



**Navigating the Underground: Assessing Vision-Based SLAM Methods in Simulated
Subterranean Scenarios**

Autonomous Exploration, Dataset recording and Evaluation

By:

Daniël Johannes Steenkamp

Supervisor:

Prof. Turgay Celik (University of the Witwatersrand)

A dissertation submitted in fulfillment of the requirements for the degree of Masters by
research

in the

School of Electrical and Information Engineering
University of the Witwatersrand

Abstract

This dissertation explores the viability of vision-based localization methods in subterranean environments, employing a variety of feature extraction techniques including traditional methods and advanced deep learning approaches. A unique dataset was generated using an autonomous exploration UAV within a simulated subterranean environment. This dataset served as the testing ground for evaluating various feature extraction methods. The ORB-SLAM3 was modified to integrate these methods, adapting its feature extraction module to accommodate alternative approaches while retaining its core pose optimization and backend components. The study includes detailed experiments and analyses of different sensor configurations and feature extraction methods, providing insights into their applicability and performance in subterranean settings.

Keywords: SLAM, computer vision, visual features, visual feature descriptors, localization, subterranean environments, UAV, MAV, deep learning, monocular SLAM, RGB-D SLAM, GPS-denied scenarios, UAV exploration.

Acknowledgments

I would like to

- Thank my supervisor Prof. Turgay Celik, for the patience, guidance and motivation in the research process and preparation of this document.
- Thank my parents for giving me financial and emotional support in pursuing my masters degree.
- Acknowledge ChatGPT[94] for language editing and grammatical checking.

Acronyms

SLAM Simultaneous Localization and Mapping	1
UAV Unmanned Aerial Vehicle	1
KF Kalman Filter	1
VO Visual Odometry	1
BA Bundle Adjustment	1
DLNN Deep Learning Neural Network	2
DL Deep learning	3
GPS Global Positioning System	3
MAV Micro Air Vehicle	3
DARPA Defense Advanced Research Projects Agency	4
LiDAR Light Detection and Ranging	3
RF Radio Frequency	3
DBoW Database Bag of Words	6

BoW Bag of Words	33
IMU Inertial Measurement Unit	5
KISS Keep it Small and Simple	5
ICP Iterative Closest Point	5
IR Infrared Radiation	5
Radar Radio Detection and Ranging	5
ToF Time-of-Flight	10
FPV First-person view	10
DOF Degrees of Freedom	11
DSO Direct Sparse Odometry	12
LDSO Direct Sparse Odometry with Loop Closure	4
DSM Direct Sparse Mapping	13
SVO Semi-direct Visual Odometry	13
PBA Photometric Bundle Adjustment	13
LSD Large-Scale Direct	12
SIFT Scale-Invariant Feature Transform	4
SURF Speeded-Up Robust Feature	5

ORB Oriented FAST and Rotated BRIEF	12
BRIEF Binary Robust Independent Elementary Features	14
BRISK Binary Robust Invariant Scalable Keypoints	5
AGAST Adaptive and Generic Accelerated Segment Test	14
FREAK Fast Retina Keypoint	15
LATCH Learned Arrangements of Three Patch Code	15
CNN Convolutional Neural Network	15
FCNN Fully Convolutional Neural Network	16
AKAZE Accelerated KAZE	14
DELFI Deep Local Feature	16
CONTEXTDESC Local Descriptor Augmentation with Cross-Modality Context	16
R2D2 Repeatable and Reliable Detector and Descriptor	5
GEODESC Learning Local Descriptors by Integrating Geometry Constraints .	17
SfM Structure-from-Motion	18
RANSAC Random Sample Consensus	35
DoG Difference of Gaussians	43
DoH Determinant of the Hessian	44

HOG Histogram of Oriented Gradients	19
RNN Recurrent Neural Network	20

Contents

Abstract	i
Acknowledgments	iii
Acronyms	v
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Background and Motivation	2
1.1.1 Research Objectives	3
1.1.2 Method	4
1.1.3 Deliverables	5
1.2 Dissertation Outline	5
2 Literature Review	9
2.1 Research Topic	9
2.2 Related Work	9
2.2.1 SLAM Surveys	9
2.2.2 SLAM Datasets	10
2.2.3 Brief Discussion of SLAM Methods	12
2.2.4 Visual Features	13
2.2.5 Deep Learning Visual Feature Extraction and Description Techniques	15
2.2.6 Deep Learning Visual Feature Description-only Techniques	17
2.2.7 Visual Loop Closure Detection Methods	19
2.3 SLAM Methods	20
2.3.1 ORB-SLAM	21
2.3.2 LDSO	22
2.4 Summary	22

3	Methodology	25
3.1	Autonomous Exploration and Data Collection	26
3.1.1	Depth-based Path Detection	26
3.1.2	Transforming Local Translation to Global Translations	28
3.1.3	Positional Mapping and Navigation	29
3.1.4	Collision Avoidance	31
3.1.5	Exploration Strategy	32
3.2	DBoW Place Recognition	33
3.2.1	Descriptor Vocabulary	33
3.2.2	Image Database Construction	33
3.2.3	Loop Closure Detection	34
3.3	SLAM Techniques in Subterranean Environments	35
3.3.1	Feature Extraction Comparison	35
3.3.2	Loop Closure Challenges	35
3.3.3	Graph-Based SLAM Techniques	36
3.3.4	ORB-SLAM	37
3.3.5	Direct Sparse Odometry with Loop Closure	39
3.4	Feature Extraction Methods	41
3.4.1	Python C++ Network Link	41
3.4.2	Scale Invariant Feature Extraction	42
3.4.3	SIFT	43
3.4.4	SURF	44
3.4.5	ORB	46
3.4.6	BRISK	47
3.4.7	Superpoint	47
3.4.8	Zippypoint	48
3.4.9	R2D2	49
3.4.10	D2NET	50
3.5	Keep it Small and Simple Iterative Closest Point Lidar Odometry	50
3.5.1	Motion Prediction and Scan Deskewing	50
3.5.2	Point Cloud Subsampling	51
3.5.3	Local Map and Correspondence Estimation	51
3.5.4	Alignment Through Robust Optimization	51
3.6	Evaluation	51
3.6.1	Umeyama Alignment	52
3.6.2	Evaluation Metrics	53
3.6.3	Evaluation and Visualization Library	53
3.7	ORB-SLAM3 and HF-Net SLAM Modifications	54
3.8	Summary	55

4	Results and Interpretation	57
4.1	Data Trials Setup	57
4.1.1	Data Trials Overview	57
4.1.2	Trial Setup	57
4.1.3	Evaluation Metrics	58
4.2	Computing Platform Specifications	58
4.3	EURoC Results on Modified Versions	59
4.4	Recorded Datasets Map Size	60
4.5	KISS ICP Results on Recorded Datasets	61
4.6	Localization Accuracy's and Processing Times on Recorded Datasets	61
4.6.1	Dataset 1	61
4.6.2	Dataset 2	70
4.6.3	Dataset 3	78
4.6.4	Dataset 4	86
4.6.5	Dataset 5	94
4.6.6	Dataset 6	102
4.6.7	Dataset 7	109
4.7	Visual Feature Comparison	113
4.8	Reference Localization Accuracy	113
4.9	Discussion and Analysis	114
4.10	Summary	115
5	Conclusion	117
5.1	Future Work	118
	References	121
A	Data Format	131
A.1	RGB Images	131
A.2	Depth	131
A.3	LiDAR	131
A.4	IMU	132
A.5	Magnetometer	132
A.6	Pose	132
A.7	Camera Calibration	132
A.8	IMU Noise	133

List of Figures

3.1	Depth-based Path Detection	27
3.2	3D Representation of Local Translation	28
3.3	Illustration of Positional Mapping	30
3.4	Collision Avoidance Illustration	32
3.5	DBoW Vocabulary Tree Illustration	33
3.6	Graph-Based SLAM Illustration	36
3.7	LDSO Graph Construction Illustration	40
3.8	Python-C++ communication	41
3.9	Scale-space pyramid	42
3.10	SIFT feature extraction illustration	43
3.12	Haar Wavelet Filters	45
3.11	DoH Implementation Using Box Filter Illustration.	45
3.13	Fast Corner Detection	46
3.14	BRISK Sampling Pattern	47
3.15	Superpoint DLNN Architecture	48
3.16	Zippypoint DLNN Architecture	49
3.17	R2D2 DLNN Architecture	49
3.18	Modifications in the ORB-SLAM3 Framework	54
3.19	Modifications in the HF-Net-SLAM Framework	55
4.1	Dataset 1 Key Locations	62
4.2	Monocular cave-circuit-01 Trajectory Plot	64
4.3	Monocular cave-circuit-01 Timing Box Plot	65
4.4	RGB-D cave-circuit-01 Trajectory Plot	67
4.5	RGB-D cave-circuit-01 Timing Box Plot	68
4.6	Dataset 2 Key Locations	70
4.7	Monocular cave-circuit-02 Trajectory Plot	72
4.8	Monocular cave-circuit-02 Timing Box Plot	73
4.9	RGB-D cave-circuit-02 Trajectory Plot	75
4.10	RGB-D cave-circuit-02 Timing Box Plot	76
4.11	Dataset 3 Key Locations	78
4.12	Monocular cave-circuit-03 Trajectory Plot	80
4.13	Monocular cave-circuit-03 Timing Box Plot	81

4.14 RGB-D cave-circuit-03 Trajectory Plot	83
4.15 RGB-D cave-circuit-03 Timing Box Plot	84
4.16 Dataset 4 Key Locations	86
4.17 Monocular cave-circuit-04 Trajectory Plot	88
4.18 Monocular cave-circuit-04 Timing Box Plot	89
4.19 RGB-D cave-circuit-04 Trajectory Plot	91
4.20 RGB-D cave-circuit-04 Timing Box Plot	92
4.21 Dataset 5 Key Locations	94
4.22 Monocular cave-circuit-05 Trajectory Plot	96
4.23 Monocular cave-circuit-05 Timing Box Plot	97
4.24 RGB-D cave-circuit-05 Trajectory Plot	99
4.25 RGB-D cave-circuit-05 Timing Box Plot	100
4.26 Dataset 6 Key Locations	102
4.27 Monocular cave-circuit-07 Trajectory Plot	104
4.28 Monocular cave-circuit-07 Timing Box Plot	105
4.29 RGB-D cave-circuit-07 Trajectory Plot	107
4.30 RGB-D cave-circuit-07 Timing Box Plot	108
4.31 Dataset 7 Key Locations	109
4.32 Monocular cave-circuit-08 Trajectory Plot	111
4.33 Monocular cave-circuit-08 Timing Box Plot	112

List of Tables

3.1	Inverse Index Word Table	34
3.2	Direct Index Word Table	34
3.3	Python-C++ Link Image Serialization	42
3.4	Python-C++ Link Keypoints and Descriptors Serialization	42
3.5	Python-C++ Link Keypoint Serialization	42
3.6	Evaluation Metrics Descriptions and Equations	53
4.1	Comparison of Processing Platforms	58
4.2	Monocular EUroC MH-01-easy Results	59
4.3	Dataset Map Sizes	60
4.4	KISS-ICP[95] Lidar odometry Results	61
4.5	Monocular cave-circuit-01 Results	63
4.6	Monocular cave-circuit-01 Timing Results	64
4.7	RGB-D cave-circuit-01 Results	66
4.8	RGB-D cave-circuit-01 Timing Results	67
4.9	Monocular cave-circuit-02 Results	71
4.10	Monocular cave-circuit-02 Timing Results	73
4.11	RGB-D cave-circuit-02 Results	74
4.12	RGB-D cave-circuit-02 Timing Results	76
4.13	Monocular cave-circuit-03 Results	79
4.14	Monocular cave-circuit-03 Timing Results	81
4.15	RGB-D cave-circuit-03 Results	82
4.16	RGB-D cave-circuit-03 Timing Results	84
4.17	Monocular cave-circuit-04 Results	87
4.18	Monocular cave-circuit-04 Timing Results	89
4.19	RGB-D cave-circuit-04 Results	90
4.20	RGB-D cave-circuit-04 Timing Results	92
4.21	Monocular cave-circuit-05 Results	95
4.22	Monocular cave-circuit-05 Timing Results	97
4.23	RGB-D cave-circuit-05 Results	98
4.24	RGB-D cave-circuit-05 Timing Results	100
4.25	Monocular cave-circuit-07 Results	103
4.26	Monocular cave-circuit-07 Timing Results	104

4.27 RGB-D cave-circuit-07 Results	106
4.28 RGB-D cave-circuit-07 Timing Results	107
4.29 Monocular cave-circuit-08 Results	110
4.30 Monocular cave-circuit-08 Timing Results	112
4.31 Best Performing Methods	113
4.32 Best Performing RMSE Errors of Different Sensor Configurations.	114
A.1 Camera Calibration Parameters	132
A.2 IMU Noise Added by Simulation	134

Chapter 1

Introduction

Simultaneous Localization and Mapping (**SLAM**) was initially proposed in 1986 [2], [5] as a method for constructing a map containing various objects while estimating the confidence and relationships between object positions in the map. The term **SLAM** was first coined to describe this group of algorithmic methods [7]. The **SLAM** problem involves mapping an unknown environment with observations taken at unknown locations within it. The **SLAM** algorithm utilizes the mapped environment to localize current sensor observations, finding applications in scenarios involving robotics and Unmanned Aerial Vehicle (**UAV**)s to facilitate autonomous exploration.

One of the earliest solutions to the **SLAM** problem, known as probabilistic **SLAM**, was proposed in [3], [4]. This method employs a probability distribution function describing the joint posterior density of object and observer positions. Statistical filtering methods, such as Kalman Filter (**KF**) variants and Rao-Blackwellized particle filters, are proposed to solve the **SLAM** problem in [9], [16], [17]. In these methods, the observed objects and the observer's position are included in the filter state. The confidence of these positions is also estimated, and the confidence in object positions improves with an increase in observations.

A well-established approach to addressing the **SLAM** problem involves using a graph to represent past positions and observations, known as graph-based **SLAM** methods [32], [40], [51], [57], [65], [88]. In these methods, a graph is constructed with various poses, each comprising an expected position, orientation, and associated uncertainty. Poses in the graph are interconnected based on known movements between them, such as vehicle kinematics or Visual Odometry (**VO**), among other factors. Objects observed in multiple poses may also contribute to linking poses in the graph. Bundle Adjustment (**BA**) [8] is then employed to refine the positions of all interconnected poses in the graph.

Visual graph-based **SLAM** methods typically use feature extractors and descriptors to associate features with poses. Features, identifiable and unique in the environment, are described by feature descriptors that characterize the image area surrounding a feature

CHAPTER 1. INTRODUCTION

point. These descriptors are compared to identify matching features in multiple frames. Various feature extraction and descriptors are available for diverse applications [12], [20], [23], [26], [27], [29]. A recent advancement in feature extraction and description techniques involves the integration of Deep Learning Neural Network (DLNN) [59], [64], [71], [73]–[75], [93].

Feature-less graph-based SLAM methods, referred to as direct SLAM methods [32], [51], [65], rely on estimated texture depth maps at specific keyframes. Unlike feature-based methods, direct methods leverage all information from an image. Consecutive frames are connected in the graph by aligning the estimated texture depth maps.

DLNNs have been incorporated into various aspects of visual feature detection, including depth estimation, VO estimation, complete environment learning, and loop closure detection. For instance, DLNNs have been successfully applied to both depth and VO estimation, as demonstrated in [63], [66]. In cases where DLNNs focus solely on VO estimation, involving the calculation of movement between two frames, examples can be found in [61], [81], [87]. Furthermore, DLNNs have been utilized for learning entire environments, a process well-illustrated in research such as [50]. Another critical application of DLNNs in SLAM is loop closure detection, which is essential for optimizing the constructed pose graph. In studies like [80], [83], [89], loop closure involves matching visual features between different poses, effectively creating a loop in the graph and thereby improving the accuracy of the poses within that loop. Additionally, DLNNs contribute to the segmentation of frames, aiding in the mapping of environments by distinguishing between dynamic and static objects, as seen in works such as [48], [56], [58], [61], [70], [84].

The vast number of proposed SLAM methods encompasses various techniques in mapping, object detection, and graph optimization, among other algorithm-specific methods. Each of these techniques can incrementally or decrementally impact factors such as processing intensity, accuracy, and robustness of the system. Typically, datasets are employed to evaluate the performance of SLAM methods. These datasets comprise sequences of sensor measurements recorded in environments where the sensor positions' ground truths are known. The construction of these datasets employs a variety of platforms, including UAVs [43], [72], mobile robots [44], [54], and vehicles [55], [78]. Additionally, simulation environments [11], [60] are frequently used to generate these sequences. Simulated datasets [86] serve as another approach to evaluating SLAM methods. It is crucial to assess SLAM methods using datasets or simulation environments that closely resemble the intended application environment to ensure performance that mirrors real-world scenarios.

1.1 Background and Motivation

UAVs have become integral to our everyday lives, finding applications in security, infrastructure inspection, aerial photography, agriculture, surveying and mapping, and

1.1. BACKGROUND AND MOTIVATION

search and rescue operations, among many other uses. The recent introduction of **UAVs** in subterranean environments adds additional challenges to path planning and mapping systems.

In subterranean environments, the use of Global Positioning System (**GPS**) sensors for the global localization of **UAV** positions is restricted due to the Radio Frequency (**RF**) shielding nature of subterranean environments. Similarly, constant communication is not possible without proper infrastructure already installed in the environment. Additionally, subterranean environments lack ambient light, requiring a light source for any visual sensor to be effective. The absence of distinctive visual features in subterranean environments may influence the performance of feature-based visual **SLAM** methods.

Light Detection and Ranging (**LiDAR**)-inertial-based methods are generally preferred in subterranean environments due to not requiring light and providing scale-accurate mapping and localization compared to monocular visual **SLAM** methods. Using rotating **LiDAR** sensors imposes additional mechanical challenges in the form of constant angular momentum on the **UAV**. Solid-state **LiDARs** can also be used but restrict the sensing area compared to all-round **LiDAR** sensors. Introducing visual-based **SLAM** methods in subterranean environments has the potential to improve localization and mapping accuracy and robustness.

1.1.1 Research Objectives

Problem Statement

This study seeks to assess the usability of various visual features, encompassing both conventional and Deep learning (**DL**) visual features, within monocular visual **SLAM** methods. The primary focus is on determining their impact on relative localization accuracy when deployed on devices with limited processing capabilities. Relative accuracy, in this context, pertains to the effectiveness of cutting-edge monocular visual **SLAM** techniques. Additionally, the study aims to pinpoint the optimal monocular visual **SLAM** method for deployment in challenging environments characterized by low processing power, lack of **GPS** signals, and sparse visual features—such as underground mining tunnels.

Problem Formulation

SLAM research typically focuses on environments with well-defined and distinctive features. These include datasets tailored for autonomous driving studies involving vehicles, Micro Air Vehicle (**MAV**) datasets for **UAV** navigation and collision avoidance investigations, and datasets for mobile robots navigating dynamic environments. However, there is a notable gap in **SLAM** research concerning performance evaluation in feature-scarce environments with low lighting conditions, where utilizing the **GPS** becomes impractical. Moreover, processing constraints on **MAVs** pose additional challenges.

CHAPTER 1. INTRODUCTION

This study addresses the necessity for UAVs to autonomously inspect mining tunnels, confronting issues such as the absence of GPS signals, processing limitations, and the need for onboard processing. Effective SLAM methods are essential for automating tunnel inspections, as they integrate data from various sensors to create maps and estimate the relative positions of observations. The SLAM method intended for mining tunnels must address three critical environmental factors: the scarcity of distinctive and identifiable features, the need for long-term mapping with constrained memory, and the instability in UAV positioning and movement.

Research Objectives

The objectives of this study are to compare multiple state-of-the-art monocular visual SLAM methods capable of real-time operation on a processing-constrained device in a GPS-denied and feature-lacking environment. The study investigates the ability of a SLAM method to maintain localization accuracy reliably in this challenging environment. Additionally, the influence of DL methods in monocular visual SLAM, including deep feature extraction and DL visual odometry methods, is explored.

Research Questions

1. **Localization Accuracy:** Which state-of-the-art monocular visual SLAM methods demonstrate effective UAV position localization with sufficient accuracy in environments lacking distinctive features and deprived of GPS signals?
2. **Processing Speed:** Among the leading monocular visual SLAM methods, which ones, with an average processing time of less than 100ms per frame on a constrained device like the Jetson Nano, are suitable for deployment on UAVs?
3. **DL Impact:** Do DL visual odometry monocular visual SLAM methods, requiring less than 100ms per frame on a processing-constrained device like the Jetson Nano, exhibit notable robustness and consistency compared to state-of-the-art monocular visual SLAM methods when implemented in feature-lacking and GPS-denied environments?

1.1.2 Method

The methodology for addressing the research questions involves dataset generation utilizing the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge simulation environment. Recorded datasets are then employed to evaluate various SLAM methods, utilizing both monocular and RGB-D sensor configurations. These methods include standalone SLAM methods such as ORB-SLAM3 [88] and Direct Sparse Odometry with Loop Closure (LDSO) [65]. Additionally, modified versions of ORB-SLAM3 and HF-Net-SLAM (a modified version of ORB-SLAM3) are introduced to incorporate alternative feature extraction methods. The added feature extraction and description methods for ORB-SLAM include Scale-Invariant Feature Transform

1.2. DISSERTATION OUTLINE

(**SIFT**)[12], Speeded-Up Robust Feature (**SURF**)[20], and Binary Robust Invariant Scalable Keypoints (**BRISK**) [26]. Furthermore, **DL** feature extraction and description methods, namely SuperPoint[64], ZippyPoint[93], Repeatable and Reliable Detector and Descriptor (**R2D2**) [75], and D2Net[73], are introduced to assess any potential improvements **DL** may offer over conventional feature extraction and description methods. As a baseline reference for accuracy, Keep it Small and Simple (**KISS**) Iterative Closest Point (**ICP**) **LiDAR** odometry is employed.

1.1.3 Deliverables

The deliverables include the recording of seven datasets [98] using the **DARPA** Subterranean Challenge simulation environments, employing the file format specified in Section A. The sensor values obtained in these datasets comprise three RGB-D cameras, **LiDAR**, and an Inertial Measurement Unit (**IMU**), using an autonomous exploration algorithm [106]. Additionally, modified versions of ORB-SLAM3 and HFNet-SLAM [107] are provided, wherein the feature extraction and description methods are replaced by various alternatives. This approach facilitates the comparison of feature usefulness and descriptor matching. Localization accuracy is then compared between the modified versions of ORB-SLAM3 and HFNet-SLAM using the recorded subterranean datasets.

The ORB-SLAM and HF-Net-SLAM frameworks were selected over other methods due to their superior performance, even when compared to intensive **DL** methods. DeepSLAM[81] conducted a comparison of localization accuracy among various **DL**-based methods, as well as other conventional monocular and stereo configurations. The Root Mean Square Error (RMSE) of ORB-SLAM is 0.67, significantly lower than other methods which range between 6.72 to 15.82. While this performance advantage is also noted in [88], it may not be to the same extent. The combination of superior or comparable performance, along with the lightweight nature of the **SLAM** background application, makes ORB-SLAM and its modified version HF-Net-SLAM the obvious choice.

1.2 Dissertation Outline

The remainder of the dissertation is organised as follows. Chapter 2 outlines various aspects of **SLAM** research and related technologies in the context of subterranean exploration. It addresses the limitations and challenges of traditional sensors like **LiDAR**, Radio Detection and Ranging (**Radar**), and Infrared Radiation (**IR**) cameras, and discusses the potential of visual-based methods for mapping and localization in such environments. The text highlights the drawbacks of monocular and RGB-D sensor configurations, emphasizing their limitations in different scenarios. To overcome these challenges, the section suggests utilizing the **DARPA** Subterranean challenge simulation environment for more accurate ground truth pose estimates and inertial measurements. The section also details the modification of a well-known visual feature-based **SLAM** method for various sensor configurations and examines the effectiveness of different visual feature

CHAPTER 1. INTRODUCTION

extractors and descriptors in subterranean environments. This exploration covers surveys of unmanned subterranean exploration and autonomous robotic deployments in confined spaces, along with a review of visual **SLAM** algorithms and **SLAM** datasets from various sources such as UZH-FPV Drone Racing, EuRoC MAV, NCLT Vision and **LiDAR**, and more. The section concludes by analyzing different **SLAM** methods like ORB-SLAM and **LDSO** and their specific features, tracking techniques, and loop closure detection methods.

Chapter 3 delves into autonomous exploration and data collection in the **DARPA** Subterranean Challenge simulation environment, highlighting the deployment of an explorer **UAV** armed with RGB-D cameras and a **LiDAR** system. It explains crucial adjustments for enhanced efficiency, including the omission of battery management and the boost in pose update frequency. The chapter presents an autonomous exploration algorithm, detailing depth-based path detection and the conversion of local movements into global translations, alongside strategies for positional mapping, navigation, path optimization, and collision avoidance. It underscores the exploration strategy's focus on prioritizing unexplored paths and significant depth map areas. The data collection methodology is elaborated, specifying the storage of visual, depth, **LiDAR**, **IMU**, and magnetometer data, as well as ground truth pose and camera calibration information. Furthermore, it discusses the application of Database Bag of Words (**DBoW**) [31] for place recognition, including the development of descriptor vocabularies and image databases for loop closure detection. The chapter concludes with an exploration of **SLAM** techniques apt for subterranean settings, comparing various feature extraction methods, addressing loop closure detection challenges, reviewing different deep learning architectures for feature extraction and description, and evaluating **SLAM** methods with the EVO [53] library. The EVO library being an open source **SLAM** evaluation library.

Chapter 4 assesses the viability of visual-based mapping and localization in subterranean environments with different datasets. It outlines the computing platforms used, detailing their specifications, and compares the performance of modified visual **SLAM** methods using the EuRoC[43] dataset. The chapter examines map sizes from datasets recorded in **DARPA** Subterranean Challenge [85] simulation environments and analyzes localization accuracy and processing times across various sensor configurations and **SLAM** platforms. Each dataset, characterized by its unique environmental challenges and outcomes, offers insights into the effectiveness of **SLAM** methods in diverse subterranean contexts.

Finally, the dissertation concludes with detailed discussions in Chapter 5, on the evaluation of visual **SLAM** methods in subterranean environments, focusing on various feature extraction and description techniques. It highlights the challenges and comparative performance of monocular visual methods, RGB-D sensor configurations, and **LiDAR**-based methods. The integration of different feature extraction methods into the ORB-SLAM3 framework showed improved accuracy, with deep learning methods like Zippypoint

1.2. DISSERTATION OUTLINE

and Superpoint performing well. The report suggests a fusion of **LiDAR** and visual sensors, potentially enhanced by **IMU** sensors, to address diverse environmental conditions. However, challenges in assessing loop closure detection effectiveness and obtaining accurate ground truth pose data in expansive subterranean environments are noted. Future work involves using the recorded dataset for testing and developing **SLAM** methods that combine visual-based techniques with **LiDAR** and **IMU** sensors, with an emphasis on validating accuracy with real-world data.

Chapter 2

Literature Review

2.1 Research Topic

Subterranean exploration primarily relies on **LiDAR**, **Radar**, and **IR** cameras, yet these sensors exhibit several drawbacks. **LiDAR** sensors, for instance, are known for their weight and cost, and rotation-based **LiDAR** can impede **UAV** maneuverability due to the angular momentum of its rotating components. Similarly, **Radar** sensors share the issues of being heavy, expensive, and having high power requirements. Additionally, **IR** cameras, while providing accurate odometry predictions, face challenges due to variations in **IR** concerning the time of day, season, and equipment usage.

The adoption of visual-based methods is not without its limitations. In monocular sensor configurations, the mapping lacks scale accuracy in the environment, necessitating additional measurements for scale accuracy. RGB-D sensor configurations provide scale-accurate mapping, but optimal accuracy requires features within the sensor's depth range, posing challenges in large open spaces common in subterranean environments. Furthermore, all visual methods require sufficient light, contributing to increased weight and power requirements as light sources need to be mounted on the **UAV**.

2.2 Related Work

2.2.1 SLAM Surveys

SLAM surveys that explore the application of **UAVs** in subterranean environments are critical for understanding both the challenges and solutions specific to these contexts. This discussion aims to provide an overview of the sensor configurations mentioned in current literature and evaluate the localization results to determine their accuracy and applicability in complex underground settings.

CHAPTER 2. LITERATURE REVIEW

Survey of Unmanned Subterranean Exploration, Navigation, and Localization[82]

This comprehensive survey delves into the intricacies of navigating and mapping subterranean environments, such as mines and tunnels, with an emphasis on overcoming the limitations posed by the absence of **GPS** signals. The survey categorizes its findings into three primary aspects: modes of locomotion, strategies for **GPS**-denied navigation, and techniques for effective localization and communication. It evaluates various locomotion strategies, including wheeled, aerial, and aquatic **UAVs**, and examines a range of localization techniques, from beacon-based and Time-of-Flight (**ToF**) methods to landmark-based and visual-based approaches, highlighting their respective advantages and limitations in subterranean exploration [91].

A Survey on the Autonomous Exploration of Confined Subterranean Spaces: Perspectives from Real-world and Industrial Robotic Deployments[92]

Focusing on the unique challenges posed by confined and subterranean spaces found in civilian and industrial contexts, this survey investigates the hazards such as toxic gases, extreme temperatures, narrow passageways, dangerous oxygen levels, and risks of flooding and collapse. It presents an analysis of state-of-the-art robotic exploration techniques tailored for these environments, discussing both single and multi-robot systems. The survey emphasizes the importance of autonomous exploration capabilities in ensuring safety and efficiency during the assessment and mapping of hazardous underground areas.

2.2.2 SLAM Datasets

SLAM datasets play a pivotal role in the development and evaluation of **SLAM** algorithms. These datasets comprise a comprehensive collection of sensor data, which may include but is not limited to visual imagery, encoder readings, **IMU** data, and **LiDAR** scans. Each dataset meticulously records the known relative positions of the sensor measurements at specific instances throughout the acquisition period.

The intrinsic value of **SLAM** datasets lies in their utility for assessing the localization accuracy of various **SLAM** methodologies. These methodologies leverage the sensor observations to predict the relative pose at which each observation was captured. By providing a ground truth comparison, **SLAM** datasets enable researchers to quantitatively evaluate the performance of **SLAM** algorithms, focusing on their ability to accurately deduce the sensor's position and orientation within an environment. This evaluation is critical for advancing **SLAM** technology, as it directly impacts the reliability and effectiveness of autonomous navigation and mapping systems in complex and dynamic settings.

UZH-FPV Drone Racing

The UZH-FPV drone racing dataset [72] captures data from a First-person view (**FPV**) drone, presenting itself as the most dynamic visual-inertial dataset available. Its primary

2.2. RELATED WORK

objective is to assess the capability of current autonomous navigation algorithms to meet the speed and precision demands inherent in **FPV** drone racing. Comprising 16 indoor and ten outdoor sequences, 16 of the 26 sequences are accompanied by a 6-Degrees of Freedom (**DOF**) ground truth, providing both position and orientation information. Sequences are acquired with the camera facing either forward or at a 45° downward angle. The **FPV** drone utilized for this dataset is equipped solely with a camera and an **IMU**.

EuRoC MAV

The EuRoC MAV [43] dataset is gathered using a six-rotor drone, featuring 11 indoor sequences categorized by difficulty. Equipped with stereo cameras and an **IMU**, this drone leverages a 6-**DOF** Vicon motion capture system for accurate position and orientation ground truth determination. Additionally, a 3-**DOF** Leica MS50 laser tracker is employed exclusively for position data. The combined readings from these systems ensure a precise and comprehensive ground truth.

NCLT Vision and LiDAR

The NCLT vision and **LiDAR** [44] dataset is a prolonged observational dataset obtained using a Segway robot at the University of Michigan North Campus. Comprising 27 sequences spanning 15 months, the dataset aims to facilitate research on long-term autonomous operations in evolving environments. Sensor data captured by the Segway robot includes omnidirectional cameras, three-dimensional (3D) **LiDAR**, planar **LiDAR**, **GPS**, and **IMU** data.

Chilean Underground Mine

The Chilean underground mine [54] dataset captures data within the world’s largest copper mine, serving as a resource for **SLAM** research in challenging environmental conditions. The robot responsible for data collection is equipped with 3D **LiDAR**, 2D **Radar**, and stereo camera sensors.

FinnForest

The FinnForest [78] dataset, acquired over the long term, involves mounting sensors on a vehicle exploring a remote area. Comprising 12 sequences, seven in summer and five in winter, this dataset targets mobile robotics, autonomous driving research, and forestry operations. Data captured includes four cameras, an **IMU**, and **GPS** sensors.

Oxford Robocar

The Oxford Robocar [55] dataset, gathered over an extended period, features sensors mounted on a vehicle exploring an inner city area. With 133 sequences captured at various days and times over 16 months, 115 sequences cover the same road, while 18

CHAPTER 2. LITERATURE REVIEW

sequences explore an alternative route. Aimed at autonomous driving systems, the vehicle is equipped with six cameras, 2D and 3D **LiDAR**, as well as **GPS** and **IMU** sensors.

2.2.3 Brief Discussion of SLAM Methods

SLAM is a fundamental technology that enables robots to determine their location and map their surrounding environment, even in previously unexplored areas. It provides robots with the ability to explore unknown spaces while retaining information about their paths. Among the different approaches to **SLAM**, this study focuses on visual-based **SLAM**, which leverages cameras for navigation, particularly in subterranean environments such as caves or mines. These environments pose significant challenges due to their darkness and the inability of **GPS** signals to penetrate. Moreover, the robots often operate with limited computational resources, complicating the processing of the extensive visual data they collect. This thesis explores how visual-based **SLAM** can facilitate the navigation and mapping of these challenging underground environments, even when powerful computational resources are not available.

ORB-SLAM

ORB-SLAM, initially introduced in 2015 [40], stands out as a lightweight, feature-based, real-time **SLAM** method utilizing Oriented FAST and Rotated BRIEF (**ORB**) features and descriptors [27]. Its functionality is distributed across three concurrently running modules: tracking, local mapping, and loop closure. The tracking module handles the introduction of newly acquired frames, feature extraction, and the estimation of the initial pose for these frames. The local mapping module manages keyframe operations, introduces new map points, and conducts local bundle adjustment. The loop closure module utilizes **DBoW2** [31] to discretize frame descriptors, providing a computationally efficient method for determining descriptor overlap between frames.

The second iteration of ORB-SLAM2, introduced in 2017 [57], extended the capabilities by incorporating stereo and RGB-D cameras into the existing framework. Furthermore, global bundle adjustment was introduced to the loop closure module.

The third and latest version of ORB-SLAM3, unveiled in 2020 [88], incorporated inertial measurements into all existing sensor configurations. This iteration introduced multi-map mapping, reducing memory requirements for application in large environments. ORB-SLAM3 also implemented serialization and de-serialization, allowing for the preservation of maps and multi-session **SLAM**.

LDSO

Direct Sparse Odometry (**DSO**) [51], akin to Large-Scale Direct (**LSD**)-**SLAM**, utilizes sparse depth maps. **DSO** incorporates a comprehensive photometric calibration, accounting for exposure time, lens vignetting, and non-linear response functions. The enhancement

2.2. RELATED WORK

of **DSO** through the addition of loop closure detection, for improved graph optimization, is presented as **LDSO** [65].

DSM

Direct Sparse Mapping (**DSM**) represents a visual **SLAM** approach comprising a tracking front end and an optimizing back end. The front end manages frame point tracking, supplying a preliminary estimate for Photometric Bundle Adjustment (**PBA**). It determines which frames qualify as keyframes. Conversely, the back end optimizes the positions of other keyframes and map point locations. Both the front and back end employ separate processing threads, enabling concurrent execution for enhanced processing efficiency. **DSM** maintains a persistent map of active keyframes, relying on covisibility with the latest keyframe. **PBA** optimizes photometric error across active keyframes.

SVO

Semi-direct Visual Odometry (**SVO**) [33] combines aspects of direct and feature-based methods. Similar to sparse mapping techniques, where the depth map is calculated only for parts of the input image, semi-direct methods determine the depth map on pixels surrounding detected features. **SVO** is segregated into a motion estimation and mapping thread. Keyframe generation involves feature extraction and depth map initialization. Similar to sparse direct methods, the depth map is refined with subsequent frames.

LSD-SLAM

LSD-SLAM [32] proposed the use of a sparse depth map, wherein depth is estimated and aligned only on portions of the image. Sparse mapping demands less processing for alignment compared to dense mapping, facilitating loop closure detection by aligning mapped keyframes with the current frame.

2.2.4 Visual Features

Visual Feature Extraction and Description Techniques

Visual feature extraction and description techniques play a pivotal role in computer vision applications. Several prominent methods are discussed below:

- **SIFT** [12]:

SIFT identifies features by iteratively applying a Gaussian smoothing kernel to the image. Local minima and maxima are detected within a $(3 \times 3 \times 3)$ pixel position space, and multiple scale levels are considered to enhance scale invariance. The **SIFT** descriptor calculates image gradients in (4×4) patches, constructing a histogram of gradients that is concatenated to form the final descriptor.

CHAPTER 2. LITERATURE REVIEW

- **RootSIFT** [30]:

RootSIFT retains **SIFT** features and descriptors but introduces a square root (Hellinger) kernel instead of Euclidean distance for matching descriptors. This modification enhances the matching process.

- **SURF** [20]:

SURF adapts **SIFT** by replacing the Gaussian smoothing kernel with box filters containing integer weights, eliminating the need for computationally expensive floating-point operations. The descriptor utilizes an integral image, significantly reducing computational requirements.

- **ORB** [27]:

ORB employs the FAST corner detector for feature extraction and a modified version of Binary Robust Independent Elementary Features (**BRIEF**) as the descriptor. BRIEF compares pixel sets in the patch surrounding the feature, with rotation invariance achieved by rotating the patch according to its principal orientation.

- **BRISK** [26]:

BRISK introduces scale-space keypoint detection by searching for maximas in both image and scale-space. **BRISK** utilizes the FAST score for feature importance and a rotated sampling pattern in the descriptor image patch, enhancing robustness to variations in scale.

- **KAZE** [29]:

KAZE features are detected through a diffusion equation, computing the scale-normalized determinant of Hessian at multiple scale levels. The KAZE descriptor considers magnitude, orientation, and Hessian eigenvalues in the local neighborhood of detected features.

- **AKAZE** [24]:

Accelerated KAZE (**AKAZE**) adapts KAZE by employing anisotropic diffusion filtering for constructing a non-linear scale space. This adaptation enhances robustness to noise, making feature extraction more reliable.

Visual Feature Extraction-only Techniques

This section explores key visual feature extraction techniques crucial for computer vision, focusing on methods like FAST, Adaptive and Generic Accelerated Segment Test (**AGAST**), and STAR for efficient feature detection.

- **FAST** [13] :

FAST corner detection identifies corners by considering a circle of pixels around the pixel in question. If a sufficient number of connected pixels in this circle are brighter or darker than the center pixel, the center pixel is considered a corner.

2.2. RELATED WORK

FAST employs a decision tree to detect corners, requiring fewer comparisons and resulting in faster runtime. The notation FAST n-N is commonly used to describe the specific parameters, with n representing the number of pixels required to be significantly brighter or darker for the center to be detected as a corner, and N depicting the number of pixels in the circle.

- **AGAST** [23]:
AGAST is an adaptation of FAST[13] that employs multiple specialized decision trees depending on the neighborhood of pixels, further improving runtime.
- **STAR** [18]:
Center surround extrema (CenSurE) is a method that introduces scale-invariant center-surround detectors by computing a simplified center-surrounding filter across all scales. The Harris measure is calculated for all local extremes detected in a specific size neighborhood to eliminate features with a low corner response.

Visual Feature Description-only Techniques

This section highlights visual feature description-only techniques, essential for defining unique identifiers of detected features. Techniques like Fast Retina Keypoint (**FREAK**), DAISY, and Learned Arrangements of Three Patch Code (**LATCH**) are discussed for their innovative approaches to creating robust descriptors, vital for matching features across different images.

- **FREAK** [28]:
FREAK uses a retinal sampling pattern to obtain a descriptor. The retinal sampling grid is circular, with higher point density close to the center. The bit descriptor is constructed by comparing image intensities of different sample pairs.
- **DAISY** [22]:
The DAISY descriptor computes a number of orientation maps, one for each quantized direction. DAISY uses image gradients to obtain the descriptor. The orientation maps are convoluted several times by a Gaussian kernel of different standard deviations.
- **LATCH** [45]:
LATCH constructs a binary descriptor similar to the **ORB** descriptor, where the intensities of multiple pairs of pixels are compared to construct the descriptor. **LATCH** proposes repeatedly applying a Gaussian kernel to the described patch. Comparisons are then made between these filtered levels of the patch to obtain an improved descriptor.

2.2.5 Deep Learning Visual Feature Extraction and Description Techniques

This subsection delves into advanced deep learning techniques for visual feature extraction and description, showcasing methods that leverage Convolutional Neural Network (**CNN**)

CHAPTER 2. LITERATURE REVIEW

and self-supervised learning approaches. Techniques such as SUPERPOINT, D2NET, Deep Local Feature (**DELF**), and others illustrate the cutting-edge in generating robust, scale-invariant features and descriptors, critical for improving the accuracy and reliability of visual **SLAM** systems. These methods represent a significant evolution in feature extraction and description, offering enhanced performance in varied and challenging environments.

- **SUPERPOINT** [64]:

SUPERPOINT is a deep learning feature extraction and description method utilizing a **CNN**. The neural network is trained via a self-supervised approach, using a pre-labeled set of images with known feature points for feature extraction network training. The descriptor network is trained using the same encoding network as the feature extraction. By warping pre-labeled images with known feature points, a descriptor loss function is computed based on features detected between the original and warped images, and it is used to train the descriptor network.

- **D2NET** [73]:

D2-Net is a deep learning feature extraction and description method closely related to SUPERPOINT. D2-Net, similar to SUPERPOINT, uses a shared encoding scheme. However, D2-Net shares all parameters between the decoding schemes for detection and description. Additionally, D2-Net incorporates geometric techniques to improve the reliability and robustness of local features.

- **DELF** [59]:

DELF is a feature extraction method developed by Google. **DELF** is trained using a large-scale Google-Landmarks dataset, which includes over one million images from almost 13,000 unique landmarks worldwide. **DELF** constructs image pyramids to achieve scale-invariant features and descriptors.

- **CONTEXTDESC** [74]:

Local Descriptor Augmentation with Cross-Modality Context (**CONTEXTDESC**) separates feature extraction into preparation and augmentation modules. The preparation module computes raw local features, high-level visual features, and keypoint locations. The augmentation module, provided with results from the preparation module, computes feature vectors and keypoints.

- **LFNET** [69]:

LFNET uses separate networks for detecting feature points and location descriptors. The first network, tasked with computing feature point locations, scales, and orientation, is a Fully Convolutional Neural Network (**FCNN**). The second network, tasked with determining the feature descriptors, is only provided with a patch surrounding the detected keypoint location. LFNET achieves scale invariance by resizing the input image multiple times and convoluting it with a filter to obtain a score map for each scale.

2.2. RELATED WORK

- **R2D2** [75]:
R2D2 is a deep-learning feature extraction and description approach using an **FCNN**. The **FCNN**, provided with an image, produces a dense local descriptor map, a repeatability confidence map, and a reliability confidence map. Features are detected by selecting pixels with high repeatability and reliability. The **FCNN** of **R2D2** is trained using a novel unsupervised loss function that encourages repeatability, sparsity, and a uniform distribution across the image.
- **KEYNET** [71]:
KEYNET is a gradient-based feature extraction method. KEYNET uses the first and second-order derivatives to compute salient corner responses. The first- and second-order derivatives are provided to a convolutional layer containing filters, normalization layer, and activation function. KEYNET uses a pyramid scheme to achieve scale-invariant features.
- **ZIPPYPOINT** [93]:
Zippypoint, a feature extraction and description method rooted in the KP2D neural network architecture [76], utilizes a VGG-style encoder-decoder structure. Furthermore, the method incorporates a binary learned descriptor approach, ensuring the normalization of descriptors to maintain a consistent number of ones in each descriptor.

2.2.6 Deep Learning Visual Feature Description-only Techniques

This section introduces advanced deep learning techniques for creating feature descriptors, crucial for matching features between images. Methods like TFEAT, BOOST_DESC, and Learning Local Descriptors by Integrating Geometry Constraints (**GEODESC**) employ innovative approaches, from triplet-based learning to integrating geometric constraints, enhancing the precision and robustness of feature matching. These techniques represent significant advancements in the field, offering sophisticated solutions to the challenges of visual feature description in computer vision applications.

- **TFEAT** [42]:
TFEAT employs a triplet-based learning approach, utilizing three images for training. Two similar images (anchor and positive images) are paired with one dissimilar image (negative image). The training objective ensures that descriptors between similar images are close, while differing from the descriptor of the negative image. The process involves minimizing the descriptor distance between anchor and positive images and maximizing the descriptor distances between anchor and negative image. TFEAT incorporates shadow **CNNs** designed for simplicity and efficiency, emphasizing limited depth.
- **BOOST_DESC** [36]:
BOOST_DESC adopts a boosting approach, a grouped learning method leveraging predictions from simple models to create a robust, discriminative feature descriptor.

CHAPTER 2. LITERATURE REVIEW

Simple image descriptors, initially categorized between intensity-based (e.g., **ORB**, **BRISK**, **FREAK**) and gradient-based methods (e.g., **SIFT**, **SURF**, **KAZE**), are used. Multiple training/boosting iterations are performed, focusing on instances where descriptors were incorrectly classified in each iteration. The descriptors are combined into a strong image descriptor, incorporating their performance during training, including challenging situations. The final **BOOST_DESC** feature description method is obtained.

- **Learning Local Feature Descriptors Using Convex Optimization [35]:**

This paper proposes a method for computing discriminative descriptors from image patches. The computational pipeline involves Gaussian patch smoothing for calculating pixel intensity gradients, spatially aggregated using convolution within descriptor pooling regions. After pooling, dimensionality reduction of the descriptor is achieved through projection onto a lower-dimensional subspace, facilitated by a learned matrix for enhanced descriptor matching. Candidate pooling regions are generated, and an optimal configuration is selected using sparse learning techniques. To ensure invariance to affine intensity transformations, the descriptor undergoes cropping and normalization. The primary goal is to learn a linear projection matrix effectively separating positive and negative descriptor pairs in a lower-dimensional space. Due to the scale of optimization problems, the paper suggests regularized dual averaging as an effective method for handling large training sets. Furthermore, the paper explores the binarization of a low-dimensional real-valued descriptor using a Parseval tight frame, resulting in a binary code with a reduced memory footprint.

- **GEODESC [68]:**

The **GEODESC** method leverages the L2-Net described in 2.2.6 to acquire local feature descriptors. The network’s feature tower is deliberately constructed without pooling layers, favoring the use of strided convolutional layers for in-network downsampling. Following each convolutional layer, batch normalization layers are applied, featuring fixed weighting and bias parameters. The final feature vector is derived through L2-normalization after the last convolutional layer. The process of training data generation involves a comprehensive pipeline, commencing with 2D correspondence generation utilizing sparse reconstructions from a standard Structure-from-Motion (**SfM**) approach. A visibility check, based on 3D Delaunay triangulation, is employed to **SIFT** through outlier-contaminated correspondences. The generation of matching patches includes cropping interest regions using keypoints identified by the **SIFT** detector. Geometric similarity estimation introduces metrics for patch and image similarity based on 3D geometry. The batch construction method capitalizes on geometry constraints from **SfM**, ensuring the effective sampling of useful data for descriptor learning. The loss formulation encompasses structured and geometric loss terms, with the former computed on cosine similarity matrices and the latter encouraging proximity of matching pairs in the measure space. The training process incorporates image sets, **SfM** data, and additional

2.2. RELATED WORK

datasets, implementing data augmentation techniques and specific hyperparameters for network training. Emphasis is placed on the proposed batch construction method for generating more challenging samples, thereby contributing to the enhanced effectiveness of descriptor learning.

- **SOSNET** [77]:

SOSNet, a deep learning feature descriptor, is trained using pairs of matching patches. The **CNN** is applied to these patches to extract the descriptor, with the training objective being the minimization of both first and second-order similarity losses between matching patches and the maximization of these losses for non-matching descriptors. Additionally, a second-order similarity regularization is introduced to incorporate higher-order similarities, thereby enhancing descriptor learning. Both first and second orders are given equal weight in the training process. The architecture of L2Net, as described in Section 2.2.6, is adopted to embed local patches into descriptors, incorporating a dropout layer to prevent overfitting.

- **L2NET** [62]:

L2Net functions as a deep learning method for local patch description, utilizing an all-convolutional architecture that incorporates convolutional downsampling. Addressing the complexity posed by a significant number of potential non-matching patches, a progressive sampling strategy is employed. This strategy aims to strike a balance between matching and non-matching patch pairs by selectively emphasizing the sampling of non-matching pairs. The multifaceted loss function integrates three key objectives: descriptor similarity, compactness, and the supervision of intermediate feature maps.

2.2.7 Visual Loop Closure Detection Methods

Visual loop closure detection methods identify and match previously observed scenes by describing them in a way that allows for recognition. These methods are integral to **SLAM** systems, enabling the detection of revisited locations and refining the map to account for the detected loop closures. Common approaches to scene description include Histogram of Oriented Gradients (**HOG**), visual feature descriptor techniques, and **DL**-based methods.

HOG-based Loop Closure Detection

HOG-based loop closure detection [15], [37] calculates pixel gradients for each pixel in an image. These gradients are then used to construct a histogram representation vector of the current image. By comparing these histogram vectors, the similarity between two scenes can be determined, facilitating loop closure detection.

CHAPTER 2. LITERATURE REVIEW

Visual Feature-based Loop Closure Detection

Visual feature-based loop closure detection [110], [111], [113] represents images as sets of visual features and descriptors. A predefined vocabulary is created by sampling descriptors according to their frequency of occurrence, with less frequent descriptors being sampled more heavily. This vocabulary is then used to create bins, each representing a specific descriptor vector length. The scene descriptor is formed by counting the descriptors attributed to each bin, followed by normalizing across all bins. This process, known as **DBoW**, is explained in greater detail in Section 3.2.

DL-based Loop Closure Detection

DL-based loop closure detection follows a similar approach to that described in Section 2.2.7, but it utilizes deep learning for feature extraction and descriptor generation. Deep neural networks are trained to recognize and encode complex visual patterns into feature vectors, allowing for a more abstract and robust representation of a scene.

Unlike traditional methods, **DL**-based approaches leverage large datasets to learn hierarchical representations. This makes them more adaptable to variations in lighting, viewpoint, and scene composition, thereby improving the accuracy of loop closure detection in complex and dynamic environments.

Other **DL**-based methods include **CNNs**, which extract spatial features from images [49], and Recurrent Neural Network (**RNN**)s, which capture temporal dependencies in image sequences [38]. Some approaches combine CNNs and RNNs to enhance loop closure detection by integrating spatial and temporal features [41].

Additionally, techniques such as AutoEncoders [52] and Siamese Networks [90] have been explored for loop closure detection. These methods use unsupervised or semi-supervised learning to map images into a latent space, where similar scenes are positioned closer together, making it easier to detect loop closures. The versatility and power of **DL** techniques have significantly improved the performance and accuracy of loop closure detection in various **SLAM** systems, especially in challenging environments.

2.3 SLAM Methods

In this section of the dissertation, we delve into a comparative analysis of two prominent **SLAM** methodologies, specifically ORB-SLAM and **LDSO**, with the aim of assessing their applicability and performance in subterranean environments. This comparison is integral to our research as it provides insights into the adaptability and efficiency of feature-based and direct-based **SLAM** techniques within the unique challenges presented by underground settings, such as mining tunnels.

ORB-SLAM, a feature-based approach, relies on the identification and tracking of specific

2.3. SLAM METHODS

visual features within the environment to estimate the camera’s motion and reconstruct a map of the environment. Conversely, **LDSO** adopts a direct-based strategy, utilizing texture map alignment for localization and mapping, potentially offering advantages in environments where visual features are sparse.

Given the characteristic lack of distinct visual markers in subterranean environments, it is hypothesized that direct-based methods like **LDSO** might exhibit superior performance in comparison to feature-based methods such as ORB-SLAM. This hypothesis stems from the assumption that the textural continuity present in such environments could provide a more reliable basis for the direct-based mapping and localization processes.

2.3.1 ORB-SLAM

- **Feature Choices:**

ORB-SLAM utilizes **ORB** features as described in Section 3.4.5, incorporating an additional feature distribution method that encourages widely distributed features in the environment, thereby aiding localization and pose optimization.

- **Tracking:**

Tracking in ORB-SLAM occurs on a separate thread, responsible for localizing the latest frame and deciding when to insert a frame as a keyframe into the constructed map. The Tracking process involves initial feature matching with the previous frame and optimizing the pose using motion-only **BA**. If tracking is lost, the loop closure detection module is activated for global relocalization. Once the initial pose estimation is complete, a local visible area is retrieved using the Covisibility graph of keyframes. Feature matches between the current frame and covisibility keyframes are searched through reprojection and used to further optimize the last frame’s pose. Following this optimization, the Tracking thread decides whether to add the current frame as a keyframe to the mapping.

- **Local Mapping:**

The Local Mapping thread processes new keyframes and performs local **BA** to achieve an optimal reconstruction in the camera pose surroundings. Additionally, Local Mapping searches for feature matches between the processed keyframe and the covisibility graph to optimize keypoint triangulation. Local Mapping is also responsible for removing redundant keyframes.

- **Loop Closure Detection:**

The Loop Closure Detection continuously searches for loops with every newly added keyframe. Upon detecting a loop, a similarity transformation is computed between the keyframes involved in the loop. This transformation matrix describes an estimation of the pose difference between these keyframes. Finally, a pose graph optimization over similarity constraints is performed to achieve a more accurate pose graph. The optimization is conducted over the Essential Graph, which is a

CHAPTER 2. LITERATURE REVIEW

sparser subgraph of the covisibility graph. The method used to detect loop closures, are further described in Section 3.2.

2.3.2 LDSO

As stated in Section 2.2.3, **LDSO** is comprised of **DSO** with added **DBoW3** database construction and querying method as loop closure detection.

- **Direct Sparse Model:**

A continuous photometric error optimization method using a window of recent frames, considering a photometrically calibrated model for image formulation. **DSO** introduces jointly optimizing camera intrinsic, camera extrinsics, and inverse depth values. Performing the photometric equivalent of windowed space bundle adjustment. Marginalization is also applied to manage computational complexity by removing less relevant areas of the frames.

- **Visual Odometry Front-End:**

The visual odometry frontend is tasked with frame management, keyframe selection, and keyframe marginalization. Steps are outlined for initial frame tracking and keyframe creation. A strategy to keep a certain number of keyframes active is also introduced. Outlier and occlusion detection are also added to this part of **LDSO** to improve the accuracy and efficiency of the **VO** algorithm.

2.4 Summary

In conclusion, this section has methodically navigated through the intricate landscape of **SLAM** methodologies, with a particular focus on the application of visual-based **SLAM** within the challenging confines of subterranean environments. The examination of sensor technologies—**LiDAR**, **Radar**, and **IR** cameras—revealed the limitations and potential of these tools in underground settings. Furthermore, the adaptation of visual-based methods to these contexts underscores the need for innovative approaches to overcome the inherent challenges posed by environments lacking in visual features and filled with navigational obstacles.

The Chilean underground mine dataset [54] offers a valuable resource for **SLAM** research, particularly within challenging environmental conditions such as those encountered in underground mining tunnels. However, it is important to acknowledge certain limitations of this dataset, especially in the context of evaluating visual **SLAM** methods. The dataset includes only 43 known positions, which may not be sufficient to provide a comprehensive evaluation of visual **SLAM** algorithms, particularly those requiring extensive ground truth data for accurate performance assessment. Additionally, the frequent low-light conditions observed during the data collection process further diminish the usability of this dataset for visual methods, as such conditions can significantly impair the effectiveness of visual sensors like cameras. Given these factors, while the Chilean underground mine dataset

2.4. SUMMARY

remains a critical resource for **SLAM** research, its applicability to visual **SLAM** evaluation is limited.

The comparative analysis between ORB-SLAM and **LDSO** has been instrumental in shedding light on the advantages and disadvantages inherent to feature-based and direct-based **SLAM** methods. This examination not only highlights the importance of selecting and configuring sensors appropriately for effective **SLAM** implementation but also demonstrates the nuanced understanding required to optimize **SLAM** applications in subterranean environments. The challenges faced with the Chilean underground mine dataset, particularly the low-light conditions and sparse pose updates, underscore the difficulties in deploying visual-based **SLAM** methods in such environments. Consequently, the reliance on the **DARPA** subterranean challenge simulation environment for evaluating these methods illustrates the critical need for suitable datasets that accurately reflect the conditions encountered in underground exploration.

We selected **ORB**, ORB-SLAM3, **BRISK**, **SIFT**, and **SURF** to represent a range of intensity and gradient-based conventional methods, while Zippypoint, SuperPoint, R2D2, and D2NET were included as deep learning approaches. This selection was based on their balance between performance and computational requirements suitable for the proposed embedded processing platform. Other features were excluded due to their computational demands or irrelevance to the specific constraints of the study

This exploration into the capabilities and limitations of visual-based **SLAM** in subterranean settings reinforces the importance of continuous innovation and adaptation in the field of **SLAM** technology. As this research progresses, it is anticipated that further advancements and refinements in **SLAM** methodologies will emerge, enhancing the efficacy and reliability of autonomous navigation and mapping systems in the complex and varied terrains of subterranean environments.

Chapter 3

Methodology

In this chapter, we detail the essential components of our research, focusing on the methodologies and technologies that underpin our advancements in Simultaneous Localization and Mapping (**SLAM**). Our discussion begins with the data acquisition process, facilitated by a straightforward exploration algorithm (Section 3.1), foundational for evaluating **SLAM** techniques.

We then examine the **DBoW** place recognition system (Section 3.2), a critical element for identifying locations within the **SLAM** framework. This examination is followed by an analysis of various **SLAM** methodologies (Section 3.3), where we explore the algorithms driving current innovations in the field.

The conversation shifts to the role of visual features in **SLAM** (Section 3.4), emphasizing their importance in environmental mapping and navigation. Although our primary focus is on visual data, we briefly discuss the utility of **LiDAR**-based odometry for reference accuracy (Section 3.5), acknowledging its relevance in achieving precise measurements.

We also discuss methods used by the EVO library [53] in the evaluation (Section 3.6) of **SLAM** methods, including timestep alignment, pose map alignment, and scaling, in addition to the evaluation metrics used in comparisons.

Concluding this chapter, we highlight modifications applied to ORB-SLAM (Section 3.7) to accommodate various visual features, showcasing our contributions towards enhancing **SLAM** technologies.

This chapter aims to provide a comprehensive overview of the components critical to our research, elucidating the integrated approach that propels our work in **SLAM**. Through a detailed exploration of each element, we intend to offer insights into the complexity and innovation driving advancements in this field.

3.1 Autonomous Exploration and Data Collection

This research involved creating datasets in the **DARPA** Subterranean Challenge simulation environment [85]. Data was recorded using the explorer ds1 **UAV**, which is outfitted with three RGB-D cameras facing upward, downward, and forward. Additionally, the **UAV** is equipped with a Velodyne-16 100m **LiDAR** and an **IMU**, which includes an accelerometer, a gyroscope, and a magnetometer. The simulation environment accurately tracks the **UAV**'s position and orientation.

To enhance the efficiency of the exploration, two key modifications were made to the **UAV**: the battery management system was removed to allow continuous operation, and the pose update frequency was increased from once per second to 20 times per second. This change was made to synchronize with the frame rate of the cameras, which is 20 frames per second.

A simplistic autonomous exploration algorithm, described in following sections, was implemented for navigating the simulated environment. The algorithm uses the data from the depth cameras and **LiDAR** to identify potential paths based on the **UAV**'s current location (Section 3.1.1), while simultaneously updating a map of its position (Section 3.1.3). A collision avoidance system (Section 3.1.4) is also integrated to prevent accidents.

3.1.1 Depth-based Path Detection

The **UAV** employs depth maps and **LiDAR** data, presented as a 2D array of depth values, to locate navigable paths. The system uses mean thresholding on the depth maps to create a binary image. In this image, regions where the depth exceeds the mean depth plus a product of a constant and the standard deviation are marked. These distinct regions in the binary image are labeled to highlight visible paths, with the center points of these regions indicating the direction and distance of each detected path.

To facilitate path detection, the **UAV** utilizes all depth-sensing sensors, compiling their outputs into a comprehensive depth map. The process applies mean and standard deviation thresholding to identify areas that are sufficiently distant and thus represent viable paths. Pixels that are out of the sensor's range, denoted as infinity (∞) or not a number (nan), are treated as being at the sensor's maximum range.

The thresholding process uses the mean distance (μ) and standard deviation (σ) of the depths in the map. The parameter τ can be adjusted to modify the sensitivity of the path detection process, allowing for greater or lesser responsiveness to variations in depth values. **THR** and **DEPTH** represent the calculated threshold map and the depth map obtained from the RGB-D camera, respectively.

$$\mathbf{THR} = \begin{cases} 1 & \text{if } \mathbf{DEPTH} \geq \mu + \tau \times \sigma \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

3.1. AUTONOMOUS EXPLORATION AND DATA COLLECTION

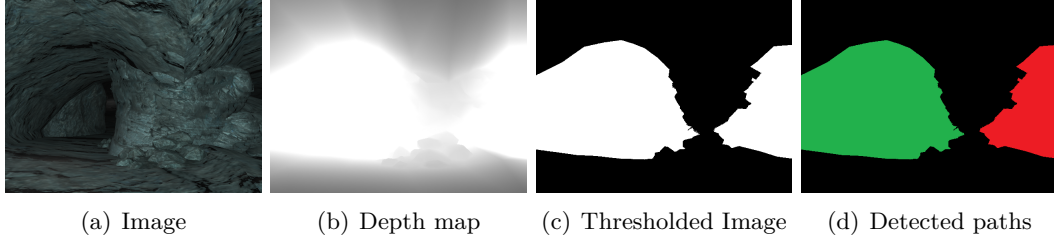


Figure 3.1: This figure illustrates the sequential steps involved in extracting navigable paths from depth maps. Panel (a) displays the original image captured by the sensor, providing the visual context of the environment. Panel (b) presents the corresponding depth map, which encodes the distance of objects from the sensor in grayscale intensity, where darker tones represent closer objects and lighter tones indicate objects further away. In Panel (c), a thresholded version of the depth map highlights areas within a specific distance range, effectively segregating potential navigable spaces from obstacles based on depth criteria. Finally, Panel (d) visualizes the detected paths: one potential path is marked in **green**, indicating a preferred or selected route, while an alternative path is marked in **red**, representing another viable option yet not selected.

The centroid of the detected path, represented as $\mathbf{c} = \begin{bmatrix} x_c & y_c \end{bmatrix}$, is used to determine the direction of the path. Here, $\mathbf{c}$ denotes the centroid's coordinates, where x_c and y_c represent the x and y coordinates of the centroid relative to the detected path. As illustrated in Figure 3.1, this concept is demonstrated using RGB-D data, but the methodology is similar for other types of data. For constructing the depth map, the process involves calculating the Euclidean distance for each point in the point cloud.

The approach to path detection varies between RGB-D and **LiDAR** in terms of computing the local translation, represented by the vector $\Delta_L = \begin{bmatrix} x_L & y_L & z_L \end{bmatrix}$. Here, Δ_L denotes the local translation vector, where x_L , y_L , and z_L correspond to the x , y , and z coordinates of the path relative to the **UAV**'s current position and orientation. For paths detected using **LiDAR**, the local translation is determined by averaging the x , y , and z values of all the 3D points in the **LiDAR** point cloud associated with the path.

In the case of RGB-D cameras, the local translation to the detected path is calculated using several factors. These include the vertical (θ_V) and horizontal (θ_H) angles of the camera's field of view, the average depth (d) within the area of the detected path, and the dimensions of the image, namely its width (w) and height (h). This method allows for a precise calculation of the path's location relative to the **UAV**, as depicted in Figure 3.2.

$$\theta_h = \theta_H \times \left(\frac{w}{2} - x_c \right) \quad (3.2)$$

$$\theta_v = \theta_V \times \left(\frac{h}{2} - y_c \right) \quad (3.3)$$

Using θ_h and θ_v determines Δ_L .

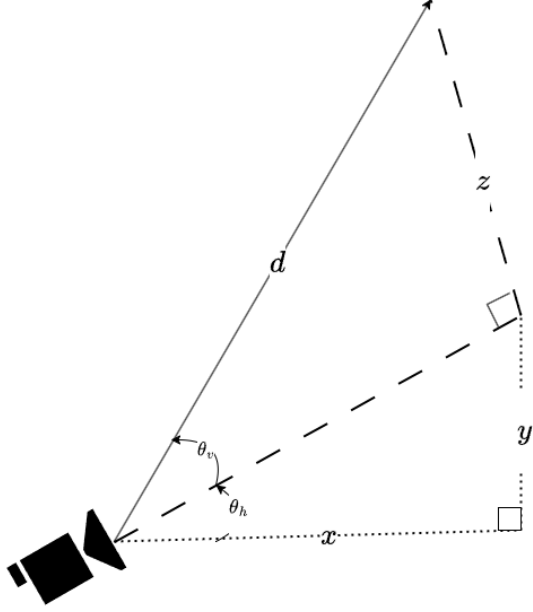


Figure 3.2: This figure demonstrates the decomposition of local translation into its x , y , and z components, facilitated by the utilization of horizontal (δ_h) and vertical (δ_v) angles. By projecting the navigational vector into a 3D space, the illustration clearly delineates how the local translation vector is segmented into its constitutive directional components. The δ_h angle characterizes the rotation around the vertical axis, influencing the x and y components, while δ_v accounts for the elevation or depression angle, directly impacting the z component of translation.

$$x_L = d \times \cos \theta_v \cos \theta_h \quad (3.4)$$

$$y_L = d \times \cos \theta_v \sin \theta_h \quad (3.5)$$

$$z_L = d \times \sin \theta_v \quad (3.6)$$

3.1.2 Transforming Local Translation to Global Translations

In Section 3.1.1, the process of detecting the local translation of a path is described. However, the global translation (Δ_G), representing the global, map-accurate position of

3.1. AUTONOMOUS EXPLORATION AND DATA COLLECTION

a path, is determined using the local translation (Δ_L), the last known pose orientation quaternion ($\mathbf{q}_p = [q_w \ q_x \ q_y \ q_z]$), the last known pose translation ($\mathbf{T}_p = [x_p \ y_p \ z_p]^T$), and the rotation of the observed depth with respect to the UAV ($\mathbf{q}_s = [q_w \ q_x \ q_y \ q_z]$). In the case of the LiDAR and front-facing depth camera, the rotation of the sensors is aligned with the UAV and doesn't require alignment with the UAV's orientation. For upward or downward-facing cameras, quaternion multiplication between $\mathbf{q}_s$ and $\mathbf{q}_p$ is performed to rotate $\mathbf{q}_p$ to align with the direction facing the sensor as follows:

$$\mathbf{q}_r = \begin{cases} \mathbf{q}_p & \text{LiDAR and Front-facing depth cameras} \\ \mathbf{q}_s * \mathbf{q}_p & \text{Upward and downward-facing depth cameras} \end{cases} \quad (3.7)$$

where

$$\mathbf{q}_s = \begin{cases} \begin{bmatrix} 0.707 & 0.0 & -0.707 & 0.0 \end{bmatrix} & \text{Upward-facing depth camera} \\ \begin{bmatrix} 0.707 & 0.0 & 0.707 & 0.0 \end{bmatrix} & \text{Downward-facing depth camera} \end{cases} \quad (3.8)$$

The local translation Δ_L can then be transformed into the global translation Δ_G by multiplying Δ_L by a rotation matrix ($\mathbf{R}$) determined using $\mathbf{q}_r$ [1] as follows:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y + 2q_wq_z & 2q_xq_z - q_wq_y \\ 2q_xq_y - 2q_wq_z & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - q_wq_x \\ 2q_xq_z + 2q_wq_y & 2q_yq_z - 2q_wq_x & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix} \quad (3.9)$$

$$\mathbf{T}_G = \mathbf{T}_p + \Delta_G \quad (3.10)$$

Where the global path position in the environment ($\mathbf{T}_G$) is calculated by summing the global translation and the last known pose. The local translation rotated by the current orientation (Δ_G) is calculated as follows:

$$\Delta_G = \mathbf{R}(\mathbf{q}_r)\Delta_L^T \quad (3.11)$$

3.1.3 Positional Mapping and Navigation

This section details the method for tracking the UAV's previous positions and managing exploration nodes. When multiple potential paths are detected, exploration nodes are added to specific positions on the map. The system also maintains a list of positions that lead to previously detected but unexplored paths.

Pose Graph Construction

The simulation environment provides poses (position and orientation data), which are used to construct and update a pose graph. Each node in this graph represents a specific location in 3D space (x, y, and z coordinates) and is linked to a parent node and possibly

CHAPTER 3. METHODOLOGY

several child nodes. Nodes fall into two categories: positional nodes, which represent specific locations, and exploration nodes, which are added when the **UAV** identifies multiple potential paths. The detection of paths, especially those on the horizontal plane, primarily utilizes **LiDAR** measurements. For paths above or below the **UAV**, depth cameras oriented in the respective directions are used, as shown in Figure 3.3.

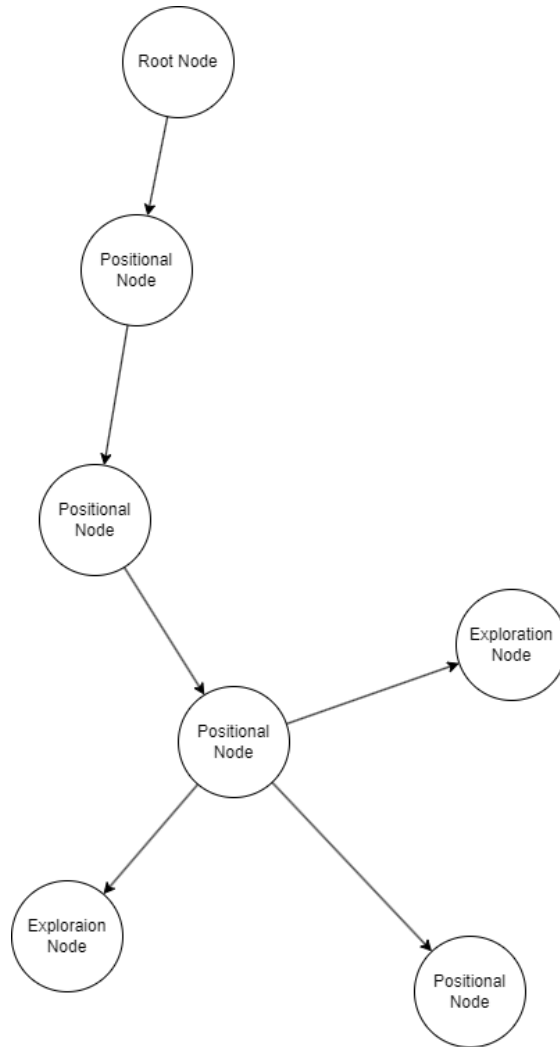


Figure 3.3: This image illustrates the concept of positional mapping via graph construction, emphasizing the graph’s role in defining the connections between positional nodes and exploration nodes. This approach facilitates the revisiting of paths that were not explored initially but were identified in previous time steps.

3.1. AUTONOMOUS EXPLORATION AND DATA COLLECTION

Traversing to Unexplored Areas

When it's time to navigate back to an unexplored exploration node or return to the starting point, the system generates a list of poses to follow. This list starts with the most recently added positional node, and the parent node's position is recursively added until a node with an unexplored child node is reached or the root node (starting point) is encountered. The decision to return to the starting location or proceed to uncharted areas depends on whether there are any unexplored exploration nodes left in the map. If none are available, the UAV is directed back to its initial location.

Path Optimization during Traversal

As the UAV retraces its path, it continually recalculates distances between its current pose and each pose on the planned traversal path. These distances are compared to a predefined threshold. The system selects the pose that is furthest along the planned path but still within this threshold, and then removes all preceding poses from the path. This approach is designed to discover any shorter routes to the traversal path's end goal. Additionally, when deciding which path to take among all visible paths,

3.1.4 Collision Avoidance

Front-facing RGB-D camera's depth data is downsized and converted to a 3D point cloud. Positions between the current UAV position and the path it is following are checked for collisions with the point cloud, using a bounding box around the UAV. In case of a potential collision, the UAV adjusts its path to avoid the obstacle.

A subsample of the front depth map is transformed into a point cloud using a methodology similar to that outlined in Section 3.1.1. The path to the intended destination is discretized to yield several intermediate positions. At each of these positions, a virtual bounding box is employed, starting from the point closest to the UAV. This process is repeated for each position, progressing until a collision is detected. A collision occurs when a point from the occupancy point cloud intersects with the bounding box established around the intermediate positions.

Upon detecting a collision, the position is iteratively adjusted in the opposite direction of the collision until the collision is circumvented, as illustrated in Figure 3.4. This updated position is then incorporated into the path planning process to proactively avoid potential collisions.

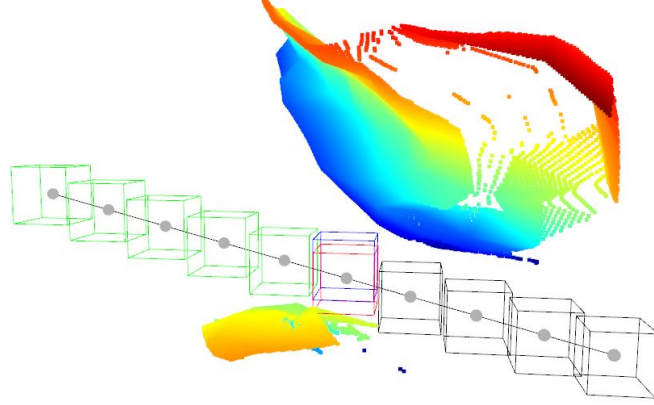


Figure 3.4: Collision avoidance illustration utilizing spheres for intermediate pose positions. **Green** bounding boxes indicate intermediate poses without detected collisions. **Red** bounding boxes highlight poses with detected collisions within the area. **Blue** bounding boxes show adjusted positions to circumvent collisions. Black bounding boxes represent poses not considered further for collision detection. The point cloud is depicted using a color spectrum, starting from **red** for points closest to the viewpoint, to **blue** for points furthest from the viewpoint.

3.1.5 Exploration Strategy

The **UAV** begins in a random exploration mode, prioritizing paths that cover the largest depth map area and giving preference to those that have not been explored. Upon reaching a point where only previously explored paths are visible, the algorithm identifies and follows the path leading to the most recent unexplored area on the map.

When multiple paths are available, they are ranked based on their distance from the already mapped area. The path furthest from the existing mapped areas is selected to ensure that the route most likely to be unexplored is chosen. The front-facing cameras are primarily utilized for collision avoidance rather than for detecting multiple paths on the horizon. The **LiDAR** is used exclusively for identifying horizontal paths due to its extended range and omnidirectional sensing capabilities.

3.2 DBoW Place Recognition

The Bag of Words (**BoW**) [110], [111], [113] based place recognition relies on a bag of visual words approach, constructing a vocabulary based on visual feature descriptors generated offline.

3.2.1 Descriptor Vocabulary

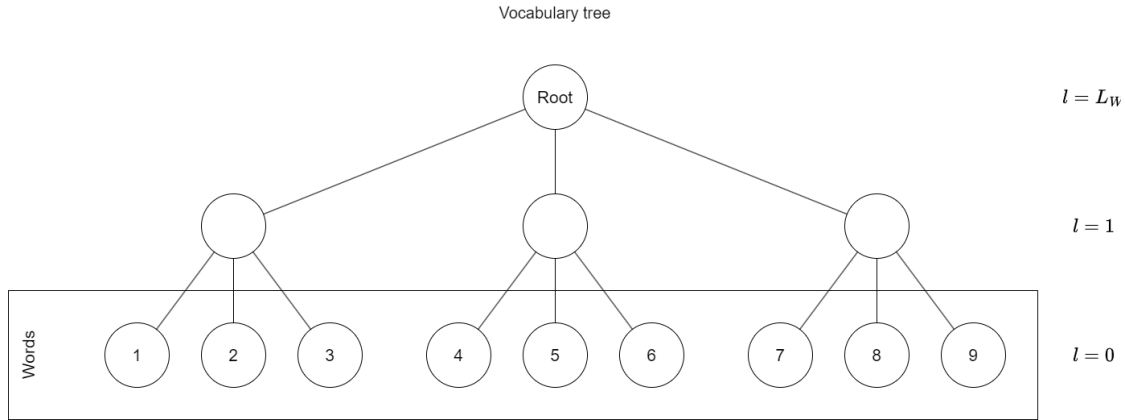


Figure 3.5: DBoW Vocabulary Tree Illustration

The vocabulary is structured as a hierarchical tree, as illustrated in Figure 3.5. Each node within the tree, with the exception of the root, is associated with a specific descriptor. These descriptors are generated from features extracted from the training images. The k-means++ clustering algorithm[14] is employed to organize these features based on their similarity to their parent node. The median values of these clustered groups are then designated as the descriptors for the nodes. In the context of binary descriptors, any value that is not binary is converted to zero. The tree's final layer consists of the vocabulary words for the **BoW** approach, where each word is assigned a weight based on its rarity in the training images. This weighting process utilizes the term frequency-inverse document frequency (tf-idf) method[10], which ensures that words less frequently occurring in the training set are given higher importance.

3.2.2 Image Database Construction

DBoW utilizes two indexing methods: inverse indexing word (Table 3.1) and direct indexing image (Table 3.2).

CHAPTER 3. METHODOLOGY

Inverse index Word n_w		
Image n_{i1}	Image n_{i2}	...
$v_{n_{i1}}^{n_w}$	$v_{n_{i2}}^{n_w}$	...

Table 3.1: Inverse index Word table where $v_{n_{i1}}^{n_w}$ is the weight of features associated with Word n_w from all features contained in image n_{i1} . Similarly, $v_{n_{i2}}^{n_w}$ is the weight of features associated with Word n_w contained in image n_{i2} .

Direct index Image n_i		
Node n_{w1}	Node n_{w2}	...
$f_{f1,f2}^{n_i}$	$f_{f3,f4}^{n_i}$	...

Table 3.2: Direct index Word table illustration where node n_{w1} and node n_{w2} are two leaf nodes of the vocabulary. $f_{f1,f2}^{n_i}$ represents two features detected in image n_i associated with node n_{w1} . Similarly, $f_{f3,f4}^{n_i}$ are features detected in image n_i , but associated with node n_{w2} .

In order to determine image similarity $s(\mathbf{v}_1, \mathbf{v}_2)$, the word weights are used in the following equation:

$$s(\mathbf{v}_1, \mathbf{v}_2) = 1 - \frac{1}{2} \left| \frac{\mathbf{v}_1}{|\mathbf{v}_1|} - \frac{\mathbf{v}_2}{|\mathbf{v}_2|} \right| \quad (3.12)$$

Where, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the **BoW** vectors and is comprised of all the weight values associated with the image in question from the first to the last word.

The inverse index provides a good indication of similar images when two images have similarly high weights for one particular word. Whereas the direct index can provide similar features between distant images, removing the need for time-consuming brute-force matching to get feature matching and only computing descriptor distances for a subset of descriptors.

3.2.3 Loop Closure Detection

Loop closures are detected through a multi-step process that begins with a database search for similar images. This search yields a list of potential matches, each accompanied by a similarity score. To calculate a normalized similarity score, we divide all scores by the similarity score between the current and previous frame. This calculation is based on the assumption that two consecutive frames are similar and serves as a reference point for identifying other similar images. For this measure to be effective, a minimum threshold of similarity is required between the last two images.

In the second step, match grouping aggregates similar images that are temporally close into clusters, referred to as ‘islands’, and treats each cluster as a single match. The island with the highest similarity score is then evaluated for temporal consistency. This requires the same island to maintain the highest score across a sequence of previous

3.3. SLAM TECHNIQUES IN SUBTERRANEAN ENVIRONMENTS

images to qualify as a potential loop closure.

The final step in determining a loop closure involves verifying geometrical consistency. This is done by employing Random Sample Consensus (**RANSAC**) to identify a fundamental matrix that is supported by at least 12 correspondences. Only when geometrical consistency is confirmed can a loop closure be definitively detected.

3.3 SLAM Techniques in Subterranean Environments

In our study, we evaluated the effectiveness of visual features in subterranean environments using two well-known **SLAM** (Simultaneous Localization and Mapping) systems: ORB-SLAM3 [88] and HFNet-SLAM. HFNet-SLAM is a modified version of ORB-SLAM3, where its standard **ORB** feature extraction component is replaced with alternative methods. This modification preserves the core **SLAM** functionalities, including pose optimization and backend processing, but introduces different feature extraction techniques.

3.3.1 Feature Extraction Comparison

For a fair comparison of various feature extraction methods, we maintained default parameter settings as provided in their respective implementations. We used OpenCV [39] for implementing **ORB**, **BRISK**, **SIFT**, and **SURF** methods. For other methods like **R2D2**, D2Net, Superpoint, and Zippypoint, we adhered to default values from their respective GitHub repositories.

Given the diverse nature of feature descriptors—ranging from intensity-based (e.g., **ORB** [27], **BRISK** [26]) to gradient-based (e.g., **SIFT** [12], **SURF** [20])—we disabled the loop closure threads in **ORB-SLAM3**. This decision was made to prevent false loop closures, a common issue observed in subterranean environments due to repetitive textures. The simulated environment data highlighted these occurrences.

3.3.2 Loop Closure Challenges

Evaluating loop closure detection with simulation data proved challenging. ORB-SLAM3’s reliance on a specific vocabulary for each descriptor method (**DBoW2** [31]) complicates its application with gradient-based descriptors, which require additional steps for compatibility. In contrast, HFNet-SLAM replaces the **DBoW2** method with a deep learning-based global descriptor integrated into its **CNN** architecture. This approach necessitates training a unique global descriptor for each deep learning feature descriptor method.

Other discussed loop closure methods, such as **HOG** and **DL**-based approaches, might be better suited for detecting loop closures in subterranean environments. However, changing the loop closure method of the well-known **ORB-SLAM3** framework would require significant modifications, as well as adequate training data for **DL**-based approaches.

CHAPTER 3. METHODOLOGY

Thus, this would fall outside the scope of this study and will be considered in future work.

3.3.3 Graph-Based SLAM Techniques

Graph-based **SLAM** methods, particularly suitable for low processing environments and monocular visual **SLAM**, are categorized into direct, feature-based, and semi-direct (hybrid) approaches. These techniques involve constructing a graph where nodes represent observed poses, and edges signify expected movements between poses. The optimization of this graph minimizes positional errors from various observations, thereby adjusting the vertices.

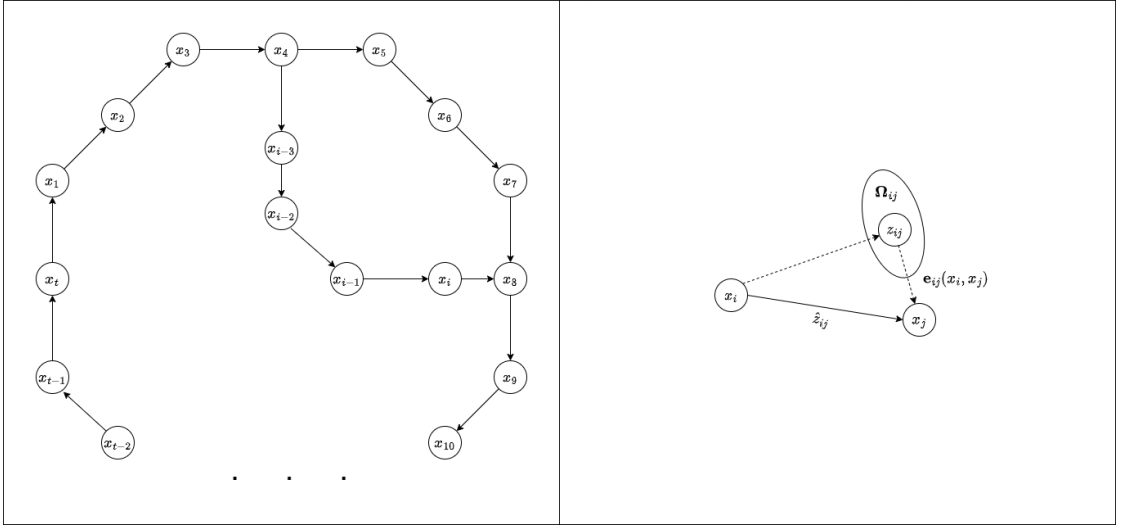


Figure 3.6: An illustration of graph-based **SLAM** showing a constructed graph with two explored paths (left), and constraints between two poses (right). Each vertex in the graph represents estimated motion $\hat{z}_{ij}$ from pose x_i to x_j . The error between the actual position z_{ij} and the estimate $\hat{z}_{ij}$ is denoted as $\mathbf{e}_{ij}$. The information matrix Ω_{ij} quantifies the uncertainty in the estimation $\hat{z}_{ij}$.

Initial pose movements in monocular settings are typically estimated using vehicle kinematics or **VO**. **VO** triangulates expected movements by matching features across consecutive frames. When loop closures or repeated object observations occur, the graph is adjusted or optimized to align more closely with these observations. Common optimization techniques include maximum likelihood and least squares methods, with a General Graph Optimization (g^2o) framework [25] being a prevalent non-linear least squares approach in contemporary **SLAM** methods. The primary aim of graph optimization is to minimize error $\mathbf{e}_{ij}$ illustrated in Figure 3.6, and maximize pose certainty, enhancing pose accuracy based on cumulative observational data.

3.3. SLAM TECHNIQUES IN SUBTERRANEAN ENVIRONMENTS

The g2o library is a versatile framework designed for optimizing graph-based models, widely used in the context of **SLAM**. It facilitates efficient and precise optimization of complex models by representing them as graphs, where nodes correspond to variables such as sensor states or environmental features, and edges represent constraints or observations between these variables. In **SLAM**, g2o is instrumental for solving the optimization problems inherent in constructing a map of an unknown environment while simultaneously determining the location of the sensor within it. This library supports various optimization techniques, making it a powerful tool for researchers and developers working on robotics and augmented reality applications where accurate and real-time mapping and localization are critical.

3.3.4 ORB-SLAM

ORB-SLAM has developed through three iterations. The first iteration [40], focused solely on monocular setups, did not include the sensor configurations introduced in later versions. ORB-SLAM2 [57] added support for stereo and RGB-D sensor configurations. This version utilizes feature points from RGB-D sensors to calculate stereo coordinates and includes full **BA** in the loop closure thread. The third version, ORB-SLAM3 [88], integrates inertial measurements for aiding initial pose estimation when new frames are added to the system and introduces multi-map mapping and map serialization, allowing for multi-session exploration by saving and reloading the created map for future exploration.

Map Point Projection

ORB-SLAM performs optimization of 3D map point locations ($\mathbf{X}_{w,i} \in \mathbb{R}^3$) and keyframe poses ($\mathbf{T}_{w,j} \in SE(3)$), where w indicates the reference to the world frame, u refers to the i^{th} frame, and j the j^{th} keyframe. This process is crucial for accurately mapping and navigating within an environment. The group $SE(3)$ represents the space of 3D poses, combining rotations and translations, to describe the spatial relationship of the camera to the world.

The optimization aims to minimize the re-projection error of matched feature points, applicable in both monocular ($\mathbf{x}_m^i \in \mathbb{R}^2$) and stereo ($\mathbf{x}_s^i \in \mathbb{R}^3$) configurations. This is achieved through projection functions π_m for monocular and π_s for stereo or RGB-D sensors, respectively. Where, f_x and f_y denote the camera's focal lengths, c_x and c_y represent the coordinates of the principal point, and b is the baseline of the camera, with these parameters being derived from the camera's calibration data. The optimization of keyframe poses within $SE(3)$ is critical for maintaining accurate and consistent mapping and localization, enabling ORB-SLAM to effectively navigate and reconstruct

CHAPTER 3. METHODOLOGY

the environment.

$$\pi_m \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}, \quad (3.13)$$

$$\pi_s \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix}. \quad (3.14)$$

Motion-only Bundle Adjustment

Motion-only **BA** optimizes the camera orientations ($\mathbf{R}_{iw}$) and positions ($\mathbf{t}_{iw}$), minimizing the projection error between visual features matched to map points. The robust Huber cost function (ρ) is used in the optimization, where Σ represents the covariance associated with the scale at which the keypoints were extracted.

$$\{\mathbf{R}_{iw}, \mathbf{t}_{iw}\} = \underset{\mathbf{R}_{iw}, \mathbf{t}_{iw}}{\operatorname{argmin}} \sum_{i \in \mathbf{X}} \rho \left(\|\mathbf{x}^i - \pi(\mathbf{R}_{iw} \mathbf{X}^i + \mathbf{t}_{iw})\|_{\Sigma}^2 \right) \quad (3.15)$$

where

$$\rho(a) = \begin{cases} \frac{1}{2} s^2 & |a| < \delta \\ \delta(|a| - \frac{\delta}{2}) & |a| \geq \delta \end{cases} \quad (3.16)$$

Local Bundle Adjustment

The Local **BA** optimization function, similar to Motion-only **BA**, adjusts not only the rotation and translation but also the locations of map points and keyframe positions in covisible keyframes (K_L). All other map points observed in P_L but not part of K_L (K_F) contribute to the cost function but are not adjusted during Local **BA**.

$$\{\mathbf{x}^i, \mathbf{R}_l, \mathbf{t}_l \mid i \in P_L, l \in K_L\} = \underset{\mathbf{x}^i, \mathbf{R}_l, \mathbf{t}_l}{\operatorname{argmin}} \sum_{k \in K_L \cup K_F} \sum_{j \in \mathbf{X}_K} \rho \left(\|\mathbf{x}^j - \pi(\mathbf{R}_k \mathbf{X}^i + \mathbf{t}_k)\|_{\Sigma}^2 \right) \quad (3.17)$$

Full Bundle Adjustment

Full **BA** follows a similar optimization problem to Local **BA**, including all keyframes and map points in the adjustments, except the origin keyframe, which remains fixed in the map.

3.3. SLAM TECHNIQUES IN SUBTERRANEAN ENVIRONMENTS

Visual-Inertial SLAM

The integration of inertial measurements in visual **SLAM** extends the state vector. Originally, the state vector only tracked the pose ($\mathbf{T}_i$), encompassing translation and rotation ($\begin{bmatrix} \mathbf{R}_i & \mathbf{p}_i \end{bmatrix}$). With inertial measurements, the state vector additionally includes constant velocity ($\mathbf{v}_i$) and estimates biases for the accelerometer ($\mathbf{b}_i^a$) and gyroscope ($\mathbf{b}_i^g$), which are presumed to drift slowly over time.

The expanded state vector is represented as $\mathbf{S}_i = \begin{bmatrix} \mathbf{T}_i & \mathbf{v}_i & \mathbf{b}_i^a & \mathbf{b}_i^g \end{bmatrix}$. In visual-inertial **SLAM**, inertial measurements between frames are pre-integrated due to the high sampling frequencies of **IMU** sensors. This approach ensures more accurate and timely pose updates between frames, maintains the constant velocity model, and provides initial pose estimates. By leveraging both accelerometer and gyroscope data, translation and rotation differences between frames can be effectively estimated.

3.3.5 Direct Sparse Odometry with Loop Closure

LDSO employs **DSO** which is a direct-based **SLAM** using a gradient-based inverse depth estimation, which is aligned between many frames. **LDSO** also employs a loop closure detection method similar to what ORB-SLAM use, with a different version. **LDSO** use the third implementation of **DBoW**[113], whereas ORB-SLAM use the second implementation of **DBoW**[111].

CHAPTER 3. METHODOLOGY

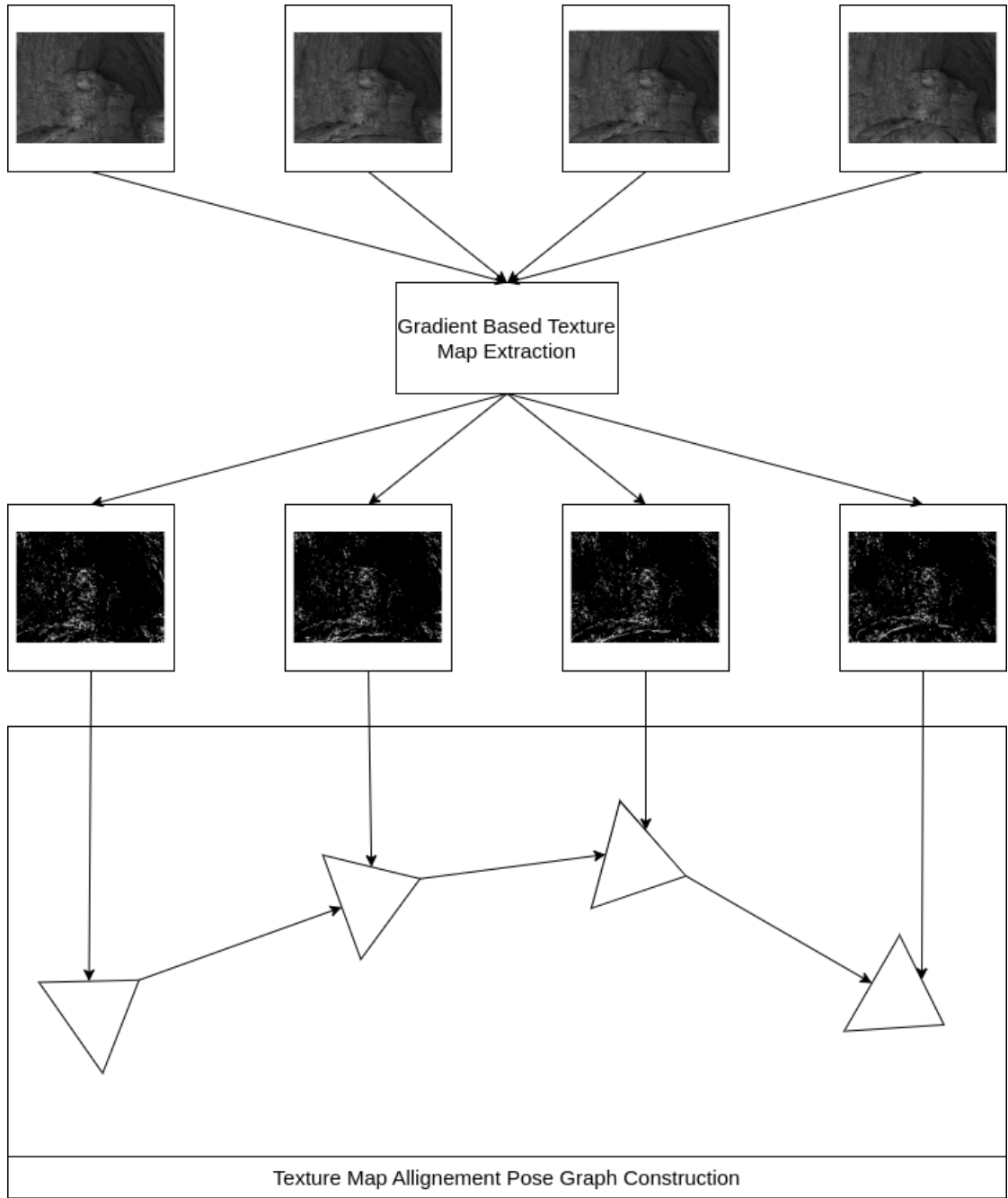


Figure 3.7: The image visually simplifies how **LDSO** estimates motion across video frames. It focuses on the technique of extracting texture maps from image gradients, a method that emphasizes the structural variations within a scene. These texture maps are meticulously aligned, forming the basis for estimating motion between frames.

3.4 Feature Extraction Methods

This section provides detailed descriptions of visual features utilized in comparing the usability of visual features in subterranean environments. It encompasses conventional gradient-based methods discussed in Sections 3.4.3 and 3.4.4, alongside conventional intensity-based methods in Sections 3.4.5 and 3.4.6. Additionally, DLNN approaches are explored in Sections 3.4.7, 3.4.8, 3.4.9, and 3.4.10. None of the DLNN models were self-trained, and this study relies on existing pre-trained models.

To facilitate the use of DLNN methods, commonly implemented in the Python programming language, within the ORB-SLAM architecture—a C++ framework—a method for communication between C++ (ORB-SLAM) and Python (feature extraction) is proposed and implemented, as outlined in Section 3.4.1.

Moreover, Section 3.4.2 discusses the most common methods to achieve scale-invariant features.

3.4.1 Python C++ Network Link

To incorporate deep learning-based feature extraction and description methods into the modified version of ORB-SLAM, most of the deep learned feature extraction methods have been implemented in Python. The localhost loopback mechanism serves as the communication bridge for exchanging images and feature keypoints and descriptors between Python and C++.

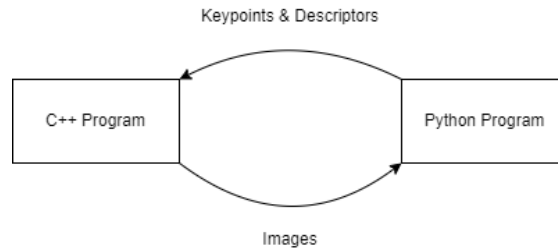


Figure 3.8: This diagram illustrates the integration of Python-based feature description methods within a C++ environment. It shows the bidirectional communication flow where images are sent from the C++ program to a Python script for analysis using advanced feature description algorithms. The script then returns visual feature descriptors and keypoints to the C++ application. This setup allows the C++ program to leverage Python’s extensive libraries for enhanced image analysis without implementing complex algorithms directly in C++.

CHAPTER 3. METHODOLOGY

Bytes	Height 4 (integer)	Width 4 (integer)	Channels 4 (integer)	Concatenated Image Height $\times$ Width $\times$ Channels
-------	-----------------------	----------------------	-------------------------	---

Table 3.3: Python-C++ Link Image Serialization

Keypoints and descriptors are communicated individually once extracted by the Python program.

Bytes	Number of Keypoints 4 (integer)	Descriptor Length 4 (integer)	Concatenated Keypoints (Number of Keypoints) $\times$ 7 $\times$ 4	Concatenated Descriptors (Number of Keypoints) $\times$ (Descriptor Length) $\times$ 4 (float)
-------	------------------------------------	----------------------------------	---	---

Table 3.4: Python-C++ Link Keypoints and Descriptors Serialization

Each keypoint comprises all parameters of the OpenCV Keypoint class.

Bytes	x position 4 (float)	y position 4 (float)	size 4 (float)	angle 4 (float)	response 4 (float)	octave 4 (integer)	class_id 4 (integer)
-------	-------------------------	-------------------------	-------------------	--------------------	-----------------------	-----------------------	-------------------------

Table 3.5: Python-C++ Link Keypoint Serialization

The above tables describe how the data is serialized to transmit the images and visual features, enabling the use of features implemented in Python with the ORB-SLAM3 architecture in C++.

3.4.2 Scale Invariant Feature Extraction

Achieving scale invariance in feature extraction is crucial, as it allows for consistent feature matching across images taken from varying distances. This consistency improves matches and forms more robust constraints between distant keyframes, thereby enhancing pose graph construction and optimization. The most common method to attain scale invariance involves constructing a scale-space pyramid[19]. Here, the original image is down-sampled using a scale factor to create multiple scaled versions. Features are then extracted from each level of this pyramid, scaling the keypoint positions and sizes relative to the original image dimensions.

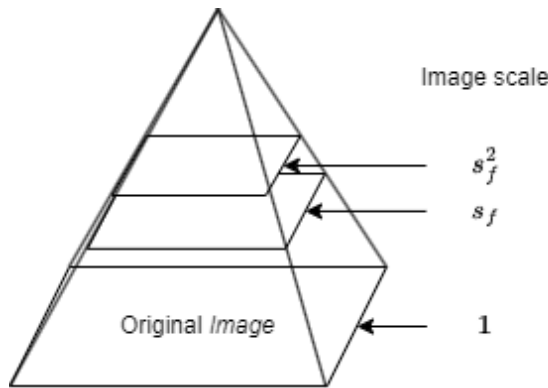


Figure 3.9: Scale-space pyramid constructed with the scale factor s_f .

3.4. FEATURE EXTRACTION METHODS

3.4.3 SIFT

Feature Extraction

SIFT[12] achieves feature extraction by estimating the Difference of Gaussians (**DoG**). This process involves repeatedly applying a Gaussian filter to the image, generating several blurred versions. Subtracting consecutive blurred images yields the **DoG**. Local extremes are detected using three consecutive difference frames, where a point is considered a local extreme if its center position is the maximum or minimum within a $3 \times 3 \times 3$ patch of the stacked difference maps.

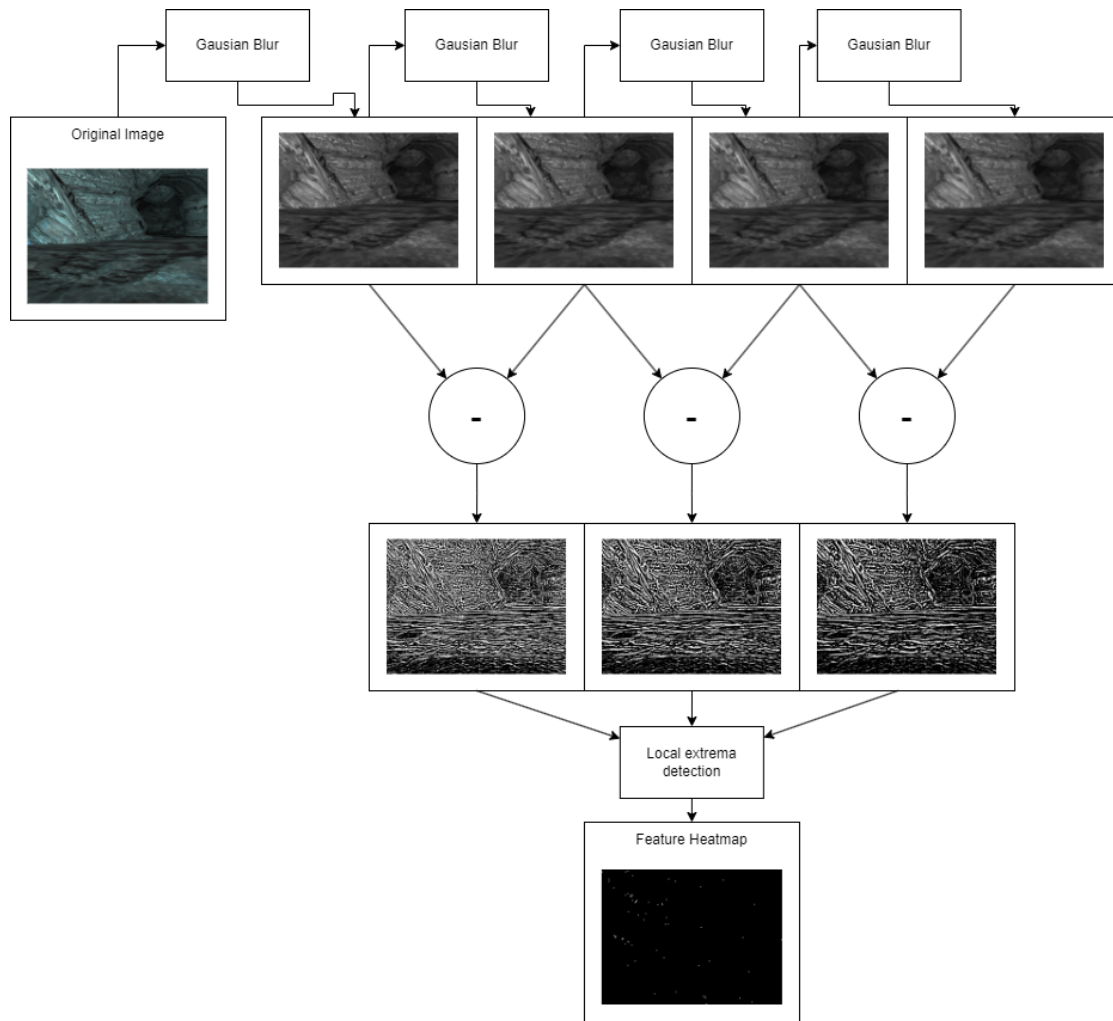


Figure 3.10: SIFT feature extraction illustration using an image from the recorded dataset.

CHAPTER 3. METHODOLOGY

Feature Description

SIFT uses an image gradient-based descriptor. It calculates image gradients in a 16×16 patch surrounding the detected interest point. These gradients populate four histograms, each with eight orientation bins to cover a full rotation. The patch is divided into 4×4 sub-patches to obtain these histograms. Concatenating the bin values from these histograms forms the descriptor, resulting in a 128-element descriptor vector ($4 \times 4 \times 8$).

3.4.4 SURF

SURF[20] features represent an enhancement over the **SIFT** feature extraction method, primarily focusing on reducing the time required for feature extraction and description. Unlike **SIFT**'s **DoG** response, **SURF** utilizes the Determinant of the Hessian (**DoH**) response. This approach leverages an integral image concept, significantly reducing the number of floating-point operations compared to **SIFT**'s Gaussian kernel.

Feature Extraction

SURF employs an integral image for feature extraction instead of time-consuming convolution. An integral image is created by summing pixel intensities up to a certain position (x, y) . Combined with box filters, this method significantly speeds up computation. The Hessian matrix $\mathbf{H}(\mathbf{x}, \sigma)$ at a point $\mathbf{x} = (x, y)$ and scale σ is defined as:

$$\mathbf{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (3.18)$$

Here, L_{xx} , L_{xy} , and L_{yy} are computed using box filters applied to the integral image.

3.4. FEATURE EXTRACTION METHODS

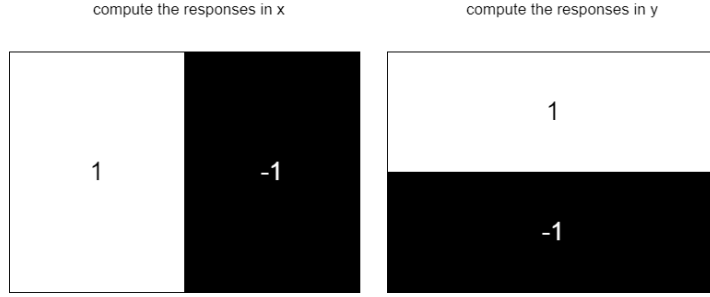


Figure 3.12: Haar wavelet filters used for computing x and y responses, replacing image gradient orientation in descriptor calculation.

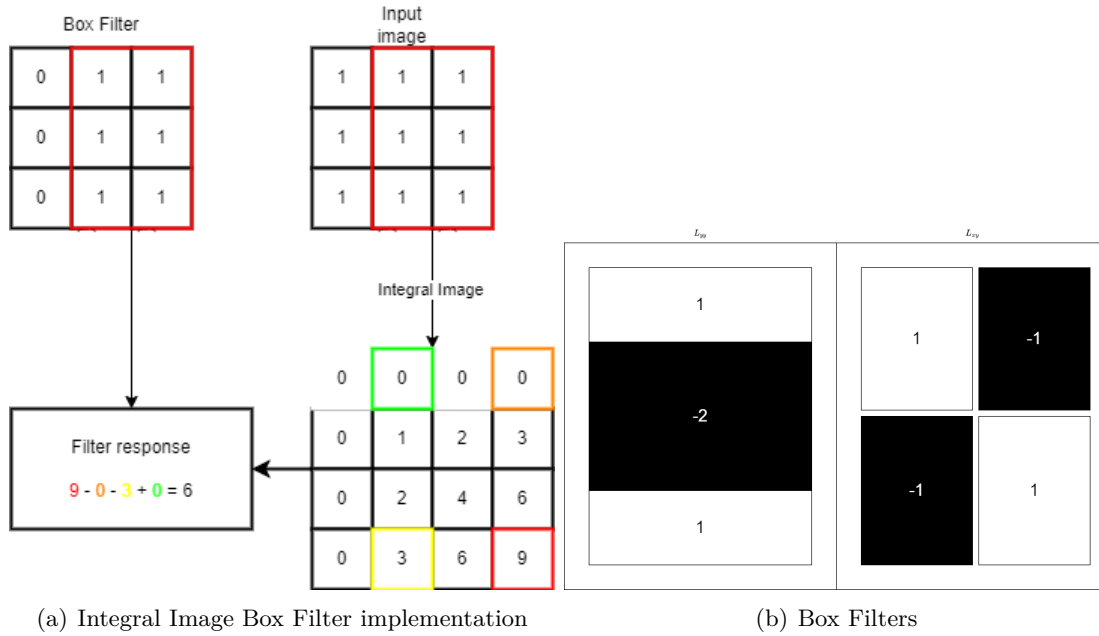


Figure 3.11: The image illustrates the application of **DoH** box filters for calculating L_{xx} , L_{xy} , and L_{yy} highlighting the specific box filters used for these calculations. Furthermore, it demonstrates the use of integral images to reduce the computational effort required to compute the responses of box filters. Notably, L_{yy} is derived through the application of the L_{xx} box filter, rotated by 90° .

Feature Description

The feature description in **SURF**, while similar to **SIFT**, replaces the computationally intensive image gradient calculation with Haar wavelet filters for x and y direction responses.

The **SURF** descriptor vector consists of the sum of magnitudes in the x and y directions, as well as their absolute values. These values from 4×4 sub-patches are

CHAPTER 3. METHODOLOGY

concatenated to form the descriptor, resulting in a descriptor half the size of **SIFT**'s due to replacing eight orientation bins with four magnitude-based values.

3.4.5 ORB

Feature Extraction

ORB [27] feature extraction employs the FAST corner detection method[13], which evaluates each pixel individually for corner characteristics.

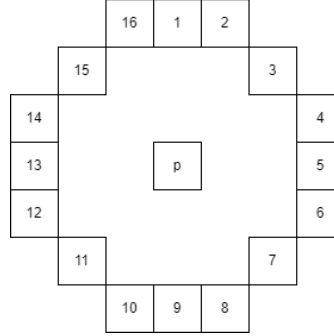


Figure 3.13: Fast Corner Detection

For a pixel p to qualify as a corner, a set number of consecutive pixels (from 1 to 16) around it must be significantly darker or brighter than p itself, determined using an intensity threshold.

Feature Description

ORB utilizes a modified version of the **BRIEF** for intensity-based description, incorporating orientation into the process. **BRIEF** achieves a binary descriptor by comparing the intensities of pixel pairs surrounding the detected feature.

ORB computes the orientation of a patch by calculating the pixel intensity (I) moments:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (3.19)$$

The centroid (C) and orientation (θ) of the patch are then determined:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (3.20)$$

$$\theta = \text{atan2}(m_{01}, m_{10}) \quad (3.21)$$

A predefined set of pixel locations ($\mathbf{S}$) is rotated using a 2D rotation matrix ($\mathbf{R}_\theta$) to achieve rotation invariance:

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \quad (3.22)$$

The descriptor is then obtained by comparing pixel intensities within the rotated set $\mathbf{S}_\theta$.

3.4. FEATURE EXTRACTION METHODS

3.4.6 BRISK

Feature Extraction

BRISK [26], similar to **ORB**, uses $FAST_{16}^9$ for feature detection. This approach requires that 9 out of the 16 pixels, arranged in a circle, be significantly darker or brighter than the central pixel to qualify as a feature.

Feature Description

BRISK employs a feature description method akin to **ORB**, but with a distinct sampling approach. Instead of comparing single pixel values, **BRISK** applies a Gaussian smoothing kernel around each sampling point. The standard deviation of this kernel, determined by the distance to the center of the patch or the detected keypoint location, influences the radius of the circle and thus affects the smoothed value at each sampling point.

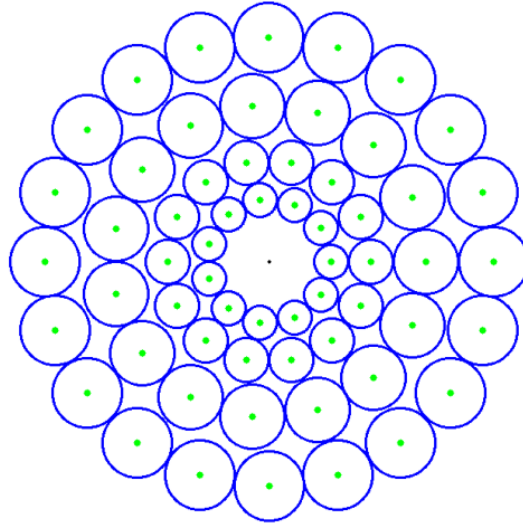


Figure 3.14: **BRISK** Sampling Pattern, illustrating sampled pixel positions in green and the area affected by the Gaussian kernel in smoothing of intensity values at sampling points in blue.

3.4.7 Superpoint

Deep Learning Neural Network Architecture

Superpoint[64] utilizes a shared **CNN** encoder, which serves both the interest point detection and feature descriptor network.

CHAPTER 3. METHODOLOGY

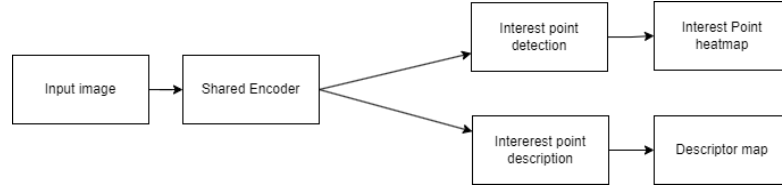


Figure 3.15: Superpoint DLNN Architecture

This shared encoder streamlines processing by merging the initial feature extraction steps for both interest point detection and feature description. The interest point heatmap provides a response for each pixel in a processed version of the input image, including image resizing. The image descriptors are computed not for individual pixels but for patches of pixels. When an image of size (w, h) is fed into the **CNN**, it produces a descriptor vector of size $(\frac{w}{8}, \frac{h}{8})$, which is then upsampled to generate descriptors for each pixel.

Deep Learning Neural Network Training

Training Superpoint, as described but not performed in this work, involves creating a synthetic dataset with well-defined shapes and feature points. These shapes undergo various transformations, such as cropping, translation, scaling, rotation, and perspective distortion, while maintaining accurate feature point locations. Initially, the model is trained on this synthetic dataset. Subsequently, iterative homographic adaptations are applied to real-world images to refine the training. This process trains the descriptor to match feature positions detected by the synthetically trained **CNN** with those identified in homographically adapted iterations of real-world images.

3.4.8 Zippypoint

Deep Learning Neural Network Architecture

Zippypoint[93], akin to Superpoint, employs a shared encoding scheme for both feature point detection and description. It utilizes the KP2D architecture[76], which, unlike Superpoint, does not generate a heatmap. Instead, it produces two lists: one containing the locations of interest points, and the other containing the corresponding keypoint scores.

3.4. FEATURE EXTRACTION METHODS

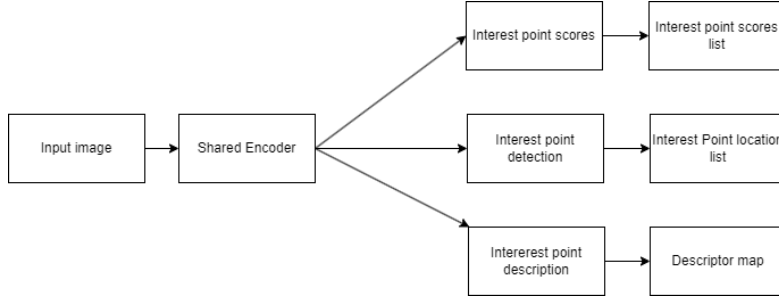


Figure 3.16: Zippypoint DLNN Architecture

Deep Learning Neural Network Training

Zippypoint, as well as Superpoint, utilizes the Microsoft COCO dataset[34] for training. This dataset offers segmented images, each accompanied by segmentation masks, which provide valuable data for the training of these feature extraction models.

3.4.9 R2D2

Deep Learning Neural Network Architecture

In line with other architectures, **R2D2** [75] also shares its encoder between the descriptor and interest point detector components. The interest point detection in **R2D2** is designed to calculate both reliability and repeatability scores. Notably, **R2D2** employs a **FCNN**, which means all its layers are convolutional.

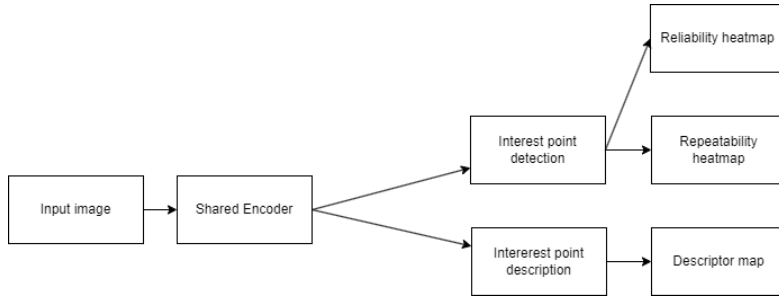


Figure 3.17: R2D2 DLNN Architecture

Deep Learning Neural Network Training

R2D2 's training involves image sets with known correspondences. It employs two methods to achieve this. The first involves generating a second image from the first by applying a known transformation. The second method uses datasets that provide accurate structure-from-motion information, allowing for the determination of correspondences between consecutive images.

CHAPTER 3. METHODOLOGY

3.4.10 D2NET

Deep Learning Neural Network Architecture

D2Net’s[73] architecture also integrates joint detection and description. When provided with an image, the network produces an interest point heatmap and corresponding descriptors.

Deep Learning Neural Network Training

D2-Net is trained using the MegaDepth dataset[67], which consists of over a million photographs across 196 different scenes. The COLMAP structure-from-motion technique [46], [47] is used to extract image correspondences. These correspondences are then utilized to train D2-Net.

3.5 Keep it Small and Simple Iterative Closest Point Lidar Odometry

KISS-ICP [95] is a widely recognized **LiDAR** odometry method known for its lightweight implementation and high accuracy.

3.5.1 Motion Prediction and Scan Deskewing

KISS-ICP employs a constant velocity model for initial pose estimation and compensating for sensor movement during sampling. This model, while effective, may be less accurate during acceleration or deceleration phases. Such inaccuracies are mitigated through registration. The velocities at timestep t , including angular velocity (ω_t) and translational velocity (v_t), are crucial. We utilize previous pose estimates, specifically $\mathbf{T}_{t-1} = (\mathbf{R}_{t-1}, \mathbf{t}_{t-1})$ and $\mathbf{T}_{t-2}$, where $\mathbf{R}$ denotes the rotation matrix and $\mathbf{t}$ represents the translation vector. The relative pose between these time steps is predicted using the following formula:

$$\mathbf{T}_{pred,t} = \begin{bmatrix} \mathbf{R}_{t-2}^T \mathbf{R}_{t-1} & \mathbf{R}_{t-2}^T (\mathbf{t}_{t-1} - \mathbf{t}_{t-2}) \\ \mathbf{0} & 1 \end{bmatrix} \quad (3.23)$$

The calculated velocities, where Δt is the time taken for one **LiDAR** sweep, are:

$$\mathbf{v}_t = \frac{\mathbf{R}_{t-2}^T (\mathbf{t}_{t-1} - \mathbf{t}_{t-2})}{\Delta t} \quad (3.24)$$

$$\omega_t = \frac{\log(\mathbf{R}_{t-2}^T \mathbf{R}_{t-1})}{\Delta t} \quad (3.25)$$

Scan deskewing involves using estimated velocities and times (s) for each detected point ($\mathbf{P}$), compensating for sensor movement during **LiDAR** sweeps.

In the **KISS-ICP** method, the *Exp* function is utilized to correct for the motion of the **LiDAR** sensor during the time it takes to complete a full sweep, denoted by δt . This

3.6. EVALUATION

correction is essential for both angular and translational velocities. The *Exp* function, based on the axis-angle representation, generates a rotation matrix that reflects the sensor’s orientation at the exact moment each point in the **LiDAR** scan was collected.

For a given point $\mathbf{P}_i$ detected at time $\mathbf{s}_i$, where $\mathbf{s}_i$ varies from 0 to δt across the duration of the **LiDAR** sweep, the function is defined as:

$$\mathbf{P}_i^* = \text{Exp}(\mathbf{s}_i \boldsymbol{\omega}_t) \mathbf{P}_i + \mathbf{s}_i \mathbf{v}_t \quad (3.26)$$

Where, $\text{Exp}(\mathbf{s}_i \boldsymbol{\omega}_t)$ calculates the rotation matrix for the point $\mathbf{P}_i$ based on the angular velocity $\boldsymbol{\omega}_t$ and the specific time $\mathbf{s}_i$ at which the point was sampled. The term $\mathbf{s}_i \cdot \mathbf{v}_t$ accounts for the translational movement of the **LiDAR** sensor, ensuring that each point is accurately adjusted for both the rotation and translation of the sensor. This comprehensive correction is what allows **KISS-ICP** to effectively deskew the **LiDAR** points, ensuring accurate point cloud registration and odometry calculations.

3.5.2 Point Cloud Subsampling

LiDAR generates a vast number of 3D points, necessitating significant processing time for alignment and correspondence identification between scans. The original point cloud is typically subsampled to decrease the number of correspondences. **KISS-ICP** employs voxel downsampling, dividing the 3D space into cubic units called voxels. Each voxel contains a limited number of points, with only one point per voxel being sampled. This voxel downsampling technique is applied twice to the deskewed point cloud.

3.5.3 Local Map and Correspondence Estimation

KISS-ICP maintains an up-to-date global point cloud map. With each new **LiDAR** scan, the map is refreshed. Correspondences are established between this global point cloud map and the downsampled **LiDAR** scan. After computing correspondences and estimating pose translation and rotation, points not involved in the correspondences are added to the map. These points are then utilized in finding correspondences with subsequent **LiDAR** scans.

3.5.4 Alignment Through Robust Optimization

ICP is a widely-used method for iteratively aligning point clouds, starting from an initial estimate of pose and rotation. It utilizes the distances between point pairs as correspondences. The alignment process involves iteratively adjusting the pose in the direction of these correspondences, which are recalculated in each iteration.

3.6 Evaluation

To evaluate the accuracy of **SLAM** methods, we initially align the predicted poses with the ground truth poses using the Umeyama Alignment technique, as outlined in Section

CHAPTER 3. METHODOLOGY

3.6.1. Following this alignment, we apply the performance metrics detailed in Section **3.6.2** to accurately and reliably compare the performance across different methods. This evaluation is conducted using both Absolute Pose Errors (APE) and Relative Pose Errors (RPE). APE assesses the discrepancy of each predicted pose relative to the global ground truth, whereas RPE measures the error in the predictions between consecutive timesteps.

3.6.1 Umeyama Alignment

Umeyama Alignment, as described in [6], to scale and align the predicted **SLAM** pose with the ground truth obtained from the simulation environment.

Umeyama alignment computes the least squares solution that minimizes the distance between registered points. In this case, the registered points are 3D pose points between the predicted and ground truth poses. The alignment is performed between two matrices of poses, namely, $\mathbf{x}$ and $\mathbf{y}$, with means $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_y$, standard deviations σ_x and σ_y in poses, and Σ_{xy} as the covariance matrix of x and y .

$$\Sigma_{xy} = \frac{1}{N} \sum_{i=0}^N (\mathbf{y}_i - \boldsymbol{\mu}_y)(\mathbf{x}_i - \boldsymbol{\mu}_x)^T \quad (3.27)$$

The singular value decomposition of Σ_{xy} is denoted as $\mathbf{UDV}^T$.

$$\mathbf{S} = \begin{cases} I & \text{if } \det \Sigma_{xy} \geq 0 \\ \text{diag}(1, 1, \dots, 1, -1) & \text{if } \det \Sigma_{xy} \leq 0 \end{cases} \quad (3.28)$$

Where $\det$ refers to the determinant of the matrix. The rotation ($\mathbf{R}$), translation ($\mathbf{t}$), and scale (c) can be calculated using the following equations.

$$\mathbf{R} = \mathbf{USV}^T \quad (3.29)$$

$$\mathbf{t} = \boldsymbol{\mu}_y - c\mathbf{R}\boldsymbol{\mu}_x \quad (3.30)$$

$$c = \frac{1}{\sigma_x^2} \text{tr}(\mathbf{DS}) \quad (3.31)$$

The tr operator represents the trace operator, determined by the summation of the diagonal elements in the provided matrix.

3.6.2 Evaluation Metrics

Metric	Description	Equation
rmse	Root Mean Squared Error (<i>RMSE</i>)	$\sqrt{\frac{\sum_{i=0}^N \ (p_i)\ ^2}{N}}$
mean	Mean Error (μ)	$\frac{\sum_{i=0}^N p_i }{N}$
median	Median Error	$\frac{[(\frac{N}{2})^{th} \text{ term} + (\frac{N}{2} + 1)^{th} \text{ term}]}{2}$
std	Standard Deviation in Error Values (σ)	$\sqrt{\frac{\sum_{i=0}^N (p_i - \mu)^2}{N}}$
min	Minimum Error	$\min(p)$
max	Maximum Error	$\max(p)$
sse	Sum Squared Errors	$\sum_{i=0}^N (p_i)^2$

Table 3.6: Evaluation Metrics Descriptions and Equations, where N is the number of poses with common timesteps between the estimated poses and ground truth. p_i is the pose difference at the i^{th} index of timestep correspondences.

- **rmse (Root Mean Squared Error)**: RMSE quantifies the square root of the average of the square differences between the predicted and actual values. It measures the standard deviation of prediction errors, representing how far from the regression line data points are.
- **mean (Mean Error)**: The Mean Error calculates the average error made by the model across all predictions, indicating the central tendency of the errors.
- **median (Median Error)**: The Median Error finds the middle value of error when all errors are arranged in ascending order. For an even number of observations, it is the average of the two middle numbers.
- **std (Standard Deviation in Error Values)**: This metric measures the dispersion of error values from their mean, indicating the variability of the errors.
- **min (Minimum Error)**: This metric identifies the smallest error made by the model, indicating the best prediction accuracy.
- **max (Maximum Error)**: Conversely, the Maximum Error indicates the largest error in the model's predictions, highlighting the worst prediction accuracy.
- **sse (Sum Squared Errors)**: SSE sums up the square of all errors, providing a cumulative measure of the error magnitude, useful for overall model performance assessment.

3.6.3 Evaluation and Visualization Library

The EVO library [53] serves as a pre-existing framework tailored for the evaluation of odometry and SLAM trajectories. This tool is not only equipped with a variety of

CHAPTER 3. METHODOLOGY

metrics, as detailed in Section 3.6.2, but also excels in producing detailed visualizations. Specifically, EVO generates a global graph that integrates the results from all evaluated methods, providing a comprehensive comparison of their performance. Furthermore, for each individual method, EVO creates a unique graph that includes a heatmap representation. This effectively showcases the distribution of errors across the traversal path, thereby illustrating the localized error patterns unique to each method. Through this dual approach to visualization, the EVO library facilitates a deeper insight into the comparative performance of different methods and the specific error dynamics of each. Consequently, the EVO library is instrumental in computing performance metrics and generating visualization graphs.

3.7 ORB-SLAM3 and HF-Net SLAM Modifications

In both frameworks, the internal mechanisms remain consistent, with the loop closure thread being deactivated. In the HF-Net-SLAM framework, the global descriptor is a deep-learned descriptor, similar to the feature vector discussed in Section 3.2. This descriptor facilitates the detection of similar images. Subsequently, feature matches between these similar images are utilized to establish loop closures or to generate correspondences for pose optimization. The highest-scoring descriptors are merged to create a composite value, serving as the global descriptor, until it achieves the required length.

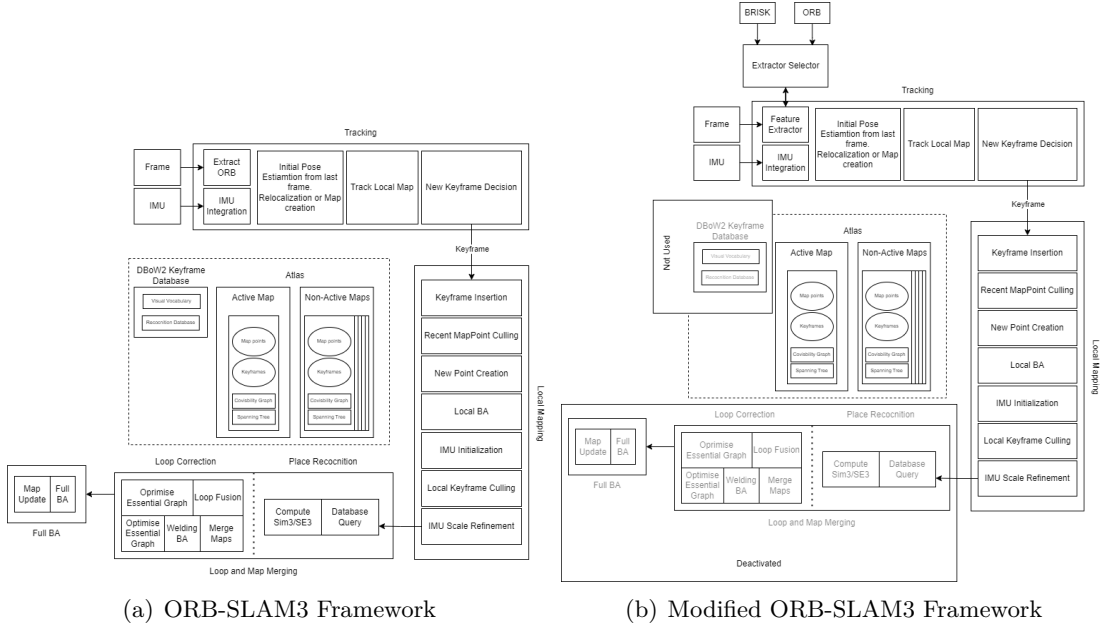


Figure 3.18: Modifications in the ORB-SLAM3 Framework

3.8. SUMMARY

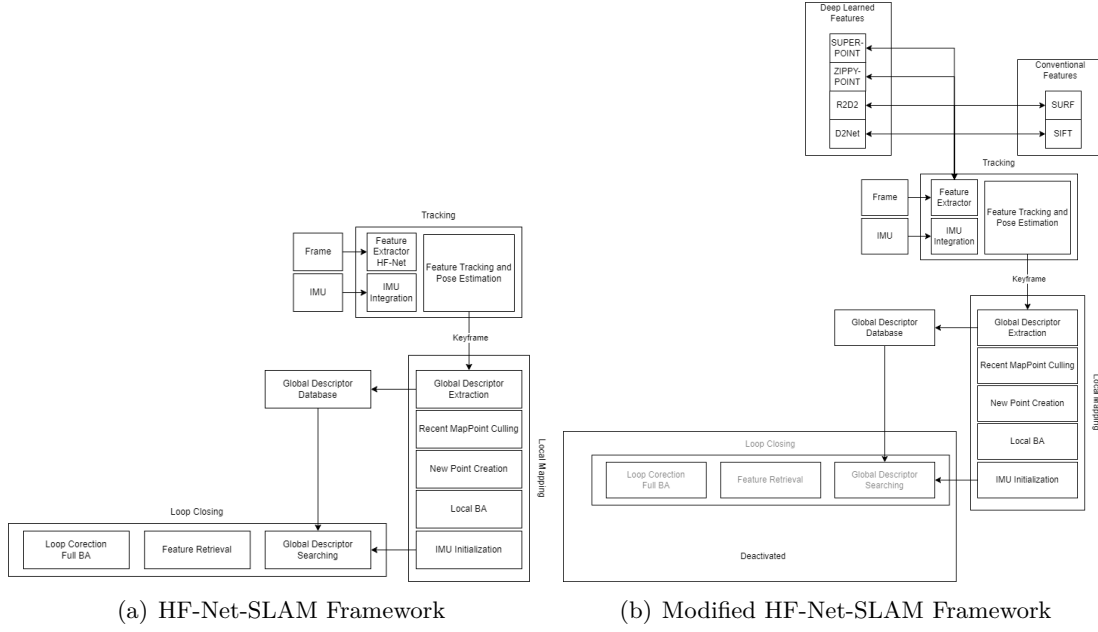


Figure 3.19: Modifications in the HF-Net-SLAM Framework

3.8 Summary

This chapter presents an in-depth exploration of the methodologies and technologies fundamental to our research in **SLAM**, with a particular emphasis on the latest advancements within this area. Initially, we introduce the data acquisition process, facilitated by an autonomous exploration algorithm detailed in Section 3.1. This process is essential for the assessment of various **SLAM** techniques. Subsequently, we investigate the **DBoW** place recognition system, outlined in Section 3.2, which plays a crucial role in location identification within the **SLAM** framework.

The discussion progresses to an examination of diverse **SLAM** methodologies, as elaborated in Section 3.3, where we delve into the algorithms that are at the forefront of innovation in the field. The narrative then shifts to the importance of visual features in **SLAM**, discussed in Section 3.4, underscoring their significance in environmental mapping and navigation. Although the primary emphasis is on visual data, the relevance of **LiDAR**-based odometry for enhancing reference accuracy is also acknowledged in Section 4.5.

Furthermore, the chapter discusses the evaluation methods utilized by the EVO library for assessing **SLAM** techniques, as detailed in Section 3.6. These methods include timestep alignment, pose map alignment, scaling, and various evaluation metrics employed for comprehensive comparisons.

CHAPTER 3. METHODOLOGY

In concluding the chapter, we highlight the modifications applied to ORB-SLAM, as described in Section 3.4, to integrate a diverse array of visual features, as mentioned in Section 3.7. These adaptations underscore our contributions towards the advancement of SLAM technologies. Overall, this chapter aims to offer a thorough overview of the critical components of our research, shedding light on the cohesive approach that drives our work in SLAM. Through a meticulous examination of each segment, we aspire to provide insights into the complexity and ingenuity propelling the evolution of this field.

Chapter 4

Results and Interpretation

In this chapter, we explore the evaluation of **SLAM** methods on lightweight processing platforms and their performance on the renowned EUroC dataset to ensure their viability and absence of critical flaws. We establish a baseline for localization accuracy by examining **LiDAR**-based methods on recorded datasets, which is considered adequate for subterranean exploration according to recent surveys. The chapter further delves into the localization accuracies achieved by all examined **SLAM** methods and features, utilizing both Monocular and RGB-D sensor configurations. Finally, we discuss and compare these results, offering insights into their implications for practical applications.

4.1 Data Trials Setup

4.1.1 Data Trials Overview

The data trials were designed to assess the performance of various visual feature extraction and description methods, integrated with the well-known ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method. These trials were conducted under controlled conditions to ensure that the methods are suitable for deployment on lightweight processing platforms, such as the ASUS FX550J laptop and the custom-built desktop computer described in Section 4.2.

4.1.2 Trial Setup

Each trial involved running the selected visual feature extractors and description methods, alongside **LDSO**, on the EUroC dataset. This dataset is a recognized benchmark for evaluating localization accuracy. The trials were conducted using two different platforms:

- **Lightweight Processing Device:** Chosen to simulate constrained processing conditions similar to those of the Jetson Nano.
- **Desktop Computer:** Used to handle methods that require higher computational resources.

CHAPTER 4. RESULTS AND INTERPRETATION

4.1.3 Evaluation Metrics

The performance of the visual feature extractors, description methods, and **LDSO** was evaluated using the following key metrics:

- **Mapping Duration:** The time required for each **SLAM** method to complete the mapping process on the dataset.
- **Absolute Pose Error (APE):** Measures the accuracy of the estimated poses relative to the ground truth. Key statistics include root mean square error (RMSE), mean error, median error, standard deviation, minimum, and maximum errors.
- **Relative Pose Error (RPE):** Evaluates the accuracy of the relative poses between consecutive frames. Similar statistics as APE are used for RPE.

4.2 Computing Platform Specifications

The processing platform comprises an ASUS FX550J laptop[112] featuring a Nvidia GTX 950m GPU[96] and an Intel Core i5-4200H CPU[108]. Additionally, a custom-built desktop computer is utilized, equipped with a Nvidia GTX 1080ti GPU[97] and an AMD Ryzen 5 3600 CPU[109]. These platforms are concisely compared in Table 4.1.

Specification	Laptop	Desktop
GPU		
Series	Nvidia GTX 950m	Nvidia GTX 1080ti
GPU Memory	2 GB	11 GB
CUDA Cores	640	3584
Core Base Speed	993 MHz	1481 MHz
Core Boost Speed	1124 MHz	1582 MHz
CPU		
Series	Intel Core i5-4200H	AMD Ryzen 5 3600
Cores	2	6
Threads	4	12
Core Base Speed	2.8 GHz	3.6 GHz
Core Boost Speed	3.4 GHz	4.2 GHz

Table 4.1: Comparison of Processing Platforms

The laptop was selected as a processing-constrained device due to its lower GPU memory (2 GB), which limits its capability to run resource-intensive models, particularly deep learning-based methods such as **R2D2** and D2Net. While the laptop can successfully run smaller models like ORB and BRISK, its limited VRAM prevented the execution of larger models. Consequently, the desktop with a higher-end GPU (11 GB of VRAM) was used for these models.

4.3. EUROC RESULTS ON MODIFIED VERSIONS

Additionally, the laptop’s reduced CUDA cores and slower clock speeds resulted in significantly longer processing times for methods that it could run. For instance, feature extraction and matching using **Superpoint!** (**Superpoint!**) on the laptop took approximately 2-3 times longer compared to the desktop. The disparity in performance became especially evident during operations requiring batch processing of frames, where the laptop frequently exhibited slowdowns and system memory bottlenecks.

Overall, the use of the desktop allowed for larger, more computationally expensive models to be run effectively, offering a better understanding of the trade-offs between performance and computational resource availability. This highlights the challenges faced when utilizing resource-constrained platforms like the laptop for advanced SLAM methods, especially those reliant on deep learning.

4.3 EUrOC Results on Modified Versions

Measurement	LDSO	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	SURF
SLAM Platform	-	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight
Mapping duration	161.95	138.35	138.05	137.95	5.15	138.2	138.4
APE							
rmse	0.933	0.041	0.042	0.065	0.197	0.066	0.041
mean	0.52	0.035	0.036	0.046	0.162	0.047	0.035
median	0.337	0.029	0.028	0.037	0.132	0.036	0.028
std	0.774	0.021	0.022	0.046	0.111	0.047	0.022
min	0.043	0.001	0.0	0.006	0.026	0.002	0.001
max	5.658	0.088	0.137	0.375	0.489	0.478	0.106
RPE							
rmse	0.17728	0.0308	0.0317	0.03263	0.06319	0.03384	0.0308
mean	0.15246	0.02695	0.02751	0.02738	0.02522	0.02751	0.02687
median	0.13907	0.02407	0.02454	0.02419	0.00915	0.02406	0.0237
std	0.09045	0.01489	0.01575	0.01775	0.05794	0.0197	0.01504
min	0.01152	0.00049	0.00121	0.00093	0.0	0.00109	0.00064
max	0.76752	0.10189	0.16001	0.37451	0.35138	0.48467	0.1252

Table 4.2: Monocular EUrOC MH-01-easy Results

In Table 4.3, we compare the performance of the modified **ORB** and HF-Net **SLAM** methods against **LDSO** and ORB-SLAM3, with all loop closure detection mechanisms deactivated. This comparison reveals a noticeable difference, attributed to the absence of loop closure detection, which delineates the results published for ORB-SLAM3 from those observed in this study. Despite not surpassing ORB-SLAM3 in performance on the EUROC dataset, the modified methods demonstrate comparable accuracy and outperform **LDSO** in terms of precision. The exclusion of Zippypoint, **R2D2**, and D2Net from this comparison is due to their requirement for RGB images, whereas the EUROC dataset comprises grayscale images.

These findings indicate that while the modifications did not enhance the performance of the framework, they did preserve accurate mapping capabilities, with the exception of the utilization of **ORB** features. This particular outcome can be ascribed to the elimination of

CHAPTER 4. RESULTS AND INTERPRETATION

techniques within ORB-SLAM3 that ensure a widespread distribution of features across the environment, rather than their aggregation. For other methods, the results align closely with those of ORB-SLAM3, whereas **LDSO** exhibits subpar localization accuracy. This discrepancy with **LDSO** can be explained by its inability to match features across widely separated frames and its bundle adjustment process not incorporating all these frames, unlike ORB-SLAM3.

4.4 Recorded Datasets Map Size

The simulation environment for the **DARPA** Subterranean Challenge [85] comprises eight cave environments labeled as cave_circuit_(01-08). However, environment 06 was excluded from the dataset generation due to a failed exploration attempt, and 08 was excluded only for the RGB-D sensor configuration. This decision was made because the environment generally fell beyond the range of the RGB-D sensor, and none of the RGB-D methods maintained mapping. In the latter case, the **UAV** executed a turnaround close to the cave wall during exploration.

To navigate through sections of each environment, the exploration algorithm outlined in Section 3.1 was employed. Random exploration persisted until reaching the target number of frames, with a preference for paths furthest from known positions. Upon achieving the target frame count, priority shifted to the path closest to the origin, emphasizing a return to the starting point instead of venturing further. This adjustment aimed to prevent excessively long explorations. This method ensures the generation of datasets within a reasonable time and distance, facilitating the evaluation of **SLAM** methods with varying visual features.

Dataset Name	Map Size ($x, m \times y, m \times z, m$)	duration [s]	Video Link
cave_circuit_01	100.026, m $\times$ 101.511, m $\times$ 7.567, m	279.5	[99]
cave_circuit_02	75.745, m $\times$ 50.212, m $\times$ 8.170, m	290.628	[100]
cave_circuit_03	241.138, m $\times$ 130.676, m $\times$ 35.261, m	508.248	[101]
cave_circuit_04	413.831, m $\times$ 264.042, m $\times$ 58.562, m	836.576	[102]
cave_circuit_05	197.065, m $\times$ 58.221, m $\times$ 26.812, m	397.904	[103]
cave_circuit_07	236.002, m $\times$ 244.061, m $\times$ 25.708, m	608.556	[104]
cave_circuit_08	153.376, m $\times$ 38.753, m $\times$ 12.117, m	296.816	[105]

Table 4.3: This table provides a concise summary of the datasets used in our **SLAM** research, including the dimensions of the maps explored (length x width x height in meters), the total time taken for exploration (in seconds), and links to corresponding videos showcasing the exploration process.

4.5. KISS ICP RESULTS ON RECORDED DATASETS

4.5 KISS ICP Results on Recorded Datasets

Measurement	cave-circuit-01	cave-circuit-02	cave-circuit-03	cave-circuit-04	cave-circuit-05	cave-circuit-07	cave-circuit-08
Mapping duration	264.0	274.8	493.3	821.1	382.1	593.6	262.1
APE							
rmse	0.601	0.403	1.164	2.916	0.841	3.336	1.041
mean	0.481	0.283	1.062	2.589	0.462	2.499	0.795
median	0.389	0.231	1.058	2.182	0.265	1.444	0.516
std	0.36	0.287	0.476	1.342	0.703	2.209	0.672
min	0.003	0.022	0.063	0.727	0.029	0.178	0.043
max	1.779	3.338	3.127	7.396	7.387	8.541	2.991
RPE							
rmse	0.08854	0.09212	0.08343	0.08331	0.10463	0.07199	0.08036
mean	0.06477	0.06156	0.06282	0.06574	0.06802	0.0561	0.05586
median	0.04633	0.04198	0.04834	0.05117	0.04527	0.04278	0.03593
std	0.06038	0.06852	0.05491	0.05117	0.0795	0.04511	0.05777
min	0.00338	0.00111	0.0009	0.00186	0.00071	0.00215	0.00182
max	1.22077	1.03424	1.001	0.80137	1.57626	0.42231	0.98505

Table 4.4: This table presents a detailed comparison of evaluation metrics across various **SLAM** datasets, with **KISS-ICP**[95] **LiDAR** Odometry serving as the baseline for satisfactory performance in subterranean exploration scenarios..

In Table 4.5, we present the results of applying **KISS-ICP**[95] **LiDAR** odometry to **LiDAR** data, establishing a reference for accuracy deemed sufficient in subterranean environments, as recognized by current research [82], [92].

These measurements of localization accuracy establish a benchmark, allowing us to compare the accuracies we obtained with those acknowledged in the literature as adequately precise for subterranean exploration [82], [92]. This comparison underscores the relevance of achieving and surpassing these baseline accuracies to ensure effective navigation and mapping in challenging subterranean environments.

4.6 Localization Accuracy's and Processing Times on Recorded Datasets

4.6.1 Dataset 1

The first dataset was recorded using the cave_circuit_01 environment of the **DARPA** Subterranean Challenge simulation environment. This dataset is unique as the return path to the start does not replicate the initial exploration path. Consequently, the focus is primarily on the capabilities of visual odometry, rather than the recall of previous keyframes. The visual data captured during this exploration can be viewed in a video available at [99]. Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.1.

CHAPTER 4. RESULTS AND INTERPRETATION

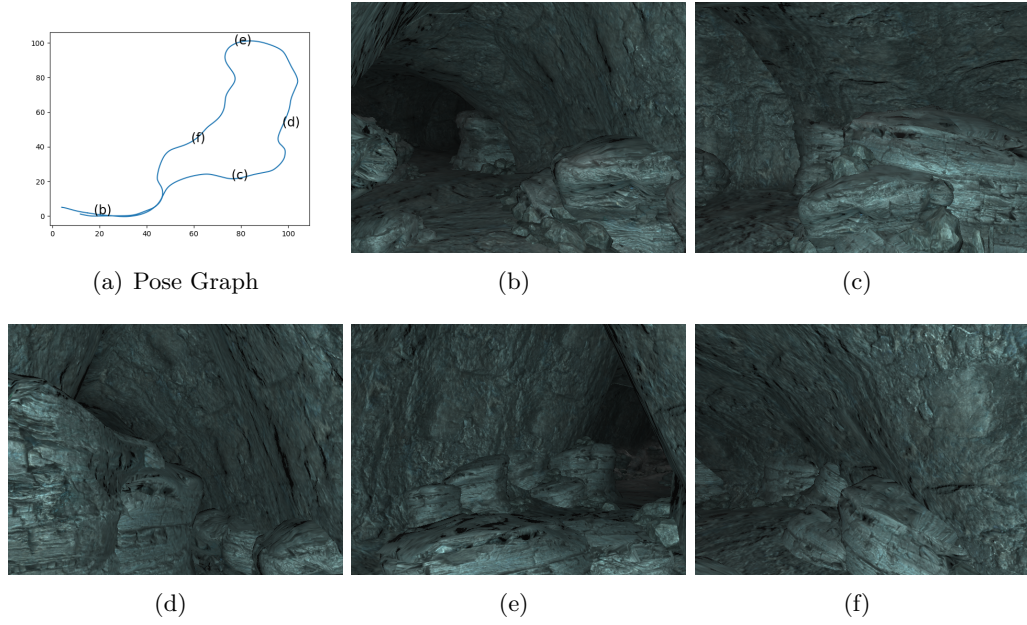


Figure 4.1: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey's path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure- ment	ZIPPY- POINT	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet- SLAM	-	-	M/ORB- SLAM3	M/HFNet- SLAM	M/ORB- SLAM3	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	257.704	264.104	258.156	251.004	258.632	255.356	253.484	256.628	254.512	258.776
APE										
rmse	2.394	1.791	1.741	3.031	1.044	7.306	2.068	6.749	2.72	23.486
mean	1.81	1.4	1.302	2.268	0.783	5.539	1.631	4.495	2.231	18.97
median	1.163	1.015	0.733	1.249	0.49	3.307	1.196	2.575	1.541	15.571
std	1.566	1.117	1.155	2.012	0.691	4.764	1.272	5.034	1.556	13.847
min	0.027	0.03	0.043	0.171	0.015	0.477	0.051	0.314	0.157	0.621
max	6.177	5.399	4.8	8.336	2.844	19.806	4.826	26.247	5.377	64.251
RPE										
rmse	0.10732	0.33803	0.10714	0.1071	0.10716	0.10686	0.10827	0.12281	0.10859	0.09748
mean	0.09803	0.29597	0.09792	0.09781	0.0978	0.09773	0.09966	0.10149	0.0995	0.08757
median	0.10365	0.25705	0.10349	0.10318	0.10413	0.1006	0.10449	0.10363	0.10436	0.08531
std	0.04367	0.1633	0.04348	0.04364	0.0438	0.04322	0.04233	0.06914	0.04349	0.04282
min	0.0	0.03297	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.37401	1.19444	0.36467	0.36847	0.36656	0.38118	0.36955	2.50687	0.47362	0.29825

Table 4.5: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping.

CHAPTER 4. RESULTS AND INTERPRETATION

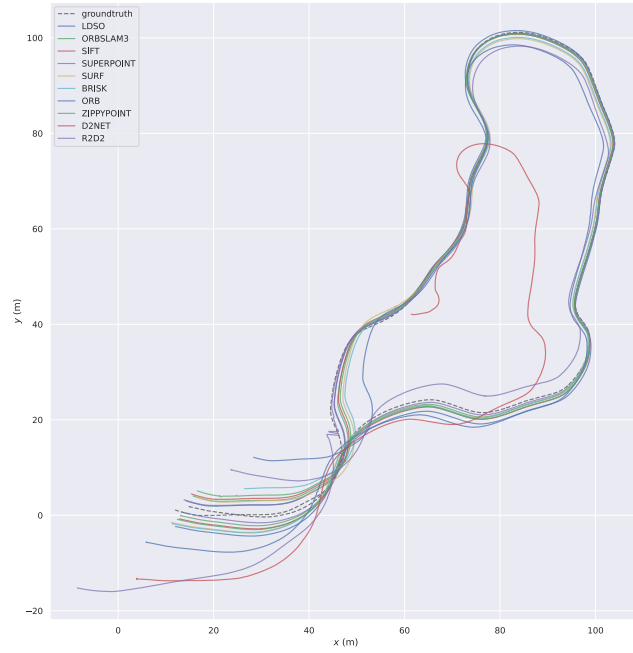


Figure 4.2: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	68.32398	29.8674	25.4021	137.4938	21.23693	111.30224	157.35848
median	11.75185	28.27055	22.71355	137.227	20.58505	111.4995	156.686
std	82.44419	9.24906	11.93443	6.85209	7.69159	14.40911	33.30675
min	4.34224	18.0857	7.27216	119.024	7.72482	70.8914	51.9988
max	294.733	156.405	91.8383	177.344	409.08	171.686	566.299

Table 4.6: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

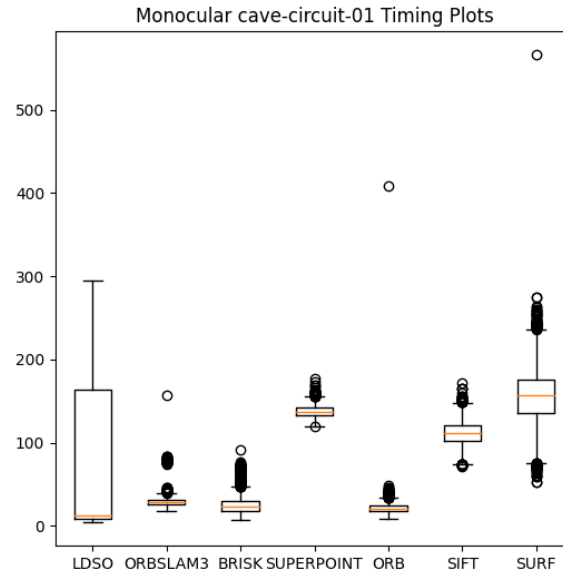


Figure 4.3: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measure- ment	ZIPPY- POINT	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet- SLAM	-	M/ORB- SLAM3	M/HFNet- SLAM	M/ORB- SLAM3	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	264.0	264.0	258.924	264.0	264.0	264.0	264.0	264.0	222.008
APE									
rmse	0.715	0.848	0.866	0.908	1.381	0.858	1.002	1.011	2.304
mean	0.589	0.687	0.708	0.724	1.119	0.702	0.801	0.803	1.423
median	0.449	0.545	0.587	0.478	0.776	0.532	0.563	0.569	1.013
std	0.405	0.497	0.5	0.548	0.809	0.493	0.602	0.614	1.812
min	0.036	0.057	0.044	0.028	0.183	0.036	0.073	0.059	0.088
max	1.576	2.079	1.989	2.015	3.093	1.977	2.288	2.209	13.623
RPE									
rmse	0.1073	0.10736	0.10781	0.10728	0.10827	0.10731	0.10746	0.10722	0.106
mean	0.097	0.09705	0.09711	0.09695	0.09774	0.09698	0.09709	0.09691	0.09495
median	0.10437	0.10411	0.10397	0.10408	0.10386	0.10442	0.10421	0.10437	0.10145
std	0.04588	0.04591	0.04681	0.04592	0.04656	0.04594	0.04605	0.04588	0.04713
min	0.00025	0.00017	0.00015	0.00029	6e-05	0.00014	0.0001	9e-05	0.00033
max	0.40188	0.39848	0.4012	0.405	0.38405	0.39352	0.40409	0.39886	0.41563

Table 4.7: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

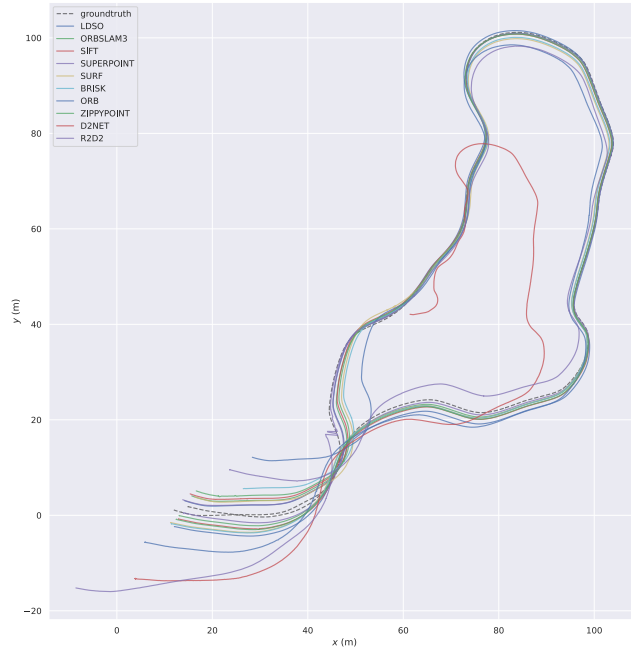


Figure 4.4: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	42.55625	27.00005	138.34912	25.95366	113.43612	161.90549
median	41.3066	24.1538	137.862	24.8854	112.316	160.214
std	6.45039	10.73897	6.76326	6.6232	15.58017	34.08526
min	28.5573	8.30335	120.844	9.12528	72.3212	57.9272
max	72.4136	76.3639	203.805	155.339	170.465	313.628

Table 4.8: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

CHAPTER 4. RESULTS AND INTERPRETATION

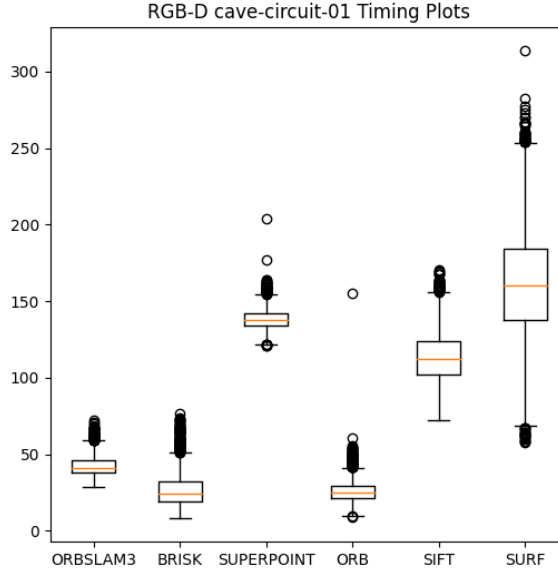


Figure 4.5: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms’ capability to perform in real-time environments and their potential impact on system resources.

Result Interpretation

Given the distinct exploration and return paths in this dataset, a decrease in localization accuracy compared to datasets with identical paths for exploration and return is anticipated. The latter scenario allows the system to use feature points matched between previous keyframes for localizing the latest frame, an advantage not present when different paths are traversed.

The percentage difference is calculated using the formula $\frac{|p_1 - p_2|}{\frac{p_1 + p_2}{2}}$. A 37.4% increase in localization accuracy is observed when comparing the best performances of Monocular and RGB-D sensor configurations. However, the improvement with Superpoint (the top performer in the Monocular (Table 4.5) category) is more modest at 13.9%. In stark contrast, the use of Zippypoint (the best in the RGB-D (Table 4.7) category) results in a 107.9% enhancement. This significant disparity in performance gains between Superpoint and Zippypoint may indicate that Zippypoint encounters more false feature matches in the Monocular sensor configuration. This could be attributed to

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

the RGB-D configuration's ability to measure relative x , y , and z point locations, while the Monocular configuration only determines x and y positions and relies on bundle adjustment to estimate the z or depth dimension, using data from multiple concurrent images. Furthermore, ORB-SLAM's match-by-projection approach might introduce false matches in initial point estimates within the Monocular setup, whereas the RGB-D scenario, being more constrained initially, is less susceptible to such inaccuracies. A similar trend is observed with **ORB** , **R2D2** , and D2Net features. The issues with **R2D2** and D2Net may stem from their training, which primarily includes well-lit images with a plethora of distinctive features.

The box-and-whisker plot reveals that most methods exhibit relatively low variability in processing speed, with notable exceptions for **LDSO** and **SURF**. For **LDSO**, the variability is due to the iterative nature of texture map alignment, which varies based on the number of iterations and specific conditions, leading to fluctuations in processing speed. In the case of **SURF**, the source of variability remains unclear without further investigation. This variability is observed in both monocular and RGB-D scenarios.

CHAPTER 4. RESULTS AND INTERPRETATION

4.6.2 Dataset 2

The second dataset was recorded in the `cave_circuit_02` environment of the **DARPA** Subterranean Challenge simulation environment. This dataset is unique in its size, being the smallest in terms of map coverage, yet it does not have the shortest exploration duration. The exploration is relatively brief and involves retracing the same path back to the start. This method ensures co-visibility is maintained with previous keyframes, thereby making the localization accuracy dependent on the ability to align the latest frame with keyframes in the co-visibility graph. The visual data from this exploration can be viewed in a video at [100]. Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.6.

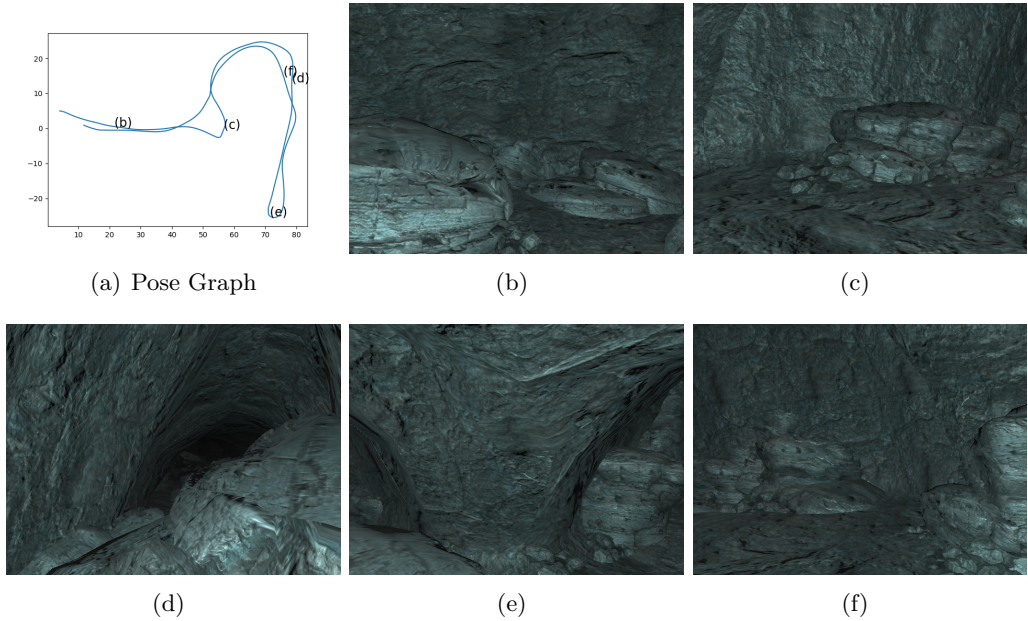


Figure 4.6: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey’s path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure-ment	ZIPPY-POINT	LDSO	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	269.0	274.788	268.084	261.36	269.752	244.968	263.652	267.248	264.796	127.5
APE										
rmse	0.869	0.572	0.821	1.161	1.045	2.205	0.94	12.772	1.167	1.623
mean	0.627	0.473	0.685	0.871	0.82	1.645	0.748	10.624	0.952	1.552
median	0.378	0.444	0.609	0.563	0.632	0.975	0.62	9.287	0.748	1.448
std	0.601	0.321	0.453	0.768	0.647	1.468	0.57	7.089	0.674	0.475
min	0.059	0.022	0.013	0.024	0.035	0.134	0.041	2.549	0.021	0.241
max	3.105	1.717	2.619	3.397	3.354	5.221	3.393	34.669	3.16	2.926
RPE										
rmse	0.09226	0.33087	0.09258	0.09203	0.09218	0.08691	0.09371	0.09724	0.09333	0.06896
mean	0.07581	0.27821	0.07611	0.07533	0.07551	0.06964	0.07743	0.07107	0.07703	0.0363
median	0.08242	0.25419	0.08289	0.08066	0.08229	0.07005	0.0838	0.07484	0.08368	0.01109
std	0.05258	0.1791	0.0527	0.05287	0.05287	0.05201	0.05278	0.06636	0.05269	0.05863
min	0.0	0.02086	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1e-05
max	0.52899	1.13616	0.52288	0.53937	0.52585	0.55571	0.51709	2.95324	0.52633	1.47078

Table 4.9: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping. **Blue** denotes incomplete mapping or the inability of a method to sustain mapping across the dataset, reflecting challenges in handling complex or sparse environments.

CHAPTER 4. RESULTS AND INTERPRETATION

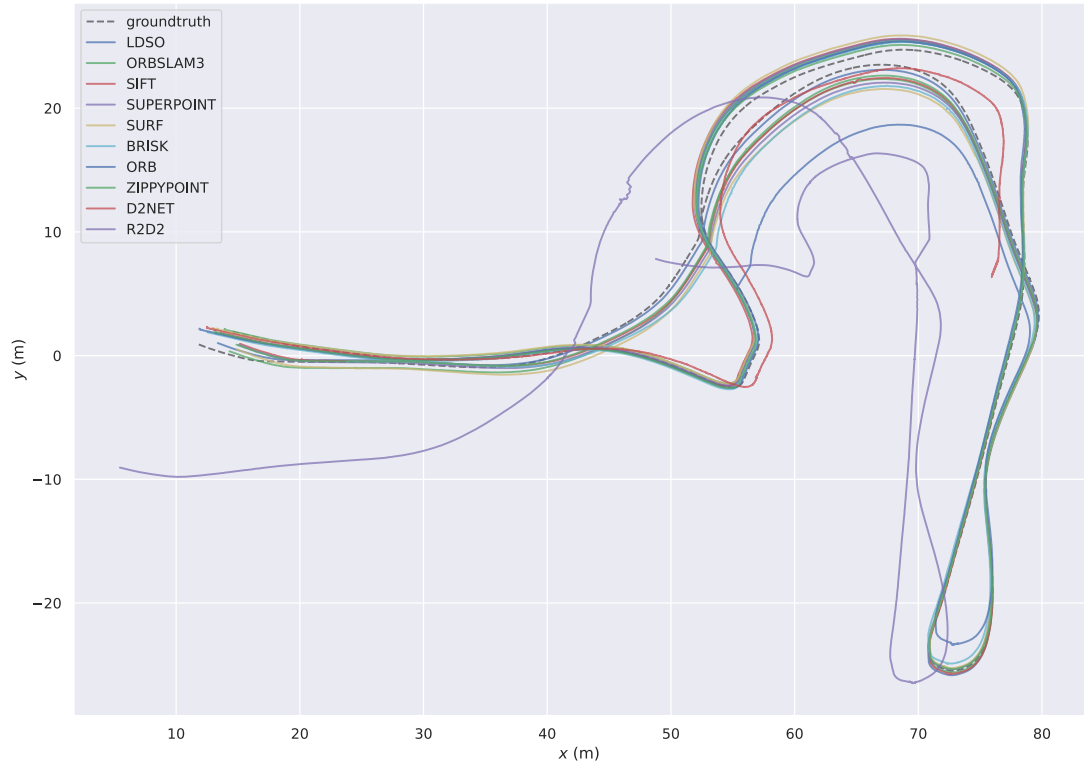


Figure 4.7: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	59.12172	29.46302	28.31572	135.90574	22.45934	112.57463	164.89987
median	10.6546	28.3396	24.793	135.212	21.7882	113.487	159.46
std	82.05702	7.91472	14.72147	6.69305	8.35711	16.46097	41.0516
min	4.2528	18.4295	7.66633	118.691	8.00634	72.3098	51.7304
max	283.738	155.546	94.7942	168.795	426.457	175.217	456.256

Table 4.10: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

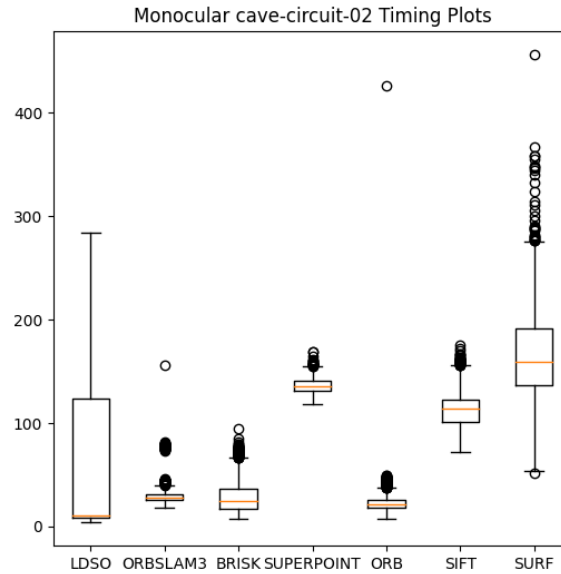


Figure 4.8: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measurement	ZIPPY-POINT	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	274.836	274.836	269.536	274.836	270.26	274.836	274.836	274.836	274.836
APE									
rmse	0.341	0.392	0.361	0.35	0.429	0.376	0.384	0.373	0.647
mean	0.333	0.361	0.341	0.336	0.369	0.354	0.357	0.346	0.585
median	0.34	0.325	0.317	0.315	0.328	0.312	0.346	0.329	0.55
std	0.072	0.153	0.119	0.098	0.219	0.128	0.14	0.138	0.278
min	0.16	0.041	0.095	0.124	0.135	0.135	0.051	0.064	0.097
max	0.572	0.81	0.659	0.672	1.95	0.909	0.765	0.772	1.303
RPE									
rmse	0.0918	0.0919	0.09178	0.09184	0.09218	0.09189	0.09174	0.09184	0.09287
mean	0.07454	0.07464	0.07413	0.07454	0.07455	0.07455	0.07445	0.07454	0.07559
median	0.08162	0.08186	0.08019	0.08183	0.08016	0.0818	0.08165	0.08185	0.08144
std	0.05359	0.05361	0.0541	0.05364	0.05422	0.05372	0.0536	0.05365	0.05395
min	0.00029	0.00011	0.0001	0.00028	0.00041	0.00014	8e-05	0.00022	0.00059
max	0.4763	0.47196	0.47083	0.48072	0.47947	0.47824	0.49023	0.46869	0.4256

Table 4.11: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping. **Blue** signals incomplete mapping or a method’s failure to sustain mapping continuity throughout the dataset, indicating challenges in managing complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

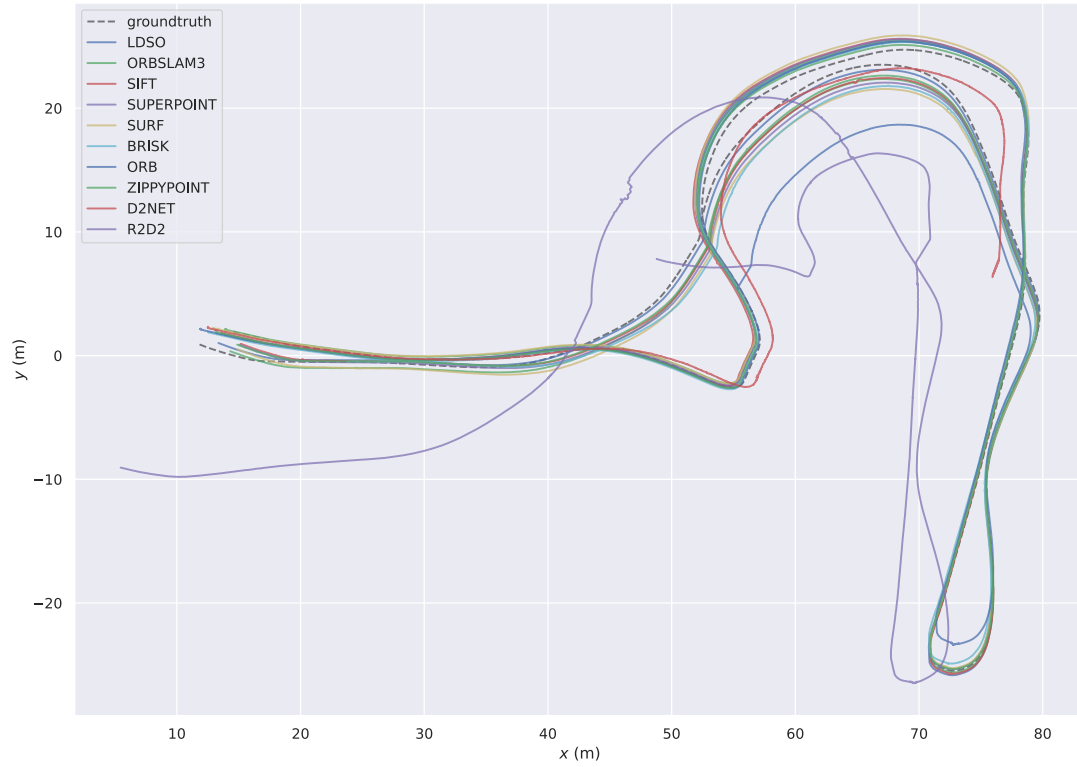


Figure 4.9: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

CHAPTER 4. RESULTS AND INTERPRETATION

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	43.39983	34.44	137.71604	26.38811	115.20312	161.75615
median	42.3395	38.64745	136.932	25.8737	114.765	159.4925
std	7.15994	9.20433	6.16814	8.42492	15.81421	36.94066
min	27.5563	12.6764	122.176	10.2881	77.6891	60.8409
max	88.6589	64.7969	174.335	416.881	177.374	300.161

Table 4.12: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

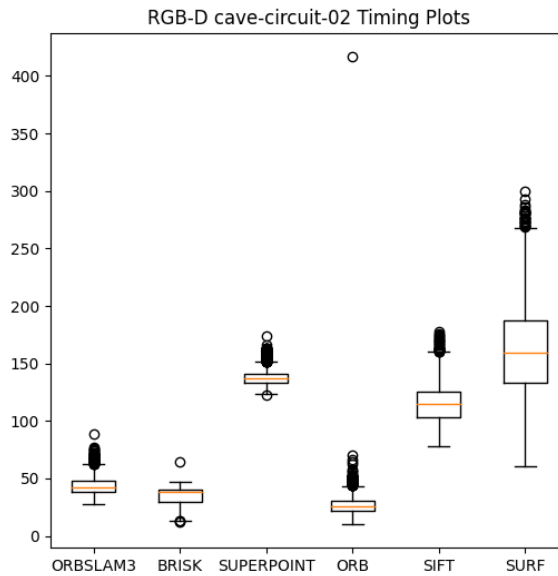


Figure 4.10: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Result Interpretation

A 58.2% increase in localization accuracy is observed when comparing the best performances of Monocular and RGB-D sensor configurations in Tables 4.9, and 4.11. However, a distinct difference emerges with Zippypoint (the best in the RGB-D (Table 4.11) category), where a 93.8% improvement is noted. This aligns with findings presented in Section 4.6.1. The smaller discrepancy may be attributed to revisiting certain keyframes, which imposes translational constraints between points and potentially corrects previously false feature point matches.

The inability to maintain mapping with D2Net features in Monocular sensor configurations arises at a point in the dataset where the UAV enters a crossing and executes a turn, seen in the video available at [100], accompanied by a rocking motion as it slowly advances. This scenario is plausible in real-world applications. Interestingly, D2Net exhibited the poorest localization accuracy in RGB-D sensor configurations. This issue could be linked to the segment mentioned above, where the camera is in close proximity to a wall and facing it. Similar to earlier discussions about false feature matches, the camera's closeness to the wall might allow these erroneous matches to fall within the point position uncertainty under these conditions, potentially causing D2Net features to lose mapping.

CHAPTER 4. RESULTS AND INTERPRETATION

4.6.3 Dataset 3

The third dataset was recorded using the `cave_circuit_03` environment from the **DARPA** Subterranean challenge simulation environment. The visual data captured during this exploration are presented in a video, which can be viewed at [\[101\]](#). Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.11.

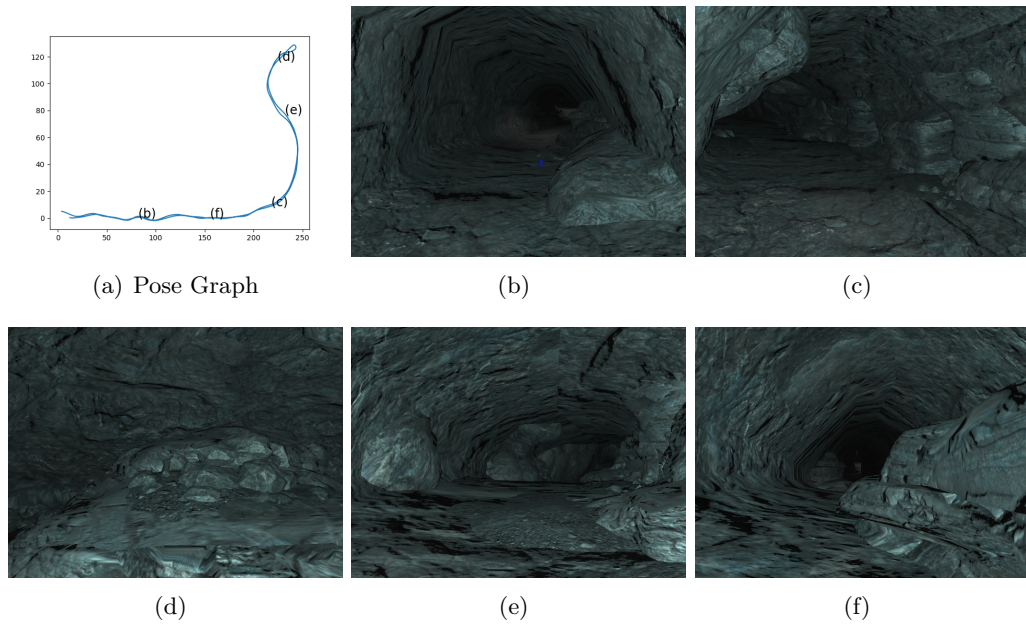


Figure 4.11: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey’s path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure-ment	ZIPPY-POINT	LDSO	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	481.6	493.296	485.832	135.652	484.848	484.444	481.7	485.044	482.096	483.7
APE										
rmse	14.206	5.579	7.657	1.309	8.973	35.573	15.169	22.947	21.463	68.314
mean	10.882	3.075	5.63	1.161	6.758	29.076	11.201	17.889	16.564	56.161
median	6.61	1.212	4.169	1.214	4.443	21.882	5.522	11.224	9.88	48.529
std	9.132	4.655	5.19	0.606	5.902	20.494	10.229	14.373	13.649	38.894
min	1.714	0.062	0.157	0.105	0.43	10.734	1.962	4.988	3.989	1.399
max	37.526	39.162	21.214	3.149	23.476	102.84	41.09	63.765	59.434	181.469
RPE										
rmse	0.11812	0.20432	0.11864	0.08365	0.11976	0.11086	0.11839	0.11568	0.11653	0.10171
mean	0.1124	0.15814	0.11263	0.05727	0.11325	0.10412	0.11247	0.10925	0.11058	0.09414
median	0.11873	0.12553	0.11974	0.00796	0.11941	0.10905	0.11853	0.11564	0.11664	0.09267
std	0.03633	0.12937	0.03729	0.06097	0.03893	0.03807	0.03697	0.03805	0.03677	0.0385
min	0.00046	0.00632	0.00047	0.0	0.00244	0.0005	0.00022	0.0004	0.0005	0.00166
max	0.30132	1.42627	0.30713	0.28291	0.6676	0.29584	0.33804	0.67462	0.30742	0.31289

Table 4.13: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping. **Blue** denotes incomplete mapping or the inability of a method to sustain mapping across the dataset, reflecting challenges in handling complex or sparse environments.

CHAPTER 4. RESULTS AND INTERPRETATION

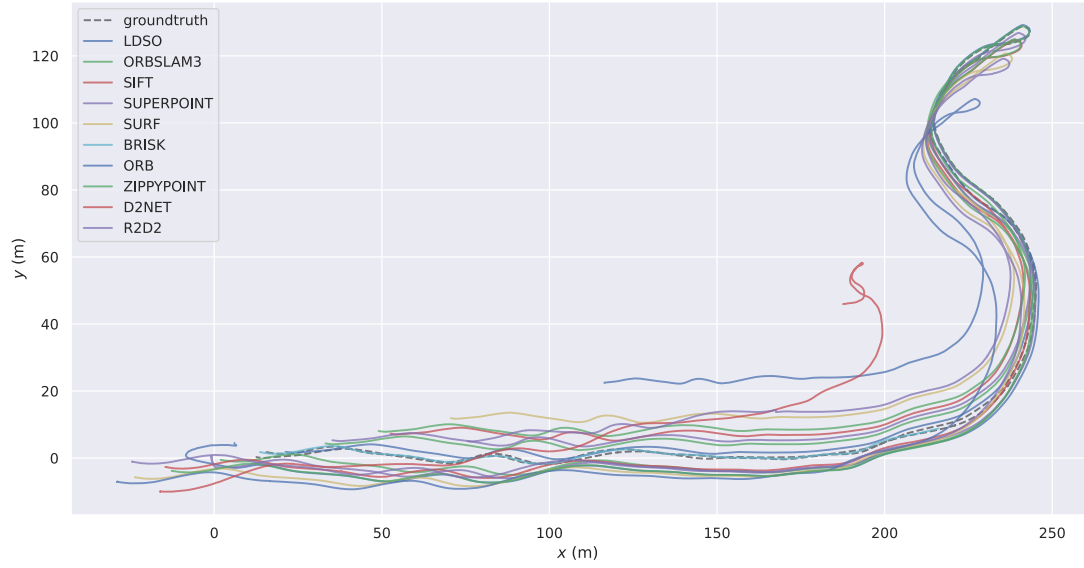


Figure 4.12: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison..

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	122.59954	28.67699	19.96632	144.48865	19.46297	107.89534	152.27402
median	152.783	27.5692	19.1706	144.166	18.6241	107.903	152.466
std	75.21113	7.3905	7.71992	8.26389	6.36769	13.84072	30.39524
min	4.36202	17.5882	7.00312	116.411	8.51744	67.042	48.0682
max	271.521	156.088	56.7315	179.818	408.425	175.038	476.383

Table 4.14: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

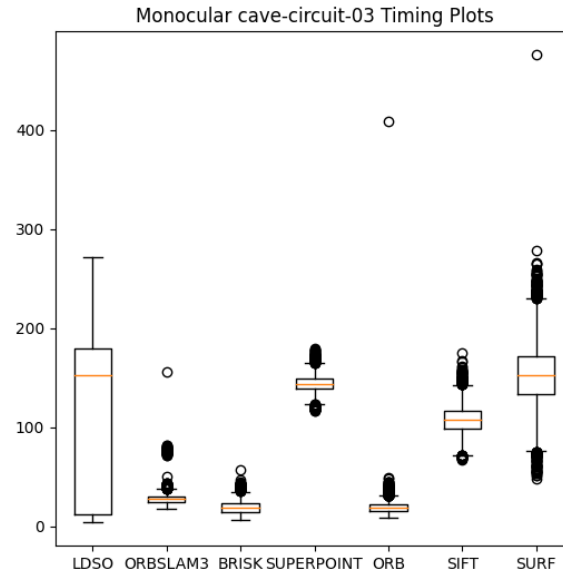


Figure 4.13: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measurement	ZIPPY-POINT	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	493.196	493.196	142.748	493.196	490.496	492.696	493.196	493.196	493.196
APE									
rmse	0.964	0.973	0.235	0.94	1.233	0.619	1.194	1.013	1.743
mean	0.844	0.848	0.218	0.852	1.062	0.598	1.029	0.897	1.536
median	0.765	0.662	0.219	0.955	0.857	0.592	0.888	0.903	1.347
std	0.464	0.476	0.089	0.398	0.628	0.163	0.606	0.469	0.823
min	0.07	0.062	0.028	0.131	0.059	0.094	0.122	0.169	0.236
max	1.718	1.928	0.585	1.701	2.872	1.044	2.355	2.174	3.683
RPE									
rmse	0.11835	0.11836	0.11527	0.11832	0.11888	0.11852	0.11838	0.11872	0.11952
mean	0.11147	0.11146	0.10593	0.11146	0.11183	0.11157	0.11143	0.11159	0.11255
median	0.11941	0.11952	0.11681	0.11945	0.11985	0.11982	0.11937	0.1194	0.12003
std	0.03977	0.03981	0.04546	0.03972	0.04032	0.03998	0.03996	0.04052	0.04022
min	0.0003	0.00039	0.00044	0.00023	0.00049	0.00037	0.00027	0.00033	0.00046
max	0.31437	0.30972	0.29612	0.31079	0.33256	0.30853	0.30966	0.51522	0.31145

Table 4.15: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping. **Blue** signals incomplete mapping or a method’s failure to sustain mapping continuity throughout the dataset, indicating challenges in managing complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

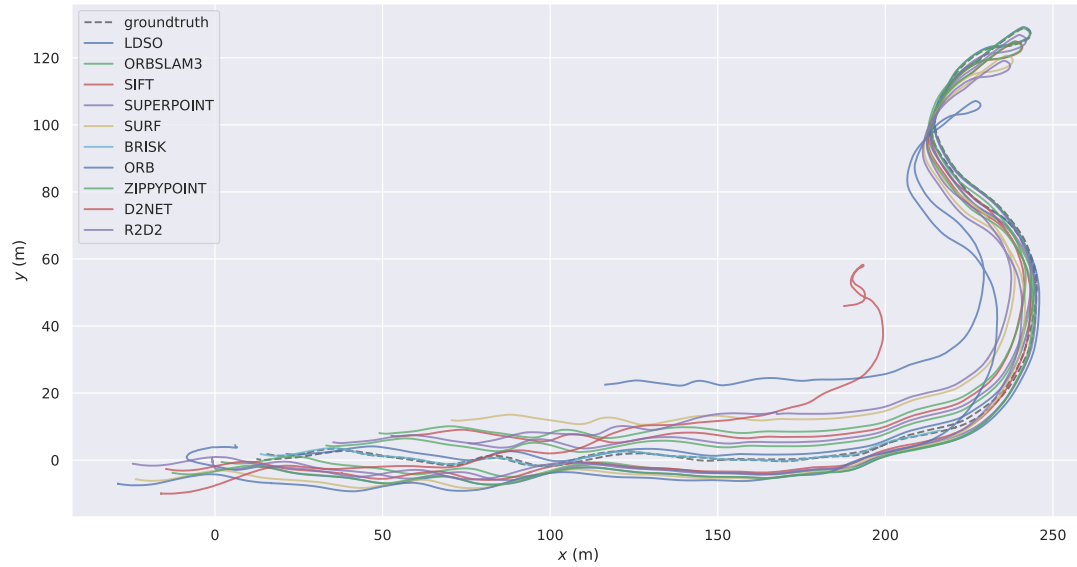


Figure 4.14: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

CHAPTER 4. RESULTS AND INTERPRETATION

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	41.66548	21.4602	145.38428	23.78875	109.40469	155.79184
median	40.35285	21.4126	145.188	22.76785	108.959	157.764
std	5.87884	7.01143	9.21793	7.22083	16.31919	33.01267
min	27.4542	7.12001	118.217	9.72062	67.6708	53.2194
max	72.294	47.2906	186.551	425.445	168.718	526.941

Table 4.16: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

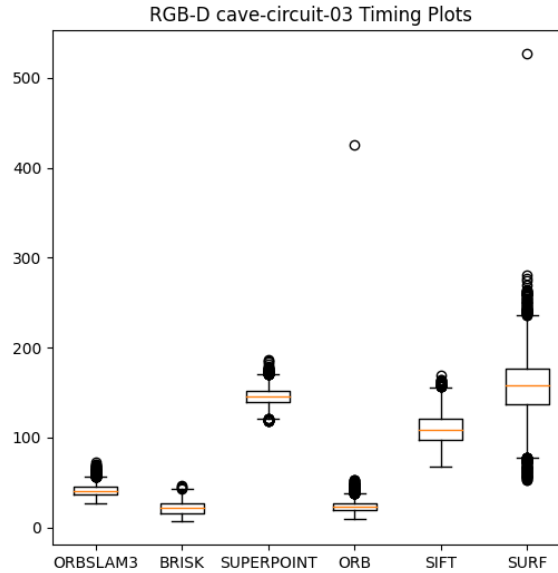


Figure 4.15: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Result Interpretation

An 160.1% increase in localization accuracy is observed when comparing the best performances of Monocular and RGB-D sensor configurations in Tables 4.13, and 4.15. Notably, with SIFT (the top performer in the RGB-D (Table 4.15) category), a 184.3% improvement is observed. Minimal differences are observed in the RPE table section between Monocular and RGB-D sensor configurations, indicating a slight discrepancy in VO error but substantial differences in absolute localization accuracies. This discrepancy is likely caused either by the enhanced point location constraints when using RGB-D compared to Monocular sensor configurations, or possibly in conjunction with a greater number of false feature matches leading to larger errors in pose graph optimization methods.

Upon reviewing the section where BRISK fails to maintain mapping in the video available at [101], no explicit reason emerges. A possible explanation is that although ORB and BRISK utilize the same keypoint detector and have somewhat similar intensity-based descriptors, ORB's descriptor is twice as long as BRISK's. This significantly limits the range of possible descriptors for BRISK. Coupled with the hypothesis that false point matches contribute to significant inaccuracies, the limited number of descriptors in BRISK could have been a substantial factor in its inability to maintain mapping in both Monocular and RGB-D sensor configurations. This theory is further supported by the observation that BRISK maintains mapping for a longer duration using RGB-D compared to Monocular sensor configurations

CHAPTER 4. RESULTS AND INTERPRETATION

4.6.4 Dataset 4

The fourth dataset was recorded in the `cave_circuit_04` environment of the **DARPA** Subterranean Challenge simulation environment. This dataset stands out for having the longest duration and the largest map size among the recorded datasets. It serves as a crucial tool for evaluating the ability to maintain accurate localization over extended periods and in large-scale maps. The visual data captured during this exploration can be viewed at [102]. Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.16.

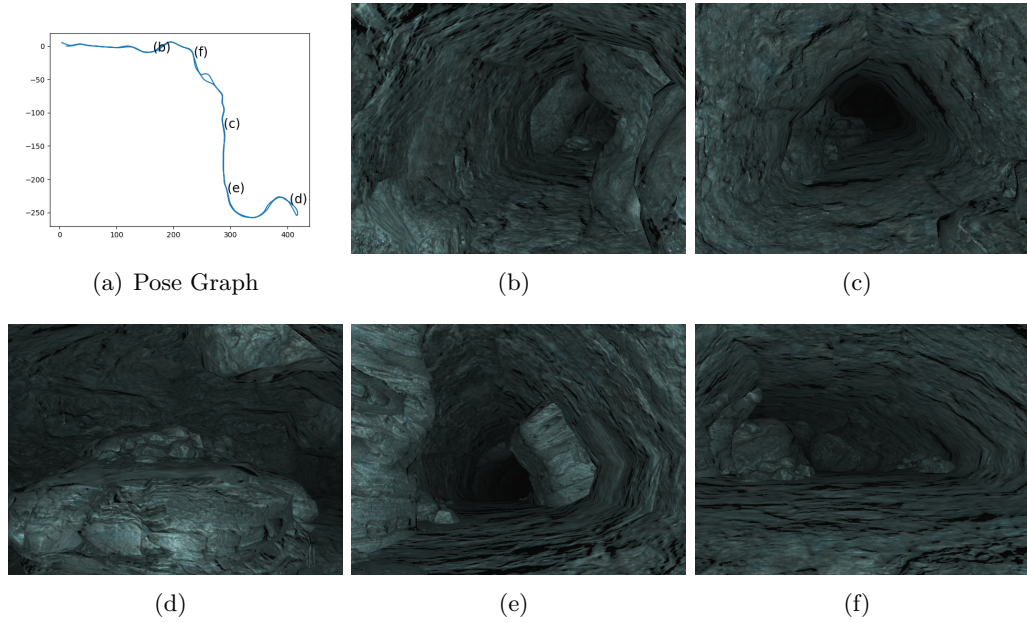


Figure 4.16: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey’s path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure-ment	ZIPPY-POINT	LDSO	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	811.6	821.152	813.2	234.976	812.048	811.5	806.44	811.4	810.3	809.264
APE										
rmse	43.984	16.45	20.484	2.514	29.029	88.323	41.795	63.547	66.057	120.088
mean	35.749	11.753	15.68	2.194	23.238	71.062	32.894	51.247	53.99	101.085
median	23.518	7.56	10.677	2.398	16.683	60.894	20.512	35.114	39.285	78.776
std	25.625	11.511	13.181	1.229	17.397	52.451	25.783	37.576	38.061	64.83
min	6.265	0.506	2.341	0.155	4.725	6.974	3.567	4.913	6.031	6.593
max	117.469	62.644	55.075	5.568	74.745	245.964	117.992	172.476	183.884	285.675
RPE										
rmse	0.12517	0.17004	0.12876	0.08633	0.12756	0.11541	0.1265	0.13016	0.12128	0.11674
mean	0.11623	0.14219	0.12006	0.0582	0.11884	0.10492	0.11762	0.11113	0.11171	0.10147
median	0.11414	0.12151	0.11793	0.00943	0.11693	0.09989	0.1151	0.10818	0.10833	0.0901
std	0.04645	0.09325	0.04654	0.06377	0.04633	0.04807	0.04655	0.06776	0.04721	0.05771
min	0.0	0.00619	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.31994	1.62944	0.33964	0.29791	0.33351	0.31199	0.3258	6.33334	0.32815	0.97759

Table 4.17: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping. **Blue** denotes incomplete mapping or the inability of a method to sustain mapping across the dataset, reflecting challenges in handling complex or sparse environments.

CHAPTER 4. RESULTS AND INTERPRETATION

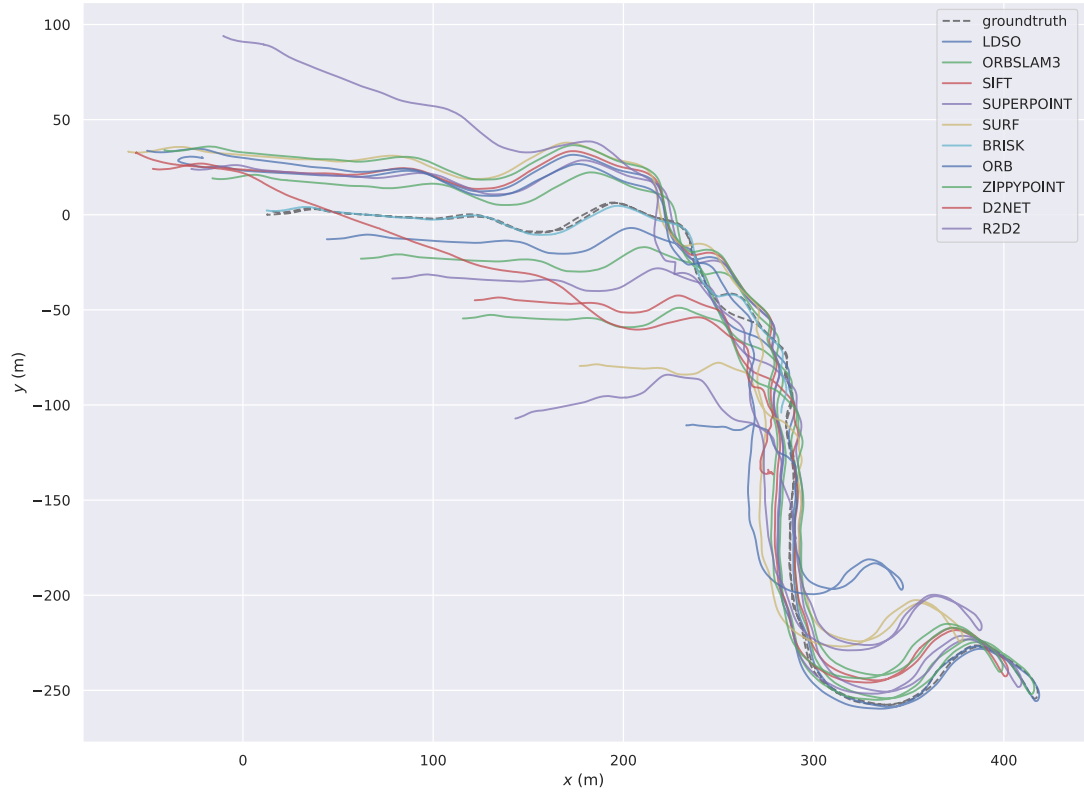


Figure 4.17: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	144.00349	28.09792	20.88697	143.50357	19.5337	107.31901	152.05633
median	161.824	27.154	19.3344	143.213	18.6233	107.022	151.278
std	62.38809	6.30364	7.79931	7.36454	5.74557	13.46864	28.28073
min	4.1197	18.194	6.46496	117.652	8.22419	68.5485	47.6497
max	290.381	151.211	67.6718	181.194	420.264	188.137	472.12

Table 4.18: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

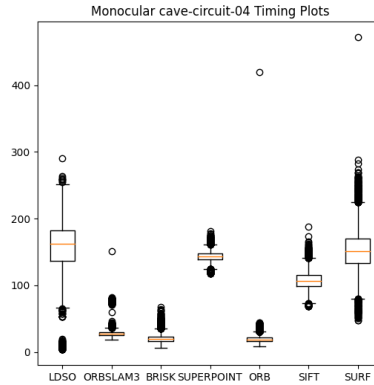


Figure 4.18: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measurement	ZIPPY-POINT	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	821.104	821.104	315.168	821.104	819.016	819.252	821.104	821.104	821.104
APE									
rmse	1.347	2.327	0.464	1.235	6.035	3.244	2.071	2.196	4.879
mean	1.168	1.84	0.396	1.046	3.877	2.471	1.835	1.671	3.883
median	0.994	1.413	0.336	0.84	2.042	1.691	1.896	1.127	2.682
std	0.669	1.424	0.242	0.656	4.624	2.103	0.959	1.424	2.954
min	0.173	0.189	0.016	0.331	0.287	0.248	0.082	0.124	0.283
max	3.602	5.756	1.123	3.458	24.458	8.668	3.986	5.386	13.053
RPE									
rmse	0.13259	0.13263	0.12553	0.13261	0.13312	0.1329	0.13255	0.13269	0.13357
mean	0.12325	0.12329	0.11617	0.12327	0.12372	0.12344	0.12317	0.12333	0.12419
median	0.11888	0.11887	0.1151	0.11895	0.11921	0.11901	0.11865	0.11897	0.11968
std	0.04888	0.04889	0.04758	0.04888	0.04913	0.04924	0.04899	0.04897	0.04917
min	0.00036	0.00037	0.00015	0.00073	0.0005	0.00029	0.00024	0.00013	0.00043
max	0.4018	0.39735	0.32898	0.44348	0.39498	0.52276	0.40421	0.39602	0.4173

Table 4.19: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping. **Blue** signals incomplete mapping or a method’s failure to sustain mapping continuity throughout the dataset, indicating challenges in managing complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

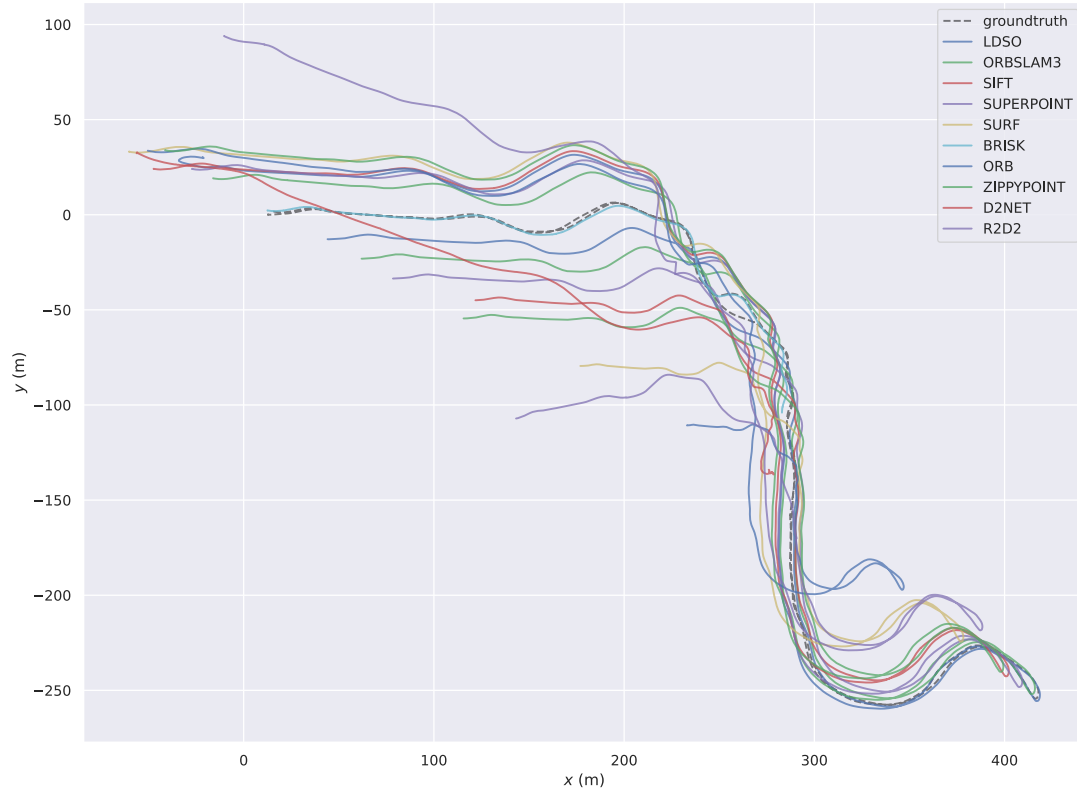


Figure 4.19: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

CHAPTER 4. RESULTS AND INTERPRETATION

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	40.831	22.00864	147.40644	23.59097	109.78693	159.22219
median	39.56835	20.8211	147.33	22.34155	108.704	158.597
std	5.92302	7.08245	7.55424	6.72	14.87923	30.97337
min	24.5909	7.64671	121.894	11.1905	70.565	54.58
max	70.1011	74.1094	194.234	438.742	179.859	466.84

Table 4.20: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

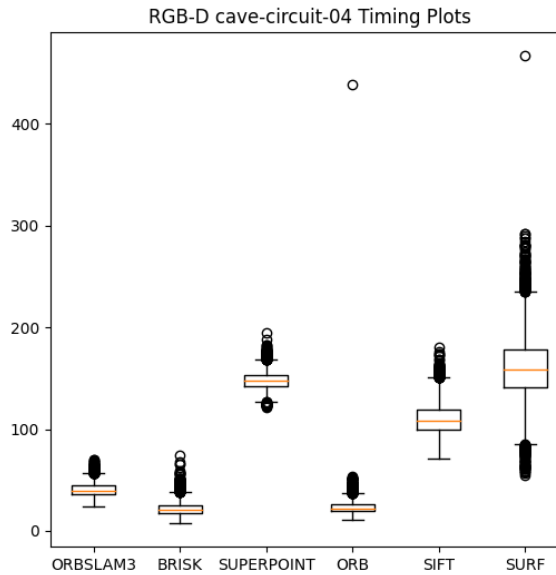


Figure 4.20: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Result Interpretation

An 172.0% increase in localization accuracy is noted when comparing the best performances of Monocular and RGB-D sensor configurations in Tables 4.17, and 4.19. In particular, with Superpoint (the top performer in the RGB-D (Table 4.19) category), a 183.7% improvement is observed. Although this represents a substantial increase in localization accuracy, the difference can be attributed to the size of the dataset, with the Monocular sensor configuration exhibiting less accurate VO compared to the RGB-D sensor configurations. This VO error accumulates across all images to the end of the dataset, resulting in significant pose errors.

Upon investigating the mapping failure with BRISK features, a conclusion similar to that in Section 4.6.3 is drawn. The limited descriptor space leads to an overwhelming number of false keypoint matches, ultimately resulting in the loss of mapping. This theory, akin to the findings in Section 4.6.3, is further substantiated by the observation that BRISK maintains mapping for a longer duration using RGB-D as opposed to Monocular sensor configurations.

CHAPTER 4. RESULTS AND INTERPRETATION

4.6.5 Dataset 5

The fifth dataset was recorded using the cave_circuit_05 environment from the **DARPA** Subterranean challenge simulation environment. The visual data captured during this exploration are presented in a video, which can be viewed at [\[103\]](#). Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.21.

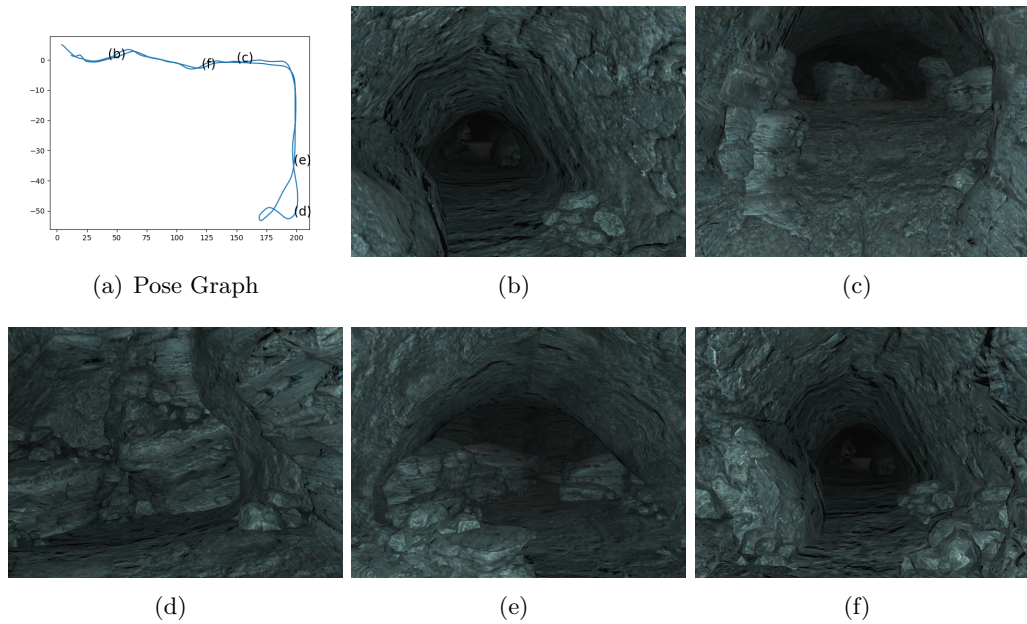


Figure 4.21: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey's path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure- ment	ZIPPY- POINT	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet- SLAM	-	-	M/ORB- SLAM3	M/HFNet- SLAM	M/ORB- SLAM3	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	374.332	382.2	375.048	371.796	375.9	371.0	370.6	374.104	376.548	378.884
APE										
rmse	9.208	4.212	5.468	9.906	3.868	23.72	7.486	16.292	11.447	52.51
mean	6.816	3.096	3.909	7.345	2.949	18.443	5.572	12.681	8.485	41.289
median	3.539	1.916	2.158	4.195	1.814	10.745	3.185	8.614	4.168	38.627
std	6.19	2.856	3.823	6.647	2.503	14.916	5.0	10.228	7.684	32.442
min	0.972	0.219	0.091	1.342	0.136	6.639	0.572	1.032	1.733	3.658
max	22.222	12.418	13.376	25.907	8.526	64.796	17.803	38.848	28.66	136.212
RPE										
rmse	0.11769	0.35915	0.11838	0.11755	0.11828	0.11328	0.11861	0.12991	0.11687	0.0979
mean	0.10705	0.32658	0.10763	0.10648	0.10746	0.10213	0.10802	0.10655	0.10589	0.08556
median	0.11127	0.32998	0.11183	0.11116	0.11201	0.10441	0.11225	0.10914	0.11058	0.08727
std	0.04889	0.14945	0.0493	0.04979	0.04942	0.04901	0.04899	0.07433	0.04946	0.04758
min	0.0	0.02207	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.34363	0.85697	0.34825	0.38779	0.34076	0.33796	0.34135	3.18132	0.33824	0.30377

Table 4.21: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping.

CHAPTER 4. RESULTS AND INTERPRETATION

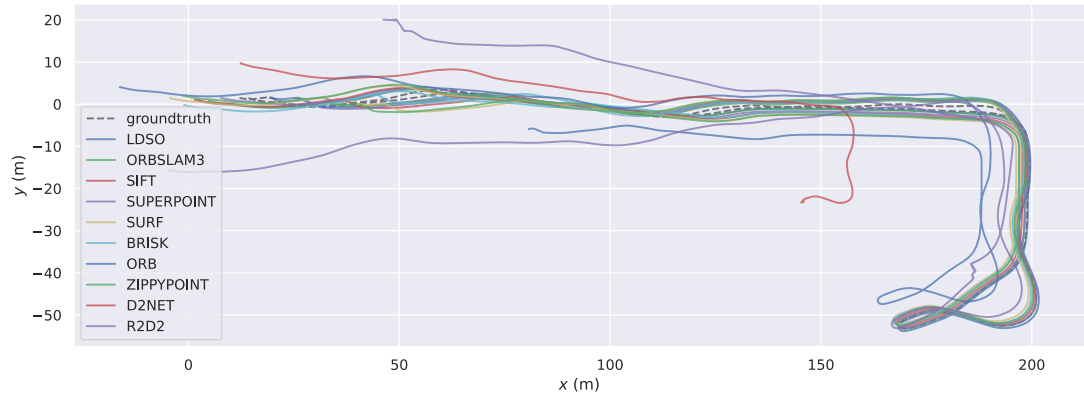


Figure 4.22: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	82.92384	29.7886	27.019	142.24349	21.40481	110.32894	162.31693
median	12.9044	28.2437	21.15135	142.067	20.322	111.053	158.086
std	87.80278	8.82785	18.11901	8.52738	7.6212	14.92729	40.25145
min	4.38382	18.9431	6.63682	117.203	8.58899	68.8986	45.5747
max	284.732	165.817	137.066	183.532	433.169	166.939	560.165

Table 4.22: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

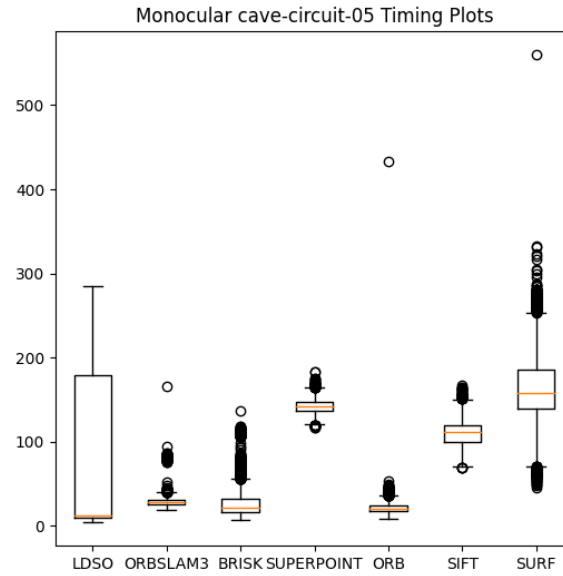


Figure 4.23: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measurement	ZIPPY-POINT	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	382.132	382.132	144.204	382.132	379.652	380.2	382.132	382.132	231.052
APE									
rmse	1.321	1.967	0.455	1.035	2.239	1.57	1.222	1.815	0.711
mean	1.147	1.732	0.403	0.91	1.897	1.363	1.06	1.588	0.653
median	1.053	1.77	0.351	0.786	1.909	1.44	1.055	1.586	0.651
std	0.654	0.932	0.211	0.492	1.19	0.779	0.607	0.879	0.281
min	0.106	0.1	0.034	0.117	0.062	0.086	0.025	0.168	0.112
max	2.491	3.279	0.92	2.08	5.307	2.805	2.201	3.095	1.268
RPE									
rmse	0.1206	0.12073	0.11959	0.12061	0.1219	0.1207	0.1205	0.12058	0.11267
mean	0.1088	0.10893	0.10606	0.10879	0.1095	0.10884	0.10867	0.10879	0.09953
median	0.11278	0.11275	0.11149	0.11262	0.11284	0.11292	0.11259	0.11272	0.10614
std	0.05203	0.05205	0.05525	0.05208	0.05357	0.05218	0.05207	0.052	0.05281
min	5e-05	0.00017	0.00021	0.00028	0.00032	0.00023	0.00017	0.0003	0.00013
max	0.34478	0.33607	0.30151	0.34167	0.76366	0.3377	0.33687	0.34697	0.30779

Table 4.23: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping. **Blue** signals incomplete mapping or a method's failure to sustain mapping continuity throughout the dataset, indicating challenges in managing complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

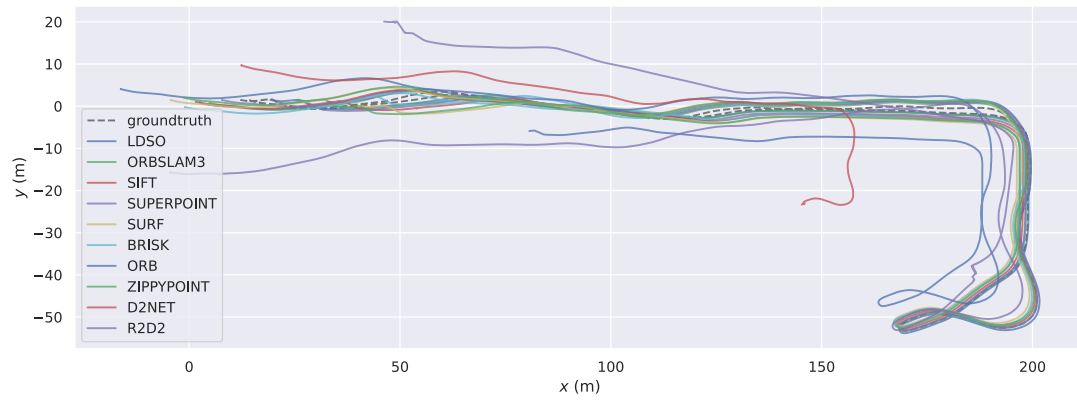


Figure 4.24: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

CHAPTER 4. RESULTS AND INTERPRETATION

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	42.02536	25.81079	141.63449	25.0835	115.36495	166.90525
median	40.85895	22.0665	141.2085	23.7685	113.481	166.5675
std	6.61016	11.98714	8.94531	6.6084	16.08921	38.59314
min	26.3241	8.84289	117.726	9.41394	71.799	47.992
max	90.99	83.3715	200.829	174.254	175.591	334.562

Table 4.24: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

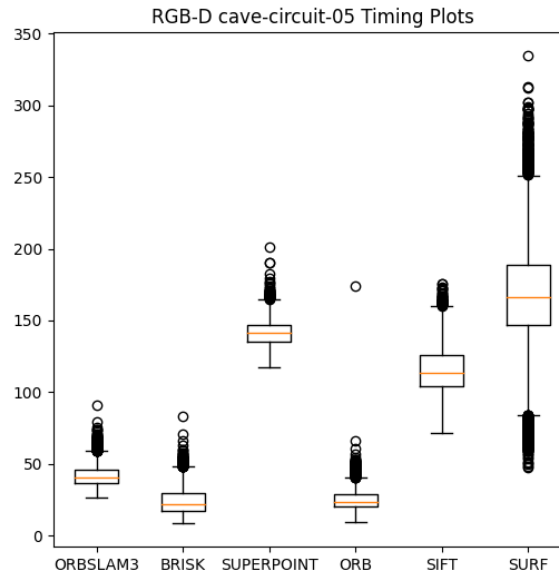


Figure 4.25: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Result Interpretation

An 115.6% increase in localization accuracy is observed when comparing the best performances of Monocular and RGB-D sensor configurations in Tables 4.21, and 4.23. Notably, Superpoint exhibits the best localization accuracy in both configurations. While all methods were able to maintain mapping with Monocular sensor configurations, both BRISK and D2Net encountered difficulties in RGB-D sensor setups. Specifically, during the section where BRISK lost mapping, the UAV entered an open area of the map, with most of the sensed region falling outside the depth sensor's range, seen upon the video available at [103]. D2Net also failed, but during the return journey, in contrast to BRISK, which lost mapping during the initial exploration. The absence of feature point constraints, normally provided by RGB-D sensors in these areas, accounts for the inability to maintain mapping.

CHAPTER 4. RESULTS AND INTERPRETATION

4.6.6 Dataset 6

The sixth dataset was recorded using the `cave_circuit_07` environment from the **DARPA** Subterranean challenge simulation environment. The visual data captured during this exploration are presented in a video, which can be viewed at [\[104\]](#). Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.26.

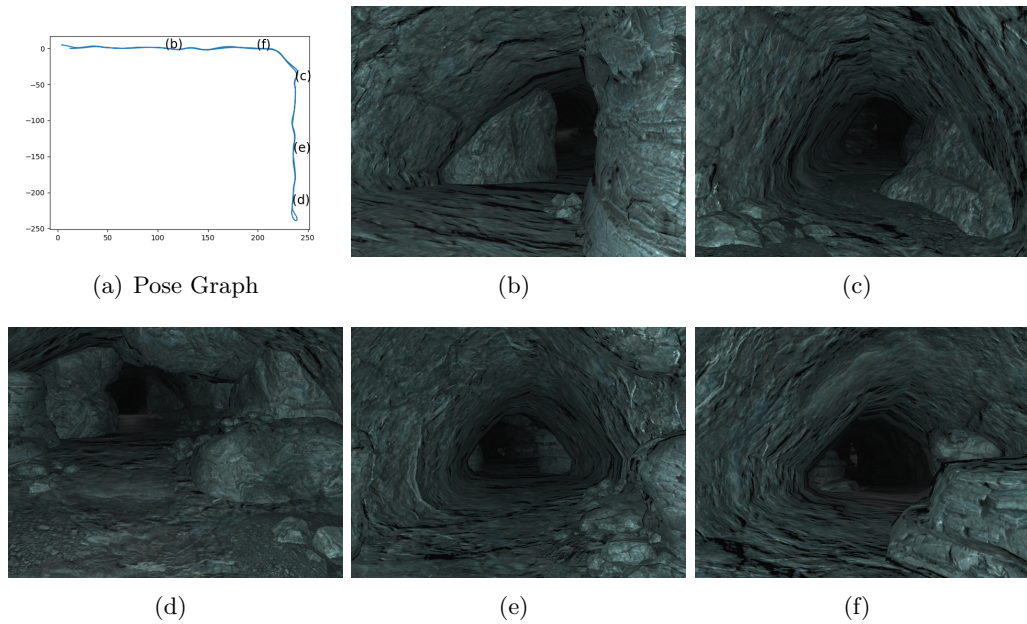


Figure 4.26: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey's path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

Monocular Sensor Configuration

Measure-ment	ZIPPY-POINT	LDSO	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	583.448	593.644	584.0	94.748	583.4	579.9	577.352	582.3	580.644	593.0
APE										
rmse	21.626	8.3	12.371	0.524	12.144	46.024	23.829	35.64	36.929	87.96
mean	17.487	5.736	10.046	0.454	9.492	38.016	19.153	29.065	30.575	73.644
median	14.402	3.28	9.274	0.44	8.75	30.367	13.965	20.918	22.429	59.149
std	12.723	6.0	7.219	0.261	7.575	25.943	14.178	20.625	20.711	48.101
min	2.232	0.79	0.72	0.047	0.776	8.583	2.516	5.097	6.257	3.217
max	56.085	39.65	28.499	1.542	31.925	125.214	62.445	98.884	97.474	208.846
RPE										
rmse	0.12221	0.19064	0.1233	0.08814	0.12312	0.11835	0.12278	0.12701	0.12018	0.11544
mean	0.11626	0.15342	0.11751	0.05955	0.11738	0.11157	0.1169	0.11364	0.11407	0.10172
median	0.11776	0.12338	0.11972	0.00743	0.11951	0.10894	0.11749	0.1124	0.11297	0.09303
std	0.03766	0.11315	0.03733	0.06499	0.03717	0.03949	0.03755	0.05671	0.03784	0.05458
min	0.00028	0.00718	0.00087	0.0	0.00064	0.0011	0.00037	0.00046	0.00032	0.00211
max	0.34602	1.3712	0.31464	0.29977	0.31028	0.33554	0.49841	4.41007	0.31415	1.3899

Table 4.25: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping. **Blue** denotes incomplete mapping or the inability of a method to sustain mapping across the dataset, reflecting challenges in handling complex or sparse environments.

CHAPTER 4. RESULTS AND INTERPRETATION

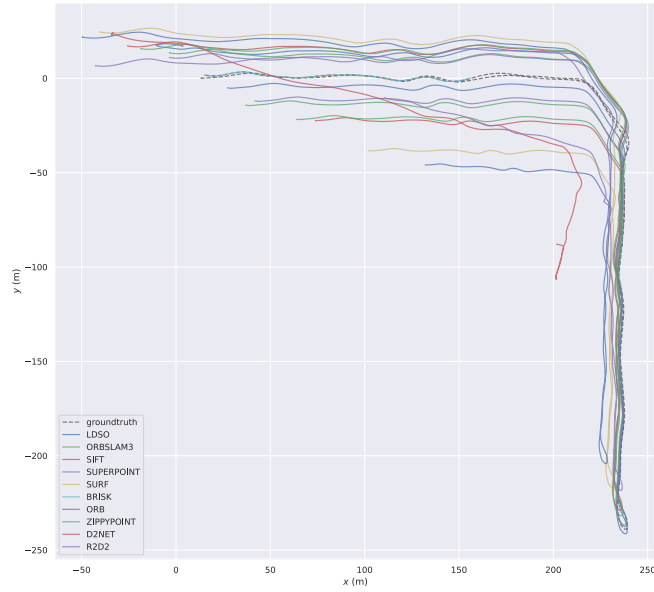


Figure 4.27: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	137.97018	28.81455	20.85218	143.82091	20.11624	109.55837	154.91538
median	161.2655	27.532	19.2157	143.5095	19.1629	108.8855	154.051
std	68.14281	7.22849	8.52483	7.25869	6.39648	13.8138	31.20445
min	4.34906	18.133	6.87089	119.212	8.76172	68.7414	49.356
max	266.196	154.291	55.3205	180.318	434.143	177.751	490.377

Table 4.26: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

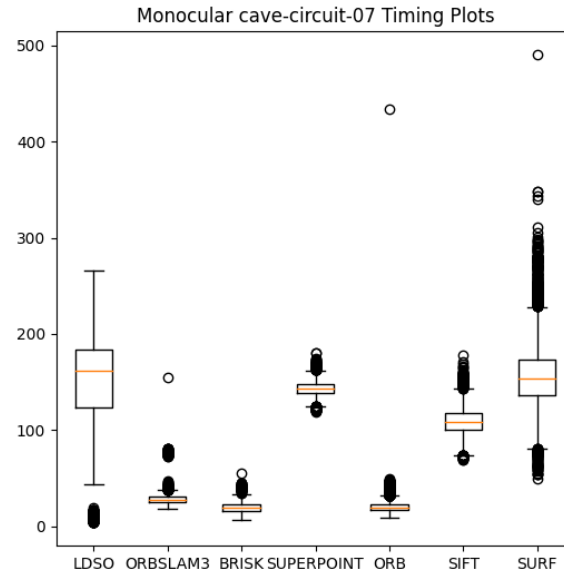


Figure 4.28: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

CHAPTER 4. RESULTS AND INTERPRETATION

RGB-D Sensor Configuration

Measurement	ZIPPY-POINT	ORB-SLAM3	BRISK	SUPER-POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet-SLAM	-	M/ORB-SLAM3	M/HFNet-SLAM	M/ORB-SLAM3	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM	M/HFNet-SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	593.544	593.544	152.8	593.544	593.5	593.3	593.544	593.544	593.544
APE									
rmse	1.057	0.714	0.395	0.859	1.691	1.145	1.356	1.323	6.258
mean	0.979	0.648	0.337	0.785	1.42	1.034	1.216	1.216	5.183
median	0.963	0.565	0.344	0.679	1.279	1.034	1.063	1.276	4.683
std	0.397	0.3	0.207	0.347	0.918	0.492	0.601	0.522	3.507
min	0.089	0.068	0.006	0.274	0.258	0.276	0.227	0.323	0.868
max	2.021	1.543	1.258	1.863	3.856	2.298	3.169	2.099	16.774
RPE									
rmse	0.12295	0.12295	0.12008	0.1229	0.12338	0.12295	0.12293	0.12288	0.12353
mean	0.11633	0.11628	0.10929	0.11627	0.11658	0.11626	0.11624	0.1162	0.11677
median	0.12079	0.12064	0.11923	0.12074	0.12063	0.12095	0.12059	0.12062	0.12086
std	0.03982	0.03996	0.04976	0.03982	0.04042	0.03999	0.03999	0.03995	0.04029
min	0.00033	0.00027	0.00036	0.00068	0.00012	6e-05	0.00024	0.00011	0.00038
max	0.31441	0.31502	0.30381	0.31255	0.32443	0.31381	0.30918	0.31293	0.3205

Table 4.27: This table showcases the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for different feature extraction and description methods utilized within the ORB-SLAM framework, specifically with RGB-D sensor configurations, across recorded datasets. Performance levels are color-coded for immediate visual interpretation: **green** illustrates the top performance, reflecting superior accuracy and dependability in feature mapping and localization. **Red** marks the lowest performance, identifying areas requiring enhancements to fulfill the precision demands of navigation and mapping. **Blue** signals incomplete mapping or a method’s failure to sustain mapping continuity throughout the dataset, indicating challenges in managing complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

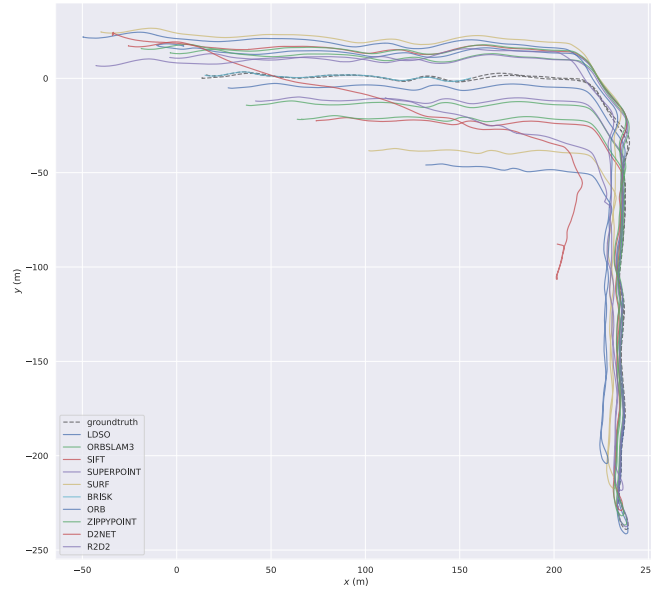


Figure 4.29: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

Measure- ment[ms]	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	41.04565	21.55503	148.00245	23.5854	112.73976	160.87334
median	39.86435	21.1821	147.7225	22.39475	111.0545	160.0375
std	5.93314	6.32412	7.84314	5.89678	15.59258	31.65151
min	26.0146	7.53929	121.358	9.63688	70.3862	55.3497
max	81.9238	42.1389	212.868	185.621	193.263	312.597

Table 4.28: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

CHAPTER 4. RESULTS AND INTERPRETATION

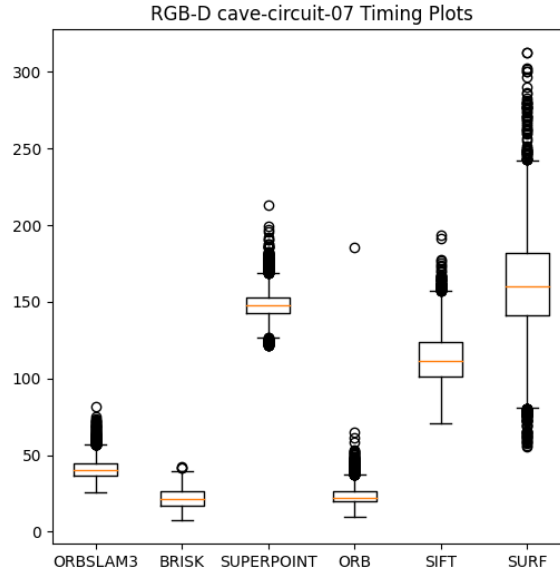


Figure 4.30: RGB-D cave-circuit-07 Timing Box Plot

Result Interpretation

An 168.3% increase in localization accuracy is observed when comparing the best performances of Monocular and RGB-D sensor configurations in Tables 4.25, and 4.27. Notably, ORB-SLAM3, excelling as the top performer in the RGB-D category (Table 4.27), shows a 178.2% improvement. This analysis reveals a consistent pattern, with BRISK failing to maintain mapping and D2Net exhibiting the poorest performance among all the tested methods.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

4.6.7 Dataset 7

The seventh dataset was recorded using the `cave_circuit_08` environment from the **DARPA** Subterranean challenge simulation environment. The visual data captured during this exploration are presented in a video, which can be viewed at the following [105]. Furthermore, a figure illustrating the pose graph, marked with RGB images from key points captured by the front-facing camera, is detailed for enhanced analysis and understanding in Figure 4.31.

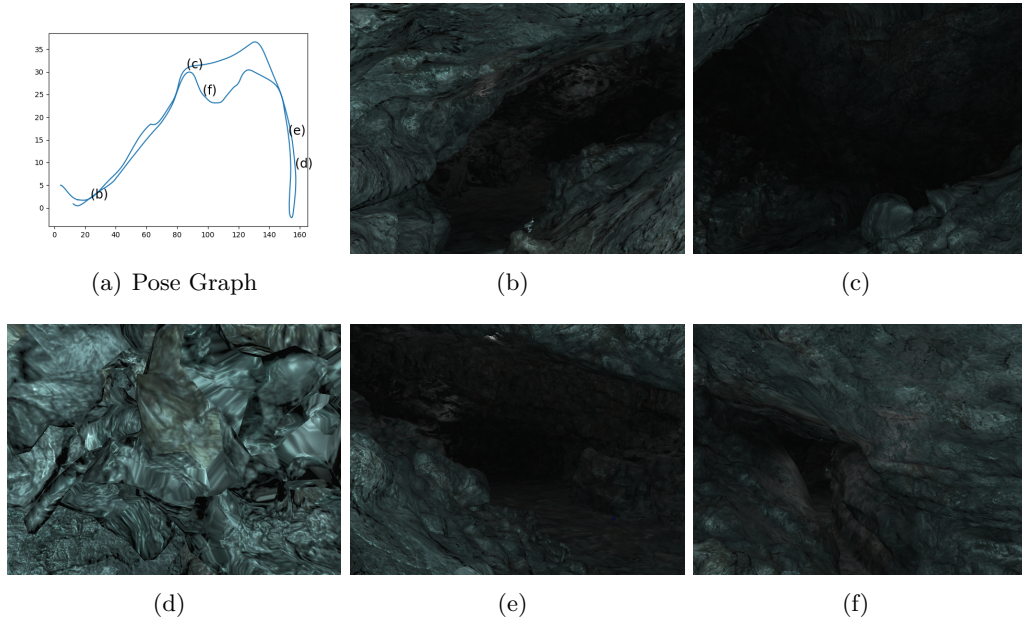


Figure 4.31: The series of front-facing images, labeled (b) through (f), are shown on the traversal pose map marked as (a). These images are taken from key points along the route, providing a visual overview of the environment. The pose map (a) outlines the journey's path and identifies the specific locations of each image (b)-(f), offering a clear understanding of their spatial relationship and enhancing the overall perspective of the exploration area.

CHAPTER 4. RESULTS AND INTERPRETATION

Monocular Sensor Configuration

Measure- ment	ZIPPY- POINT	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	R2D2	SURF	D2NET
SLAM Platform	M/HFNet- SLAM	-	-	M/ORB- SLAM3	M/HFNet- SLAM	M/ORB- SLAM3	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM	M/HFNet- SLAM
Processing Platform	desktop	lightweight	lightweight	lightweight	lightweight	lightweight	lightweight	desktop	lightweight	desktop
Mapping duration	265.792	261.956	266.936	34.348	263.0	59.508	38.616	262.04	261.992	267.64
APE										
rmse	1.49	0.691	1.419	0.216	0.512	0.317	0.178	9.583	2.111	37.235
mean	1.062	0.605	1.025	0.194	0.494	0.272	0.158	8.251	1.553	29.907
median	0.485	0.525	0.515	0.195	0.479	0.211	0.149	8.02	1.005	28.942
std	1.045	0.334	0.981	0.094	0.136	0.162	0.082	4.873	1.43	22.182
min	0.108	0.159	0.039	0.008	0.176	0.024	0.011	0.351	0.049	4.015
max	4.167	1.984	4.071	0.448	0.858	0.741	0.376	19.082	5.589	102.34
RPE										
rmse	0.1134	0.47708	0.11295	0.08329	0.11356	0.08205	0.08455	0.14577	0.17967	0.15584
mean	0.10295	0.38303	0.10235	0.0576	0.10359	0.05636	0.05828	0.10323	0.1137	0.0921
median	0.10987	0.34608	0.1101	0.0227	0.11083	0.02314	0.02544	0.10646	0.11107	0.08573
std	0.04756	0.28443	0.04778	0.06015	0.04653	0.05963	0.06126	0.10291	0.13911	0.12572
min	0.0	0.01429	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.31865	1.98651	0.31971	0.28675	0.30909	0.3999	0.29945	6.42041	3.49905	4.26391

Table 4.29: This table delineates the Absolute Pose Error (APE) and Relative Pose Error (RPE) performance metrics for a selection of visual features within the ORB-SLAM3 framework and **LDSO**, a direct-based **SLAM** method, applied to recorded datasets. Performance is visually coded to facilitate quick assessment: **green** indicates the best performance, demonstrating superior accuracy and reliability in feature mapping and localization. **Red** signifies the worst performance, highlighting areas where improvements are necessary to meet the demands of precise navigation and mapping. **Blue** denotes incomplete mapping or the inability of a method to sustain mapping across the dataset, reflecting challenges in handling complex or sparse environments.

4.6. LOCALIZATION ACCURACY'S AND PROCESSING TIMES ON RECORDED DATASETS

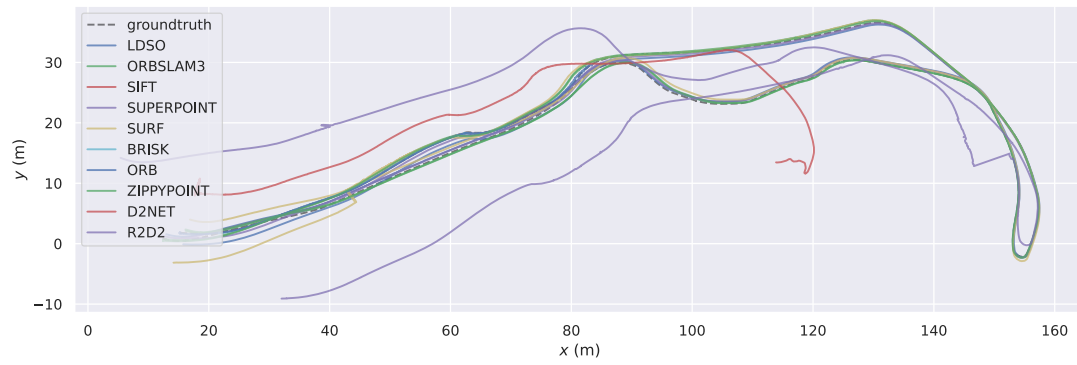


Figure 4.32: This figure displays a 2D representation of the estimated poses obtained from various **SLAM** methods and feature extraction techniques implemented within the ORB-SLAM3 framework, applied to a set of recorded datasets. Each method's pose estimates are depicted using unique markers and colors, providing a clear visual distinction that facilitates comparison.

CHAPTER 4. RESULTS AND INTERPRETATION

Measure- ment[ms]	LDSO	ORB- SLAM3	BRISK	SUPER- POINT	ORB	SIFT	SURF
mean	55.87849	30.93009	12.98055	137.19421	16.43421	86.903	108.03847
median	9.54088	28.5479	12.76385	135.824	15.9018	90.341	81.485
std	78.60344	9.8705	7.65313	10.75046	12.30123	17.50796	39.05715
min	4.09873	18.5551	3.96205	115.657	6.50793	63.3257	68.0682
max	273.873	119.353	44.7555	194.239	401.977	145.243	424.333

Table 4.30: This table outlines the processing time metrics for various **SLAM** methods, including root mean square (RMS), mean, median, standard deviation, minimum (min), and maximum (max) processing times per frame. These metrics provide a quantitative analysis of the computational efficiency and speed of each **SLAM** algorithm, highlighting their performance in handling frame-by-frame processing under various conditions. The detailed statistical data allows for an evaluation of the algorithms' responsiveness and potential bottlenecks, offering insights into their suitability for real-time applications and performance under different operational scenarios.

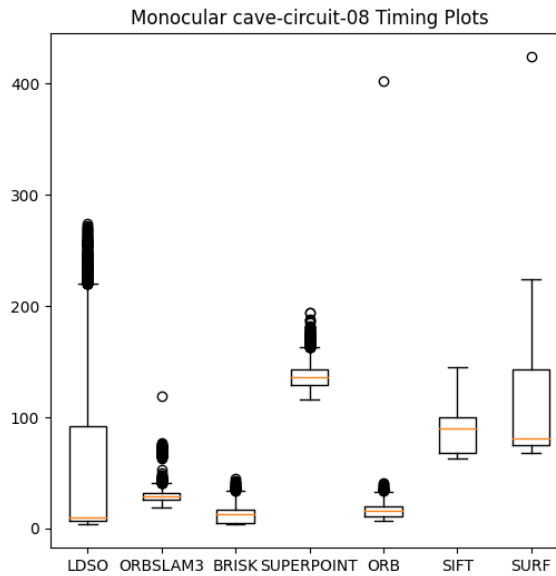


Figure 4.33: This figure presents a box and whisker plot that graphically depicts the distribution of processing time metrics—RMS, mean, median, standard deviation, and range (min to max)—for different **SLAM** methods. This visualization helps in understanding the variability in processing times, identifying any outliers, and comparing the computational efficiency across various **SLAM** algorithms. By analyzing the spread and central tendencies of processing times, we can assess the algorithms' capability to perform in real-time environments and their potential impact on system resources.

4.7. VISUAL FEATURE COMPARISON

Result Interpretation

This dataset, which is already eliminated in the RGB-D sensor configurations due to containing spread-out areas that fall outside the range of the depth sensor, presents challenges for Monocular configurations as well. These extensive areas do not come without complications for Monocular, leading to low-light images and resulting in **BRISK**, **ORB**, and **SIFT** losing mapping, as seen in Table 4.29.

4.7 Visual Feature Comparison

SLAM Method	Times best performer (Monocular)	Times best performer (RGB-D)
SUPERPOINT	3	2
LDSO	4	N/A
ZIPPYPOINT	-	2
SIFT	-	1
ORBSLAM3	-	1

Table 4.31: Number of Times Each Method Produced the Best Localization Accuracy.

The count of instances where a specific method achieved the best evaluation result is tabulated by method in Table 4.31. As anticipated, **LDSO** consistently demonstrates superior performance when considering monocular results. This is attributed to **LDSO** being a direct-based **SLAM** method, aligning estimated depth texture maps to predict odometry between frames. In contrast, feature-based **SLAM** methods rely on the ability to rediscover and match visual features between frames.

The significant variance in performance between **ORB** and ORB-SLAM3 can be attributed to two main factors. Firstly, the discrepancy in default values between OpenCV and ORB-SLAM, particularly in the edge threshold parameter, with OpenCV’s default set at 31 and ORB-SLAM using 19. Secondly, the difference in feature distribution is a contributing factor. ORB-SLAM employs a grid-based feature distribution, dividing the image into grids and stipulating a minimum number of features to be extracted. Adjustments are made to the detector threshold when the required minimum number of features is not detected. This strategic feature distribution in the environment facilitates better triangulation, resulting in improved overall performance.

The **DL**-based feature extractor and descriptor, D2-Net, consistently underperformed in these datasets. This underperformance can largely be attributed to the nature of its training images, which are predominantly well-lit and contain many distinctive features. Consequently, D2-Net is less effective in processing images characterized by low-light conditions and a lack of discriminative features.

4.8 Reference Localization Accuracy

LiDAR-based **SLAM** are generally the first choice when it comes to low-light and expansive environments [79], [82], [92], as a baseline accuracy to compare to, as the current

CHAPTER 4. RESULTS AND INTERPRETATION

implementation in a similar environment use **LiDAR**-based methods. This assume the current accuracy offered by **LiDAR**-based methods are adequate.

Dataset	Sensor Configuration		
	Lidar	Monocular	RGB-D
cave-circuit-01	0.601	1.044	0.715
cave-circuit-02	0.403	0.572	0.341
cave-circuit-03	1.164	5.579	0.619
cave-circuit-04	2.916	16.45	1.235
cave-circuit-05	0.841	3.868	1.035
cave-circuit-07	3.336	8.3	0.714

Table 4.32: Best Performing RMSE Errors of Different Sensor Configurations.

In Table 4.32, **LiDAR** is used as the benchmark for the required localization accuracy in subterranean exploration. Compared to **LiDAR**, Monocular systems consistently delivered inferior performance. This shortfall, coupled with the lack of accurate mapping scale, suggests that localization accuracy obtained through Monocular systems might not be sufficient for subterranean environments. On the other hand, RGB-D systems outperformed **LiDAR** in most scenarios, except in situations where the environment extended beyond the RGB-D sensor’s effective sensing range. This indicates that localization with RGB-D sensor configurations, which also provide scale-accurate mapping, could be suitable for subterranean exploration, provided the environment is within the RGB-D sensor’s range.

4.9 Discussion and Analysis

The results in Table 4.3 clearly shows that the modifications made to the ORB-SLAM framework performs comparably well when using a well trusted dataset compared to the unmodified version, also performing way better when compared to **LDSO**.

Among all methods, **BRISK** struggles the most to maintain consistent mapping, despite showing similar performance to other methods on the EUroC dataset. This could be attributed to the smaller size of the **BRISK** descriptor compared to, for example, **ORB**. While a smaller descriptor may suffice in feature-rich environments, it likely becomes a limiting factor in feature-sparse settings.

LDSO, a direct **SLAM** method, exhibited poor localization accuracy on the EUroC dataset but delivered one of the best accuracies on the captured datasets. This suggests that **LDSO**’s approach, which involves extracting and aligning texture maps from the entire image rather than relying on corner detectors influenced by pixel intensities or image gradients, may be more effective in certain scenarios.

In monocular scenarios, **LDSO** demonstrated the best overall performance, with Superpoint

coming in second when comparing the top performances for each dataset. In RGB-D scenarios, where **LDSO** is not applicable, the best performers were the **DL**-based feature methods. This indicates that while **DL** offers some improvements over conventional feature extraction methods, it does not consistently outperform direct-based methods in monocular scenarios

4.10 Summary

This chapter meticulously outlines the outcomes of research focused on evaluating the performance of various **SLAM** methods in constrained processing environments. Section 4.2 begins by detailing the computing platforms used in the experiments, contrasting a lightweight ASUS FX550J laptop with a more powerful custom-built desktop computer to simulate scenarios with varying computational capacities.

Section 4.3 evaluates modified **SLAM** methods on the well-known EuRoC dataset, demonstrating comparable performance between ORB-SLAM3 with loop closure detection deactivated and the modified **SLAM** methods. This indicates adequate performance on a known dataset and concludes that these modifications do not possess inherent flaws and are capable of accurately initiating and maintaining mapping.

The datasets recorded using the exploration algorithm described in Section 3.1 are discussed next, with the sizes of the obtained explorations tabulated in Table 4.3. Additionally, a link to a video of the dataset with a pose map overlay is provided.

Subterranean exploration surveys, mainly focusing on **LiDAR**-based methods, use a well-known **LiDAR**-based odometry to establish baseline localization accuracies deemed adequate in subterranean exploration. These accuracies are tabulated in Table 4.5.

The recorded datasets are then utilized to obtain performance metrics of different **SLAM** methods, including the modified ones in Section 4.6. These methods are evaluated for both Monocular and RGB-D sensor configurations across datasets that are diverse in length, map size, and distance traveled, with the first dataset discussed in Section 4.6.1 and the largest in terms of duration and size described in Section 4.6.4.

Section 4.7 compares localization accuracies, showing superior performance using **LDSO** compared to other methods, with Superpoint closely following for Monocular sensor configurations. Superpoint and Zippypoint closely compete in the case of RGB-D sensor configurations. Direct-based methods show superior performance in feature-lacking subterranean environments compared to feature-based methods.

The only methods consistently meeting the 100ms specification are ORB-SLAM3, **ORB** , and **BRISK** , with **SIFT** typically being close to the 100ms mark. **LDSO** shows the most variability in processing time, occasionally exceeding 100ms. Superpoint consistently

CHAPTER 4. RESULTS AND INTERPRETATION

requires more processing time, likely due to the computational demands of its **DL** architecture, especially when operated on a GPU with limited capabilities. The C++ to Python interface, used to implement Python features, necessitates additional processing for the serialization and deserialization of images, feature points, and feature descriptors, as well as the time required to transmit these values over localhost.

In Section 4.8, the focus shifts to evaluating localization accuracy in subterranean environments, highlighting the baseline accuracy offered by **LiDAR**-based methods as a point of comparison. This benchmark is crucial, considering the current implementations in similar environments predominantly rely on **LiDAR** for accuracy. The comparative analysis presented in Table 4.32 showcases the performance of different sensor configurations across various datasets, demonstrating that while Monocular systems consistently underperform against **LiDAR**, RGB-D systems occasionally surpass **LiDAR**'s accuracy in environments within their effective sensing range. This suggests that for subterranean exploration, RGB-D sensor configurations, with their scale-accurate mapping capabilities, could offer a viable alternative to **LiDAR**, provided the exploration environment falls within the RGB-D sensor's operational limitations.

Chapter 5

Conclusion

Our investigation has delved into the feasibility of implementing visual **SLAM** methods in subterranean environments, with a specific focus on exploring various feature extraction and description techniques. Despite the absence of scale-accurate mapping, monocular visual methods exhibited lower localization accuracy compared to **LiDAR**-based methods. In contrast, RGB-D sensor configurations provided scale-accurate mapping and demonstrated improvements over **LiDAR**-based methods, particularly within the operational range of the depth sensor. However, in less constrained environments like cave-circuit-08, RGB-D methods faced challenges in maintaining consistent mapping.

We also investigated integrating various feature extraction and description methods into the ORB-SLAM3 framework, which led to improved localization accuracy. Additional enhancements could be realized by dedicating more processing time to pose graph optimization and feature correspondences between keyframes. Notably, deep learning methods such as Zippypoint and Superpoint generally outperformed conventional methods, although there were some exceptions.

While our evaluation on the widely recognized EUroC dataset showed comparable results with no significant improvement over conventional ORB-SLAM3, our performance on subterranean datasets revealed a different picture. **LDSO** exhibited significantly lower localization accuracy on the EUroC dataset but demonstrated comparable to improved performance in subterranean scenarios.

Interestingly, our modifications to ORB-SLAM3, such as the removal of **DBoW** for keyframe correspondence, resulted in substantial performance differences. Specifically, the removal of **DBoW** was intended to reduce false positives in keyframe associations, which could otherwise adversely impact the pose graph optimization process. Instead, we introduced direct feature matching, which enhanced the robustness of keyframe matching, particularly in challenging environments. Furthermore, we transitioned from **DBoW2**, the version used in ORB-SLAM3, to **DBoW3**, a more advanced version that includes custom training vocabularies tailored to each feature extraction method (e.g., **ORB**,

CHAPTER 5. CONCLUSION

BRISK, Superpoint). This shift is anticipated to lead to notable improvements in feature matching and loop closure detection accuracy.

In addition to the **DBoW** modifications, deep learning-based feature detectors and descriptors (such as Superpoint and D2Net) were integrated into the ORB-SLAM3 framework. These methods were chosen due to their ability to handle more complex features in dynamic or texture-poor environments, which are common in the datasets used. The result was a measurable improvement in localization accuracy for both monocular and RGB-D configurations. However, the inclusion of these deep learning methods introduced an increase in computational complexity, leading to longer processing times. Future work may focus on optimizing these methods for embedded platforms by exploring feature sparsification techniques or reducing the network complexity.

To effectively address confined and expansive environments, we propose the fusion of **LiDAR** and visual sensor configurations, potentially complemented by **IMU** sensors. This approach can accommodate a wide range of scenarios, from well-lit conditions to darkness, and has the potential to further enhance localization accuracy while remaining suitable for lightweight systems.

However, it is important to note that the use of identical textures in different areas of the simulation environment led to incorrect loop closure detections. This issue is likely to be mirrored in real-world scenarios, especially in environments with similar rock formations or identical rock types, like in mining tunnels. It is difficult to determine the exact extent to which this similarity would occur. The reason for these false detections is that texture descriptors, used in the simulation, matched between different sections of the environment. This makes it challenging to accurately evaluate the effectiveness of loop closure detection methods in simulated settings. For meaningful testing of these techniques, particularly in subterranean environments, real-world data is essential. Furthermore, the acquisition of accurate ground truth pose data in vast subterranean environments poses a significant challenge.

5.1 Future Work

In future work, we aim to leverage the recorded dataset to refine and test **SLAM** methods that integrate visual, **LiDAR**, and **IMU** sensors. These datasets will also be instrumental in developing deep learning models for autonomous navigation, mapping, and localization in subterranean environments. To validate these methods with real-world data, efforts will be directed towards obtaining funding for an **UAV** equipped with the necessary sensors to collect extensive datasets in such environments. The challenge lies in acquiring accurate ground truth poses over vast distances within subterranean settings.

The potential of transformer-based models in visual processing, due to their ability to discern complex patterns and relationships, opens new avenues for enhancing visual

5.1. FUTURE WORK

localization techniques. These models could revolutionize feature extraction, description, and improve the accuracy of visual odometry and loop closure detection.

Our future research will focus on employing exploration and dataset recording programs alongside **LiDAR**-based SLAM for precise pose estimation in real-world drone explorations of mining tunnels. This approach aims to capture real-world datasets that mirror those obtained in simulated environments. The simulated datasets will then be utilized to train deep learning models for feature extraction and description, specifically designed for subterranean contexts. These real-world datasets will serve as a benchmark for evaluating the performance of SLAM methods, including the deep learning-based feature extraction techniques developed through this research.

Additionally, an area for potential future research is the evaluation of the framework's robustness to varying lighting conditions. Incorporating snapshots of darker segments along with a comparative analysis of results would provide valuable insights into how different visual feature extraction and description methods perform under challenging lighting scenarios. Implementing this would necessitate substantial modifications to the current simulation environment to accurately capture and assess the impact of lighting variations. While such an analysis was outside the scope of this study, it presents a promising direction for future work to further validate and enhance the robustness of the proposed methods.

References

- [1] K. Shoemake, “Animating rotation with quaternion curves,” in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.
- [2] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986. DOI: [10.1177/027836498600500404](https://doi.org/10.1177/027836498600500404).
- [3] R. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *The International Journal of Robotics Research*, vol. 5, Feb. 1987. DOI: [10.1177/027836498600500404](https://doi.org/10.1177/027836498600500404).
- [4] H. Durrant-Whyte, “Uncertain geometry in robotics,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988. DOI: [10.1109/56.768](https://doi.org/10.1109/56.768).
- [5] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Autonomous robot vehicles*, Springer, 1990, pp. 167–193.
- [6] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.
- [7] H. Durrant-Whyte, D. Rye, and E. Nebot, “Localization of autonomous guided vehicles,” *Robotics Research*, pp. 613–625, 1996.
- [8] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment—a modern synthesis,” in *International workshop on vision algorithms*, Springer, 1999, pp. 298–372.
- [9] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, “Fastslam: A factored solution to the simultaneous localization and mapping problem,” *Aaai/iaai*, vol. 593598, 2002.
- [10] Sivic and Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Proceedings ninth IEEE international conference on computer vision*, IEEE, 2003, pp. 1470–1477.
- [11] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3. DOI: [10.1109/IROS.2004.1389727](https://doi.org/10.1109/IROS.2004.1389727).

REFERENCES

- [12] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
- [13] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *European conference on computer vision*, Springer, 2006, pp. 430–443.
- [14] D. Arthur and S. Vassilvitskii, “K-means++ the advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.
- [15] A. Bosch, A. Zisserman, and X. Munoz, “Representing shape with a spatial pyramid kernel,” in *Proceedings of the 6th ACM international conference on Image and video retrieval*, 2007, pp. 401–408.
- [16] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [17] L. M. Paz, P. Jensfelt, J. D. Tardos, and J. Neira, “EKF slam updates in $O(n)$ with divide and conquer slam,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 1657–1663.
- [18] M. Agrawal, K. Konolige, and M. R. Blas, “Censur: Center surround extremas for realtime feature detection and matching,” in *European conference on computer vision*, Springer, 2008, pp. 102–115.
- [19] G. Klein and D. Murray, “Improving the agility of keyframe-based slam,” in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12–18, 2008, Proceedings, Part II 10*, Springer, 2008, pp. 802–815.
- [20] W. He, T. Yamashita, H. Lu, and S. Lao, “Surf tracking,” in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 1586–1592.
- [21] M. Quigley, K. Conley, B. P. Gerkey, *et al.*, “Ros: An open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.
- [22] E. Tola, V. Lepetit, and P. Fua, “Daisy: An efficient dense descriptor applied to wide-baseline stereo,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 815–830, 2009.
- [23] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, “Adaptive and generic corner detection based on the accelerated segment test,” in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5–11, 2010, Proceedings, Part II 11*, Springer, 2010, pp. 183–196.
- [24] P. F. Alcantarilla and T. Solutions, “Fast explicit diffusion for accelerated features in nonlinear scale spaces,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1281–1298, 2011.

REFERENCES

- [25] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3607–3613.
- [26] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *2011 International conference on computer vision*, Ieee, 2011, pp. 2548–2555.
- [27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.
- [28] A. Alahi, R. Ortiz, and P. Vandergheynst, “Freak: Fast retina keypoint,” in *2012 IEEE conference on computer vision and pattern recognition*, Ieee, 2012, pp. 510–517.
- [29] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “Kaze features,” in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part VI 12*, Springer, 2012, pp. 214–227.
- [30] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 2911–2918.
- [31] D. Gálvez-López and J. D. Tardós, “Bags of binary words for fast place recognition in image sequences,” *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012, ISSN: 1552-3098. DOI: [10.1109/TR0.2012.2197158](https://doi.org/10.1109/TR0.2012.2197158).
- [32] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *European conference on computer vision*, Springer, 2014, pp. 834–849.
- [33] C. Forster, M. Pizzoli, and D. Scaramuzza, “Svo: Fast semi-direct monocular visual odometry,” in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 15–22.
- [34] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, “Microsoft coco: Common objects in context,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.
- [35] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1573–1585, 2014.
- [36] T. Trzcinski, M. Christoudias, and V. Lepetit, “Learning image descriptors with boosting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 597–610, 2014.
- [37] W.-C. Chiu and M. Fritz, “See the difference: Direct pre-image reconstruction and pose estimation by differentiating hog,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 468–476.

REFERENCES

- [38] X. Gao and T. Zhang, “Loop closure detection for visual slam systems using deep neural networks,” in *2015 34th Chinese control conference (CCC)*, IEEE, 2015, pp. 5851–5856.
- [39] Itseez, *Open source computer vision library*, <https://github.com/itseez/opencv>, 2015.
- [40] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [41] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, “Netvlad: Cnn architecture for weakly supervised place recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.
- [42] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, “Learning local feature descriptors with triplets and shallow convolutional neural networks,” in *Bmvc*, vol. 1, 2016, p. 3.
- [43] M. Burri, J. Nikolic, P. Gohl, *et al.*, “The euroc micro aerial vehicle datasets,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.
- [44] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, “University of michigan north campus long-term vision and lidar dataset,” *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.
- [45] G. Levi and T. Hassner, “Latch: Learned arrangements of three patch codes,” in *2016 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2016, pp. 1–9.
- [46] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [47] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, “Pixelwise view selection for unstructured multi-view stereo,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [48] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, “Probabilistic data association for semantic slam,” in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 1722–1729.
- [49] Z. Chen, A. Jacobson, N. Sünderhauf, *et al.*, “Deep learning features at scale for visual place recognition,” in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3223–3230.
- [50] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, “Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.
- [51] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

REFERENCES

- [52] X. Gao and T. Zhang, “Unsupervised learning to detect loops using deep neural networks for visual slam system,” *Autonomous robots*, vol. 41, pp. 1–18, 2017.
- [53] M. Grupp, *Evo: Python package for the evaluation of odometry and slam*. <https://github.com/MichaelGrupp/evo>, 2017.
- [54] K. Leung, D. Lühr, H. Houshiar, *et al.*, “Chilean underground mine dataset,” *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 16–23, 2017.
- [55] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The oxford robotcar dataset,” *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
- [56] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks,” in *2017 IEEE International Conference on Robotics and automation (ICRA)*, IEEE, 2017, pp. 4628–4635.
- [57] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [58] T. Naseer, G. L. Oliveira, T. Brox, and W. Burgard, “Semantics-aware visual localization under challenging perceptual conditions,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2614–2620.
- [59] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, “Large-scale image retrieval with attentive deep local features,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3456–3465.
- [60] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in *Field and Service Robotics*, 2017. eprint: [arXiv:1705.05065](https://arxiv.org/abs/1705.05065). [Online]. Available: <https://arxiv.org/abs/1705.05065>.
- [61] K. Tateno, F. Tombari, I. Laina, and N. Navab, “Cnn-slam: Real-time dense monocular slam with learned depth prediction,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.
- [62] Y. Tian, B. Fan, and F. Wu, “L2-net: Deep learning of discriminative patch descriptor in euclidean space,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 661–669.
- [63] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, “Codeslam—learning a compact, optimisable representation for dense visual slam,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2560–2568.
- [64] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

REFERENCES

- [65] X. Gao, R. Wang, N. Demmel, and D. Cremers, “Ldso: Direct sparse odometry with loop closure,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2198–2204.
- [66] R. Li, S. Wang, Z. Long, and D. Gu, “Undeepvo: Monocular visual odometry through unsupervised deep learning,” in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 7286–7291.
- [67] Z. Li and N. Snavely, “Megadepth: Learning single-view depth prediction from internet photos,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2041–2050.
- [68] Z. Luo, T. Shen, L. Zhou, *et al.*, “Geodesc: Learning local descriptors by integrating geometry constraints,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 168–183.
- [69] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “Lf-net: Learning local features from images,” *Advances in neural information processing systems*, vol. 31, 2018.
- [70] C. Yu, Z. Liu, X.-J. Liu, *et al.*, “Ds-slam: A semantic visual slam towards dynamic environments,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1168–1174.
- [71] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, “Key. net: Keypoint detection by handcrafted and learned cnn filters,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5836–5844.
- [72] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, “Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6713–6719.
- [73] M. Dusmanu, I. Rocco, T. Pajdla, *et al.*, “D2-net: A trainable cnn for joint detection and description of local features,” *arXiv preprint arXiv:1905.03561*, 2019.
- [74] Z. Luo, T. Shen, L. Zhou, *et al.*, “Contextdesc: Local descriptor augmentation with cross-modality context,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2527–2536.
- [75] J. Revaud, P. Weinzaepfel, C. De Souza, *et al.*, “R2d2: Repeatable and reliable detector and descriptor,” *arXiv preprint arXiv:1906.06195*, 2019.
- [76] J. Tang, H. Kim, V. Guizilini, S. Pillai, and R. Ambrus, “Neural outlier rejection for self-supervised keypoint learning,” *arXiv preprint arXiv:1912.10615*, 2019.
- [77] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, “Sosnet: Second order similarity regularization for local descriptor learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 016–11 025.
- [78] I. Ali, A. Durmush, O. Suominen, *et al.*, “Finnforest dataset: A forest landscape for visual slam,” *Robotics and Autonomous Systems*, vol. 132, p. 103 610, 2020.

REFERENCES

- [79] K. Ebadi, Y. Chang, M. Palieri, *et al.*, “Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 80–86.
- [80] D. Li, X. Shi, Q. Long, *et al.*, “Dxslam: A robust and efficient visual slam system with deep features,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 4958–4965.
- [81] R. Li, S. Wang, and D. Gu, “Deepslam: A robust monocular slam system with unsupervised deep learning,” *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2020.
- [82] J. Martz, W. Al-Sabban, and R. N. Smith, “Survey of unmanned subterranean exploration, navigation, and localisation,” *IET Cyber-Systems and Robotics*, vol. 2, no. 1, pp. 1–13, 2020.
- [83] A. R. Memon, H. Wang, and A. Hussain, “Loop closure detection using supervised and unsupervised deep neural networks for monocular slam systems,” *Robotics and Autonomous Systems*, vol. 126, p. 103470, 2020.
- [84] T. Qin, T. Chen, Y. Chen, and Q. Su, “Avp-slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5939–5945.
- [85] T. Rouček, M. Pecka, P. Čížek, *et al.*, “Darpa subterranean challenge: Multi-robotic exploration of underground environments,” in *Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6*, Springer, 2020, pp. 274–290.
- [86] W. Wang, D. Zhu, X. Wang, *et al.*, “Tartanair: A dataset to push the limits of visual slam,” 2020.
- [87] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, “D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.
- [88] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [89] C. Qin, Y. Zhang, Y. Liu, and G. Lv, “Semantic loop closure detection based on graph matching in multi-objects scenes,” *Journal of Visual Communication and Image Representation*, vol. 76, p. 103072, 2021.
- [90] M. Wang, X. Zhang, Y. Ou, and Z. Chen, “Slam loop closure detection and verification based on the improved siamese network,” in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2021, pp. 1–6. DOI: [10.1109/CISP-BMEI53629.2021.9624460](https://doi.org/10.1109/CISP-BMEI53629.2021.9624460).

REFERENCES

- [91] E. Aldao, L. M. González-de Santos, and H. González-Jorge, “Lidar based detect and avoid system for uav navigation in uam corridors,” *Drones*, vol. 6, no. 8, 2022, ISSN: 2504-446X. DOI: [10.3390/drones6080185](https://doi.org/10.3390/drones6080185). [Online]. Available: <https://www.mdpi.com/2504-446X/6/8/185>.
- [92] H. Azpúrua, M. Saboia, G. M. Freitas, *et al.*, “A survey on the autonomous exploration of confined subterranean spaces: Perspectives from real-word and industrial robotic deployments,” *Robotics and Autonomous Systems*, vol. 160, p. 104304, 2023.
- [93] M. Kanakis, S. Maurer, M. Spallanzani, A. Chhatkuli, and L. Van Gool, “Zippypoint: Fast interest point detection, description, and matching through mixed precision discretization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6113–6122.
- [94] OpenAI, *Chatgpt: Conversational language model*, <https://www.openai.com>, Accessed on: Nov-Dec 2023, 2023.
- [95] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, “KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 1029–1036, 2023. DOI: [10.1109/LRA.2023.3236571](https://doi.org/10.1109/LRA.2023.3236571).
- [96] Feb. 2024. [Online]. Available: <https://www.techpowerup.com/gpu-specs/geforce-gtx-950m.c2642>.
- [97] Feb. 2024. [Online]. Available: <https://www.techpowerup.com/gpu-specs/geforce-gtx-1080-ti.c2877>.
- [98] Johan Steenkamp, *Darpa subt challenge simulation environment datasets (with pose graph)*, <https://www.youtube.com/playlist?list=PL0J0JKDfE-uYr8E0wi3EX0KnIbBlfQOLs>, Accessed: 28-Feb-2024, 2024.
- [99] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 01)*, https://youtu.be/Od_Kgq5VjII, Accessed: 28-Feb-2024, 2024.
- [100] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 02)*, <https://youtu.be/92toTpBqI-g>, Accessed: 28-Feb-2024, 2024.
- [101] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 03)*, <https://youtu.be/8iI3NBm68ZU>, Accessed: 28-Feb-2024, 2024.
- [102] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 04)*, <https://youtu.be/D75xNwUzPnM>, Accessed: 28-Feb-2024, 2024.
- [103] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 05)*, <https://youtu.be/ACokgw3IH48>, Accessed: 28-Feb-2024, 2024.
- [104] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 07)*, <https://youtu.be/ggUEAHQ08Ko>, Accessed: 28-Feb-2024, 2024.
- [105] Johan Steenkamp, *Exploring the darpa subterranean challenge (cave circuit 08)*, <https://youtu.be/NUzSDNkzL10>, Accessed: 28-Feb-2024, 2024.

REFERENCES

- [106] J. Steenkamp, *Modified_slam*, <https://github.com/JohanSteenkamp264/Subt-challenge-autonomous-exploration>, 2024.
- [107] J. Steenkamp, *Modified_slam*, https://github.com/JohanSteenkamp264/modified_SLAM, 2024.
- [108] [Online]. Available: <https://www.intel.com/content/www/us/en/products/sku/75027/intel-core-i54200h-processor-3m-cache-up-to-3-40-ghz/specifications.html>.
- [109] [Online]. Available: <https://www.amd.com/en/product/8456>.
- [110] dorian3d, *Dbow*, <https://github.com/dorian3d/DBow>.
- [111] dorian3d, *Dbow2*, <https://github.com/dorian3d/DBow2>.
- [112] Hexadesign, *Asus fx550jx (dm109t) specifications, tests and review*. [Online]. Available: <https://www.productindetail.com/pn/asus-fx550jx-dm109t>.
- [113] rmsalinas, *Dbow3*, <https://github.com/rmsalinas/DBow3>.

Appendix A

Data Format

This section outlines the formats utilized for data collection from the simulation environment.

A.1 RGB Images

Each dataset is stored in a separate folder. Subfolders, namely `rgb`, `rgb_up`, and `rgb_down`, are created to save visual images as `.png` files. Corresponding text files listing these images are generated.

```
Example:      # Time[s],file path
              1.656,/rgb/0000000000.png
              1.704,/rgb/0000000001.png
```

A.2 Depth

Depth information is stored in folders named `depth`, `depth_up`, and `depth_down`, utilizing `.npy` files.

```
Example:      # Time[s], file path
              1.656,/depth/0000000000.npy
              1.704,/depth/0000000001.npy
```

A.3 LiDAR

LiDAR data is stored in `.npy` files within a folder labeled `lidar`. A text file logs the timestep of each **LiDAR** measurement.

```
Example:      # Time[s], file path
              1.704,/lidar/0000000000.npy
              1.804,/lidar/0000000001.npy
```

APPENDIX A. DATA FORMAT

A.4 IMU

The **IMU** data is solely stored in a text file, formatted as shown below:

```
# Time[s], ax, ay, az, wx, wy, wz
Example: 0.0,0.065,-0.05,9.725,-0.018,0.0125,0.002
          0.0,0.025,-0.055,9.74,-0.00525,0.00875,0.00275
```

A.5 Magnetometer

Magnetometer data are stored in a format similar to the **IMU**.

```
# Time[s], mx[Tesla], my[Tesla], mz[Tesla]
Example: 1.652,6.04e-06,2.33e-05,-4.27e-05
          1.696,6.04e-06,2.33e-05,-4.27e-05
```

A.6 Pose

Ground truth pose information is stored in a text file named groundtruth.

```
# Time[s], px[m], py[m], pz[m], qx, qy, qz, qw
Example: 1.668,4.00,5.00,0.05,-1.57e-08,1.44e-07,-1.73e-09,1.0
          1.72,4.00,5.00,0.05,-1.57e-08,1.44e-07,-1.73e-09,1.0
```

A.7 Camera Calibration

DARPA Subterranean Challenge [85] simulation environment provides a camera calibration ROS[21] topic of message type CameraInfo containing the camera and depth sensor calibration data.

CameraInfo	
height	480
width	640
distortion_model	"plumb_bob"
D	[0.0, 0.0, 0.0, 0.0, 0.0]
K	[554.25469, 0.0, 320.5, 0.0, 554.25469, 240.5, 0.0, 0.0, 1.0]
R	[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P	[554.25469, 0.0, 320.5, -0.0, 0.0, 554.25469, 240.5, 0.0, 0.0, 1.0, 0.0]

Table A.1: Camera Calibration Parameters

The camera calibration parameters provided by the simulation environment shown as vectors for simplicity in Table A.1 include the following matrices:

A.8. IMU NOISE

- **D (Distortion Coefficients):** This matrix contains the distortion parameters. For the plumb bob model, these typically include radial distortion coefficients (k_1, k_2, k_3) and tangential distortion coefficients (p_1, p_2) . It is represented as:

$$D = [k_1, k_2, p_1, p_2, k_3]$$

- **K (Camera Matrix):** The intrinsic camera matrix that includes the focal lengths (f_x, f_y) and the optical center (c_x, c_y) of the camera. It is a 3x3 matrix defined as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **R (Rectification Matrix):** A 3×3 matrix used to rotate the camera coordinate system to align it with the ideal image plane. This is particularly useful in stereo vision systems to ensure that the epipolar lines are parallel to the horizontal axis of the image. It is given by:

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

- **P (Projection Matrix):** This matrix maps the 3D camera coordinates to 2D image coordinates, incorporating the effects of the camera intrinsic parameters (K), the rectification (R), and the cameras extrinsic parameters. For a monocular setup, it is a 3×4 matrix that can be expressed as:

$$P = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The Projection Matrix (P) combines the intrinsic parameters of the camera (K) with the extrinsic parameters, which typically include the rotation and translation matrices in more complex setups. The Rectification Matrix (R) is used in stereo vision systems to align images from multiple cameras, and it can be incorporated into P in such systems to correct for lens distortions and misalignments.

A.8 IMU Noise

The average and standard deviation of the noise added to the accelerometer (μ_a, σ_a) and gyroscope (μ_g, σ_g) are specified in the spawner.rb file within the submitted models directory for the utilized model configuration. These values are based on an ADIS16448

APPENDIX A. DATA FORMAT

IMU sensor and assume gyroscope bias and drift compensation have already been applied, resulting in a mean around each axis being zero.

Sensor	Axis		
	x	y	z
Accelerometer			
μ_a	0.0	0.0	0.05
σ_a	0.1105	0.1261	0.00947
Gyroscope			
μ_g	0.0	0.0	0.0
σ_g	0.004	0.004	0.004

Table A.2: IMU Noise Added by Simulation