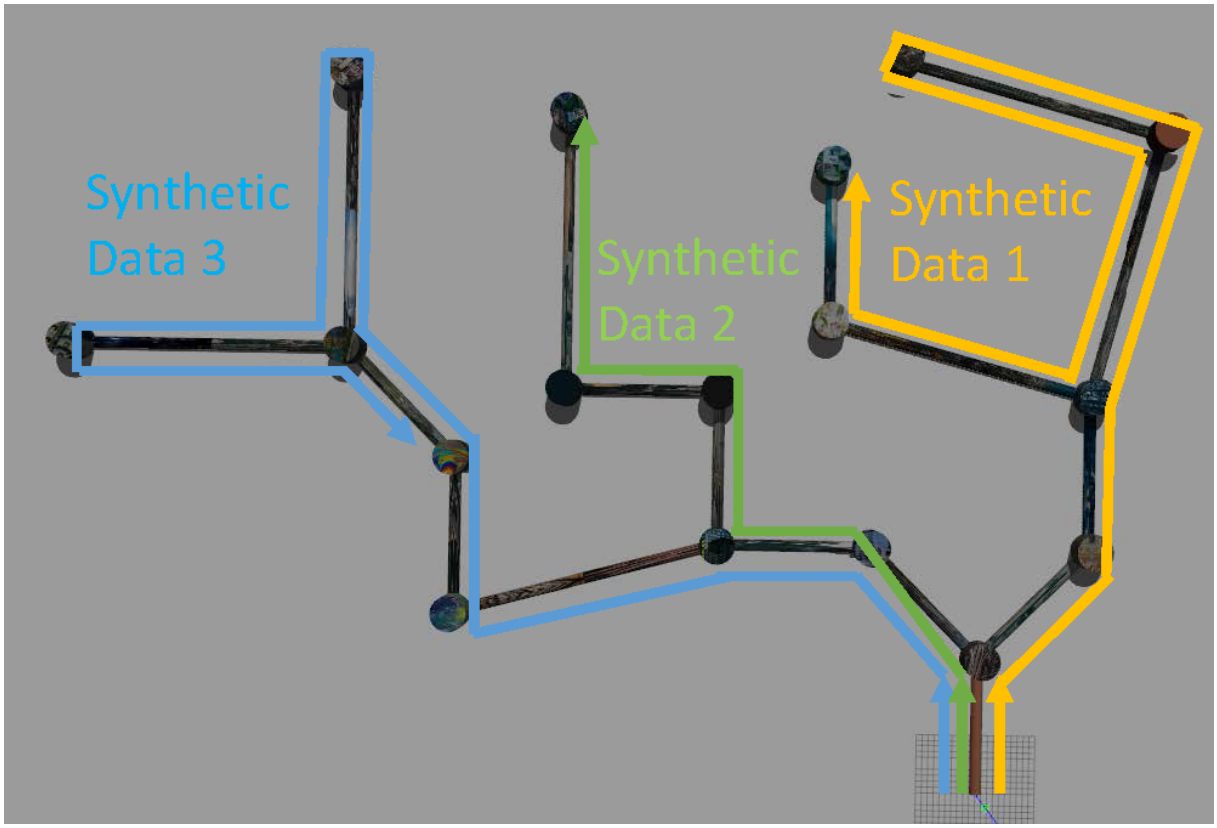



Figure 5.9: Simulation environment and the trajectories of the robot

## 5.5 Experiments and Results

This section presents a series of experiments that validate the effectiveness of the algorithm proposed in Section 5.2. The data used for these experiments consists of feature maps and keyframe poses created by ORB-SLAM2 [5]. Due to the lack of suitable real pipeline data, the vSLAM algorithm operates on the simulated data mentioned in Chapter 4. Based on a real-world pipe network, Gazebo generates a virtual pipe network that includes manholes and pipes. A robot moves within the pipe network, capturing image data. The exploration of the entire pipe network was completed in three passes, which can be seen as three independent robots exploring the pipe network. For clarity, Figure 5.9 shows the generated maps along with the approximate trajectories of the robots. Feature maps produced by vSLAM running on different datasets are labelled as Feature Maps 1, 2, and 3, respectively. Since this chapter introduces two offline algorithms, all algorithms will be executed on a computer equipped with an Intel Core i7-8700 @ 3.2GHz processor and 32GB of memory.

### 5.5.1 Map Merging Algorithm Evaluation

To demonstrate the effectiveness of the proposed algorithm, its performance is compared to the Horn algorithm [128] in terms of time efficiency and accuracy. The Horn algorithm estimates

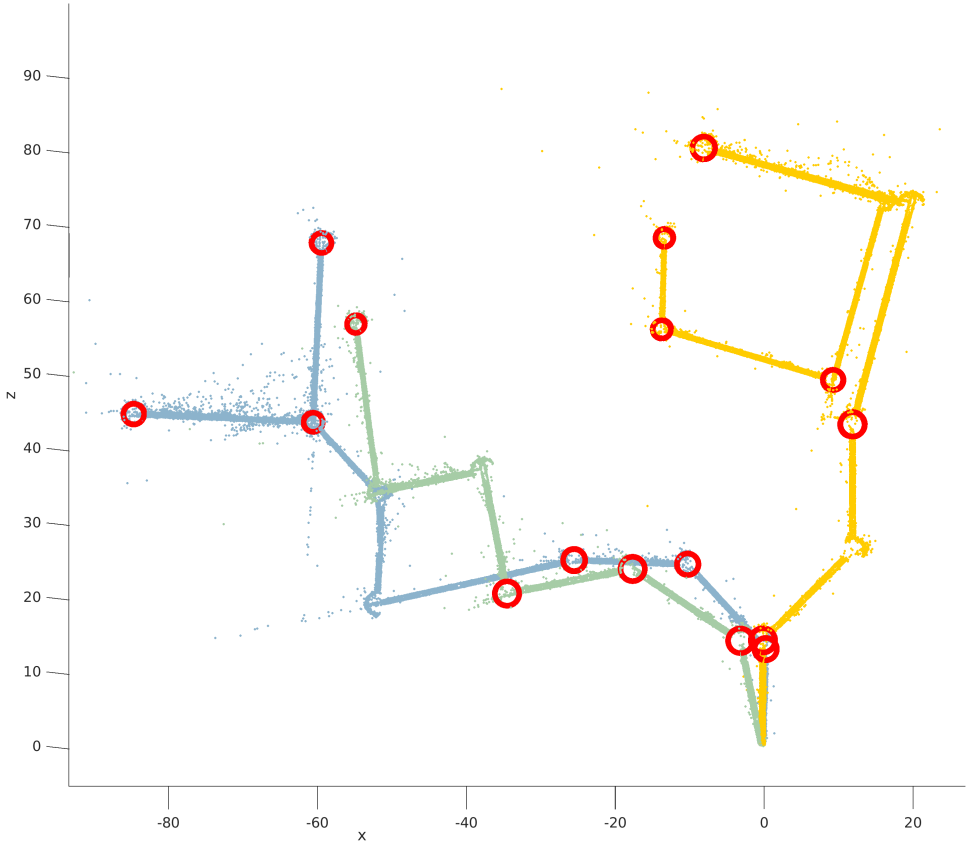



Figure 5.10: Map before Map Merging

the current frame pose on the map based on matched features. There are numerous keyframes within the two maps when merging two maps. The Horn algorithm requires sequential frame-by-frame feature matching in another map to calculate the relative pose, and its computation time is affected by the location of overlapping areas. If the overlapping area is closer to the starting point of the robot, the algorithm requires less time to complete map merging. Otherwise, it will take more time. The proposed algorithm reduces the number of keyframes involved in map merging by using the keyframes in the detected manholes.

The experiment evaluates the algorithms by comparing the error in the estimated positions of manholes on the merged maps. The algorithm computes the errors by measuring the absolute distances between detected manholes and their real positions. Given that there are multiple manholes, the average distance is used as the error metric. The scale factor of the map is defined as the ratio of the actual distance to the estimated distance of the first two keyframes in the initialisation. Additionally, the success rate of manhole detection is also presented, defined as the number of manholes detected versus the number of actual manholes on the map. Figure 5.10 shows the original point cloud maps with different coordinate systems, where yellow, green, and

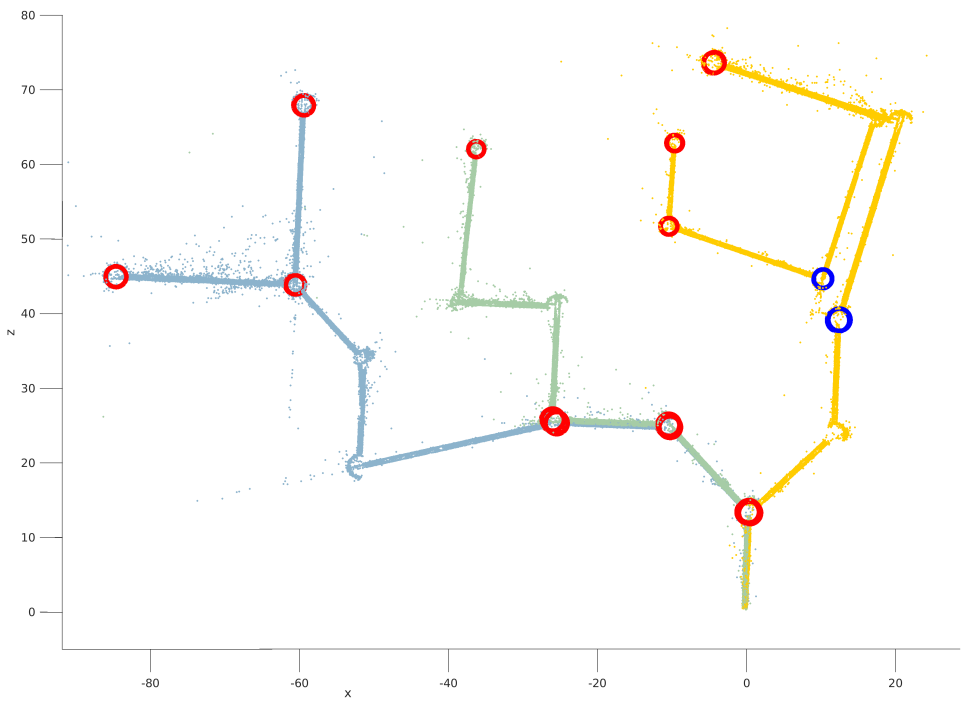



Figure 5.11: Map after Map Merging

blue point clouds correspond to synthetic data 1, 2, and 3, respectively. The red circles represent the detected manholes. Figure 5.11 displays the merging result of all feature maps, where the algorithm aligns the three maps based on shared manholes.

In Figure 5.11, two blue circles represent the same manhole. The robot went through the manhole twice from different entrances, and the vSLAM algorithm failed to recognise the loop, creating map points for the manhole and the connected pipe twice.

Three feature maps are provided, and the proposed algorithm was tested by merging them in pairs. Table 5.5 presents the experimental results as the average of ten runs. The proposed algorithm is deterministic, meaning that it does not incorporate any random elements. As a result, it consistently produces the same output for a given input, regardless of the number of times it is executed. The Horn [128] algorithm employs the RANSAC method [36] and ICP [121] to determine the relative pose between point clouds during its execution. Consequently, the results vary with each run due to the inherent randomness in these methods.

Table 5.5 compares the performance of the two algorithms in map merging. The proposed algorithm (referred to as MH) from Section 5.2 has advantages in computational efficiency, especially for large-scale maps. Since the proposed algorithm only matches features observed at the manhole, its computation is less affected by map size than the Horn method. The Horn algorithm establishes a more extensive feature matching between two maps, which enables

| Dataset | Synthetic 1 and 2 | | Synthetic 2 and 3 | | Synthetic 1 and 3 | |
|---|---|---|---|---|---|---|
| Algorithm | MH | Horn | MH | Horn | MH | Horn |
| Time (s) | 13.47 | 70.48 | 14.17 | 81.06 | 20.26 | 135.4 |
| Max Error (m) | 21.16 | 23.02 | 41.82 | 43.04 | 32.07 | 38.04 |
| Error (m) | 21.16 | 19.54 | 41.82 | 34.42 | 32.07 | 32.49 |
| Min Error (m) | 21.16 | 15.55 | 41.82 | 27.1 | 32.07 | 21.15 |
| Proportion of detected manholes (%) | 57.14 | | 57.14 | | 75.00 | |

Table 5.5: The results for map merging

the system to optimise all the keyframe poses from both maps. The result has demonstrated significant efficiency advantages through experimental validation. The accuracy of this method is comparable to existing approaches, maintaining similar error levels.

### 5.5.2 Map Alignment Algorithm Evaluation

This experiment evaluates the performance of the algorithm proposed in Sections 5.3 and 5.4 on feature maps generated in the previous experiment. The experiment aligns the merged maps with prior maps and calculates their relative similarity matrices. Since there are no similar algorithms for aligning feature maps from a pipe network with a topological map, only the proposed method is evaluated here.

The experiment evaluates the proposed algorithm by calculating the proportion of successfully matched vertices and the error in the relative similarity transformation matrix between two maps. The proportion of matched vertices is defined as the ratio of correctly matched vertices in the estimated map to all vertices in the estimated map, including missing vertices. The absolute pose error refers to the difference between the estimated similarity transformation matrix and the ground truth. Figures 5.12, 5.13, 5.14, and 5.15 illustrate the results of the initial vertex matching and the final results. The performance of manhole detection has been displayed in the previous subsection and will not be repeated here.

The evaluation involves both initial vertex matching and the extension algorithm. Initial vertex matching searches for seed matches, which serve as the starting vertex for the extension algorithm. The experiment will provide the number of seeds and the proportion of correct seeds. The extension algorithm uses all estimated vertices to refine the matching and identify the optimal solution. The experiment calculates the proportion of successfully matched vertices and the error in the relative similarity transformation matrix. The proportion of matched vertices is the ratio of correctly matched vertices in the estimated map to the corresponding vertices in the prior map. The absolute pose error is the difference between the similarity transformation

| Dataset | Feature Map 1 | Feature Map 2 | Feature Map 3 |
|---|---|---|---|
| number of seeds | 3 | 3 | 3 |
| The ratio of correct seeds | 1 | 1 | 1 |
| Ratio of Matched Vertices | 100% | 100% | 100% |
| Relative Pose Error | 3.12 | 5.99 | 4.36 |

Table 5.6: The results for map alignment

matrix and the ground truth.

The proposed algorithm transforms the feature maps into topological maps and then aligns them with the prior maps to obtain the relative similarity transformation matrices. However, due to occasional failures in manhole detection, the estimated topological map deviates from the prior map. As mentioned in the previous section, vSLAM may fail to close the loop, leading to the creation of a duplicate point cloud for a manhole. The proposed algorithm would detect manholes twice and convert them into vertices. Two vertices represent the same manhole, which exceeds the capability of the proposed algorithm. The experiment artificially merged these two vertices in the estimated topological map. The vertex with the later timestamp is deleted, and its connected edges are linked to the vertex with the earlier timestamp.

Figures 5.12, 5.13 and 5.14 illustrate the results of initial vertex matching, where the algorithm identified several seeds containing correct matches based on structural and geometric information. Initial vertex matching, which relies solely on a vertex and its adjacent edge information, often includes incorrect matches. In these figures, the green topological map represents the prior map, while the blue topological map represents the estimated map. The coloured lines connecting vertices from the two maps indicate the vertex correspondence provided by the initial matching. In Figure 5.12, the estimated map lacks corresponding vertices for Vertices $V_3'$, $V_5'$, and $V_{10}'$ from the prior map. As a result, the descriptors of the neighbouring vertices of the missing vertices differ significantly from the corresponding descriptors in the prior map. Due to scale drift in the estimated map, the edge length of Vertex $V_7'$ in the prior map differs from that of Vertex $V_4$ in the estimated map. Thus, only Vertex $V_6$ remains close to the ground truth. Because many vertices in the prior map have similar descriptors, the initial matching incorrectly matches Vertex $V_2$ with Vertices $V_3$, $V_9$, and $V_{15}$.

Similarly, in Figures 5.13 and 5.14, the estimated map lacks corresponding vertices for Vertices $V_{10}'$, $V_{11}'$, $V_{12}'$, $V_{14}'$, and $V_{15}'$ from the prior map. This leads to significant deviations in the descriptors of their neighbouring vertices from the ground truth. The algorithm yields the same results as in Figure 5.12. Despite the missing vertices, the extension algorithm successfully discards unreasonable matching sets in favour of correct ones during the extension process. Table 5.6 demonstrates that the algorithm can efficiently complete topological map matching and

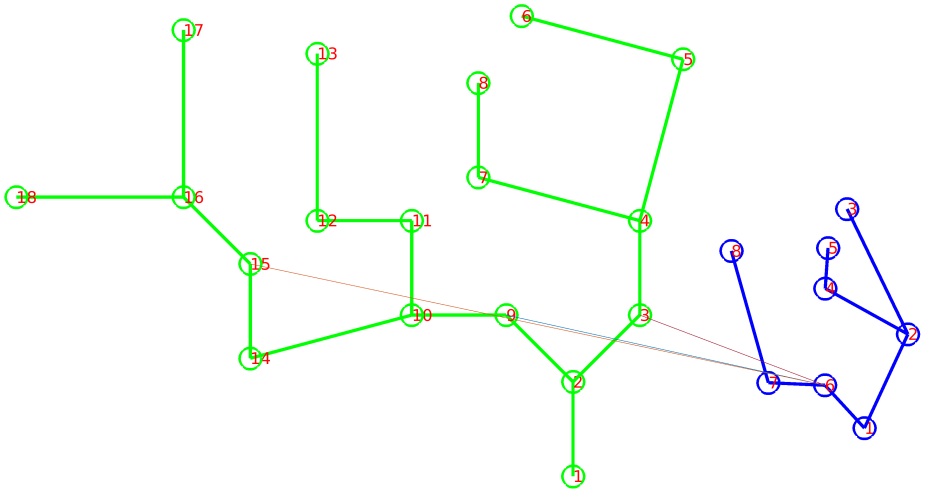


Figure 5.12: Initial match for Feature map 1

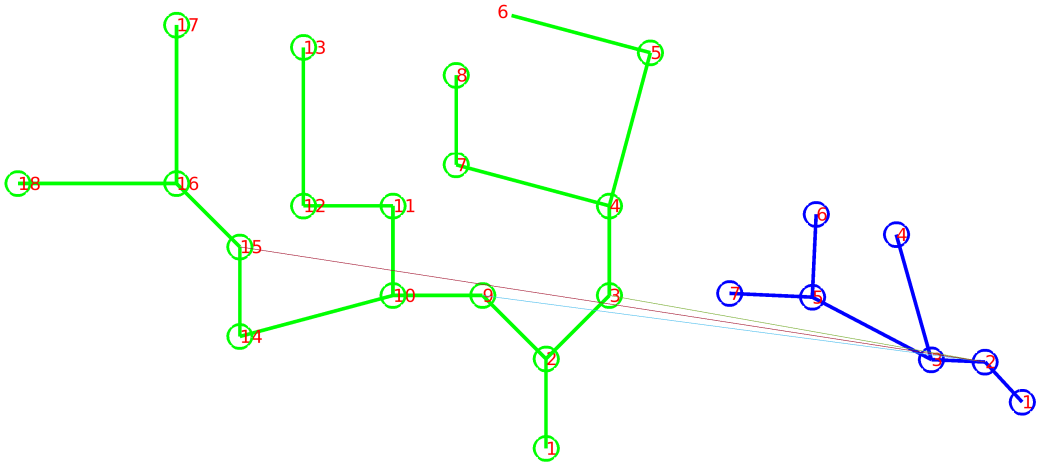


Figure 5.13: Initial match for Feature map 2

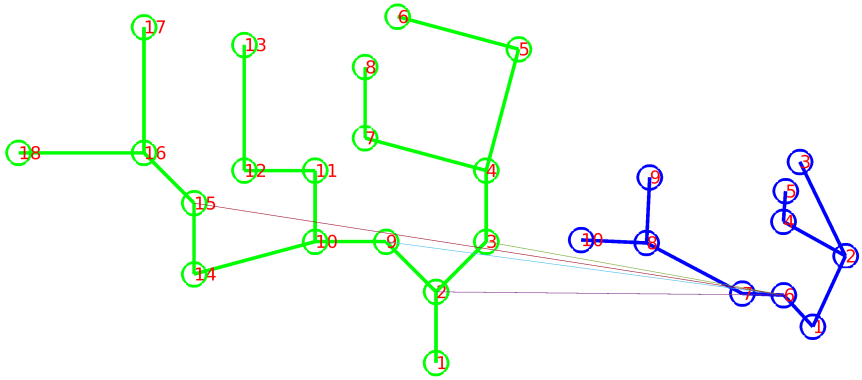


Figure 5.14: Initial match for Feature map 3

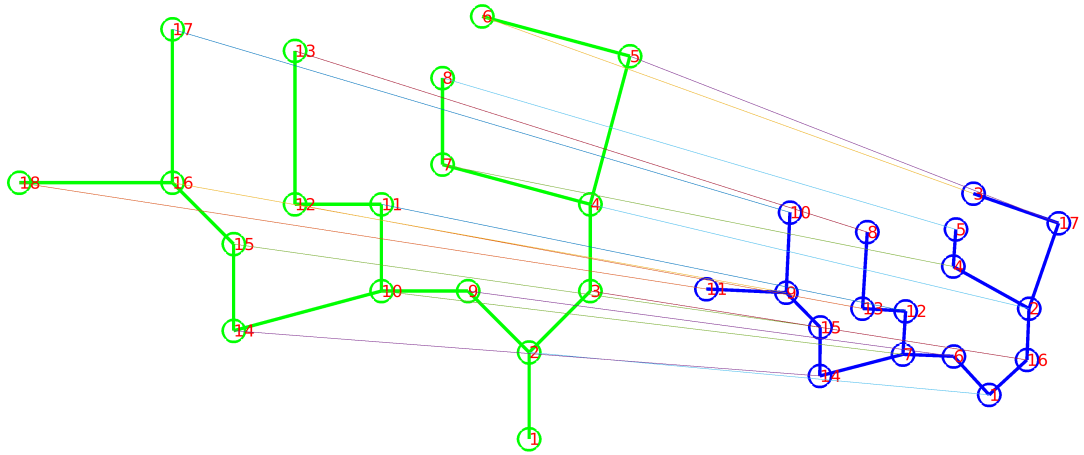


Figure 5.15: The estimated map after map alignment

estimate the relative similarity transformation matrices between maps. Figure 5.15 shows the two maps after extension. The proposed algorithm added missing vertices to the estimated map, and those vertices are coloured red for identification. The results confirm that the algorithm can efficiently complete topological map matching and estimate the relative similarity transformation matrices between maps. It can handle missing vertices in the estimated maps.

## 5.6 Summary

This chapter proposes an algorithm designed for multiple robots exploring a large pipe network. Multi-robot visual SLAM systems require merging maps created by different robots. Due to the difficulties of online communication, an offline algorithm has been chosen. The algorithm leverages key characteristics of pipe networks, selecting manhole keyframes to reduce the number of keyframes involved in map merging. This also helps avoid some unintended incorrect feature associations. Experiments prove that the proposed method is computationally efficient while maintaining accuracy.

The proposed algorithm utilises prior maps to estimate scale factors and actual positions, addressing limitations in vSLAM. It transforms point cloud maps into topological maps and aligns them with prior maps. Even when the estimated topological maps contain missing or false-positive vertices, the algorithm robustly matches the maps, ensuring precise alignment and scale estimation.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

This dissertation aims to tackle the challenges of mapping pipe networks and localising robots within these networks, which is crucial for robots performing NDT. Cameras, being inexpensive and commonly used sensors, are widely utilised in robots. This research investigates vSLAM algorithms in pipe networks, assuming that the camera can track features inside the pipes. However, vSLAM algorithms in pipes face significant drift, and the absolute pose and the scale factor are unobservable. Consequently, existing algorithms cannot estimate the absolute position of the robot within the pipe networks or build an accurate model.

The study found that vSLAM algorithms suffer from severe scale drift in pipes, with the error in the estimated trajectory increasing significantly as the robot moves forward. The estimated speed of the robot gradually decreases, and the newly generated point cloud also shrinks. This makes vSLAM unable to estimate the robot's position and map the pipeline network accurately. Existing algorithms attempt to correct pose estimation by incorporating the structural information of the pipe networks, leveraging the cylindrical shape of the pipes and the topological distribution of the network.

To utilise cylindrical information, this dissertation explores cylinder extraction algorithms and analyses the issues with existing methods. Chapter 3 introduces a robust cylinder extraction algorithm that can accurately select cylindrical points and estimate their corresponding cylinders in complex point clouds. This method is more accurate and more robust than existing algorithms. In most cases, In most cases, its directional and reprojection errors are only half those of the RLTS algorithm [3]. The dissertation summarises the reasons for the failure of cylinder fitting algorithms that rely on point-cylinder distances and proposes a new method to identify such situations based on the reprojection error, thus avoiding incorrect cylinder adoption that could lead to vSLAM failures.

Chapter 4 presents a new vSLAM algorithm that can extract cylinders and utilise cylin-

drical information. When the robot moves within a pipe, the vSLAM algorithm processes a sequence of images, tracks features, estimates the robot's pose and generates point clouds according to photography Geometry. The proposed algorithm uses the cylinder extraction algorithm from Chapter 3 to select cylindrical points and estimate cylinders from the point cloud. Upon detecting a cylinder, the algorithm incorporates cylindrical information into triangulation, local optimisation, and loop closure optimisation. The proposed algorithm can mitigate the scale drift and demonstrates better performance in both simulation and real data compared to the existing open-source algorithm ORB-SLAM2 [2]. If isodiametric information is available, the algorithm can further reduce scale drift and achieve lower errors. The proposed algorithm's error on real data is only 30% of that of ORB-SLAM2.

When the proposed vSLAM algorithm detects a cylinder in the point cloud, it evaluates the estimated cylinder based on the proportion of cylindrical points. If there are many non-cylindrical points in the environment, the algorithm will reject the estimated cylinder even if the estimated cylinder parameters are correct. Additionally, when there are many obstacles at the bottom of the pipe, the corresponding point cloud contains a large number of non-cylindrical points. In this case, the proportion of cylindrical points may be very low, and the system will not accept the estimated cylinder.

Chapter 5 introduces an offline map merging algorithm for multi-robot vSLAM to improve exploration efficiency in large-scale pipe networks. Given the vast and complex nature of pipeline networks, it is reasonable to use multiple robots for exploration. Existing algorithms estimate the relative transformation between two maps according to the overlapping regions. The system establishes manhole correspondence by matching features between the keyframes of the two maps. However, A large map contains numerous keyframes, making the feature matching computationally expensive. Additionally, it is likely to have visually similar scenes in large maps, which can lead to erroneous manhole correspondences. The system might mistakenly identify images from different locations as the same place, resulting in incorrect map merging. By utilising the structural information of the pipe network, the system selects manhole keyframes to reduce the number of keyframes involved in feature matching during map merging, thus lowering computational demands and reducing the likelihood of erroneous feature matching. Experiments demonstrate that the proposed algorithm requires only one-fifth of the time needed by existing algorithms [132].

Chapter 5 also explores the role of prior maps in vSLAM. When robots explore pipe networks, prior maps are often available. Compared to maps constructed by vSLAM, prior maps have many advantages, such as higher accuracy, no accumulated errors, inherent scale factors, and absolute positions. Using a prior map, the algorithm can calculate the absolute position and scale factor of the estimated map. However, establishing correspondences between

the two different kinds of maps is challenging. This study converts feature maps into topological maps and then aligns manholes on the two maps using graph matching. Once map alignment is complete, the algorithm can obtain the absolute position and scale factor of the estimated map. The algorithm can correctly match all vertices and accurately align the two maps in simulation experiments.

The proposed map merging and map alignment algorithms are based on manhole detection results, which impose critical requirements on manhole detection. However, the internal shapes and textures of manholes are not unique, making manhole detection challenging. Incorrect manhole detection results can lead to incorrect manhole matches or significantly erroneous topological maps, which could make these two algorithms fail.

By addressing these challenges and proposing new methodologies, this dissertation advances the field of robotic mapping and localisation in pipe networks, offering practical solutions and highlighting directions for future research.

## 6.2 Future Work

In this section, some promising approaches which may improve the current methods of vSLAM in a pipe environment will be discussed.

1. Sparse feature environment within pipelines poses a significant challenge to visual SLAM, which is a common issue in many environments. Traditional feature extraction methods often struggle to address this problem. Deep learning offers new solutions, with transformers achieving promising results in pixel tracking within sparse feature environments [168]. Additionally, techniques like SuperGlue have demonstrated their strengths [169]. SuperGlue is an advanced neural network architecture designed to match features between images. SuperGlue uses a Graph Neural Network to model the relationships between features. Such algorithms can enhance the performance of visual SLAM within pipelines, potentially expanding its application to other difficult environments significantly.

2. Feature matching for a feature with a large parallax angle often leads to failures in loop closure detection. In pipe networks, when robots traverse from different directions, vSLAM algorithms may fail to recognise loop closures due to poor feature-matching performance at large parallax angles. This issue in pipeline networks where loop closures are rare makes it unlikely for vSLAM to leverage loop closure optimisation to correct cumulative errors. Addressing this challenge could significantly improve the performance of vSLAM in pipeline networks.

3. The proposed vSLAM algorithm confirms cylinder keyframes according to the proportion of cylindrical points, which limits its ability to utilise cylindrical information in complex

scenarios. In real-world environments, pipelines or tunnels may not be perfectly cylindrical, and only certain parts may have cylindrical surfaces. Current algorithms might discard valuable cylindrical information due to map points from the non-cylindrical surface. Developing a more flexible approach to segmenting cylindrical and non-cylindrical regions within images could improve the performance of the proposed visual SLAM algorithm.

It is recommended that future researchers concentrate on advancing mapping techniques, such as creating a global map from the local maps acquired by each robot using various image processing methods. Particular attention should be given to investigating topological methods, with an emphasis on those capable of handling data association in the images.

From a practical perspective, a challenge arises when a robot moves through the pipe network and approaches a manhole, but the camera fails to capture informative data. To address these gaps, future work could explore integrating multiple sensor modalities, such as odometry and acoustic data, to bridge the gaps caused by missing information. This sensor fusion method could offer a robust solution to the data limitations commonly encountered in pipe networks.

# Bibliography


[1] Raul Mur-Artal, J. M.M. Montiel, and Juan D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31:1147–1163, 2015. ISSN 15523098. doi: 10.1109/TRO.2015.2463671.

[2] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33:1255–1262, 2017. ISSN 15523098. doi: 10.1109/TRO.2017.2705103.

[3] Abdul Nurunnabi, Yukio Sadahiro, Roderik Lindenbergh, and David Belton. Robust cylinder fitting in laser scanning point cloud data. *Measurement*, 138:632–651, 2019. ISSN 02632241. doi: 10.1016/j.measurement.2019.01.095. URL `https://doi.org/10.1016/j.measurement.2019.01.095`.

[4] Sarah Edwards and Sean Anderson. Visual Odometry for Robot Localisation in Feature-Sparse Sewer Pipes Using Joint and Manhole Detections – Data, 9 2022. URL `https://orda.shef.ac.uk/articles/dataset/Visual_Odometry_for_Robot_Localisation_in_Feature-Sparse_Sewer_Pipes_Using_Joint_and_Manhole_Detections_--_Data/21198070`.

[5] Raul Mur-Artal, Juan D. Tardos, J. M. M. Montiel, and Dorian Galvez-Lopez. ORB-SLAM2, 2016. URL `https://github.com/raulmur/ORB-SLAM2`.

[6] Rui Zhang, Rob Worley, Sarah Edwards, Jonathan Aitken, Sean Anderson, and Lyudmila Mihaylova. Video data for robot localisation within pipe networks, 2023. URL `https://dx.doi.org/10.21227/4fz3-cc46`.

[7] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Dso, 2017. URL `https://github.com/JakobEngel/dso`.

[8] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo, 2014. URL `https://github.com/uzh-rpg/rpg_svo`.

[9] Tzu Yi Chuang and Cheng Che Sung. Learning and SLAM based decision support platform

for sewer inspection. *Remote Sensing*, 12(6, article number 968), 2020. ISSN 20724292. doi: 10.3390/rs12060968.

[10] GOV.UK. Water and treated water, 2015. URL https://www.gov.uk/government/publications/water-and-treated-water/water-and-treated-water.

[11] Government Office for Science. UK water research and innovation framework: Taking responsibility for water, 2012. ISSN 1303-2917. URL https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/294176/11-1416-taking-responsibility-for-water-summary.pdf.

[12] The Consumer Council for Water. Water, water everywhere? Delivering resilient water and waste water services everywhere (2017-2018), 2017. ISSN 00130613. [Online] Available at: https://www.ccwater.org.uk/research/water-water-everywhere-delivering-resilient-water-and-wastewater-services-2017-18/ (accessed 20/07/2020).

[13] Xiaoyu Zhang, Wei Wang, Xianyu Qi, Ziwei Liao, and Ran Wei. Point-plane slam using supposed planes for indoor environments. *Sensors*, 19:1–20, 2019. ISSN 14248220. doi: 10.3390/s19173795.

[14] Sabine El Kahi, Daniel Asmar, Adel Fakih, Juan Nieto, and Eduardo Nebot. A vison-based system for mapping the inside of a pipe. In *Proc. 2011 IEEE International Conference on Robotics and Biomimetics*, pages 2605–2611. IEEE, 2011. ISBN 9781457721373. doi: 10.1109/ROBIO.2011.6181697.

[15] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2017.

[16] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE International Conference on Robotics and Automation*, pages 15–22, Hong Kong, China, 2014. doi: 10.1109/ICRA.2014.6906584.

[17] Peter Hansen, Hatem Alismail, Peter Rander, and Brett Browning. Towards a visual perception system for pipe inspection : Monocular visual odometry. Technical Report CMU-CS-QTR-104/CMU-RI-TR-10-22, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 2010. URL http://www.ri.cmu.edu/publication_view.html?pub_id=6653.

[18] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] Jianjun Gui, Dongbing Gu, Sen Wang, and Huosheng Hu. A review of visual inertial odometry from filtering and optimisation perspectives. *Advanced Robotics*, 29:1289–1301, 2015. ISSN 15685535. doi: 10.1080/01691864.2015.1057616. URL `http://dx.doi.org/10.1080/01691864.2015.1057616`.

[20] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32: 1309–1332, 2016. ISSN 15523098. doi: 10.1109/TRO.2016.2624754.

[21] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2:194–220, 2017. ISSN 23798858. doi: 10.1109/TIV.2017.2749181.

[22] Guoquan Huang. Visual-inertial navigation: A concise review. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2019-May, pages 9572–9582, 2019. ISBN 9781538660263. doi: 10.1109/ICRA.2019.8793604.

[23] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11, 2022. ISSN 22186581. doi: 10.3390/robotics11010024.

[24] Rui Zhang, Rob Worley, Sarah Edwards, Jonathan Aitken, Sean R Anderson, and Lyudmila Mihaylova. Visual simultaneous localization and mapping for sewer pipe networks leveraging cylindrical regularity. *IEEE Robotics and Automation Letters*, 8(6):3406–3413, 2023.

[25] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017. ISSN 14740508. doi: 10.1017/S096249291700006X.

[26] Sho Kagami, Hajime Taira, Naoyuki Miyashita, Akihiko Torii, and Masatoshi Okutomi. 3D pipe network reconstruction based on structure from motion with incremental conic shape detection and cylindrical constraint. In *Proc. IEEE International Symposium on Industrial Electronics*, volume 2020-June, pages 1345–1352, 2020. ISBN 9781728156354. doi: 10.1109/ISIE45063.2020.9152377.

[27] Sandro Esquivel, Reinhard Koch, and Heino Rehse. Reconstruction of sewer shaft profiles from fisheye-lens camera images. In *Pattern Recognition*, pages 332–341. Springer Berlin Heidelberg, 2009. ISBN 3642037976. doi: 10.1007/978-3-642-03798-6_34.

[28] Yuhang Zhang, Richard Hartley, John Mashford, Lei Wang, and Stewart Burn. Pipeline reconstruction from fisheye images. *Journal of WSCG*, 19(1=3):49–57, 2011.

[29] Yuanzheng Gong, Richard S. Johnston, C. David Melville, and Eric J. Seibel. Axial-stereo 3-d optical metrology for inner profile of pipes using a scanning laser endoscope. *International Journal of Optomechatronics*, 9:238–247, 2015. ISSN 15599620. doi: 10.1080/15599612.2015.1059535. URL `http://dx.doi.org/10.1080/15599612.2015.1059535`.

[30] Peter Hansen, Hatem Alismail, Peter Rander, and Brett Browning. Visual mapping for natural gas pipe inspection. *International Journal of Robotics Research*, 34:532–538, 2015. ISSN 17413176. doi: 10.1177/0278364914550133.

[31] Johannes Künzel, Thomas Werner, Peter Eisert, Jan Waschnewski, Ronja Möller, and Ralf Hilpert. Automatic analysis of sewer pipes based on unrolled monocular fisheye images. volume 2018-Janua, pages 2019–2027, 2018. ISBN 9781538648865. doi: 10.1109/WACV.2018.00223.

[32] Craig M. Shakarji. Least-squares fitting algorithms of the nist algorithm testing system. *Journal of Research of the National Institute of Standards and Technology*, 103:633–641, 1998. ISSN 1044677X. doi: 10.6028/jres.103.043.

[33] Gabor Lukács, Ralph Martin, and Dave Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1406:671–686, 1998. ISSN 16113349. doi: 10.1007/BFb0055697.

[34] Jean francis Lalonde, Nicolas Vandapel, and Martial Hebert. Automatic three-dimensional point cloud processing for forest inventory automatic three-dimensional point cloud processing for forest inventory. pages 1–16, 2006. URL `papers2://publication/uuid/D2EBE629-8322-4421-8003-004DA8614C74`.

[35] Daniel Lopez-Escogido and Luis Gerardo De La Fraga. Automatic extraction of geometric models from 3D point cloud datasets. In *Proc. of the International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–5. IEEE, 2014. ISBN 9781479962310. doi: 10.1109/ICEEE.2014.6978316.

[36] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981. ISSN 15577317. doi: 10.1145/358669.358692.

[37] Open Robotics. Robot operating system, 2007. URL `https://www.ros.org/`.

[38] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664. IEEE, 2010.

[39] Jonathan M. Aitken, Mathew H. Evans, Rob Worley, S. Edwards, Rui Zhang, Tony Dodd, Lyudmila Mihaylova, and Sean R. Anderson. Simultaneous localization and mapping for inspection robots in water and sewer pipe networks: A review. *IEEE Access*, 9:140173–140198, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3115981.

[40] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2004. doi: 10.1109/cvpr.2004.1315094.

[41] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. pages 834–849. Springer, 2014. ISBN 9781479928392.

[42] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct Sparse Odometry with Loop Closure. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. International Conference on Intelligent Robots and Systems (IROS), 2018.

[43] David Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:756–770, 2004.

[44] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81:155–166, 2009. ISSN 09205691. doi: 10.1007/s11263-008-0152-6.

[45] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on pattern analysis and machine intelligence*, 29, 2007. ISSN 01628828.

[46] Jianbo Shi and Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pages 593–600, Seattle, WA, USA, 1994. IEEE Comput. Soc. Press. ISBN 978-0-8186-5825-9. doi: 10.1109/CVPR.1994.323794.

[47] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. pages 225–234, 2007. ISBN 9781424417506. doi: 10.1109/ISMAR.2007.4538852.

[48] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. pages 1–15, 2020. URL `http://arxiv.org/abs/2007.11898`.

[49] David Nist and Henrik Stew. Scalable recognition with a vocabulary tree david. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2161–2168, 2006. URL `http://www.vis.uky.edu/dnister/http://www.vis.uky.edu/stewe/`.

[50] Arren Glover, William Maddern, Michael Warren, Stephanie Reid, Michael Milford, and Gordon Wyeth. Openfabmap: An open source toolbox for appearance-based loop closure detection. pages 4730–4735. IEEE, 2012. ISBN 9781467314039. doi: 10.1109/ICRA.2012.6224843.

[51] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. A first-estimates jacobian ekf for improving slam consistency. volume 54, pages 373–382. Springer, 2009. ISBN 9783642001956. doi: 10.1007/978-3-642-00196-3_43.

[52] Edward Rosten and Tom Drummond. Machine learning for high speed corner detection. In *In Proceedings of Computer Vision–ECCV 2006: 9th European Conference on Computer Vision*, volume Part I 9, pages 430–443, 2006.

[53] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2008.275.

[54] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[55] Shiqin Sun, Benlian Xu, Yidan Sun, and Zhen Sun. Sparse pointcloud map fusion of multi-robot system. In *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 270–274, 2018. doi: 10.1109/ICCAIS.2018.8570467.

[56] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. pages 3565–3572, 2007. ISBN 1424406021. doi: 10.1109/ROBOT.2007.364024.

[57] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. volume 2017-Janua, pages 5816–5824, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.616.

[58] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3:965–972, 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2793349.

[59] Zhenfei Yang and Shaojie Shen. Monocular visual-inertial state estimation with online initialization and camera-imu extrinsic calibration. *IEEE Transactions on Automation Science and Engineering*, 14:39–51, 2017. ISSN 15455955. doi: 10.1109/TASE.2016.2550621.

[60] D Bruce Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *In IJCAI'81: 7th International Joint Conference on Artificial Intelligence*, volume Vol. 2, pages 674–679, 1981.

[61] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In *Computer Vision – ECCV 2010*, volume 6314, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15560-4 978-3-642-15561-1. doi: 10.1007/978-3-642-15561-1_56.

[62] Raul Mur-Artal and Juan D. Tardos. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2:796–803, 2017. ISSN 23773766. doi: 10.1109/LRA.2017.2653359.

[63] Lukas deVon Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. pages 2510–2517, 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8462905.

[64] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. Structslam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology*, 64:1364–1375, 2015. ISSN 00189545. doi: 10.1109/TVT.2015.2388780.

[65] Mohan Krishna Nutalapati, Lavish Arora, Anway Bose, Ketan Rajawat, and Rajesh M. Hegde. Model free calibration of wheeled robots using gaussian process. *Proc. IEEE International Conference on Intelligent Robots and Systems*, pages 29–35, 2019. ISSN 21530866. doi: 10.1109/IROS40897.2019.8967569.

[66] Meixiang Quan, Songhao Piao, Minglang Tan, and Shi Sheng Huang. Tightly-coupled monocular visual-odometric slam using wheels and a mems gyroscope. *IEEE Access*, 7: 97374–97389, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2930201.

[67] Kejian J. Wu, Chao X. Guo, Georgios Georgiou, and Stergios I. Roumeliotis. Vins on wheels. pages 5155–5162, 2017. ISBN 9781509046331. doi: 10.1109/ICRA.2017.7989603.

[68] Alexandre Eudes, Maxime Lhuillier, Sylvie Naudet-Collette, and Michel Dhome. Fast odometry integration in local bundle adjustment-based visual slam. *Proc. International Conference on Pattern Recognition*, pages 290–293, 2010. ISSN 10514651. doi: 10.1109/ICPR.2010.80.

[69] Julien Michot, Adrien Bartoli, and Francois Gaspard. Bi-objective bundle adjustment with application to multi-sensor slam. *Proc. International Symposium on 3D Data Processing, Visualization and Transmission*, page 8, 2010.

[70] Lawrence E Widman. Computational inverse problems in electrocardiography. *Journal of Biomedical Informatics*, 35:51, 2002. ISSN 15320464. doi: 10.1016/s1532-0464(02)00008-4.

[71] Javier Hidalgo-Carrio, Daniel Hennes, Jakob Schwendner, and Frank Kirchner. Gaussian process estimation of odometry errors for localization and mapping. *Proc. IEEE International Conference on Robotics and Automation*, pages 5696–5701, 2017. ISSN 10504729. doi: 10.1109/ICRA.2017.7989670.

[72] Yan Lu, Dezhen Song, and Jingang Yi. High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment. pages 1540–1545, 2014. doi: 10.1109/ICRA.2014.6907056.

[73] Antoni Rosinol, Torsten Sattler, Marc Pollefeys, and Luca Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. volume 2019-May, pages 8220–8226, 2019. ISBN 9781538660263. doi: 10.1109/ICRA.2019.8794456.

[74] Xin Li, Yijia He, Jinlong Lin, and Xiao Liu. Leveraging planar regularities for point line visual-inertial odometry. 2020. URL `http://arxiv.org/abs/2004.11969`.

[75] Andrew P. Gee, Denis Chekhlov, Andrew Calway, and Walterio Mayol-Cuevas. Discovering higher level structure in visual slam. *IEEE Transactions on Robotics*, 24:980–990, 2008. ISSN 15523098. doi: 10.1109/TRO.2008.2004641.

[76] Bo Xu, Xin Li, JianCheng Li, Chau Yuen, JiCheng Dai, and YiQun Gong. Pvi-dso: Leveraging planar regularities for direct sparse visual-inertial odometry. 2022. URL `http://arxiv.org/abs/2204.02635`.

[77] Bo Xu, Peng Wang, Yijia He, Yu Chen, Yongnan Chen, and Ming Zhou. Leveraging structural information to improve point line visual-inertial odometry. *IEEE Robotics and Automation Letters*, 7:3483–3490, 2022. ISSN 23773766. doi: 10.1109/LRA.2022.3146893.

[78] Xin Li, Yanyan Li, Evin Pnar Ornek, Jinlong Lin, and Federico Tombari. Co-planar parametrization for stereo-slam and visual-inertial odometry. *IEEE Robotics and Automation Letters*, 5:6972–6979, 2020. ISSN 23773766. doi: 10.1109/LRA.2020.3027230.

[79] Qiang Fu, Jialong Wang, Hongshan Yu, Islam Ali, Feng Guo, and Hong Zhang. Pl-vins: Real-time monocular visual-inertial slam with point and line. *arXiv*, 2020. ISSN 23318422.

[80] Jinyu Li, Bangbang Yang, Kai Huang, Guofeng Zhang, and Hujun Bao. *Robust and efficient visual-inertial odometry with multi-plane priors*. Springer International Publishing, 2019. ISBN 9783030317256. doi: 10.1007/978-3-030-31726-3-24. URL `http://dx.doi.org/10.1007/978-3-030-31726-3-24`.

[81] Chuchu Chen, Patrick Geneva, Yuxiang Peng, Woosik Lee, and Guoquan Huang. Monocular Visual-Inertial Odometry with Planar Regularities. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6224–6231, London, United Kingdom, 2023. IEEE. ISBN 9798350323658. doi: 10.1109/ICRA48891.2023.10160620.

[82] Haoang Li, Jian Yao, Jean Charles Bazin, Xiaohu Lu, Yazhou Xing, and Kang Liu. A monocular slam system leveraging structural regularity in manhattan world. pages 2518–2525. IEEE, 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8463165.

[83] Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari. Structure-slam: Low-drift monocular slam in indoor environments. *IEEE Robotics and Automation Letters*, 5:6583–6590, 2020. ISSN 23773766. doi: 10.1109/LRA.2020.3015456.

[84] F. Cheng, C. Liu, H. Wu, and M. Ai. Direct sparse visual odometry with structural regularities for long corridor environments. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 43:757–763, 2020. ISSN 16821750. doi: 10.5194/isprs-archives-XLIII-B2-2020-757-2020.

[85] Yan Lu and Dezhen Song. Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. *IEEE Transactions on Robotics*, 31:736–749, 2015. ISSN 15523098. doi: 10.1109/TRO.2015.2424032.

[86] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. pages 4503–4508, 2017. ISBN 9781509046331. doi: 10.1109/ICRA.2017.7989522.

[87] Yijia He, Ji Zhao, Yue Guo, Wenhao He, and Kui Yuan. Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. *Sensors (Switzerland)*, 18:1–25, 2018. ISSN 14248220. doi: 10.3390/s18041159.

[88] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35:925–938, 2019. ISSN 19410468. doi: 10.1109/TRO.2019.2909168.

[89] Danping Zou, Yuanxin Wu, Ling Pei, Haibin Ling, and Wenxian Yu. Structvio: Visual-inertial odometry with structural regularity of man-made environments. *IEEE Transactions on Robotics*, 35:999–1013, 2019. ISSN 19410468. doi: 10.1109/TRO.2019.2915140.

[90] Federico Nardi, Bartolomeo Della Corte, and Giorgio Grisetti. Unified representation and registration of heterogeneous sets of geometric primitives. *IEEE Robotics and Automation Letters*, 4:625–632, 2019. ISSN 23773766. doi: 10.1109/LRA.2019.2891989.

[91] Hyunjun Lim, Yeeun Kim, Kwangik Jung, Sumin Hu, and Hyun Myung. Avoiding degeneracy for monocular visual slam with point and line features. *Proceedings - IEEE International Conference on Robotics and Automation*, 2021-May:11675–11681, 2021. ISSN 10504729. doi: 10.1109/ICRA48506.2021.9560911.

[92] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:722–732, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2008.300.

[93] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24:794–805, 2013. ISSN 10473203. doi: 10.1016/j.jvcir.2013.05.006. URL `http://dx.doi.org/10.1016/j.jvcir.2013.05.006`.

[94] Xiao Shan Gao, Xiao Rong Hou, Jianliang Tang, and Hang Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:930–943, 2003. ISSN 01628828. doi: 10.1109/TPAMI.2003.1217599.

[95] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003. ISBN 9780511188955. doi: 10.1016/S0143-8166(01)00145-2.

[96] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[97] David G Low. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004. URL `https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf`.

[98] Peter Hansen, Hatem Alismail, Peter Rander, and Brett Browning. Pipe mapping with monocular fisheye imagery. In *Proc. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5180–5185, 2013. ISBN 9781467363587. doi: 10.1109/IROS.2013.6697105.

[99] Peter Hansen, Hatem Alismail, Brett Browning, and Peter Rander. Stereo visual odometry

for pipe mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4020–4025. IEEE, 2011.

[100] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment —a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Proc. of the International Workshop on Vision Algorithms*, volume 1883, page 298–372, Berlin, Heidelberg, 1999. Springer.

[101] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:1115–1138, 1991.

[102] B. Paláncz, J. L. Awange, A. Somogyi, N. Rehány, T. Lovas, B. Molnár, and Y. Fukuda. A robust cylindrical fitting to point cloud data. *Australian Journal of Earth Sciences*, 63 (5):665–673, 2016. ISSN 0812-0099, 1440-0952. doi: 10.1080/08120099.2016.1230147.

[103] Filippo Bergamasco, Mara Pistellato, Andrea Albarelli, and Andrea Torsello. Cylinders extraction in non-oriented point clouds as a clustering problem. *Pattern Recognition*, 107, 2020. ISSN 00313203. doi: 10.1016/j.patcog.2020.107443.

[104] Christian Beder and Wolfgang Förstner. Direct solutions for computing cylinders from minimal sets of 3d points. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:135–146, 2006. ISSN 03029743. doi: 10.1007/11744023_11.

[105] Petko Faber and Bob Fisher. A buyer's guide to euclidean elliptical cylindrical and conical surface fitting. pages 54.1–54.10, 2001. doi: 10.5244/c.15.54.

[106] Soon-Wook Kwon, Frederic Bosche, Changwan Kim, Carl T. Haas, and Katherine A. Liapi. Fitting range data to primitives for rapid local 3D modeling using sparse range point clouds. *Automation in Construction*, 13(1):67–81, 2004. ISSN 09265805. doi: 10.1016/j.autcon.2003.08.007.

[107] Tahir Rabbani and Frank van den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. In *Proceedings of the ISPRS Workshop Laser Scanning, ISPRS Archives, Netherlands*, volume 36, pages 60–65, Entschede, 2005.

[108] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26:214–226, 2007. ISSN 14678659. doi: 10.1111/j.1467-8659.2007.01016.x.

[109] Abner M.C. Araújo and Manuel M. Oliveira. Connectivity-based cylinder detection in unorganized point clouds. *Pattern Recognition*, 100(April):107161, 2020. ISSN 00313203. doi: 10.1016/j.patcog.2019.107161.

[110] Trung Thien Tran, Van Toan Cao, and Denis Laurendeau. Extraction of cylinders and estimation of their parameters from point clouds. *Computers and Graphics (Pergamon)*, 46:345–357, 2015. ISSN 00978493. doi: 10.1016/j.cag.2014.09.027. URL `http://dx.doi.org/10.1016/j.cag.2014.09.027`.

[111] Mia Hubert, Peter J Rousseeuw, and Karlien Vanden Branden. ROBPCA: A New Approach to Robust Principal Component Analysis. *Technometrics*, 47(1):64–79, 2005. ISSN 0040-1706, 1537-2723. doi: 10.1198/004017004000000563.

[112] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Advances in neural information processing systems*, 22, 2009.

[113] Ashok Kumar Patil, Pavitra Holi, Sang Keun Lee, and Young Ho Chai. An adaptive approach for the reconstruction and modeling of as-built 3D pipelines from point clouds. *Automation in Construction*, 75:65–78, 2017. ISSN 09265805. doi: 10.1016/j.autcon.2016.12.002.

[114] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. ISSN 0001-0782, 1557-7317. doi: 10.1145/361237.361242.

[115] Thomas Chaperon and Francois Goulette. Extracting cylinders in full 3d data using a random sampling method and the Gaussian image. In *Proceedings of the Vision Modlling and Visualisation Conference (VMV) 2001*, pages 35–42, 2001.

[116] Monica Ballesta, Arturo Gil, Oscar Reinoso, and Miguel Julia. Alignment of Visual Maps in Multirobot FastSLAM. In *Proceedings of the 8th WSEAS International Conference on Computational Intelligence*, pages 92–97, 2009.

[117] Isaac Deutsch, Ming Liu, and Roland Siegwart. A framework for multi-robot pose graph SLAM. In *Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572, Angkor Wat, Cambodia, 2016. IEEE. ISBN 978-1-4673-8959-4. doi: 10.1109/RCAR.2016.7784092.

[118] S. Se, D.G. Lowe, and J.J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.839228.

[119] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Barcelona, Spain, 2011. IEEE. ISBN 978-1-4577-1102-2 978-1-4577-1101-5 978-1-4577-1100-8. doi: 10.1109/ICCV.2011.6126544.

[120] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[121] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Proceedings of SPIE: Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. Spie, 1992.

[122] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE, 2011.

[123] Monica Ballesta, Oscar Reinoso, Arturo Gil, Miguel Julia, and Luis Paya. Analysis of Map Alignment techniques in visual SLAM systems. In *Proceedings of the 2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pages 825–832, Hamburg, Germany, 2008. IEEE. ISBN 978-1-4244-1505-2. doi: 10.1109/ETFA.2008.4638493.

[124] Monica Ballesta, Arturo Gil, Oscar Reinoso, and Miguel Julia. Multi-robot map alignment in visual SLAM. *WSEAS Transactions on Systems*, 9(2):213–222, 2010.

[125] Gajamohan Mohanarajah, Vladyslav Usenko, Mayank Singh, Raffaello D'Andrea, and Markus Waibel. Cloud-Based Collaborative 3D Mapping in Real-Time With Low-Cost Robots. *IEEE Transactions on Automation Science and Engineering*, 12(2):423–431, 2015. ISSN 1545-5955, 1558-3783. doi: 10.1109/TASE.2015.2408456.

[126] Kin Leong Ho and Paul Newman. Multiple Map Intersection Detection using Visual Appearance. *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2005.

[127] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. ISSN 00222836. doi: 10.1016/0022-2836(81)90087-5.

[128] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, 1987. ISSN 1084-7529, 1520-8532. doi: 10.1364/JOSAA.4.000629.

[129] Arren Glover, William Maddern, Michael Warren, Stephanie Reid, Michael Milford, and Gordon Wyeth. OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4730–4735, St Paul, MN, USA, 2012. IEEE. ISBN 978-1-4673-1405-3 978-1-4673-1403-9 978-1-4673-1578-4 978-1-4673-1404-6. doi: 10.1109/ICRA.2012.6224843.

[130] Laurent Kneip and Paul Furgale. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, Hong Kong, China, 2014. IEEE. ISBN 978-1-4799-3685-4. doi: 10.1109/ICRA.2014.6906582.

[131] Adrian Garcea, Jiazhen Zhu, Dominik Van Opdenbosch, and Eckehard Steinbach. Robust Map Alignment for Cooperative Visual SLAM. In *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP)*, pages 4083–4087, Athens, 2018. IEEE. ISBN 978-1-4799-7061-2. doi: 10.1109/ICIP.2018.8451717.

[132] Dominik Van Opdenbosch and Eckehard Steinbach. Collaborative Visual SLAM Using Compressed Feature Exchange. *IEEE Robotics and Automation Letters*, 4(1):57–64, 2019. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2018.2878920.

[133] Andrey Bokovoy, Kirill Muraviev, and Konstantin Yakovlev. Map-Merging Algorithms for Visual SLAM: Feasibility Study and Empirical Evaluation. In Sergei O. Kuznetsov, Aleksandr I. Panov, and Konstantin S. Yakovlev, editors, *Artificial Intelligence*, volume 12412, pages 46–60. Springer International Publishing, Cham, 2020. ISBN 978-3-030-59534-0 978-3-030-59535-7. doi: 10.1007/978-3-030-59535-7_4.

[134] Benjamin Kuipers. Modeling Spatial Knowledge. *Cognitive Science*, 2(2):129–153, 1978.

[135] G Dudek, M Jenkin, E Milios, and D Wilkes. Topological Ecploration with Multiple Robots. In *Proceedings of the International Symposium on Robotics with Application (ISORA)*, Anchorage, Alaska, USA, 1998.

[136] Jan Oliver Wallgrün. Voronoi Graph Matching for Robot Localization and Mapping. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Marina L. Gavrilova, C. J. Kenneth Tan, and François Anton, editors, *Transactions on Computational Science IX*, volume 6290, pages 76–108. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-16006-6 978-3-642-16007-3. doi: 10.1007/978-3-642-16007-3_4.

[137] Sajad Saeedi, Liam Paull, Michael Trentini, Mae Seto, and Howard Li. Group Mapping: A Topological Approach to Map Merging for Multiple Robots. *IEEE Robotics & Automation Magazine*, 21(2):60–72, 2014. ISSN 1070-9932. doi: 10.1109/MRA.2014.2304091.

[138] Horst Bunke. Graph Matching: Theoretical Foundations, Algorithms, and Applications. In *Proceedings of Vision Interface Conference*, pages 82–88, Montreal, 2000.

[139] Wesley H. Huang and Kristopher R. Beevers. Topological Map Merging. *The International Journal of Robotics Research*, 24(8):601–613, 2005. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364905056348.

[140] Gorkem Erinc and Stefano Carpin. Anytime merging of appearance based maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1656–1662, St Paul, MN, USA, 2012. IEEE. ISBN 978-1-4673-1405-3 978-1-4673-1403-9 978-1-4673-1578-4 978-1-4673-1404-6. doi: 10.1109/ICRA.2012.6225131.

[141] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, 1983. ISSN 0018-9472, 2168-2909. doi: 10.1109/TSMC.1983.6313167.

[142] Malcolm Mielle, Martin Magnusson, and Achim J. Lilienthal. Using sketch-maps for robot navigation: Interpretation and matching. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 252–257, Lausanne, Switzerland, 2016. IEEE. ISBN 978-1-5090-4349-1. doi: 10.1109/SSRR.2016.7784307.

[143] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu, and Yung-Hsiang Lu. Multi-robot SLAM with topological/metric maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1467–1472, San Diego, CA, USA, 2007. IEEE. ISBN 978-1-4244-0911-2 978-1-4244-0912-9. doi: 10.1109/IROS.2007.4399142.

[144] Göksel Dedeoglu and Gaurav S. Sukhatme. Landmark-based Matching Algorithm for Cooperative Mapping by Autonomous Robots. In Lynne E. Parker, George Bekey, and Jacob Barhen, editors, *Distributed Autonomous Robotic Systems 4*, pages 251–260. Springer Japan, Tokyo, 2000. ISBN 978-4-431-67991-2 978-4-431-67919-6. doi: 10.1007/978-4-431-67919-6_24.

[145] Taigo Maria Bonanni, Bartolomeo Della Corte, and Giorgio Grisetti. 3-D Map Merging on Pose Graphs. *IEEE Robotics and Automation Letters*, 2(2):1031–1038, 2017. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2017.2655139.

[146] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944. ISSN 0022-0574.

[147] Mohamed Rahayem, Naoufel Werghi, and Johan Kjellander. Best ellipse and cylinder parameters estimation from laser profile scan sections. *Optics and Lasers in Engineering*, 50:1242–1259, 2012. ISSN 01438166. doi: 10.1016/j.optlaseng.2012.03.014. URL `http://dx.doi.org/10.1016/j.optlaseng.2012.03.014`.

[148] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual SLAM algorithms. *Robotics*, 11(1):24, 2022.

[149] Paulo Debenest, Michele Guarnieri, and Shigeo Hirose. PipeTron series - Robots for pipe inspection. In *Proceedings of the 3rd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 1–6, 2014.

[150] Amir A F Nassiraei, Yoshinori Kawamura, Alireza Ahrary, Yoshikazu Mikuriya, and Kazuo Ishii. A new approach to the sewer pipe inspection: Fully autonomous mobile robot "KANTARO". In *Proceedings of the 32nd Annual Conference on IEEE Industrial Electronics (IECON)*, pages 4088–4093. IEEE, 2006.

[151] Erich Rome, Joachim Hertzberg, Frank Kirchner, Ulrich Licht, and Thomas Christaller. Towards autonomous sewer robots: the MAKRO project. *Urban Water*, 1(1):57–70, 1999. ISSN 1462-0758.

[152] Gao Xiang. ORB-YGZ-SLAM, 2017. URL `https://github.com/gaoxiang12/ORB-YGZ-SLAM`.

[153] Qi Xuxiang, Yang Shaowu, and Yan Yuejin. ORB-SLAM2 with semantic label, 2019. URL `https://github.com/qixuxiang/orb-slam2_with_semantic_label`.

[154] Kun Qian, Wei Zhao, Kai Li, Xudong Ma, and Hai Yu. Visual slam with boplw pairs using egocentric stereo camera for wearable-assisted substation inspection. *IEEE Sensors Journal*, doi: 10.1109/JSEN.2019.2947275, 2019.

[155] James M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. *Proceedings of the IEEE International Conference on Computer Vision*, 2:941–947, 1999. doi: 10.1109/iccv.1999.790349.

[156] D. Galvez-López and J. D. Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2012.2197158.

[157] Rainer Kuemmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o - general graph optimization, 2011. URL `https://github.com/RainerKuemmerle/g2o`.

[158] Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154, 2004.

[159] Michael Grupp. evo: Python package for the evaluation of odometry and slam, 2017. URL `https://github.com/MichaelGrupp/evo`.

[160] Zutao Jiang, Jihua Zhu, Congcong Jin, Siyu Xu, Yiqiong Zhou, and Shanmin Pang. Simultaneously merging multi-robot grid maps at different resolutions. *Multimedia Tools and Applications*, 79(21-22):14553–14572, 2020. ISSN 1380-7501, 1573-7721. doi: 10.1007/s11042-018-7109-8.

[161] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Zhen Liu, Henrik I Christensen, and Frank Dellaert. Multi robot object-based SLAM. In *Proceedings of the 2016 International Symposium on Experimental Robotics*, pages 729–741. Springer, 2017.

[162] Xieyuanli Chen, Huimin Lu, Junhao Xiao, and Hui Zhang. Distributed monocular multi-robot SLAM. In *Proceedings of the 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 73–78. IEEE, 2018.

[163] Haoyao Chen, Hailin Huang, Ye Qin, Yanjie Li, and Yunhui Liu. Vision and laser fused slam in indoor environments with multi-robot system. *Assembly Automation*, 39(2):297–307, 2019.

[164] Youngseok Jang, Changsuk Oh, Yunwoo Lee, and H Jin Kim. Multirobot collaborative monocular SLAM utilizing rendezvous. *IEEE Transactions on Robotics*, 37(5):1469–1486, 2021.

[165] Andersone. Heterogeneous Map Merging: State of the Art. *Robotics*, 8(3):74, 2019. ISSN 2218-6581. doi: 10.3390/robotics8030074.

[166] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009. ISSN 09218890. doi: 10.1016/j.robot.2009.06.010.

[167] F. Amigoni, S. Gasparini, and M. Gini. Building Segment-Based Maps Without Pose Information. *Proceedings of the IEEE*, 94(7):1340–1359, 2006. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2006.876925.

[168] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[169] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023.